Thèse de Doctorat
de l'université Pierre et Marie Curie (Paris VI)
Spécialité : **Informatique**
Option : **Intelligence Artificielle**

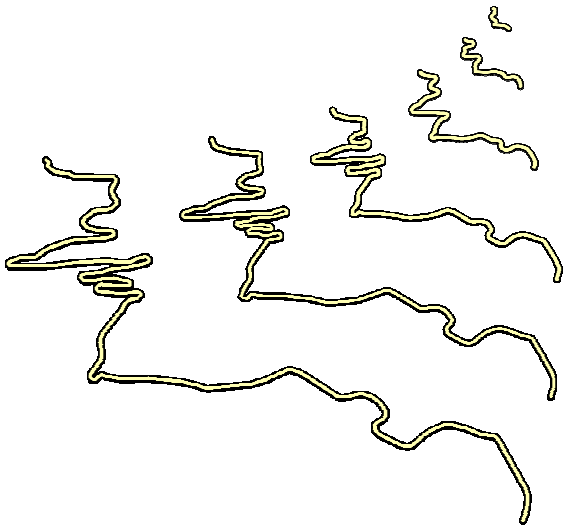
Présentée par :
Sébastien Mustière

Pour obtenir le grade de :
Docteur de l'Université de Paris VI

Intitulée :

Apprentissage Supervisé pour la Généralisation Cartographique

Soutenue le **8 juin 2001**



Jury :

Patrice Boursier	Examinateur
Jean-Gabriel Ganascia	Directeur
Anne Ruas	Présidente
Lorenza Saitta	Rapporteur
Robert Weibel	Rapporteur
Jean-Daniel Zucker	Co-directeur, Encadrant

Images en couverture : résultats, sur une route représentée à différentes échelles, du traitement de généralisation cartographique issu du processus d'apprentissage présenté dans ce mémoire.

Remerciements

Mes travaux doivent beaucoup à Jean-Daniel Zucker en matière d'apprentissage automatique et à Anne Ruas en matière de généralisation cartographique automatique. J'espère que cette thèse reflète au moins en partie la rencontre de leurs idées

.

Merci à Jean-Daniel Zucker, de l'équipe ACASA du LIP6, pour avoir suivi de près cette thèse, pour avoir porté un grand intérêt à ce travail, pour m'avoir prodigué de nombreux et pertinents conseils, et pour m'avoir fait découvrir autant la logique que les techniques d'apprentissage. Merci pour son enthousiasme communicatif et revigorant qui n'a jamais diminué et a toujours été d'un grand soutien.

Merci à Anne Ruas, directrice du laboratoire COGIT, pour avoir inspiré le sujet de cette thèse ainsi que les grandes lignes en généralisation de ce travail. Merci pour son soutien et ses remarques toujours pertinentes et constructives.

Merci à François Lecordix, de l'IGN, pour son travail sur le formidable outil d'expérimentation PlaGe, pour son travail réalisé en généralisation du réseau routier, pour sa chasse au bugs, pour ses idées qui ont fortement influencées GALBE, et pour avoir permis que cet algorithme soit utilisé dans les services de production de l'IGN.

Merci à Jean-Gabriel Ganascia pour m'avoir accueilli au sein de l'équipe ACASA du LIP6.

Merci à Gérald Weger, mon premier professeur de cartographie à l'ENSG, pour m'avoir transmis sa passion de ce domaine, et pour avoir accepté de commenter certains des résultats de cette thèse durant sa retraite.

Merci à Nicolas Regnauld, de l'université d'Edimbourgh, pour m'avoir fourni de nombreux exemples nécessaires à mon travail et pour son aide de manière générale.

Merci à Lorenza Saitta, spécialiste de l'apprentissage automatique, pour avoir accepté de poser son regard très pertinent sur mes travaux.

Merci à Robert Weibel, directeur du groupe de travail de l'ACI en généralisation, pour avoir accepté d'évaluer cette thèse inspirée de ses premiers travaux en apprentissage automatique pour la généralisation cartographique.

Merci à Bénédicte Bucher la KADSienne, pour ses conseils dans ce domaine, et son soutien permanent.

Merci aux membres de l'équipe ACASA et du laboratoire COGIT, et plus particulièrement à tous ceux avec qui j'ai travaillé. Merci à tous ceux qui ont relu tout ou partie de cette thèse : Anne, Bénédicte, Cécile, François, Jean-Daniel, Jean-François, Jenny, Hakima, Nicolas, Sylvain, Yolène.

Merci Cécile…

INTRODUCTION

◆ Dans le contexte actuel de l'information géographique numérique, l'automatisation de la généralisation cartographique est de plus en plus utile.

◆ Comment recueillir les connaissances nécessaires au guidage de ce processus complexe ? Est-ce possible de le faire par apprentissage automatique ?

i. Contexte : la cartographie

Jusqu'au milieu du XX^{ème} siècle, l'acquisition des données géographiques constituait le principal enjeu scientifique de la cartographie. A l'époque, le rôle du cartographe couvrait les domaines de la géographie, de la découverte et de la politique, restreignant ainsi la conception même de la carte –ce que nous appelons aujourd'hui la cartographie – à un art ou un savoir-faire graphique.

Durant le siècle dernier, la précision et la vitesse d'acquisition des données ont été accrues par l'invention de la photogrammétrie aérienne[1], des satellites d'observation de la terre, des systèmes de positionnement par satellites et de l'informatique. Dans le même temps, la conception de la carte a elle aussi profité des révolutions technologiques, mais surtout elle a cessé d'être considérée comme relevant du seul domaine artistique. Afin de remédier aux nombreuses erreurs d'interprétation rencontrées et à l'inefficacité de certaines cartes, les cartographes ont plaidé pour une approche plus scientifique de la conception cartographique.

Ceci a amené à considérer la carte sous plusieurs angles théoriques [Pravda 95], du paradigme de la carte *communication* à celui de la carte *représentation* ou *modèle* qui parfois considère la cartographie comme un *langage*. Ces approches s'opposent sur certains points, se complètent sur d'autres, et chacune a privilégié certaines directions de recherche.

L'approche *communication* considère que le cartographe "émetteur" transmet un message au lecteur "récepteur" par l'intermédiaire de la carte "médium". Elle a conduit à étudier en particulier les influences du "filtre" du choix de la représentation et du "filtre" de la perception de la carte par le lecteur sur la perception du message. Ce lecteur est alors le plus souvent considéré, dans une approche béhavioriste, comme une boite noire réagissant aux stimuli des symboles de la carte [Board 67 ; Kolácňý 69 ; Dobson 85].

L'approche *représentation* considère la carte comme une représentation particulière de l'espace qui permet de faire apparaître des relations spatiales inconnues. Elle a en particulier guidé l'étude du choix des objets à représenter et de la manière dont le lecteur, dans une approche cognitive, construit de l'information à partir de la carte [Petchenik 75 ; Eastman 85 ; MacEachren 95]. Dans ce contexte, l'approche *langage* se concentre sur les "mots" de la carte que sont les symboles et leur sémiologie [Bertin 67 ; Eastman 87].

L'approche *communication* suppose l'existence d'un message prédéfini à transmettre et s'applique donc surtout à certains types de cartes comme les cartes présentant les résultats d'une analyse, la visualisation sur les navigateurs embarqués, ou encore les cartes soutenant un message de propagande délibérément trompeur [Monmonnier 91]. Dans ce travail nous nous plaçons au contraire dans le contexte des cartes comme représentations du monde géographique, et dont la fonction s'attache donc moins à véhiculer un message prédéfini qu'à répondre à un besoin d'information.

Information géographique numérique et cartographie

Parallèlement à son entrée dans la communauté scientifique, la cartographie, comme de nombreux domaines, a été bouleversée par l'avènement de l'informatique et de la gestion numérique de l'information.

Aujourd'hui l'information géographique est stockée sous la forme de bases de données géographiques. C'est à partir de ces bases de données que l'on réalise les cartes, elles-mêmes

[1] Technique d'exploitation de photographies aériennes stéréoscopiques, mise en œuvre vers 1930.

sous forme numérique avant d'être imprimées (cf. Figure 1, extrait de la BDTopo et d'une carte Top25 de l'IGN).

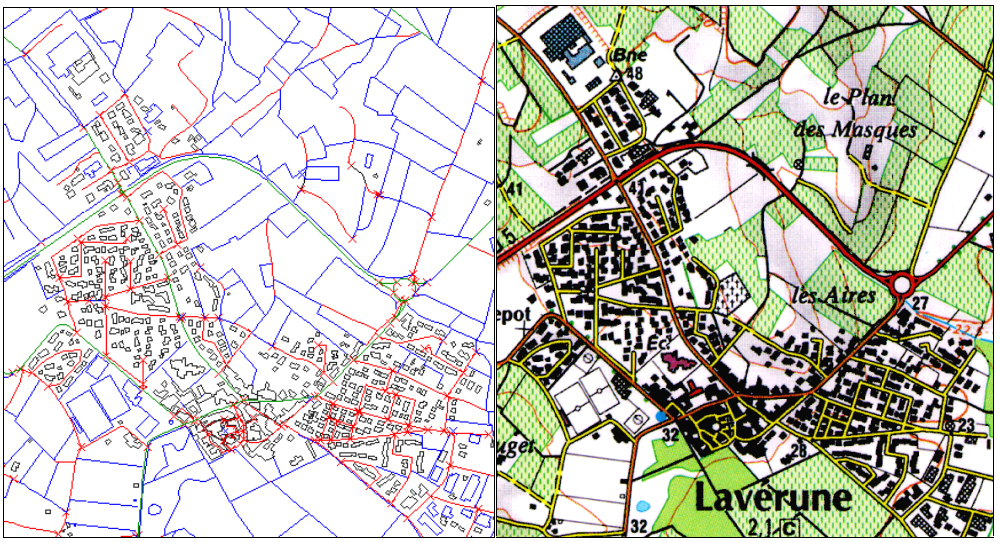


Figure 1. vue d'une BDG et carte dérivée de la BDG

Il est donc fondamental de différencier les bases de données géographiques (BDG) des bases de données cartographiques (BDC). Dans une base de données géographique on représente et localise dans l'espace des objets géographiques (maisons, rivières…), dans une base de données cartographique on représente et localise sur un plan les objets graphiques d'une carte (des points, des lignes, des surfaces, du texte) avec leur symbolisation (couleur, largeur…). Une base de données cartographique est une base prête à être affichée ou imprimée à une certaine échelle.

Si en théorie cette distinction est très nette, les bases de données géographiques actuelles sont en fait souvent plus cartographiques que géographiques car elles ont été créées en numérisant des cartes existantes [Brassel et Weibel 88]. On peut même ajouter que certaines bases de données géographiques constituées indépendamment de toute numérisation de carte ont souvent été conçues dans un esprit cartographique, soit par habitude des modèles cartographiques, soit dans le but de produire des cartes. Elles ont donc été pensées pour faciliter la future production d'une ou plusieurs cartes particulières, et non pour faciliter les analyses géographiques. Cependant, on devrait se diriger peu à peu vers une véritable dissociation des notions de BDG et de BDC.

L'existence des bases de données géographiques n'a pas seulement pour conséquence d'apporter aux données géographiques les avantages des données numériques (facilité de stockage, de duplication, de diffusion et de manipulation), elle modifie aussi le rôle même de la carte. Celle-ci ne joue plus le rôle d'archive, de recensement le plus exhaustif possible des phénomènes géographiques présents en un lieu donné, à une certaine échelle d'analyse. Les bases de données assurent dorénavant ce rôle.

Ceci signifie tout d'abord la fin du conflit entre la précision ou la résolution des données et leur visualisation [Rase 91 ; Morrison 94]. Auparavant, il était inutile de recueillir des données avec une précision ou une résolution qui ne pouvait être représentée et perçue sur la carte limitée par des contraintes graphiques. Par ailleurs, la carte devait être de la meilleure précision possible pour ne pas trop dégrader la précision des données recueillies qui sinon aurait été perdue à tout jamais. Dorénavant, peu importe de produire une carte avec une faible précision (si cela n'a pas d'influence sur l'utilisation qui en est faite) puisque la précision

originale des données est conservée dans les bases de données et est donc disponible au besoin.

Cela signifie aussi la fin du rôle central de la notion d'échelle. Celle-ci s'estompe au profit des notions, d'une part de précision et résolution (de localisation et de contenu) des bases de données, et d'autre part de but, contenu et lisibilité de la carte. "Le cartographe se concentre désormais plus sur le contenu de la carte que sur la région couverte" [Morrison 94].

Les bases de données géographiques, loin de contraindre le cartographe, devraient lui laisser une plus grande liberté, à la condition cependant de savoir réaliser des cartes à partir des bases de données géographiques. "Après la phase d'introduction de l'informatique dans le domaine de la cartographie, puis celle de l'adaptation de l'informatique à la cartographie on entre maintenant dans la phase d'adaptation de la cartographie à l'ordinateur" [Morrison 94].

Du papier à l'écran

Une autre conséquence de l'apparition des données géographiques numériques, et des capacités croissantes des ordinateurs en graphisme, est la possibilité d'afficher les cartes sur écran et de s'affranchir de la carte papier.

De nombreux travaux en visualisation s'intéressent à ce nouveau support [MacEachern et Taylor 94]. Par exemple, il est nécessaire de redéfinir pour l'écran les connaissances cartographiques concernant la perception des formes et des couleurs [Brown et Van Elzaker 93 ; Brewer 94].

Par ailleurs la visualisation de cartes sur écran ouvre de nouvelles possibilités d'expression, telles que les cartes dynamiques [Koussoulakou et Kraak 92 ; Kraak et MacEachren 94 ; Midtbø 2000], les cartes intégrant des liens hypermédia [Buttenfield, Weber et Jelinski 95], les cartes utilisant le son [Krygier 94], et les cartes mises à jour en temps réel [Clausen et Mark 91].

Toutefois, le papier reste pour l'instant de rigueur. Par exemple, si les cartes sur ordinateur peuvent être utiles pour préparer une randonnée, les cartes sur papier restent encore préférées pendant les randonnées [Wood et Goodwin 95]. Mais on peut penser que lorsqu'on pourra manipuler les cartes sur écran avec autant de facilité que les cartes papier (par exemple pouvoir facilement écrire dessus) les cartes sur écran seront davantage présentes, à l'instar de ce que seront les livres électroniques aux livres papiers.

Néanmoins, que le support soit le papier ou l'écran, la carte reste une représentation graphique du monde géographique et, si le champ des recherches s'élargit, les travaux sur la conception cartographique restent nécessaires, voire de plus en plus importants.

L'utilisateur cartographe

Puisque les données numériques peuvent être facilement diffusées, cela signifie que tout utilisateur de données géographiques peut posséder ces données[2] et donc fabriquer lui-même ses cartes. L'utilisateur devient donc cartographe. Si la difficulté et le coût de la production cartographique ont eu pour conséquence la centralisation des producteurs de cartes et la standardisation (une carte étant coûteuse elle doit être largement diffusée), l'informatique a changé cela : les cartes sont moins chères, plus personnalisées et leur production est décentralisée [Chrisman 91]. Plus précisément, les cartes sont toujours coûteuses à produire,

[2] Nous faisons abstraction des problèmes légaux et commerciaux liés à la diffusion des données

mais on peut dissocier, par l'intermédiaire des bases de données géographiques, la saisie des données et la création de la carte. Cette dernière phase peut donc être plus facilement personnalisée et décentralisée.

Au premier abord il paraît simple de créer une carte à partir d'une base de données géographique. Il semble suffire d'appliquer un symbole prédéfini à chaque objet de la base de données. Cependant, il est toujours nécessaire de modifier ces objets afin de les rendre compatibles avec les besoins de la carte : il est nécessaire de choisir les objets à représenter, de les simplifier, de placer les toponymes, etc., ce qui est traditionnellement le travail du dessinateur cartographe. L'utilisateur qui s'improvise cartographe n'a parfois pas les connaissances, les outils ou le temps pour créer des cartes efficaces. On voit ainsi apparaître (dans les journaux, les rapports d'étude, les dépliants touristiques, etc.) des cartes certes personnalisées, mais contenant également toutes les maladresses constatées au début des recherches en cartographie. Ces cartes sont donc parfois inefficaces voire involontairement trompeuses.

De nombreux cartographes identifient l'importance du besoin d'automatiser efficacement la fabrication de la carte, non seulement pour accélérer la production des cartes en série des instituts de cartographie, mais aussi pour permettre aux utilisateurs de créer eux-mêmes des cartes efficaces et adaptées à leurs besoins [Morrison 94 ; Taylor 94 ; Ruas et Lagrange 95 ; João 98 ; Forrest 99].

Ainsi, certains travaux entreprennent de créer des systèmes experts automatisant l'application de règles de cartographie guidant le choix de la symbolisation (telles que celles définies par Jacques Bertin en 1967) [Wang 92 ; Forrest 95 ; Andrienko et Andrienko 99]. D'autres travaux se concentrent sur le choix des thèmes à représenter [Martynenko et Leontiev 95 ; Ishizaki et Lokuge 95]. Enfin, une fois le choix des thèmes à représenter et de la symbolisation arrêté, il reste à réaliser la carte. Dans ce domaine, certains travaux se sont penchés sur le placement des écritures [Imhof 75 ; Ahn et Freeman 83 ; Jones 89 ; Barrault 98] et d'autres, plus nombreux, sur la généralisation cartographique [McMaster 83 ; Brassel et Weibel 88 ; Müller 91 ; Ruas 99 ; etc.]. C'est dans ce contexte de l'automatisation de la généralisation cartographique que se situent nos travaux.

Généralisation cartographique

Afin d'éviter toute confusion de langage, précisons avant tout que ***la généralisation cartographique n'est pas une généralisation au sens commun du terme, ni au sens utilisé en Intelligence Artificielle***.

Autant de travaux dans ce domaine, autant de définitions de la généralisation cartographique. L'ACI (Association Cartographique Internationale) la définissait en 1973 comme *"la sélection et la représentation simplifiée de détails en fonction de l'échelle et des objectifs de la carte"*.

Avant l'apparition des bases de données numériques, la généralisation cartographique désignait la création d'une carte à une échelle donnée à partir d'une autre carte à une échelle plus grande. Comme on le voit dans la Figure 2, cette généralisation cartographique est beaucoup plus qu'une simple réduction photographique de l'image de la carte.

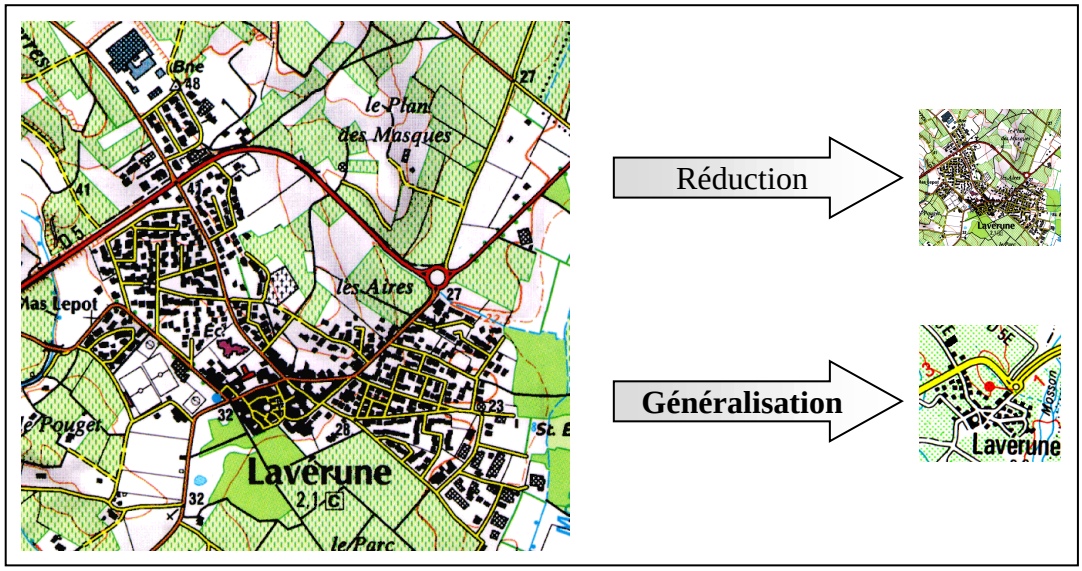


Figure 2. Réduction et Généralisation de cartes du 1:25.000 au 1:100:000

Désormais, la généralisation désigne aussi la création d'une carte à partir de données numériques contenant un surplus d'information par rapport aux besoins ou par rapport à la symbolisation choisie (cf. Figure 3). Autrement dit, la généralisation désigne le passage d'une base de données géographique à une base de données cartographique.

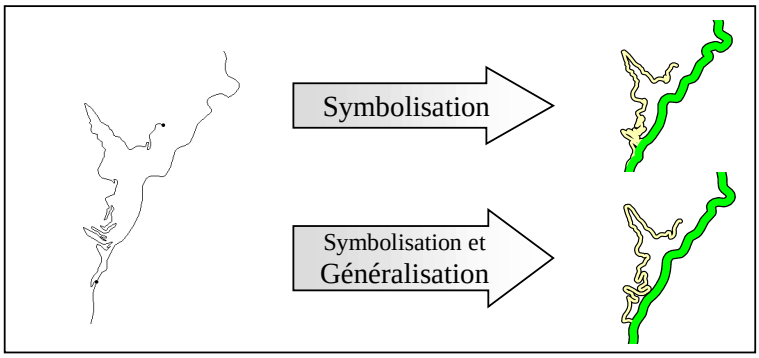


Figure 3. Passage d'une BDG trop détaillée à une BDC routière, avec et sans généralisation.

La généralisation cartographique est un processus rendu nécessaire par les limites de la perception humaine. Même si l'on disposait de techniques d'impression avec une résolution infiniment fine, l'œil ne pourrait percevoir des détails trop petits. Ainsi, lors de la mise à l'échelle des objets sur une carte, certains deviennent illisibles (cf. Figure 2, en haut à droite). Il est donc nécessaire de les agrandir si on veut les représenter. Mais alors les objets trop proches qui ont été agrandis se superposent, et il devient impératif de les simplifier, de les écarter, d'en éliminer certains, etc. C'est le rôle de la généralisation : simplifier et caricaturer l'espace afin d'en faire ressortir les caractères importants.

Mais cette limite de la perception n'est en fait qu'une cause secondaire de la généralisation. Les limites de la perception sont beaucoup moins importantes que les limites des capacités d'analyse. Pour qu'une carte soit efficace elle doit faire ressortir uniquement l'information nécessaire, sinon son analyse peut être soit considérablement ralentie soit impossible. Or, une carte est considérée complexe et difficile à analyser non seulement quand elle contient trop d'éléments, mais également quand elle n’a pas de structure globale claire (c'est-à-dire quand l'organisation générale de l'espace ne ressort pas clairement) [Eastman 85]. Pour que la généralisation soit efficace, elle ne doit donc pas seulement consister à diminuer le nombre

d'objets représentés, mais doit aussi analyser l'espace pour en faire ressortir les caractères importants.

La généralisation interactive, qu'elle soit réalisée manuellement avec des outils de dessin traditionnels ou par dessin assisté par ordinateur, est un processus long et coûteux pouvant exiger plusieurs mois de travail pour une seule carte. Par exemple, des estimations et des réalisations effectives de cartes IGN au 1:100.000 et 1:1.000.000 à partir de la BDCarto[3] ont montré que la généralisation peut nécessiter de six mois à deux années-homme de travail interactif par carte ! Pour s'en convaincre, il suffit de constater que les instituts de cartographie ont modifié au fil des ans les spécifications de leurs cartes pour réduire cette tâche (qui reste néanmoins longue), parfois au détriment de la lisibilité. Par exemple, la Figure 4 présente trois extraits de cartes d'une même région au 1: 25,000 dont les spécifications ont été définies respectivement en 1993, 1972 et 1922 (de gauche à droite sur la figure). On y voit que la largeur du symbole des routes de type "route de bonne viabilité 2 voies larges" est passé de 1,1 mm en 1922 à 0,7 mm en 1972 et 0,6 mm en 1993 (de même les chemins représentés par un tireté large de 0,5 mm en 1922 sont ensuite représentés par un trait noir continu de 0,2 mm). Cette diminution des largeurs de symbole permet de réduire les conflits de superposition entre les objets cartographiés, et donc de réduire le temps et le coût de la généralisation, mais elle rend plus difficile la lecture de la carte.

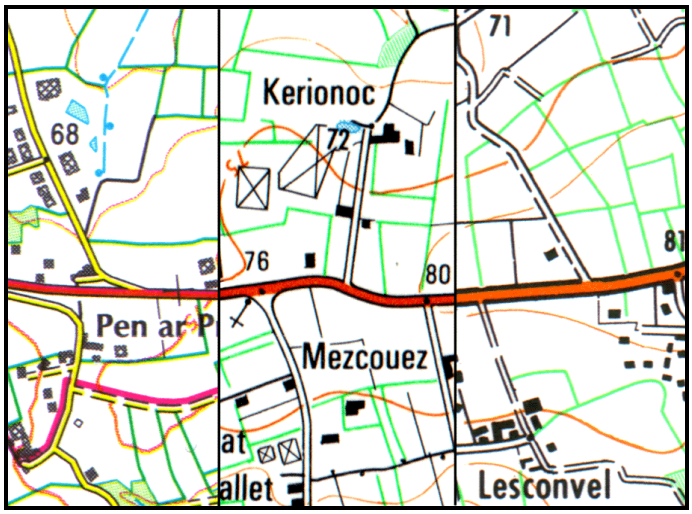


Figure 4. Trois factures de la carte au 1:25.000 de l'IGN: types 1993, 1972, 1922

Si les données numériques présentent d'indéniables facilités de manipulation par rapport aux cartes sur papier, la généralisation interactive sur écran à partir de bases de données reste très longue. Ceci est en particulier dû aux limites de résolution des écrans qui, contrairement au papier, ne permettent pas une vision simultanée de la carte à de nombreux niveaux d'analyse et obligent à énormément zoomer sur de petites zones lorsque l'on y travaille [Mustière et Lecordix 2000].

[3] Base de données de l'IGN de précision décamétrique, originellement fabriquée en numérisant les cartes topographiques au 1 : 50.000.

L'automatisation de la généralisation cartographique est donc nécessaire pour qu'une seule opération d'acquisition des données géographiques puisse permettre de fabriquer de nombreuses cartes (cf. Figure 5[4]). Plus précisément elle est requise pour aider :

- la fabrication et la mise à jour de cartes en série à différentes échelles à partir d'une même BDG. Ce contexte des cartes en série a été à l'origine des recherches en généralisation automatique, et reste toujours un problème non résolu.
- la fabrication de cartes personnalisées à la demande. Ceci est particulièrement important pour les études nécessitant plusieurs visions de l'espace à différents niveaux d'analyse, comme par exemple les études d'impact sur l'environnement [João 98].

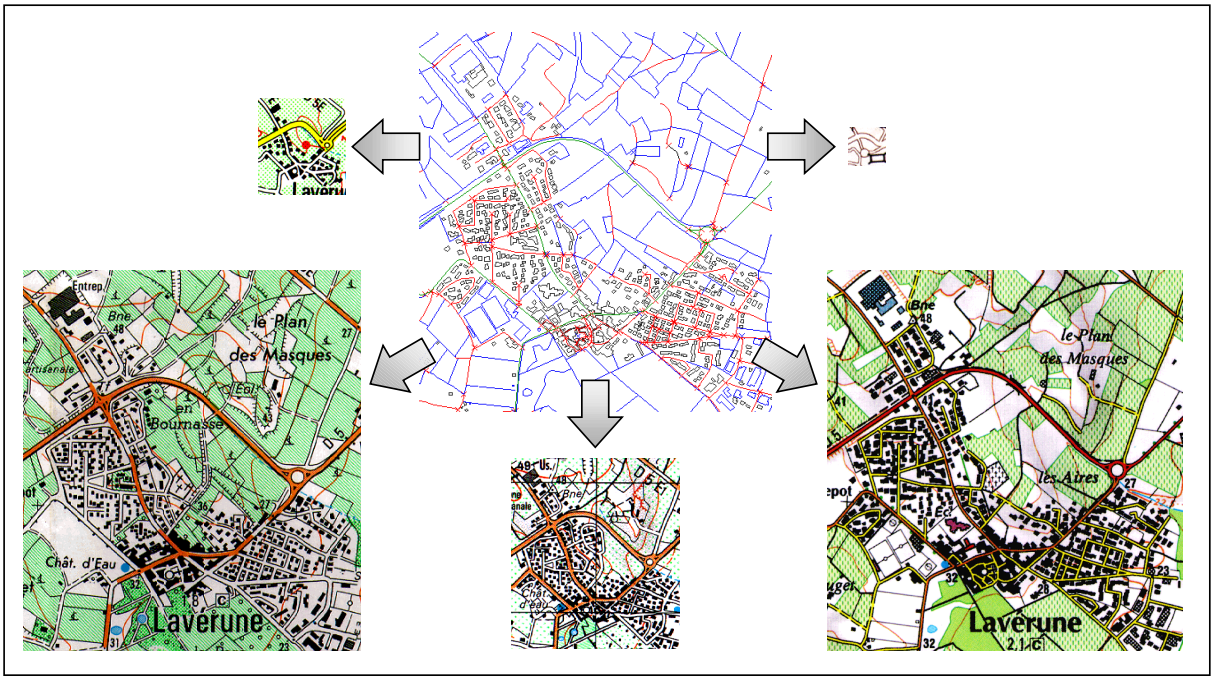


Figure 5. Une BDG et différentes cartes

Par ailleurs, cette automatisation a d'autres applications, comme par exemple:

- L'affichage de données géographiques transmises via Internet. Lorsque l'on reçoit une image par Internet, on la voit s'afficher grossièrement et s'affiner peu à peu. De même pour les données géographiques, on pourrait imaginer de transmettre dans un premier temps des données simplifiées (généralisées) puis des données de plus en plus détaillées [Buttenfield 99 ; Bertolotto et Egenhofer 99].
- L'aide à la visualisation de données sur écran. Le multi-fenêtrage (et en particulier la visualisation simultanée de données à plusieurs niveaux de détail [Stephan 95]) et les zooms efficaces [Spiess 95] sont des outils de première importance pour visualiser efficacement des données géographiques. Chacun de ces outils peut profiter de l'existence d'outils de généralisation automatique.
- L'analyse spatiale. La généralisation manipule la complexité du monde géographique représenté dans les bases de données et constitue donc un excellent terrain d'application des outils d'analyse spatiale. Les recherches en généralisation participent ainsi de manière plus générale aux recherches en analyse spatiale.

[4] Les différences entre ces cinq extraits de cartes de l'IGN ne sont pas seulement dues à des différences de généralisation mais aussi à des différences d'actualité des données.

Mais l'automatisation de la généralisation est un problème difficile :

> *"L'automatisation du processus de généralisation cartographique reste un des défis principaux en recherche sur les systèmes d'information géographique. […] Ce processus a montré toute la difficulté de son automatisation, parce qu'il est difficile d'imiter une démarche si subjective et intuitive. Ceci explique pourquoi, de nos jours, aucun logiciel SIG ne réussit à traiter efficacement le problème de la généralisation"* [João 98, p.5]

ii. Sujet

Problème cartographique

Nos travaux entrent dans le cadre des recherches en automatisation de la généralisation cartographique effectuées depuis une dizaine d'années au sein du laboratoire COGIT de l'Institut Géographique National (IGN). Comme mentionné précédemment, ces recherches ont pour but de faciliter à la fois la fabrication des cartes en série et la dérivation par les utilisateurs de données personnalisées à partir des bases de données géographiques.

Les recherches en automatisation de la généralisation cartographique ont abouti en particulier à l'élaboration d'un ensemble d'algorithmes géométriques de transformation des données, chacun étant conçu pour s'appliquer dans des conditions particulières et pour effectuer une opération particulière. La généralisation nécessite en effet de réaliser des actions différentes sur des données différentes et des configurations différentes. Si de nombreux travaux méritent encore d'être effectués sur la mise au point d'algorithmes, un problème crucial pour une véritable automatisation de la généralisation est de savoir enchaîner de manière efficace ces algorithmes.

Plus précisément, nous nous plaçons dans un modèle de généralisation où le traitement d'un objet nécessite de focaliser sur des espaces de travail pertinents et d'appliquer sur chaque espace un ou plusieurs algorithmes de transformation [Ruas et Plazanet 96 ; Ruas 99].

La généralisation cartographique doit rendre compte et donc tenir compte de la complexité du monde géographique, tout en intégrant un ensemble de contraintes graphiques. Ce processus est donc complexe et dirigé par un ensemble de connaissances variées. Les règles cartographiques qui le concernent sont nombreuses, concurrentes entre elles, et peu formalisées. Dans ces conditions, le recueil des connaissances[5] nécessaires à l'automatisation de ce processus est difficile. Par exemple, dans la règle "Il faut élargir suffisamment les virages non lisibles pour qu'ils soient lisibles, tout en conservant leur forme et leur position planimétrique autant que possible", comment formaliser les notions de "virage", "suffisamment", "non lisibles", "forme", "position", "autant que possible", et surtout comment formaliser le "tout en" qui régit la combinaison des différentes parties de cette règle ?

Dans ce travail nous étudions comment recueillir ces connaissances. Nous concentrons notre étude sur la question suivante : étant donné un objet géographique en cours de généralisation, comment acquérir les connaissances déterminant quel prochain algorithme lui appliquer ?

[5] Nous utilisons ici le terme général de "recueil de connaissances" plutôt que celui "d'acquisition de connaissances" car ce dernier terme, relativement général au début des années 80, s'est connoté au fil des ans pour s'orienter vers la désignation du problème de la modélisation des connaissances [Thomas 96]

L’apprentissage automatique, une solution ?

Une approche possible pour résoudre le problème de l'automatisation de la généralisation cartographique est de concevoir des systèmes experts. Ceux-ci ont en effet prouvé leur efficacité dans de nombreux domaines où des connaissances complexes doivent être introduites et manipulées (comme en médecine avec le système MYCIN [Shortliffe 1976]). Dans de nombreuses situations il est difficile de recueillir auprès d'experts les connaissances nécessaires au système. Ce problème est bien connu en Intelligence Artificielle sous le terme de "goulot d'étranglement de l'acquisition des connaissances" [Feigenbaum 81], et a été souligné dans le domaine de la généralisation cartographique [Weibel et al. 95].

L'apprentissage automatique supervisé est une des solutions développées en Intelligence Artificielle pour élargir le goulot d'étranglement de l'acquisition de connaissances. Son but est de construire automatiquement, à partir d'exemples fournis par un expert, des fonctions représentant une partie de la connaissance de l'expert. Ces fonctions sont appelées *hypothèses* en apprentissage automatique et peuvent être exprimées, entre autres, sous la forme de bases de règles ou d'arbres de décision.

L'expert fournit des exemples sous la forme, d'une part, d'une description d'un objet et, d'autre part, d'une classification de cet objet. Les techniques d'apprentissage automatique supervisé construisent alors automatiquement des hypothèses pour expliquer la classification en fonction de la description. Ces hypothèses peuvent alors servir à classer tout nouvel objet décrit dans le même langage de description.

Parce que la cartographie est guidée par des connaissances diverses et complexes, et parce qu'elle est souvent enseignée par des exemples, l'apprentissage automatique supervisé semble être une technique particulièrement bien adaptée à notre problème. Dans cette thèse, nous nous interrogeons sur l'opportunité de recueillir par apprentissage automatique supervisé les connaissances nécessaires au guidage des algorithmes de généralisation cartographique.

iii. Guide de lecture

Plan

Afin de situer et définir précisément le sujet de notre travail, nous effectuons un rapide état de l'art des travaux en automatisation de la généralisation cartographique dans le chapitre A. Nous nous situons dans une approche pas à pas (la généralisation étant vue comme un processus de changement d'état) où la recherche d'un espace de travail adapté à l'analyse et à l'application des algorithmes est primordiale [Ruas 99]. Nous cherchons à répondre à la question suivante: disposant d'une part de nombreux algorithmes de généralisation avec chacun son propre domaine d'application et d'autre part de nombreuses mesures d'analyse spatiale, comment relier ces algorithmes à ces mesures pour déterminer quel algorithme appliquer à un moment donné du processus ?

En s’appuyant sur l’exemple du processus GALBE de généralisation des routes que nous avons conçu, le chapitre B montre la faisabilité et l'efficacité de l'approche pas à pas, mais aussi les difficultés qui y sont associées. Le travail de mise au point du processus est long et, si les connaissances introduites ne sont pas assez nombreuses et pertinentes, les cas complexes sont mal gérés. Ces constatations motivent l'idée de déterminer les liens entre mesures et algorithmes par apprentissage automatique à partir d'exemples.

Le chapitre C présente les aspects de l'apprentissage supervisé automatique qui nous intéressent pour répondre à notre problème. Nous insistons notamment sur les points suivants : Comment poser un problème d'apprentissage ? Comment valider un résultat d'apprentissage ? Comment guider l'apprentissage automatique par les connaissances du domaine traité ? Les réponses à ces questions nous permettent de déterminer les grandes lignes du processus d'apprentissage que nous souhaitons mettre en place et qui possède les contraintes suivantes : apprendre des règles *compréhensibles* à partir de *peu* d'exemples.

Dans le chapitre D, nous définissons le processus d'apprentissage que nous mettons en œuvre. Partant des travaux intégrant l'acquisition des connaissances et l'apprentissage automatique [Ganascia, Thomas et Laublet 93 ; Thomas 96], nous décomposons notre problème grâce à une *méthode de résolution de problème* afin de guider l'apprentissage automatique. Ceci nous permet de distinguer les différents types de connaissances mis en jeu en généralisation cartographique : les connaissances d'*abstraction* pour raisonner sur le monde géographique et les connaissances de *représentation* pour en choisir une représentation cartographique. Ainsi nous proposons en particulier d'apprendre tour à tour à abstraire puis à représenter nos données géographiques.

Le chapitre E présente les expérimentations et résultats du processus défini au chapitre précédent. Nous décrivons comment ce processus est mis en place dans le cas de la généralisation cartographique des routes. Nous analysons en détail les résultats de l'apprentissage, à la fois d'un point de vue apprentissage et d'un point de vue cartographique. Nous montrons l'utilité de notre processus, dissociant les étapes d'abstraction et de représentation, pour obtenir des règles plus compréhensibles et plus efficaces que lors d'un apprentissage en une seule étape. Nous comparons également les résultats cartographiques issus de l'application de nos règles apprises avec ceux du processus GALBE présenté au chapitre B, et montrons ainsi que l'apprentissage nous permet d'améliorer la qualité cartographique de nos résultats.

Nous concluons en résumant la méthodologie de recueil de connaissances mise en place et les résultats obtenus lors de son application.

Le lecteur familier de l'automatisation de la généralisation cartographique pourra survoler le chapitre A sans préjudice pour la compréhension des autres parties, sauf la partie A.5 qui présente le sujet. Le lecteur familier de l'apprentissage automatique pourra pour sa part survoler le chapitre C, sauf la partie C.4 qui, expliquant les liens entre apprentissage et acquisition de connaissances, guide particulièrement les propositions du chapitre D. Le lecteur intéressé uniquement par les résultats en apprentissage de cette thèse pourra passer le chapitre B.

Note sur les cartes et figures de ce document

L'intégralité des extraits de cartes présentés dans ce mémoire sont issus des cartes de l'IGN. Celles-ci sont originellement en couleurs mais restent globalement lisibles en noir et blanc pour ce que nous voulons y montrer, même si certains éléments tels que les rivières ne sont plus mis en valeurs.

Nous n'avons pas reproduit ces cartes à leur échelle mais les avons agrandies, soit légèrement pour compenser l'absence de couleur et la résolution des imprimantes classiques plus faible que celle des impressions offset, soit exagérément pour mettre en valeur certains détails.

Pour les mêmes raisons, lorsque nous montrons des images d'objets extraits de bases de données[6] et symbolisés, ceux-ci sont agrandis par rapport à l'échelle pour laquelle la symbolisation est conçue. Quand nous indiquons que les objets cartographiques d'une figure sont symbolisés à telle échelle, il faut comprendre que la symbolisation de ces objets est celle qui pourrait être utilisée pour réaliser une carte à cette échelle, et non que notre figure est à cette échelle.

Ainsi, en cartographie nous manipulons en permanence trois unités de longueur : la longueur sur le terrain, la longueur sur la carte imprimée et la longueur à l'échelle d'affichage lorsque nous travaillons sur la fabrication de la carte. Afin de familiariser le lecteur avec ces notions, la Figure 6 suivante montre un ensemble de routes extrait de la BDCarto et :
a/ représenté en filaire (axe des routes sans symbolisation) et imprimé au 1:100.000,
b/ symbolisé et imprimé au 1:100.000,
c/ symbolisé au 1/250.000 et imprimé successivement au 1:100.000 et 1:250.000,
d/ symbolisé au 1/500.000 et imprimé successivement au 1:100.000 et 1:500.000.

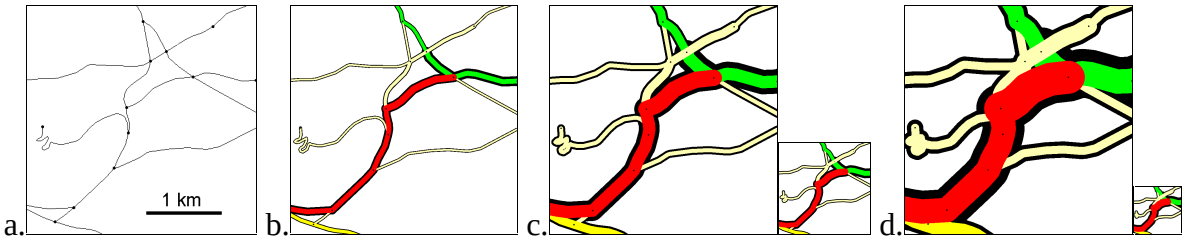


Figure 6. Extrait de BD à différentes échelles d'impression et de symbolisation

Enfin, pour montrer en détail les changements de géométrie créés par la généralisation, certaines figures contiennent en filaire les axes des objets non symbolisés, éventuellement superposés avec l'axe de l'objet original en pointillé (cf. Figure 7).

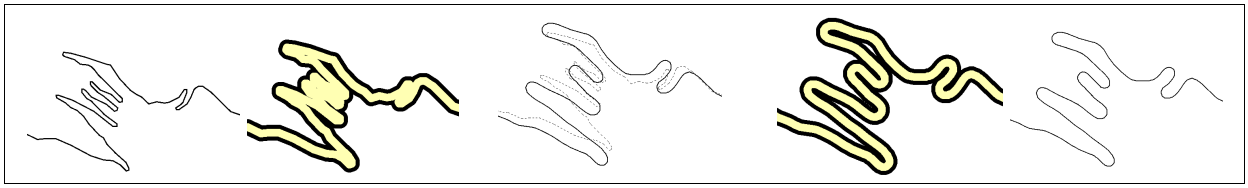


Figure 7. Comparaison d'une route symbolisée avant et après généralisation

Note sur le vocabulaire utilisé

La cartographie et l'intelligence artificielle utilisent des langages différents. Par exemple, les termes de *généralisation* et de *sémantique* ont un sens différent dans ces deux communautés. Nous avons essayé d'éviter toute confusion dans le texte tout en conservant les termes consacrés. Un glossaire à la fin de ce document permet néanmoins de lever les ambiguïtés de langage et de préciser le sens que nous donnons à certains termes.

■

[6] Nous utilisons essentiellement des extraits de la BDCarto de l'IGN.

A GENERALISATION CARTOGRAPHIQUE AUTOMATIQUE

◆ Après avoir décrit les opérations élémentaires utilisées en généralisation cartographique manuelle, nous exposons quelques grandes lignes des travaux en automatisation de la généralisation : comment sont conçus les algorithmes ? Comment les utiliser ?

◆ En nous plaçant dans une approche à base de connaissances, et en voyant la généralisation comme un processus de changement d'état, nous définissons plus précisément notre sujet : comment recueillir les connaissances nécessaires au guidage des algorithmes de généralisation cartographique ?

A.1 Représentation de l'Information Géographique Numérique

Avant de décrire plus en détail la généralisation cartographique et son automatisation, il nous faut décrire brièvement comment l'information géographique est modélisée dans les systèmes d'information géographique (SIG), et le vocabulaire employé.

La spécificité des bases de données géographiques (BDG) repose sur la représentation de la localisation et de la forme des objets géographiques (regroupées sous le terme de *géométrie*), et des relations *topologiques* entre ces objets (adjacence, inclusion). Les informations non relatives à la géométrie et à la topologie sont appelées les informations *sémantiques*, terme consacré en SIG mais qui peut faire penser à tort que la géométrie d'un objet ne contient pas de sens propre [Ruas 99, p.28].

Il existe deux grands modes de représentation de la géométrie dans les BDG : les représentations *vecteur* et *rasteur* (cf. Figure 8).

En mode rasteur l'espace est représenté par une partition complète de cellules, en général carrées et appelées alors *pixels*. Chaque pixel possède un ou plusieurs attributs qui décrivent son contenu (e.g. un morceau de route, son numéro, l'altitude, etc.). Ce mode de représentation est aussi celui des images, qui sont souvent les sources de saisie des BDG (photographies aériennes ou images satellites).

En représentation vecteur la géométrie de chaque objet géographique est décrite par une ou plusieurs primitives géométriques : des points, des lignes, des surfaces, et éventuellement des volumes. Une surface est délimitée par un ensemble de lignes, elles-mêmes décrites comme des ensembles de segments (ou plus rarement d'autres primitives telles que des cubiques), eux-mêmes représentés par deux points. La géométrie est donc décrite par un ensemble de couples (x,y) - ou triplets (x,y,z) - de coordonnées de ces points. Les relations topologiques entre les objets peuvent être explicitement représentées dans les BDG vecteur, par l'intermédiaire d'une structure de graphe.

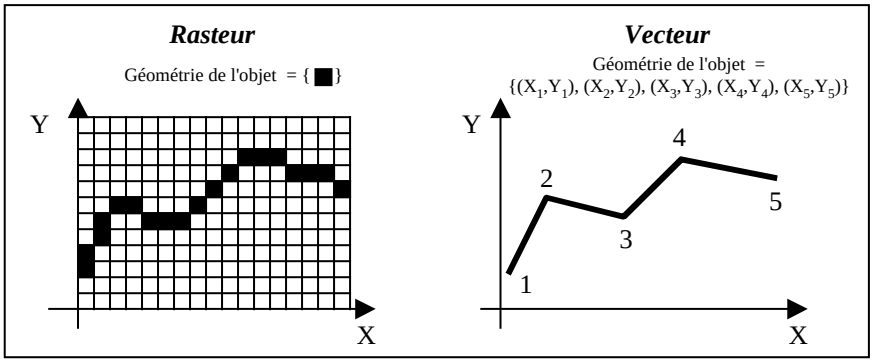


Figure 8. Représentation vecteur et rasteur de la géométrie

Même si les deux représentations sont possibles, la représentation rasteur est souvent utilisée pour modéliser des phénomènes géographiques couvrant entièrement tout ou partie de l'espace (e.g. le relief, l'occupation du sol), alors que la représentation vecteur est souvent utilisée pour représenter des phénomènes moins étendus (e.g. routes, bâtiments).

Notre travail concerne le traitement des BDG en représentation vecteur, à la fois parce que cela répond aux besoins en généralisation cartographique de l'IGN, et parce que cette

représentation est mieux adaptée à une modélisation objet des entités manipulées et de leurs relations.

Lorsque nous utilisons le terme d'objet géographique dans ce mémoire, nous ne faisons pas référence à la notion informatique d'objet mais à un sens beaucoup plus large de toute entité géographique considérée à un moment donné. Nous utilisons ce terme essentiellement pour différencier l'objet géographique de l'objet cartographique (i.e. un objet géographique représenté sur une carte).

Nous reprenons la terminologie de [Ruas 99] et appelons *caractère* un critère spatial ou sémantique qui qualifie un objet géographique ou cartographique (e.g. la taille, la position, la forme, la proximité, la densité, la nature). De même nous appelons *contrainte* cartographique la traduction des spécifications de généralisation sur les caractères des objets, par exemple "ne pas dégrader la précision planimétrique de plus de x mètres" est une contrainte sur le caractère position. Enfin nous appelons *conflit* le non respect d'une contrainte cartographique.

A.2 Opérations de généralisation cartographique

Nous avons brièvement présenté en introduction ce qu'est la généralisation cartographique. Nous la détaillons maintenant en présentant les opérations qui y sont réalisées.

De nombreuses classifications des opérations de base existent. Ces classifications peuvent être réalisées selon plusieurs axes : par exemple en fonction du but de ces opérations, en fonction des objets traités, ou en fonction du type de transformation géométrique réalisé. Par ailleurs, les cartographes travaillent dans des domaines divers, avec différents types de données, différentes échelles et différents buts. Une étude des termes utilisés dans la littérature montre que différentes définitions sont données pour un même terme, et que différents termes sont utilisés pour désigner le même opérateur [Rieger et Coulson 93]. En plus de ces problèmes terminologiques, les cartographes ne décomposent pas leurs actions en opérations élémentaires de la même manière. Il n'y a donc pas de consensus sur une terminologie ni sur une classification des opérations de généralisation.

Cette absence de consensus, que l'on retrouve même dans les communautés de cartographes aux préoccupations très proches, montre la difficulté qu'il y a à décomposer la généralisation en un ensemble de tâches élémentaires. Parmi les nombreuses classifications existantes on peut citer :

- D'un point de vue algorithmique, McMaster et Shea [1992] mettent l'accent sur les primitives géométriques manipulées (points, surfaces, lignes) : *classification* (sur les attributs sémantiques), *filtrage* (élimination de points), *lissage* (déplacement de points), *agrégation* (de points en une surface), *amalgamation* (de surfaces en une surface), *fusion* (de lignes en une surface), *squelettisation* (d'une surface en une ligne), *sélection* (parmi un ensemble d'objets), *exagération* (élargissement de parties d'un objet), *mise en valeur* (par changement de symbolisation) et *déplacement*.
- D'un point de vue gestion de bases de données, Peng et Tempfli [1996] mettent l'accent sur les éléments manipulés (classe, objet, attribut) : *sélection* (de classes, d'objets, ou de parties d'un objet), *universalisation* (généralisation d'attributs), *homogénéisation* (agrégation d'objets connexes de même classe), *simplification thématique* (élimination d'attributs), *reclassification* (redéfinition des classes), *combinaison* (agrégation d'objets connexes de classes différentes), *squelettisation* (changement de modélisation de la

géométrie), *simplification géométrique*, *agrégation* (d'objets non connexes) et *élimination* (d'objets sur des critères graphiques).

- D'un point de vue enseignement de la cartographie Weger [1994], met l'accent sur les résultats obtenus : *sélection des objets*, *schématisation des formes*, *harmonisation du contenu*, *décalages* et *substitution* (suggestion d'informations non explicitement décrites).

Classer en fonction du type d'objets en entrée et en sortie et/ou du type de modification effectuée crée des frontières relativement nettes entre les classes. Mais cela n'est néanmoins pas toujours le cas et ce type de classification, qui a principalement un intérêt algorithmique, est mal adapté à la description du but des opérations effectuées.

A l'inverse, classer en fonction du but crée des frontières très floues car beaucoup d'opérations réalisent plusieurs buts à la fois et le but principal est différent selon les points de vue. Néanmoins, afin de détailler la généralisation, nous proposons une classification très générale guidée par le but (résumée dans le Tableau 1) qui peut être affinée voire remise en question pour différents contextes.

Grandes familles d'opérations	Sous-familles d'opérations	Actions effectuées pour réaliser l'opération
Simplifier *éliminer de l'information* *(cf. A.2.1)*	Généralisation de modèle *rendre les objets similaires*	Reclassification thématique Fusion Squelettiser / Couvrir
	Sélection / Elimination *diminuer le nombre d'objets*	Suppression d'objets
	Lissage / Filtrage *atténuer ou éliminer les parties d'un objet*	Déformation d'objet Suppression de parties d'un objet
Caricaturer *exagérer des caractères* *(cf.* A.2.2*)*	Amplification *exagérer la forme et la taille*	Dilatation Déformation
	Structuration *exagérer la structure*	Sélection, fusion, translation Typification, schématisation
	Agrégation *exagérer la forme de l'ensemble*	Fusion Amalgamation
	Déplacement *exagérer les distances*	Translation / Décalage Déformation / Erosion
	Différenciation *exagérer les différences*	Dilatation Elimination
	Amélioration de la géométrie *exagérer la nature de l'objet*	Lissage Fractalisation Equarrissage
Harmoniser *gommer des différences* *(cf. A.2.3)*	Equilibrage *atténuer les différences*	Lissage, élimination, fusion, etc.
	Rééquilibrage *corriger les différences créées*	Lissage, élimination, fusion, etc.

Tableau 1. Récapitulatif des opérations de généralisation suivant notre classification

A.2.1 Simplifier

Tout ne peut pas être représenté sur une carte. Il faut donc réduire la densité d'information, simplifier l'espace en fonction du but de la carte. Pour cela il faut éliminer les informations "secondaires" et conserver les informations importantes. Cette notion d'importance d'une information est primordiale en cartographie. Elle n'est souvent pas exprimée explicitement dans les bases de données géographiques, parce qu'elle dépend du but de la carte, et parce que

"c'est notre regard sur les données géographiques représentées qui reconstruit ces informations" [Ruas 99, p.33]. L'importance peut dépendre à la fois de la taille des objets (une petite maison est moins importante qu'une grande), de leur forme (une maison rectangulaire est moins particulière et donc moins importante qu'une maison circulaire), de leur fonction (une gare a une fonction plus importante qu'un hangar) et surtout de leur contexte (une maison isolée est plus importante qu'une des maisons d'un lotissement). Les différentes façons de simplifier sont illustrées sur la Figure 9 suivante et décrites après.

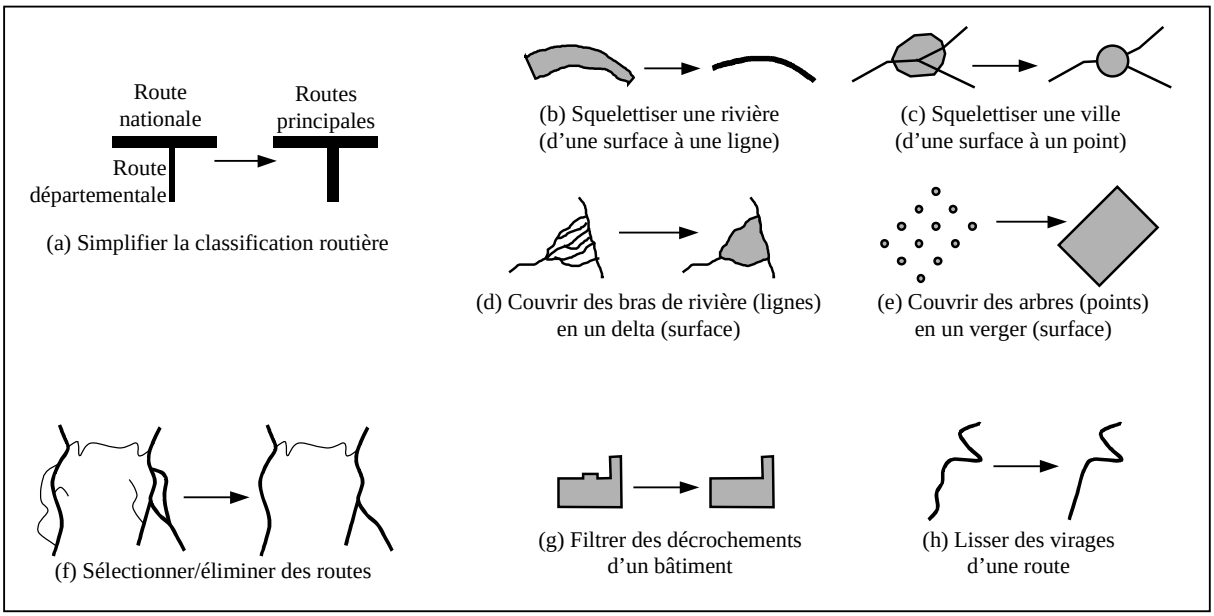


Figure 9. Simplifier

Généralisation de modèle

Une première simplification est réalisée lors du choix de la légende d'une carte qui détermine ce que l'on veut représenter et avec quelle symbolisation. La spécification de la légende néglige de fait un ensemble d'informations présentes dans la base de données géographique. Par exemple les routes nationales et départementales sont représentées par un même symbole de "routes principales", négligeant ainsi leur distinction. Cette simplification est souvent appelée *généralisation de modèle* (cf. Figure 9, a). Elle est réalisée avant tout autre traitement pour simplifier la base de données géographique à cartographier (c'est, en terminologie SIG, un changement de terrain nominal de la BDG).

La plupart des opérations de généralisation de modèle concernent les objets pris isolément et sont indépendantes de leur géométrie. Ce sont alors des opérations de *reclassification thématique*. Quelques opérations de reclassification peuvent cependant dépendre de l'importance des objets et donc éventuellement de leur géométrie et de leur contexte. Par exemple, les routes départementales peuvent n'être classées "routes principales" que si elles sont importantes, avec toute la difficulté qu'il y a à définir ce critère.

D'autres opérations de généralisation de modèle peuvent nécessiter l'analyse et la modification de plusieurs objets à la fois et de leur géométrie. Par exemple des opérations de *fusion* sont effectuées pour remplacer par un seul objet plusieurs objets connexes qui, après la reclassification thématique, n'ont plus lieu d'être distincts. Par exemple, deux parcelles connexes initialement classées "blé" et "maïs" sont reclassées "céréales" puis fusionnées.

Des opérations, dites de *changement de dimension*, changent aussi la modélisation de la géométrie des objets. Elles consistent par exemple à représenter une ou plusieurs surfaces par un point, plusieurs points par une ligne, etc. Ce changement de dimension est plus qu'un changement de modélisation dans la base de données cartographique. Par exemple pour le passage de surface à point, un symbole ponctuel est bien sûr une surface sur la carte (par exemple un disque), mais lors de la lecture de la carte on différencie clairement les symboles surfaciques (de forme quelconque) des symboles ponctuels (tous de taille restreinte et de même forme simple et symbolique). Parmi les opérations de changement de dimension, on peut distinguer les opérations de *squelettisation* (cf. Figure 9, b et c), qui consistent à diminuer la dimension géométrique des objets (une surface devient une ligne ou un point, et une ligne un point), des opérations de *couverture* (cf. Figure 9, d et e), qui augmentent la dimension (des points deviennent une ligne ou une surface, des lignes deviennent une surface).

Sélection/élimination et lissage/filtrage

Un autre type de simplification est la *sélection* ou l'*élimination*. Ce sont deux visions complémentaires d'une même opération. La sélection choisit les informations importantes à conserver alors que l'élimination choisit les informations secondaires à éliminer. Si nous distinguons les deux c'est que leurs approches sont différentes et que certaines règles de cartographie régissent la sélection et d'autres l'élimination. Par exemple on simplifiera le réseau routier de deux vallées proches en éliminant les routes de faible importance et les petits culs-de-sac, mais en sélectionnant les routes qui permettent de relier les deux vallées (cf. Figure 9, f).

La sélection ou l'élimination peuvent s'opérer sur un ensemble d'objets (e.g. sélectionner les routes importantes), ou sur les parties d'un objet (e.g. éliminer les petits décrochements d'un bâtiment). Lorsque l'on sélectionne/élimine les parties d'un objet, selon le point de vue on peut parler de *sélection/élimination* des parties ou de *lissage* ou *filtrage* de l'objet (cf. Figure 9, g et h). Le lissage désigne une atténuation des parties (typiquement le lissage d'une route pour atténuer les petits virages), et le filtrage une élimination de parties de l'objet.

Selon le type d'objet considéré, l'élimination peut s'effectuer soit en supprimant réellement les parties d'un objet, soit en lissant cet objet de manière à ce que ces parties ne soient plus distinctes du reste de l'objet. En effet éliminer un virage au milieu d'une ligne droite ne veut pas dire interrompre la route avant et après ce virage mais le lisser de manière à ce qu'il ne se distingue plus de la ligne droite qui le contient.

A.2.2 Caricaturer

Simplifier n'est pas suffisant pour généraliser, il faut aussi caricaturer, c'est à dire accentuer certains caractères des objets considérés au détriment d'autres caractères. Les différentes manières de caricaturer sont illustrées sur la Figure 10 suivante.

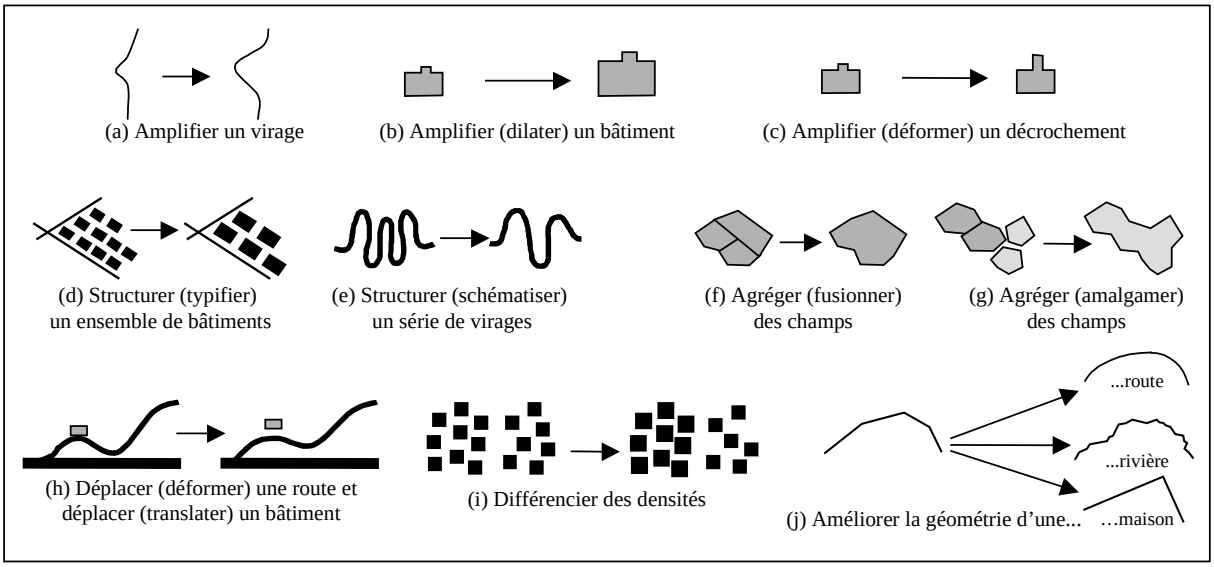


Figure 10. Caricaturer

Amplification

La caricature peut tout d'abord consister à exagérer certaines caractéristiques de forme ou de taille des objets, cette opération est alors l'*amplification* (Figure 10, a). Par exemple si une surface contient des décrochements considérés importants mais trop petits pour être lisibles, il faut les exagérer. Une exagération de la taille de l'objet peut se faire par une simple *dilatation* de celui-ci (Figure 10, b), une exagération de détails de forme d'un objet peut se faire par une *déformation* de celui-ci (Figure 10, c). Là encore, selon le point de vue on peut parler de dilater une partie d'un objet ou de déformer un objet pour en accentuer une partie.

Structuration

La caricature peut concerner l'organisation ou la structure d'un groupe d'objets, trop nombreux et/ou trop petits pour être tous représentés, cette opération est la *structuration* (aussi appelée *typification* ou *schématisation* selon les objets considérés, cf. Figure 10, d et e). On représente alors ce groupe par un ensemble plus restreint d'objets tout en conservant son organisation générale (on exagère la structure générale au détriment de l'information sur chaque objet). En cartographie, cet ensemble restreint est généralement un sous-ensemble de l'ensemble de départ, cela afin de conserver une partie de l'information sur les objets initiaux et en particulier les objets remarquables. Mais on ne peut néanmoins pas toujours savoir de quels objets initiaux dérive un nouvel objet.

Agrégation

La caricature peut aussi concerner la forme générale d'un groupe d'objets, cette opération est l'*agrégation*. Ce qui importe là ce n'est pas, comme pour la structuration, la répartition des objets au sein de l'ensemble, mais la forme générale de l'ensemble des objets. On peut différencier la *fusion* qui ne fait que négliger la frontière entre deux objets (Figure 10, f), de l'*amalgamation* qui recouvre un groupe d'objets par un autre objet (Figure 10, g). Cette opération peut faire penser à la couverture présentée dans les opérations de simplification, mais l'agrégation n'est pas dirigée par un changement de mode de représentation des objets (des surfaces sont agrégées pour former une autre surface). L'agrégation ne change pas non plus la nature de l'objet dans la légende. Par exemple, on ne spécifiera pas dans la légende que

le symbole surfacique "verger" peut en fait désigner sur le terrain soit un seul verger, soit une amalgamation de plusieurs vergers, car cela est considéré comme naturellement compris.

Déplacement

La caricature peut enfin concerner les distances entre plusieurs objets, cette opération est le *déplacement*. Si deux objets non connectés sont trop proches, la dissociation entre ces deux objets n'est pas perceptible. Il s'agit alors de l'exagérer (on exagère la dissociation des objets au détriment des positions absolues). Si un objet est uniformément déplacé, le déplacement est réalisé par une *translation*. Si seulement une partie d'un objet est déplacée, on réalise une *anamorphose* ou une *déformation* (Figure 10, h). Pour les objets surfaciques de grande taille, il s'agit de déformer la frontière des objets pour les rendre distincts, ce qui est alors vu comme une *érosion* des surfaces.

Le déplacement peut également servir à dissocier sur la carte deux objets superposés sur le terrain. C'est le cas par exemple des limites administratives qui sont souvent portées par d'autres objets comme des chemins, des routes ou des rivières. On effectue alors un *décalage* de la limite administrative pour que celle-ci soit lisible. Le lecteur interprétera naturellement que la limite est en fait superposée à l'objet (par un effet dit de substitution).

Différenciation

Il faut parfois caricaturer les différences entre les caractères de plusieurs objets ou plusieurs groupes d'objets. Ceci est réalisé en particulier pour mettre en valeur les différences de densité d'objets entre deux zones, par exemple pour faire ressortir le centre d'une ville par rapport à sa banlieue (Figure 10, i). Cela se réalise par exemple en éliminant des objets dans la zone la moins dense et en élargissant ceux de la zone la plus dense.

Amélioration de la géométrie

Un autre type de caricature est l'accentuation de la nature des objets. On exagère alors certains caractères de la forme d'un objet pour que sa nature soit plus facilement identifiée, on parle alors d'*amélioration de la géométrie* (Figure 10, j). Par exemple une construction artificielle linéaire comme une voie ferrée ou une route est en général lisse, on peut la lisser pour accentuer ce caractère ; un abri artificiel tel qu'une maison est en général à angles droits, on peut l'équarrir ; un objet naturel tel qu'une rivière est en général très sinueux, on peut le fractaliser, etc.

A.2.3 Harmoniser

L'harmonisation, c'est à dire l'élimination des différences entre plusieurs objets, peut tout d'abord être vue comme un objectif guidant les opérations de simplification ou de caricature. Par exemple, on peut lisser deux lignes avec des niveaux de sinuosité proches, de manière à ce que ceux-ci deviennent similaires ; dans ce cas on lissera donc un peu plus la ligne la plus sinueuse des deux.

Mais l'harmonisation est parfois une opération en soi et non un but secondaire. Deux raisons peuvent pousser à harmoniser : gommer les différences peu ou pas perceptibles entre plusieurs objets vis à vis de certains caractères (on peut alors parler d'*équilibrage*, Figure 11, a), ou corriger les écarts entre les caractères des objets qui auraient été involontairement accentués par les opérations de lissage ou de caricature (on peut parler alors de *rééquilibrage*, cf. Figure 11, b). L'harmonisation est parfois utilisée parallèlement à l'opération de caricature

des différences, car harmoniser les caractères de certains objets permet de faire ressortir les exceptions d'un groupe.

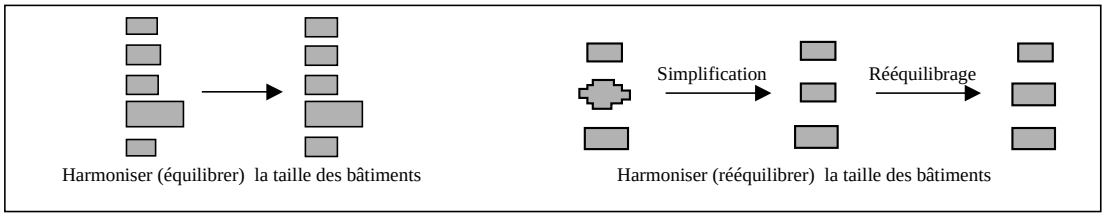


Figure 11. Harmoniser

A.3 Algorithmes de généralisation cartographique

A.3.1 De la compression aux premiers algorithmes de généralisation

Les travaux en automatisation de la généralisation cartographique ont débuté dans les années 1960, à une époque où la manipulation de fichiers volumineux était un problème important. Les problèmes liés au volume des données sont particulièrement présents dans les données géographiques. En effet, celles-ci contiennent beaucoup d'objets et la taille de chaque objet est grande, ceci en partie à cause du stockage de la géométrie et par conséquent de nombreuses coordonnées. De plus, selon la méthode de saisie employée, ces coordonnées sont parfois très redondantes. Ainsi, historiquement, les premiers algorithmes considérés comme réalisant de la généralisation automatique sont en fait des algorithmes de compression de données géométriques, c'est à dire de diminution du nombre de coordonnées nécessaires pour représenter la géométrie d'un objet sans perdre trop d'information sur la forme.

La compression n'est pas une opération de généralisation cartographique en soi. En généralisation cartographique on cherche à diminuer la quantité d'information perçue sur la carte et non la quantité d'information stockée dans la base de données. Cependant, même si ce n'est pas son but premier, la compression diminue *a posteriori* aussi la quantité d'information perçue. Ceci a contribué pendant de nombreuses années (et encore actuellement) au succès de ces algorithmes de compression en généralisation cartographique. Ce succès peut aussi être attribué, d'une part, au fait que ces algorithmes de compression sont simples à implémenter et, d'autre part, au succès obtenu par le paradigme de la théorie de l'information en cartographie. On les retrouve encore (et principalement ceux-là) implémentés dans les logiciels SIG commerciaux sous le nom d'algorithmes de généralisation. Le plus célèbre parmi les nombreux algorithmes de compression, l'algorithme dit de "Douglas et Peucker"[7] [Douglas et Peucker 73] (cf. Figure 12), reste une référence et fait encore l'objet d'une somme importante de travaux de variation ou d'adaptation [de Berg, van Kreveld et Schirra 95 ; Zhang et Tian 97 ; Saafeld 99]

[7] Cet algorithme est appelé "generalize" dans le SIG très répandu Arc/Info. Ceci montre bien la confusion, toujours présente, entre compression et généralisation cartographique des données.

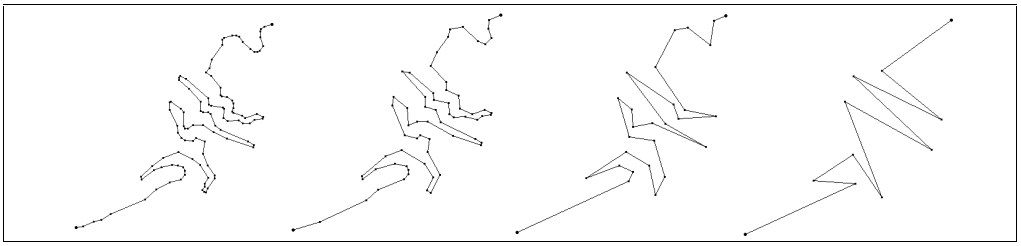


Figure 12. Algorithme de [Douglas et Peucker 73] appliqué avec différents paramètres

Dans le cadre de la généralisation cartographique, qui nous intéresse ici, les algorithmes de compression peuvent être utiles en tant qu'outils d'analyse spatiale (l'algorithme de Douglas et Peucker met bien en évidence les sommets de virages par exemple), ou en tant que post-traitement si "à quantité d'information perçue égale, ils permettent de réduire la taille de stockage". Mais ils ne peuvent être considérés comme des algorithmes à but de transformation graphique. Lorsqu'ils sont utilisés dans le but de réduire l'information perçue graphiquement ils ne sont en général pas efficaces. Dans un test de lecture de carte, Wood et Clifford [1995] présentent à différents lecteurs plusieurs cartes traitées par l'algorithme de Douglas et Peucker avec différents paramètres. Les principaux résultats de ce test sont que la rapidité et la précision de lecture de la carte ne sont pas ou peu améliorées sur les cartes simplifiées, ce qui montre l'inefficacité de cet algorithme dans le but de simplifier la lecture de la carte[8]. En conséquence, nous ne considérerons pas ces algorithmes comme des algorithmes réalisant des opérations de généralisation cartographique dans la suite de notre étude.

Le deuxième type d'algorithme développé concerne le lissage de lignes. Beaucoup d'entre eux fonctionnent par moyenne ou convolution sur une fenêtre glissante inspirée du traitement d'image [Brophy 73 ; Lowe 88 ; McMaster 89]. Chaque point de la ligne est déplacé vers une position fonction des points qui l'entourent. Un algorithme donnant des résultats visuellement satisfaisants est le lissage dit gaussien, étudié notamment dans le domaine du traitement d'image [Badaud et al. 86]. Ce type d'algorithme de lissage est utile lorsque l'on désire conserver la forme générale d'un objet en négligeant ses détails. Cependant si ces algorithmes conservent bien la forme générale d'un objet, ils ont le défaut d'atténuer autant les petits détails de l'objet que les grands, ce qui est parfois gênant en cartographie où l'on cherche à mettre en valeur les détails significatifs.

Les algorithmes de caricature, opération la plus importante en généralisation, ont été les derniers à être étudiés. Les premiers algorithmes de caricature se sont inspirés des algorithmes de simplification. Par exemple [Dutton 81] a adapté l'algorithme de lissage de [Brophy 73] pour amplifier plutôt que diminuer les virages d'une ligne. Alors que l'algorithme de Brophy déplace chaque point vers le centre du cercle inscrit dans le triangle qu'il forme avec ses deux voisins, l'algorithme de [Dutton 81] éloigne chaque point de ce centre. Depuis, de nombreux algorithmes de caricature ont été conçus.

[8] Si nous utilisons cette étude pour illustrer l'inadéquation des algorithmes de compression dans un but graphique, il nous faut préciser que Wood et Clifford ne l'analysent pas ainsi. Sans se prononcer clairement, ils sous-entendent dès le titre de leur article que la généralisation (qu'ils réduisent à la compression) n'est peut-être pas utile : "Do map readers really notice and use generalization? : The perceptual consequences of line simplification in a task-oriented thematic map analysis experiment".

A.3.2 Propriétés des algorithmes de généralisation

Notre but n'est pas de faire un état de l'art complet des algorithmes de généralisation cartographique mais de montrer les différentes approches utilisées et comment ces algorithmes intègrent les différentes contraintes cartographiques.

On pourra trouver des états de l'art détaillés des algorithmes de généralisation, en particulier dans diverses thèses réalisées au laboratoire COGIT : [Ruas 99] pour un état de l'art général, [Plazanet 96 ; Fritsch 97] pour le traitement des objets linéaires routiers, [Monier 97] pour le traitement du relief, [Regnauld 98] pour le traitement des bâtiments et groupes de bâtiments. D'autres présentations d'algorithmes de généralisation peuvent être trouvés dans [McMaster et Shea 92 ; AGENT 99].

Nous avons présenté la généralisation comme une combinaison d'opérations élémentaires. Les algorithmes développés pour automatiser la généralisation sont plus ou moins ciblés, au niveau de leur champ d'application, des opérations qu'ils réalisent, et des contraintes qu'ils intègrent.

Les algorithmes les plus atomiques sont conçus pour ne traiter qu'un type très particulier d'objet, pour ne réaliser qu'une opération, et se concentrent sur la résolution d'une contrainte principale même s'ils intègrent de manière plus ou moins explicite d'autres contraintes à un niveau secondaire. A l'opposé, les algorithmes les plus généraux s'appliquent à un champ plus large d'objets, réalisent plusieurs opérations à la fois, et intègrent au premier plan un ensemble de contraintes variées. Avant de détailler cette analyse, nous présentons brièvement trois algorithmes de traitement des routes développés au laboratoire COGIT pour illustrer ces différentes approches.

A.3.2.1 Trois algorithmes représentatifs de différentes approches

Accordéon [Plazanet 96, p.166]

Accordéon est conçu pour s'appliquer principalement à une série de virages en épingles à cheveux d'une route. Il ne réalise qu'une seule opération : amplifier cette série de virages en l'étirant. Il se concentre sur la résolution de la contrainte du respect des critères de lisibilité graphique, plus précisément il vise à dissocier graphiquement chaque virage des autres. Il intègre de manière explicite les contraintes secondaires d'une part de position de l'ensemble des virages et d'autre part de respect de l'orientation générale et de la forme de chaque virage. Il intègre de manière implicite le respect de l'orientation entre les virages et de l'aspect général de la série de virages. Il n'intègre pas de contrainte de précision planimétrique.

> Fonctionnement simplifié : on délimite d'abord les virages principaux de la série de virages. On élargit ensuite chaque virage perpendiculairement à sa direction principale, de manière à ce que la base de ce virage soit assez large pour que, une fois symbolisé, l'intérieur du virage soit lisible. On recale l'ensemble des virages de manière à ce que le point d'inflexion central de la série soit localisé à sa position d'origine.

Plâtre [Fritsch 97]

Plâtre s'applique en théorie à toute route. Il réalise à la fois des opérations de simplification (atténuation des petits virages) et d'amplification (élargissement des virages serrés). Il intègre explicitement tour à tour plusieurs contraintes : une contrainte de lisibilité (représentée sous la

forme d'une contrainte de courbure maximale de la route), une contrainte de diminution de la granularité (taille des plus petits détails), et une contrainte sur la position des virages principaux.

Fonctionnement simplifié : On transforme la représentation initiale d'une ligne vecteur (suite de coordonnées) sous la forme d'une fonction "courbure en fonction de l'abscisse curviligne". Cette fonction est lissée de manière à diminuer la courbure des points. Une transformation inverse est réalisée pour revenir à une représentation vecteur. Pour corriger les erreurs de positionnement créées, chaque virage de forte courbure est recalé sur la position de son homologue de la ligne initiale.

Approche Mécanique [Fritsch 97 ; Kyndt 97]

L'*Approche Mécanique* s'applique en théorie à toute route. Elle réalise à la fois des opérations de simplification (élimination de petits virages), d'amplification (élargissement de certains virages) et de déplacement (écartement de parties de lignes trop proches). Elle intègre simultanément et explicitement des contraintes de lisibilité (courbure et proximité), de respect de forme locale et de précision planimétrique. Elle intègre plus implicitement des contraintes de respect de forme globale.

Fonctionnement simplifié : On considère une ligne comme une suite de poutres extensibles (avec une certaine rigidité) articulées par des rotules (des ressorts avec une certaine rigidité). Les contraintes cartographiques sont représentées sous la forme d'un ensemble de forces s'exerçant sur les poutres et les rotules : des forces de répulsion incitent les rotules à s'écarter les unes des autres, des forces de rappel incitent les rotules à ne pas s'éloigner de leur position originale, des moments incitent les rotules à s'ouvrir, la rigidité des poutres les incite à ne pas trop se déformer, etc. La ligne est transformée peu à peu sous l'influence de ces forces mécaniques.

La comparaison de ces trois algorithmes nous permet d'illustrer ci-dessous les différences essentielles entre les algorithmes, non pas d'un point de vue de leur résultat, mais d'un point de vue des opérations effectuées, des contraintes cartographiques prises en compte, et de leur champ d'application.

A.3.2.2 Contraintes, opérations, et champ d'application des algorithmes

Contraintes gérées

Certains algorithmes considèrent sur le même plan un ensemble de contraintes, alors que d'autres sont conçus pour traiter en priorité une contrainte même si d'autres contraintes sont aussi utilisées pour guider le traitement.

Pour considérer sur le même plan plusieurs contraintes, celles-ci doivent être représentées dans un formalisme commun. Par exemple pour l'*Approche Mécanique* décrite ci-dessus, toutes les contraintes sont exprimées sous la forme de forces mécaniques. Une autre approche consiste à représenter ces contraintes comme des équations sur les coordonnées (éventuellement pondérées selon l'importance des contraintes), et à résoudre cet ensemble d'équations par exemple par la méthode des moindres carrés [Sester 2000 ; Harrie et Sarjakoski 2000]. En pratique ces approches ne traitent qu'un ensemble restreint de contraintes car l'hétérogénéité des contraintes cartographiques rend souvent impossible leur expression dans un formalisme commun.

Dans l'algorithme *Accordéon*, le but premier est d'écarter la base des virages, c'est donc cette contrainte de lisibilité des virages qui est traitée avant tout. Cet écartement est géré de manière à conserver explicitement l'orientation de chaque virage, c'est une contrainte secondaire. Implicitement, l'orientation relative entre les virages est relativement maintenue, c'est une contrainte secondaire implicite. Par contre la taille de la série de virages n'entre pas en jeu dans le traitement, c'est une contrainte qui n'est pas prise en compte par l'algorithme.

Le changement de représentation est une façon de mettre en avant un caractère de l'objet considéré et donc certaines contraintes par rapport aux autres. Par exemple *Plâtre* représente une ligne comme sa courbure fonction de l'abscisse curviligne. Ce faisant il met en avant la notion de courbure et la contrainte sur le respect d'une courbure minimale. Le même auteur a aussi essayé de représenter la ligne par analyse de Fourrier ou par ondelettes ; là ce sont les formes récurrentes qui sont mises en avant et la notion de localisation est alors mise en arrière plan [Fritsch 97, p.C30]. La deuxième étape de *Plâtre* (le recalage des virages principaux) travaille en représentation classique de la ligne comme une suite de vecteurs. Ce faisant la contrainte de position est mise en avant, et la contrainte de courbure est alors reléguée en arrière plan.

On peut distinguer pour chaque algorithme les contraintes de premier plan (que l'on doit absolument résoudre), les contraintes de second plan explicites ou implicites (que l'on doit respecter au mieux), et les contraintes non prises en compte. Considérer simultanément plusieurs contraintes de premier plan présente des avantages et des inconvénients par rapport au fait de ne considérer qu'une seule contrainte de premier plan.

Considérer plusieurs contraintes présente l'avantage de se placer dans une approche holiste plus proche de l'approche manuelle. On peut donc espérer au premier abord obtenir des résultats plus proches de ceux d'une généralisation manuelle. Par ailleurs, si un seul algorithme permet de respecter plusieurs contraintes, le problème de l'enchaînement de différents algorithmes pour résoudre plusieurs contraintes est moindre.

Mais d'un autre côté, pour gérer plusieurs contraintes il est souvent nécessaire de représenter toutes les contraintes dans un formalisme commun. Ce formalisme est parfois inadapté à certaines contraintes qui sont alors mal représentées. Ensuite le paramétrage des algorithmes devient difficile car chaque contrainte peut être paramétrée, et les effets de ces nombreux paramètres sont rarement indépendants. Plus généralement le contrôle de ce type d'algorithme est souvent difficile. De plus, quand un algorithme réalise un compromis entre différentes contraintes, on oublie que souvent en cartographie "prendre la moyenne est la mauvaise solution, la plus fausse et la moins expressive" [Weger 94]. Enfin, ces approches ne permettent en général pas d'effectuer des transformations géométriques avec des sauts discontinus (par exemple représenter un groupe de quatre bâtiments par un groupe de trois bâtiments), ils sont donc mal adaptés aux opérations de structuration.

Les algorithmes qui considèrent simultanément plusieurs contraintes de premier plan se révèlent efficaces surtout lorsque la généralisation à effectuer n'est pas trop importante. Cela se rencontre soit dans le cas où la base de données géographique à traiter possède un niveau de détail proche de la base de données cartographique à réaliser, soit en tant que post-traitement final de la généralisation après que l'espace a été suffisamment simplifié et caricaturé. Par exemple ce type d'algorithme appliqué au déplacement de bâtiments [Højholt 98 ; Harrie 98] est essentiellement utilisé pour créer des cartes dont l'échelle est de l'ordre du 1:10.000, où les déplacements nécessaires sont faibles [Ruas 99, p.179].

Notons que les contraintes, reliées aux spécifications de la carte, peuvent être contrôlées par des paramètres, qui peuvent être de différentes natures. Certains paramètres expriment la quantité de déformation à effectuer ("de combien ?"), et d'autres expriment des conditions sur le résultat à obtenir ("jusqu'où ?"). Par exemple la largeur minimale requise pour chaque virage dans l'algorithme *Accordéon* est un paramètre de type "jusqu'où?". Par contre la force du lissage de l'algorithme *Plâtre* est un paramètre de type "de combien?". De même, l'algorithme *Approche Mécanique* effectue à chaque étape une micro-déformation de la ligne sans aboutir nécessairement à un état d'équilibre, il possède donc un paramètre "de combien" qui correspond au nombre d'étapes à effectuer.

Du point de vue de l'utilisation des algorithmes, il est en général plus facile d'utiliser des paramètres de type "jusqu'où ?" que des paramètres de type "de combien ?", plus difficiles à relier aux spécifications de la carte à obtenir. Cependant, du point de vue de la réalisation des algorithmes, ce premier type de paramètre suppose de savoir évaluer la propriété à atteindre (le "ou`?"), ce qui est parfois difficile.

Opérations effectuées

Certains algorithmes réalisent plusieurs opérations à la fois et d'autres n'en réalisent qu'une seule. Par exemple l'algorithme *Accordéon* ne réalise qu'une opération d'amplification. A l'inverse l'algorithme *Plâtre* réalise à la fois une amplification des virages serrés et un lissage des virages peu importants, et l'*Approche Mécanique* réalise à la fois des opérations de lissage, d'amplification et de déplacement.

Les avantages et inconvénients des algorithmes réalisant plusieurs opérations sont du même ordre que ceux intégrant plusieurs contraintes de premier ordre. Si plusieurs opérations sont effectuées par un algorithme il est nécessaire de gérer moins d'algorithmes pour effectuer une généralisation complète, mais le contrôle de chaque algorithme peut être difficile. En particulier le compromis entre les différentes opérations réalisées est contrôlé à l'intérieur de l'algorithme et donc pas nécessairement adapté aux besoins de la carte à réaliser.

Champ d'application

Le champ d'application des algorithmes peut être plus ou moins large. Par exemple l'algorithme *Accordéon* n'est pertinent que sur une série de virages serrés, alors que les algorithmes *Plâtre* et *Approche Mécanique* peuvent s'appliquer sur tout type de ligne.

Là encore, utiliser des algorithmes possédant un champ d'application restreint possède des avantages et des inconvénients.

Les algorithmes ciblés sont dédiés à un type d'objet bien défini et sont donc en général très efficaces et plus simples à concevoir. Ces algorithmes sont aussi en général plus simples à contrôler car les objets sur lesquels on les applique sont relativement semblables.

Mais pour utiliser des algorithmes ciblés il faut savoir délimiter automatiquement leur espace de travail. De plus il devient alors nécessaire de concevoir de nombreux algorithmes, un pour chaque type d'objet que l'on peut rencontrer. Ensuite, il faut contrôler les divers effets de bords de ces algorithmes, et les éventuels problèmes d'harmonisation dus à l'application de différents algorithmes. Enfin, l'application par erreur d'un algorithme non adapté à l'objet traité peut créer des problèmes importants.

A.3.3 Enchaînement des algorithmes

Utiliser plusieurs algorithmes

Les algorithmes de transformation géométrique sont des briques élémentaires nécessaires à l'automatisation de la généralisation cartographique. Des tests d'utilisation des algorithmes existants montrent la nécessité d'enchaîner de nombreux algorithmes pour réaliser une généralisation satisfaisante [Ruas 98b ; Regnauld, Edwardes et Barrault 99], et ce même si on se restreint à des sous-problèmes tels que le traitement d'un thème particulier indépendamment du reste de la carte. Ces tests montrent également que l'enchaînement idéal des algorithmes n'est pas le même pour différents objets. Par exemple, le traitement des petits bâtiments globalement rectangulaires et celui des bâtiments de taille moyenne en forme de L nécessitent l'application de séquences d'algorithmes différentes. Cette vision de la généralisation comme l'enchaînement de nombreux algorithmes est partagée dans les principaux modèles de généralisation [Brassel et Weibel 88 ; McMaster et Shea 88 et 92 ; Beard 91 ; Ruas 99].

La détermination de cet enchaînement est cruciale pour automatiser la généralisation cartographique. Pour cela, nous nous plaçons dans une approche similaire à celle des Systèmes Experts, développés dans le domaine de l'Intelligence Artificielle. Ces systèmes ont prouvé leur efficacité à résoudre des problèmes où des connaissances complexes doivent être introduites. Dans ces systèmes, les connaissances sont introduites sous la forme de règles de production qui sont granulaires : chaque règle contient relativement peu de connaissances et est considérée comme une partie indépendante des connaissances utilisées. De même nous pensons qu'il est nécessaire pour automatiser la généralisation cartographique d'isoler au mieux chaque connaissance impliquée. Pour cela, il faut utiliser des algorithmes relativement atomiques et guider leur enchaînement par une base de connaissances, qui peut contenir un ensemble de règles spécifiant quand appliquer quel algorithme. Si les algorithmes combinent indistinctement trop de connaissances, il devient difficile de les contrôler et de les manipuler. Il vaut donc mieux situer le maximum de connaissances dans les moteurs d'utilisation des algorithmes.

Il est néanmoins difficile de concevoir de telles bases de connaissances pour guider la généralisation. Tout d'abord les règles cartographiques sont nombreuses, contradictoires et peu formalisées, il est donc difficile de les intégrer dans un système automatisé. De plus ces règles sont générales et ont été posées pour guider la généralisation manuelle. Elles ne sont donc pas nécessairement adaptées au guidage des algorithmes.

Ensuite ces règles doivent s'appuyer sur des mesures de description des objets géographiques et plus généralement de l'espace. En effet, "les divers algorithmes de généralisation fonctionnent correctement si, et seulement si, ils sont utilisés dans un contexte approprié" [Brassel et Weibel 88]. Il faut donc savoir décrire le contexte d'utilisation d'un algorithme, c'est-à-dire les types d'objets et les configurations sur lesquels il peut s'appliquer. Cependant "un des principaux problèmes en généralisation automatique est notre incapacité à savoir mesurer" [João 91]. Un problème tout aussi important réside dans la difficulté à utiliser efficacement les mesures conçues pour la généralisation. Ces problèmes (que nous évoquerons un plus en détail au chapitre B pour ce qui concerne les routes) sont encore source de nombreuses recherches pour mieux mesurer les objets [Buttenfield 91 ; Plazanet 96] et mieux utiliser ces mesures en les reliant aux contraintes cartographiques [Ruas et Lagrange 95 ; Weibel 96 ; Ruas 99].

Un autre problème réside dans la validation des résultats. Si on dispose de méthodes fiables de validation des résultats, et si on fait abstraction des problèmes de temps de calcul, on peut essayer de nombreuses combinaisons d'algorithmes jusqu'à retenir la plus efficace [Regnauld 2001]. Cependant les méthodes de validation ne sont pour l'instant pas assez robustes pour cela. Il est faut donc trouver des heuristiques d'utilisation des algorithmes assez fiables pour éviter que les algorithmes effectuent des transformations aberrantes que nous ne saurions détecter.

Déterminer le bon espace de travail

D'une part, les règles cartographiques s'appliquent généralement à une partie bien précise de l'espace (une ville, un quartier, des bâtiments alignés, un bâtiment, une aile de bâtiment…) et, d'autre part, les algorithmes peuvent avoir un champ d'application tout aussi ciblé. Il est donc nécessaire de déterminer l'espace de travail adapté à chaque opération.

> *"Le découpage des entités géographiques en objets dans une base de données est dû à des changements sémantiques et à des intersections avec d'autres entités. Ces découpages ne sont pas satisfaisants pour la généralisation. En effet pour appliquer des opérations il faut parfois travailler sur un groupe d'objets et d'autres fois sur une partie d'objet."* [Ruas 99, p.51]

Cet espace de travail n'est pas explicitement défini dans les bases de données géographiques. Pour pouvoir appliquer automatiquement et efficacement des algorithmes, il faut donc savoir le déterminer automatiquement. D'un point de vue algorithmique, ceci est relativement immédiat si cet espace est nettement défini à partir des données présentes dans les BDG, comme par exemple un îlot urbain défini par un cycle minimal du réseau de rues [Ruas 99, p.112]. Par contre ceci est beaucoup plus difficile à réaliser quand cet espace est défini par des critères d'homogénéité ou des critères perceptuels, comme les quartiers [Boffet 2000], les groupes linéaires de bâtiments [Regnauld 98, p.86], les groupes de bâtiments desservis par une route [Hangouët 98, p.197], les limites d'une ville [Edwardes et Regnauld 2000 ; Boffet 2000], les parties de sinuosité homogène d'une route [Plazanet 96, pp.118-119], les virages à différents niveaux de détail [Ai, Guo et Liu 2000], les zones de relief homogène [Weibel 89 ; Monier 97], etc.

Même si cela n'est pas immédiat, la nécessité d'enrichir les données de bases pour détecter ces espaces de travail à différents niveaux d'analyse est mise en avant dans les principaux modèles de généralisation. Brassel et Weibel [1988] introduisent une phase de "reconnaissance des structures" dans leur modèle. Ruas et Plazanet [1996] introduisent la notion de plan global et de plan local. Ruas [1999, p.72] représente les niveaux d'analyse avec les notions d'objets micro / méso / macro, et ces notions ont été utilisées dans le projet européen AGENT (ESPRIT/LTR/24939, cf.[Lamy et al. 99]).

Nous partageons également cette vue, et pensons que la focalisation sur le bon espace de travail est nécessaire pour automatiser efficacement le processus de généralisation cartographique. Le besoin de focalisation et d'un enchaînement adapté de différents algorithmes peut être résumé dans le modèle de généralisation proposé par Ruas et Plazanet [1996] illustré en Figure 13.

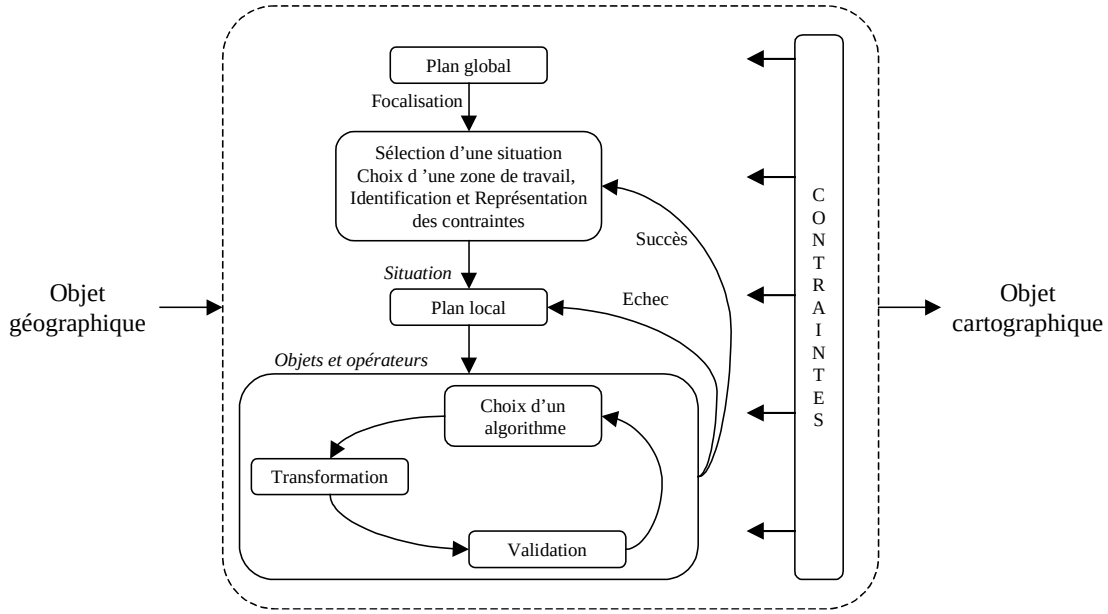


Figure 13. Focaliser et enchaîner les algorithmes, d'après [Ruas et Plazanet 96]

A.4 Recueil des connaissances de généralisation

Que les connaissances utilisées soient situées à l'intérieur ou à l'extérieur des algorithmes, il est nécessaire de les recueillir. En particulier il est nécessaire de recueillir des connaissances pour guider le processus :

> *"En théorie, une approche à base de connaissances nécessite que notre connaissance de la généralisation soit formalisée en une chaîne de raisonnement [Müller 91]. […] Cependant, en pratique, il y a peu de consensus sur comment la généralisation doit être conduite ; c'est à dire qu'il existe un manque de connaissances procédurales"* [João 98]

Plusieurs approches, de l'analyse des cartes existantes à l'apprentissage automatique, ont été envisagées pour recueillir ces connaissances dans le but d'automatiser l'application d'algorithmes, en répondant à diverses questions. Quel(s) algorithme(s) utiliser sur un objet donné ? Quelles opérations doivent être réalisées sur tel objet ? Sur quels objets tel algorithme est-il efficace ? Comment choisir les paramètres d'un algorithme ? Quelle mesure est adaptée à la qualification de tel problème ? etc.

Identification des connaissances impliquées

Des travaux se sont concentrés sur l'identification des connaissances impliquées dans le processus de généralisation cartographique. Plusieurs auteurs plaident pour une différenciation des connaissances "géométriques", "structurelles" et "procédurales" [Armstrong 91 ; McMaster 95 ; Weibel, Keller et Reichenbacher 95], en y distinguant parfois les connaissances dites de "Gestalt"[9]. D'autres auteurs distinguent les connaissances

[9] La théorie du Gestalt est issue de travaux en psychologie en particulier sur la perception de groupes. Elle statue que l'on perçoit un ensemble d'objets comme un groupe si ces objets satisfont des critères de similarité, de proximité, de symétrie, de fermeture ou d'alignement (cf. par exemple [Rock 83] pour une description de cette théorie et [Thórisson 94] ou [Regnauld 98] pour des utilisations informatiques de celle-ci). Le terme Gestalt est souvent un peu abusivement utilisé en cartographie pour désigner tout ce qui concerne les critères perceptuels, tous les effets d'optique.

"géométriques", "topologiques", "contextuelles" et "culturelles" [Kilpeläinen 2000]. Ces types de connaissances restent très généraux, et ne sont pas associés à un contexte d'utilisation. Leur utilisation reste pour l'instant du domaine de l'aide à la réflexion.

Les travaux sur l'identification des opérations effectuées en généralisation, que nous avons déjà cités dans la partie A.2, participent aussi à cette identification des connaissances impliquées dans la généralisation.

Analyse de cartes

Un premier type de recueil de connaissances opérationnelles a été effectué par dénombrement des objets présents sur des cartes réalisées manuellement à différents niveaux de généralisation [Töpfer et Pillewizer 64]. Ceci a permis de définir des règles très générales sur le nombre d'objets à représenter, mais cette approche ne donne que très peu de connaissances sur le choix des objets à conserver.

Plus généralement, si les cartes existantes sont indéniablement des sources d'information riches, il est souvent difficile d'en faire ressortir des règles utiles pour guider le processus. Ces cartes sont issues d'une très grande quantité d'opérations sur de très nombreux objets, il est donc difficile d'en extraire des règles granulaires concernant un ou quelques objets et une ou quelques opérations. De plus, ces cartes ont été réalisées par des techniques traditionnelles de dessin, très éloignées de notre approche algorithmique : beaucoup d'opérations élémentaires ont été effectuées simultanément avec une vision très globale de la carte.

Les algorithmes de généralisation ont été conçus par analyse de cartes : une opération récurrente est détectée sur les cartes, des algorithmes sont développés pour obtenir un résultat similaire. Mais on constate que savoir quand et comment utiliser ces algorithmes n'est pas assez formellement identifié par cette approche.

L'analyse de carte reste donc un moyen d'obtenir des connaissances cartographiques générales ou d'identifier des contraintes sur le résultat final à obtenir, mais cette approche est difficilement envisageable pour acquérir des connaissances assez fines pour suffire à l'automatisation.

Analyse de textes

Les cours de cartographie, où un effort de formalisation a été réalisé, sont une autre source d'information. Des analyses de cours ont été réalisées, elles concluent que les descriptions n'y sont pas assez détaillées pour en déduire des règles de généralisation [Müller et Mouwes 91]. De même, des analyses des manuels opérateurs réalisés dans les instituts de cartographie n'aboutissent qu'à des principes très généraux, comme par exemple des déterminations de seuils sur la taille des objets pour guider leur sélection [Nickerson 91]. La limite de l'analyse de textes pour recueillir des connaissances utiles à l'automatisation de la généralisation est résumée dans la citation suivante :

> *"Les descriptions contenues dans ces cours ou dans d'autres types de documents explicatifs sont souvent vagues, incomplètes et présentées sous forme de graphiques proposant des exemples sans explication (présentation de cas de généralisation favorables / défavorables). […] On ne peut donc qu'espérer extraire des documents textuels existants des règles de base simples qui pourraient également être énoncées par les cartographes" [Plazanet 96, p.157]*

Observation

Une autre possibilité est d'observer des cartographes en action, voire de tracer les opérations qu'ils effectuent lors de la réalisation de cartes sur ordinateur pour exploiter cette information. Malheureusement (pour nous) toutes les cartes ne sont pas encore réalisées sur ordinateur et surtout, lorsqu'elles le sont, les techniques numériques n'utilisent pas d'algorithmes tels que nous les concevons en recherche : ce sont des outils de dessin qui sont utilisés. La trace obtenue serait donc peu exploitable. On peut évidemment plaider pour une réalisation semi-automatique des cartes, c'est à dire par utilisation d'algorithmes. Mais cela n'est envisageable en pratique que si un fort travail d'ergonomie est réalisé et en particulier si les algorithmes sont conçus dans un objectif d'utilisation semi-interactive, ce qui n'est pas le cas actuellement. Si ces conditions ne sont pas réunies, l'utilisation d'outils de dessin est beaucoup plus rapide que celle d'algorithmes, et les contraintes de productivité plaident définitivement pour l'utilisation d'outils de dessin.

Seuls des tests ponctuels pour les besoins de recherche peuvent donc être actuellement réalisés. Des travaux ont étudié les composants nécessaires dans une plate-forme adaptée à de tels tests, comme par exemple : des algorithmes paramétrables, des outils de dessin classique, une interface permettant d'afficher des résultats intermédiaires et d'effectuer de nombreux retours en arrière, des outils de traçage, des mesures de description des objets, des outils d'aide comme des suggestions d'actions [Weibel 91 ; McMaster et Mark 91 ; McMaster 95].

De tels tests ont été réalisés par le groupe de travail en généralisation de l'OEEPE [Ruas 98b], sur des plates-formes de généralisation commerciales ou de recherche. Ils imposaient à des cartographes, sur différentes plates-formes, de généraliser des jeux de données en utilisant uniquement des algorithmes au lieu des outils de dessin classiques. Ces tests étaient participatifs car ils demandaient aux cartographes de décrire les raisons de leurs actions ainsi que d'en évaluer le résultat. Ces descriptions étaient guidées : les cartographes utilisaient une liste prédéfinie de conflits cartographiques potentiels pour décrire les raisons qui incitaient à l'action. Ces tests ont permis de mettre en évidence des relations nettes entre conflit détecté et action réalisée et de mieux comprendre l'enchaînement des algorithmes. Mais chaque plate-forme contenait ses propres algorithmes, ce qui a parfois limité les possibilités de comparaison entre les différents tests. Par ailleurs, la principale limite de ces tests a résidé dans le fait que chaque plate-forme contenait un nombre restreint d'algorithmes. Les cartographes contournaient donc le manque d'un algorithme pour leur besoin par l'application d'un ou plusieurs algorithmes en théorie peu adaptés au problème, ce qui relevait de l'astuce difficilement automatisable (e.g. utiliser un algorithme de déplacement pour élargir un virage). Ces tests ont permis d'améliorer la compréhension du processus de généralisation, mais l'absence de collecte de mesures sur chaque objet traité n'a pas permis d'établir de critères numériques d'utilisation des algorithmes. Il serait sûrement très profitable de réitérer de tels tests quand des plates-formes plus complètent existeront.

Interviews guidées

D'autres travaux ont effectué des interviews d'expert ou leur ont donné des exercices à réaliser, en leur demandant d'expliquer leurs choix [Kilpeläinen 2000]. Ces travaux ont abouti à l'obtention de règles générales, mais pas encore assez formalisées, ni assez granulaires pour être reliées à l'utilisation d'algorithmes. Deux raisons peuvent expliquer les limites de ces approches. Tout d'abord, "les cartographes travaillent de façon holiste et ont des problèmes pour décomposer leur travail en séries d'opérations" [Weibel 91]. Ensuite, "ce qui rend le savoir cartographique très spécial, c'est qu'il est essentiellement graphique et donc difficile à décrire avec des mots" [Weibel, Keller et Reichenbacher 95].

Apprentissage automatique

Enfin, pour dépasser les limites des approches présentées ci-dessus, plusieurs travaux (issus en majorité de l'université de Zurich) ont suggéré d'utiliser des techniques d'apprentissage automatique pour recueillir ces connaissances. Certains proposent de concevoir des algorithmes sous la forme de réseaux de neurones, ou de déterminer avec des réseaux de neurones les paramètres d'algorithmes existants [Werschlein et Weibel 94 ; Keller 95]. Ils montrent alors que le choix de la représentation des entrées et sorties des réseaux est crucial et difficile pour les données en représentation vecteur.

D'autres travaux proposent d'utiliser le raisonnement à partir de cas pour choisir quels algorithmes appliquer sur un objet géographique [Keller 94 ; Weibel, Keller et Reichenbacher 95]. Certains travaux proposent aussi l'utilisation d'algorithmes d'apprentissage symbolique pour déterminer le paramètre d'un algorithme ou l'algorithme à appliquer sur un objet [Weibel, Keller et Reichenbacher 95 ; Plazanet, Martini-Bigolin et Ruas 98]. Enfin, certains travaux utilisent des algorithmes de clustering ou d'apprentissage supervisé symbolique pour enrichir la description des données [Plazanet, Martini-Bigolin et Ruas 98 ; Sester 98].

Tous ces travaux insistent sur la mise en évidence, d'une part, des outils nécessaires à la collecte d'information pour l'apprentissage (des mesures de description des objets et un environnement de SIG adapté) et, d'autre part, des domaines pour lesquels l'apprentissage pourrait être utile (choisir un algorithme, choisir un paramètre d'algorithme, ou enrichir les données). Mais ces travaux restent principalement théoriques et peu nombreux. Les quelques travaux ayant réalisé des implémentations ont principalement montré comment ces approches pourraient être implémentées et non leur efficacité, les règles apprises ayant été peu validées :

> *"Etant donné que l'application de techniques d'apprentissage automatique à la cartographie est un sujet de recherche relativement nouveau, seules quelques publications existent pour présenter une validation théorique de ces techniques, et encore moins présentent de réelles implémentations et expériences"* [Weibel, Keller et Reichenbacher 95]

Ces approches par apprentissage, même si elles n'ont pas encore totalement prouvé leur efficacité dans le domaine de la cartographie, sont néanmoins prometteuses. Leur étude sera le cœur de cette thèse.

A.5 Sujet et approche

Nous nous plaçons dans une approche où la généralisation est vue comme un processus de changement d'état progressif par application successive d'algorithmes. Ceci nécessite de savoir appliquer chaque algorithme sur un espace de travail adapté. En ce sens nous nous plaçons dans le cadre des principaux modèles de généralisation automatique [Brassel et Weibel 88 ; Ruas 99], dont celui qui a été utilisé dans le projet européen AGENT (ESPRIT/LTR/24939).

Nous désirons contrôler ce processus grâce à une base de connaissances. Celle-ci doit permettre, à un moment donné du processus, de guider le choix d'un algorithme à utiliser sur un objet en fonction de mesures le décrivant. Elle doit également permettre de guider la focalisation sur les espaces de travail adaptés aux algorithmes. Elle doit enfin permettre de décider quand arrêter le processus.

Cette approche nécessite en premier lieu une boîte à outils d'algorithmes de transformation géométrique, d'algorithmes de focalisation, et de mesures de description des objets.

Nous prenons pour *a priori* que nous ne disposons pas de mesures fiables pour évaluer la qualité d'un résultat quelconque de généralisation. L'évaluation des effets d'une généralisation des données sur leur utilisation a déjà fait l'objet d'études (e.g. [João 98]) mais, à notre connaissance, nous ne disposons pas d'outils fiables pour l'évaluation de la qualité d'une quelconque généralisation. Ce problème fait l'objet d'une thèse de Sylvain Bard, en cours depuis novembre 2000 au laboratoire COGIT.

Le but de notre thèse est de déterminer des méthodes pour recueillir les connaissances nécessaires au guidage de la généralisation cartographique automatique. D'un point de vue applicatif, les connaissances recueillies doivent permettre d'automatiser tout ou partie d'un processus de généralisation cartographique. L'apprentissage automatique est une des méthodes développées en Intelligence Artificielle pour recueillir des connaissances difficiles à formaliser. Suivant les pistes ouvertes dans [Weibel, Keller et Reichenbacher 95], nous nous intéresserons principalement à l'étude de cette approche.

Puisque nous concentrons nos travaux sur le recueil des connaissances de guidage, nous utiliserons autant que possible des algorithmes de transformation et des mesures de généralisation existants. Nous ne développerons de tels outils que lorsqu'un manque nous paraît rédhibitoire pour effectuer nos expérimentations.

Restrictions du problème

Afin de cibler nos recherches nous apportons quelques restrictions à ce problème général.

Nous restreignons tout d'abord le type d'objet traité. Si en théorie nous nous intéressons à tout type d'objet géographique, nous nous restreindrons en pratique à des objets "simples" (une route, un bâtiment…) et non à des groupes d'objets dont la structure des composants est une information primordiale (une ville, un réseau routier…).

Ensuite, nous ne prétendons pas recueillir toutes les connaissances nécessaires à la généralisation. Nous nous intéressons à la manière de généraliser un objet, c'est à dire au choix des transformations à lui appliquer. Nous éludons les problèmes liés au traitement simultané de plusieurs objets, et en particulier nous ne recherchons pas à déterminer des connaissances pour régir l'ordre optimal de traitement des objets, même si nous savons que cet ordre influence la qualité des résultats. Nous éludons aussi en partie les problèmes liés à l'adéquation du traitement avec le but de la carte. Ainsi, nous nous intéressons à un type de carte particulier, les cartes topographiques. Nous n'étudions pas directement les cartes sur lesquelles sont ajoutées des informations thématiques (démographiques, politiques, économiques, etc.) et qui utilisent les cartes topographiques comme fond. Ces problèmes de gestion de l'ordre de traitement et de la prise en compte du but ont fait l'objet de la thèse de Anne Ruas [1999] puis du projet européen AGENT.

Contraintes du problème

Nous posons quelques contraintes sur la source et le type des connaissances que les méthodes proposées doivent permettre de recueillir.

Tout d'abord, afin que les connaissances acquises soient réutilisables et évaluables, nous désirons obtenir des connaissances sous une forme très compréhensible pour les experts du domaine. Cette compréhensibilité nous paraît nécessaire pour évaluer les connaissances recueillies et ainsi évaluer la technique employée pour les recueillir. Elle est aussi nécessaire dans une approche système expert car un tel système est supposé être capable d'expliquer le raisonnement qu'il emploie. La justification des choix réalisés par le système est en effet essentielle dans un contexte d'aide à la décision, ou pour identifier les forces et faiblesses du

système dans une optique de maintenance ou d'évolution du système. Enfin, même si nous concentrons notre travail sur la conception de cartes topographiques, il nous paraît essentiel qu'une partie de ce travail soit réutilisable pour réaliser d'autres types de cartes, ou d'autres cartes topographiques que celles de l'IGN qui nous servent de référence. Il nous paraît aussi nécessaire que les résultats de ce travail puissent évoluer en fonction des avancées dans le domaine de la recherche en généralisation. Pour que ces besoins de souplesse et d'évolution soient satisfaits, il est nécessaire que les connaissances apprises soient modifiables et donc compréhensibles.

Ensuite, pour que la méthodologie mise en place dans nos travaux soit réutilisable pour d'autres problèmes (autres objets traités, autres types de cartes), il est nécessaire que le recueil des connaissances ne soit pas trop lourd. En terme d'apprentissage à partir d'exemples cela se traduit par la recherche d'un nombre faible d'exemples à recueillir. En effet, si l'apprentissage est automatique, le recueil des exemples nécessite l'action d'un expert du domaine et peut être long.

La problématique de cette thèse peut donc être illustrée par la Figure 14 suivante et résumée ainsi :

Soit un objet géographique à un moment quelconque du processus de généralisation cartographique. Comment recueillir des connaissances efficaces et compréhensibles pour déterminer quel prochain algorithme lui appliquer ? Comment cela peut-il être réalisé grâce aux techniques d'apprentissage automatique ?

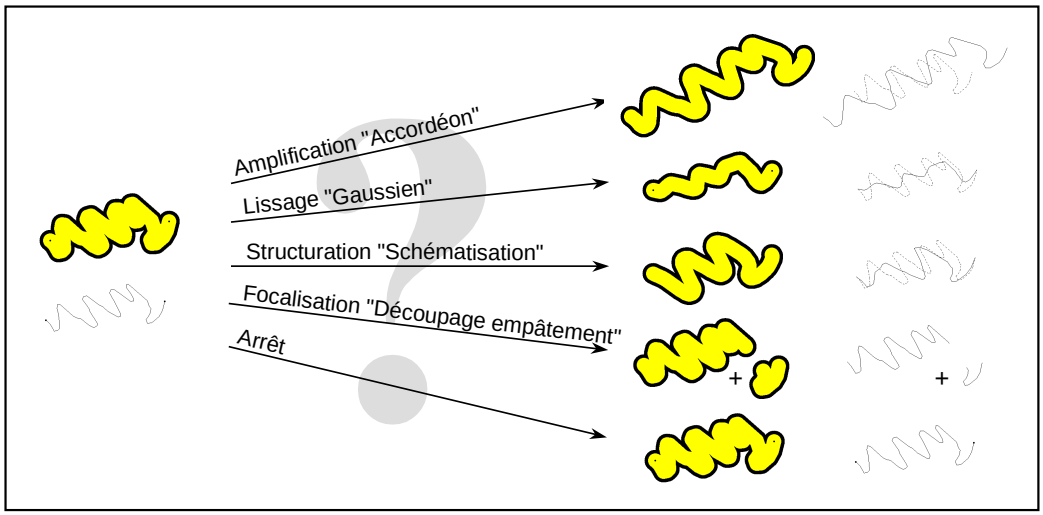


Figure 14. Quelle prochaine étape effectuer sur cette route ?

Première étude : définition empirique d'un processus

Nous continuerons tout d'abord dans le chapitre B les travaux initiés dans la thèse de Corinne Plazanet [1996]. Au cours de ces travaux de nombreuses connaissances sur la généralisation indépendante des routes ont été recueillies et traduites en partie en algorithmes de mesure, en algorithmes de transformation géométrique, et en définition de processus. Ces travaux, dont une grande partie a été consacrée au problème complexe de la détermination de mesures caractérisant une route, n'ont pas abouti à un processus complet de généralisation des routes faute de temps et, à notre avis, faute d'outil stable de focalisation sur le bon espace de travail des algorithmes. Cette première étude nous permettra de déterminer la pertinence de

l'approche générale (traitement par applications successives d'algorithmes sur des espaces de travail adéquats) et les limites liées au recueil des connaissances effectué.

Apprentissage automatique

Nous étudierons ensuite l'utilisation de l'apprentissage automatique dans les chapitres C, D et E. Le chapitre C présente les grandes lignes de l'apprentissage automatique supervisé, le chapitre D définit l'approche proposée pour utiliser l'apprentissage dans le contexte de la généralisation cartographique, et le chapitre E expérimente cette approche sur le cas particulier du traitement des routes.

■

B GENERALISATION CARTOGRAPHIQUE DES ROUTES : LE PROCESSUS GALBE

◆ Dans ce chapitre, nous appliquons les principes définis dans le chapitre A au cas de la généralisation cartographique des routes. Nous définissons un processus pour mettre en valeur les virages importants d'une route et éliminer les autres, en prenant une approche focalisée et pas à pas. Nous aboutissons ainsi au processus GALBE (Généralisation Adaptative du Linéaire Basée sur l'Empâtement).

◆ Nous montrons l'efficacité du processus, mais aussi ses limites : les cas complexes sont mal gérés et la mise au point du processus est difficile. Nous proposons de dépasser ces limites grâce à l'apprentissage automatique à partir d'exemples.

B.1 Domaine d'application : les routes pour les cartes routières

Nous prenons les routes comme champ d'application principal de nos recherches dans ce chapitre pour plusieurs raisons :

- De nombreuses recherches ont été effectuées sur les objets linéaires en général et sur les routes en particulier. Plus particulièrement, le traitement des routes a fait l'objet de nombreuses recherches au laboratoire COGIT. Deux thèses [Plazanet 96 ; Fritsch 97] ainsi que de nombreux stages et autres travaux y ont été entièrement consacrés, et la plate-forme de recherche PlaGe [Lecordix, Plazanet et Lagrange 97] a été principalement utilisée pour ce thème. Nous disposons donc à la fois de bonnes connaissances et de nombreux outils pour cette problématique.
- Malgré ces nombreuses recherches, l'automatisation de la généralisation cartographique des routes, même considérées une à une indépendamment de leur contexte, reste un problème non résolu. Ce problème est donc d'une complexité suffisante pour y consacrer nos recherches.
- Les routes sont des objets particulièrement importants sur les cartes. En tant qu'objets géographiques elles sont des voies de communication influençant fortement l'espace géographique lié aux activités humaines. Elles sont aussi représentatives de l'espace géographique naturel (sinueuses en montagne, droites en plaine). Elles sont donc représentées sur la quasi totalité des cartes. En tant qu'objets graphiques elles structurent l'espace et sont utiles à une lecture aisée de la carte [Eastman 85], la qualité de leur représentation est donc importante pour que la lecture de la carte entière soit efficace.
- 80% des éléments présents sur une carte topographique d'échelle moyenne sont des lignes [Müller 91] et une partie de nos travaux sur les routes pourront certainement être réutilisés pour d'autres objets linéaires.

Nous nous concentrons sur la généralisation des routes de la BDCarto de l'IGN (BD de précision décamétrique issue à l'origine de la numérisation des cartes au 1:50.000) pour la création des cartes routières au 1:250.000. Nous nous restreignons au traitement de chaque route prise isolément. C'est-à-dire que nous considérons les routes une à une et que nous ne cherchons pas à résoudre les problèmes liés à la représentation cartographique d'un ensemble de routes, ou à la représentation des routes conjointement à d'autres thèmes (l'hydrographie, l'orographie, l'occupation du sol…). En terme d'opérations, ceci signifie que nous ne traitons pas les déplacements ni la sélection. En interactif, ce problème peut représenter la moitié du temps de travail nécessaire pour rédiger les cartes routières à partir de la BDCarto, principalement à cause du traitement des zones de montagne. La généralisation au 1:250.000 à partir de la BDCarto est considérée comme un champ d'expérimentation des recherches intéressant et est identifiée comme un besoin par l'IGN. C'est ce cadre qui a guidé les recherches en généralisation des routes au COGIT jusqu'à présent.

Dans ce cadre, le but est de rendre lisible chaque route considérée : il faut éliminer les petits virages considérés comme du bruit qui brouillent la lecture de la route, et il faut rendre les virages conservés lisibles. Les principales difficultés apparaissent en zone de montagne où les routes sont très sinueuses et où les virages, une fois symbolisés, se superposent.

B.2 Règles de généralisation cartographique des routes

La généralisation manuelle est guidée par un ensemble de règles et de techniques peu formalisées. Un recensement des règles et techniques mises en œuvre pour le traitement des routes a été fait en détail par C. Plazanet [1996, pp.26-36] à partir d'analyses de cartes, de cours de cartographie [Weger 94 ; Cuenin 72], d'articles, ou par interview de cartographes (principalement G. Weger). Nous en relevons ici les principaux points.

Contraintes à respecter [Plazanet 96, pp.26-32] :

- Il faut assurer la lisibilité du symbole. C'est-à-dire que le symbole ne doit pas être empâté (conflit de superposition du symbole sur lui même), et que les virages trop petits qui peuvent être perçus comme du bruit doivent être éliminés.
- Il faut maintenir la forme globale de la route (sinueuse à tendance fractale, composée de lignes droites, composée de courbes régulières, etc.) [Ruas et Lagrange 94].
- Il faut maintenir l'aspect local de la route. En particulier il faut conserver les sinuosités relatives entres différentes portions de routes, une route faiblement sinueuse ne doit pas apparaître complètement lisse, et tout élément rectiligne doit le rester.
- Il faut conserver voire mettre en valeur les virages isolés et les séries de virages en épingles à cheveux.
- Il faut assurer une certaine précision planimétrique. Cette contrainte peut être représentée sous la forme d'une précision absolue minimale lors de la généralisation, ou d'une précision moyenne lors d'un contrôle post-traitement [Mustière 95].
- Il ne faut pas que l'axe de la route traitée s'auto-intersecte. Cette règle est nécessaire à la conservation de l'intégrité topologique de la base de données. Elle n'est pas liée à des contraintes réellement cartographiques, qui concernent le symbole et non l'axe de la route.

Solutions pour respecter les contraintes [Plazanet 96, pp.26-36] :

- Un détail peut être conservé et amplifié s'il est caractéristique (i.e. isolé où de forme particulière), même s'il est en réalité trop petit pour être représenté à l'échelle finale.
- On peut éliminer certains détails compatibles avec l'échelle mais qui altèreraient inutilement la valeur du message [Swiss Society of Cartography 87]
- Pour rendre lisible tout en élargissant au minimum un virage, on peut faire se recouvrir le bord du symbole (appelé le couteau) à l'intérieur du virage.
- On peut élargir une série de virages pour la rendre lisible, mais il faut conserver dans une certaine mesure l'amplitude et la forme générale de cette série.
- Des virages peuvent être éliminés dans une série de virages de formes équivalentes. On évitera néanmoins de supprimer les virages en début et en fin de série, ainsi que les virages les plus importants et les plus particuliers.
- Plus précisément, pour le cas d'une série de virages sujette à une violation de contrainte de symbolisation, l'exagération (élargissement) est préférable à la structuration (élimination de virages) puisqu'elle conserve le nombre de virages et donc l'information. Mais si le décalage généré est trop fort, alors la structuration est nécessaire. Cependant si il y a de la place autour des virages l'élargissement est préféré. [Plazanet 96, pp.171 et 185]

Par ailleurs, nous disposons d'un ensemble de connaissances sur la manière d'utiliser les algorithmes de généralisation des routes. Ces connaissances relatives à chaque algorithme seront précisées par la suite en fonction des algorithmes utilisés. Nous citons ici des connaissances très générales :

- Certains algorithmes s'appliquent sur un seul virage, d'autres sur une série de virages de forme relativement homogène, d'autres sur n'importe quelle portion de ligne.
- Certains algorithmes s'appliquent uniquement sur des virages ou séries de virages serrés, et d'autres uniquement sur des virages moins sinueux.
- Le paramétrage idéal des algorithmes dépend souvent de la symbolisation et de la forme de la route considérée.

Le but de la suite de ce chapitre est de définir un processus intégrant au mieux ces règles.

B.3 Le bon espace de travail pour les routes

B.3.1 Focalisation idéale

Comme expliqué au chapitre A, nous considérons l'automatisation de la généralisation comme l'application successive d'algorithmes sur des portions de ligne à un niveau de détail adéquat. Il nous faut donc tout d'abord définir quel est ce niveau de détail.

Pour identifier quels critères peuvent définir ce niveau de détail adéquat, plusieurs concepts peuvent être relevés dans les règles de généralisation du routier présentées ci-dessus. Il s'agit des concepts de lisibilité, de virage, de série homogène de virages, et d'homogénéité de sinuosité. Dans l'idéal, il serait donc utile de focaliser selon les zones lisibles et non lisibles, selon les limites d'un virage et d'une série homogène de virages, et selon la sinuosité. En effet, d'une part, le choix d'une opération est lié aux problèmes rencontrés (les zones non lisibles) et à la forme de l'objet à traiter (nombre de virages, sinuosité) et, d'autre part, les algorithmes effectuent des traitements homogènes qui ne se justifient que sur des objets homogènes.

Pour mettre au point un processus intégrant tous ces concepts il est nécessaire de savoir définir et mesurer chacun de ces concepts. Il est également nécessaire de savoir manipuler simultanément tous ces concepts, donc de savoir manipuler des focalisations multiples et non nécessairement hiérarchiques.

Corinne Plazanet [1996] a montré toute la difficulté qu'il y a à manipuler les notions de sinuosité, de virage, et de série homogène de virages. Tout d'abord, ces concepts sont difficiles à formaliser. Ensuite, les algorithmes de mesure implémentant ces concepts sont très sensibles au bruit. Elle propose des solutions pour résoudre ces problèmes, comme une analyse multi-niveaux de la sinuosité. Cependant des tests montrent que l'instabilité des outils proposés est toujours un problème.

Pour illustrer cela nous prenons un seul exemple, le découpage d'une ligne en virages d'un point de vue cartographique. Dans la ligne de la Figure 15 suivante, un cartographe pourra considérer qu'il n'y a qu'un seul virage de forme complexe si ce virage est son espace de travail (ici, le virage "cartographique" peut être ce qui permet de franchir un talweg). Ce concept est alors difficile à formaliser et à implémenter, même sans évoquer les problèmes dus au caractère discret du mode de représentation des lignes. Par exemple, si l'on définissait un virage comme la limite entre deux points d'inflexion, on détecterait aussi des virages sans importance ou dus à un bruit lors de la saisie. La limite entre virage pertinent et virage non pertinent est un premier problème. De plus un virage "cartographique" peut contenir plusieurs

inflexions, et même ne pas être délimité par des points d'inflexions. De même la définition d'un virage comme une zone de forte courbure ne correspond pas nécessairement à la définition du cartographe.

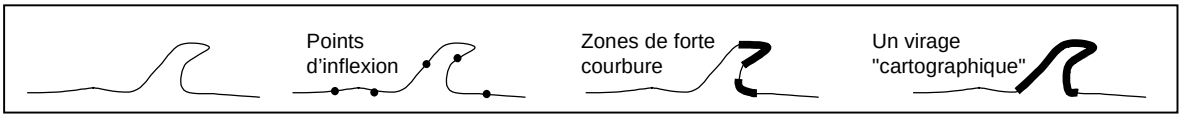


Figure 15. Qu'est-ce qu'un virage ?

La définition de la notion de virage est dépendante du point de vue et est difficile à formaliser. Il est difficile de concevoir des outils stables de délimitation de virages, il en est de même pour des outils de description de virages. Si la notion de virage est déjà difficile à manipuler, on conçoit aisément que les notions de série homogène de virages et de sinuosité sont d'autant plus complexes à manipuler.

Par souci de simplicité, nous proposons donc de nous intéresser uniquement à la notion de lisibilité, qui se traduit en partie pour les objets linéaires par la notion d'empâtement. Tout d'abord, éliminer l'empâtement est un des buts principaux de la généralisation de routes, cette notion est donc de toute première importance. Ensuite, comme nous l'expliquons ci-dessous, il est possible de définir une formalisation et un outil de détection des empâtements. Enfin, en première approximation, on pourra relier les notions de virage, série de virages, et de sinuosité à celle d'empâtement. En effet nous considérerons une zone non empâtée comme une zone peu sinueuse, une zone empâtée d'un seul coté comme un virage important serré, et une zone empâtée des deux côtés comme une série sinueuse de virages serrés (ces notions seront précisées par la suite).

Autrement dit, plutôt que de focaliser selon la forme des objets considérés (virage, sinuosité), nous proposons de le faire selon les conflits à traiter (empâtement). Nous verrons que si cette décomposition selon l'empâtement n'est pas une focalisation homogène idéale comme la définirait un cartographe, elle est néanmoins assez robuste et homogène pour être bien adaptée aux algorithmes disponibles et donc pour être utilisée en pratique.

B.3.2 Focalisation selon l'empâtement

La notion d'empâtement est liée à la fois à des notions de lisibilité et d'esthétisme, elle désigne les zones où le symbole d'une route se superpose "trop" à lui même (cf. Figure 16). Cette première définition est trop vague et subjective pour pouvoir être utilisée en pratique. Il nous faut donc définir plus formellement ce qu'est l'empâtement.

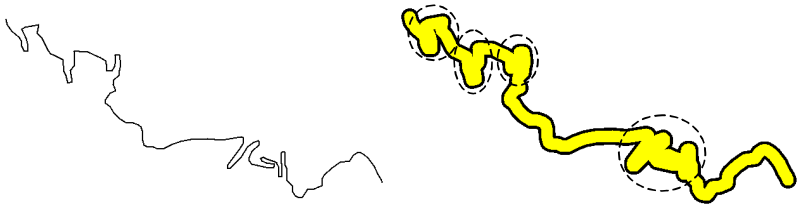


Figure 16. Empâtement interne d'une ligne symbolisée

B.3.2.1 Définitions théoriques de l'empâtement

L'empâtement d'une ligne dépend du symbole utilisé pour la représenter, à l'inverse de la notion de lisibilité d'une surface ou d'un groupe de surfaces. En effet lorsque l'on symbolise une surface on la "remplit", mais on ne "l'élargit" pas. Lorsque l'on symbolise une ligne on "l'élargit", et c'est cet élargissement qui est source de conflits de lisibilité. Par exemple, une route représentée avec symbole large de 1mm sur une carte au 1:1M correspondrait, sur le terrain, à une route large de 1 km. Cet élargissement est tel qu'il peut provoquer sur la carte la superposition de la route sur elle même : l'empâtement.

Les travaux cherchant à détecter les conflits de lisibilité à l'intérieur d'une surface ou d'un groupe de surfaces (e.g. goulots dans une surface [Bader et Weibel 97] ou proximités entre surfaces [Ware, Jones et Bundy 95 ; Ruas 98a]) étudient explicitement le contour des surfaces considérées pour détecter les conflits de lisibilité. Ce contour est directement disponible puisque les surfaces sont représentées par leur contour dans les BD cartographiques.

A l'inverse, la notion de symbole étant centrale dans la notion d'empâtement de ligne, nous préférons pour notre part rechercher les empâtements en étudiant explicitement le symbole de la ligne considérée, même si une ligne est représentée par son axe dans une BD cartographique. Nous proposons un premier pas vers une définition formelle de l'empâtement dans la définition suivante.

> *Première définition de l'empâtement : une ligne symbolisée est considérée empâtée si l'axe de la ligne qu'elle représente est difficile voire impossible à reconstruire à partir de la vision de sa symbolisation. C'est-à-dire qu'une ligne est empâtée si il y a perte d'information sur la forme de son axe lors de la lecture de son symbole.*

Cette définition nous permet de considérer différents niveaux d'empâtement pour une ligne (cf. Figure 17), selon que la reconstruction de son axe à partir de la lecture de son symbole est : immédiate (i.e. pas d'empâtement, cas a/), longue ou imprécise (cas b/), impossible (cas c/), trompeuse (cas d/ et e/).

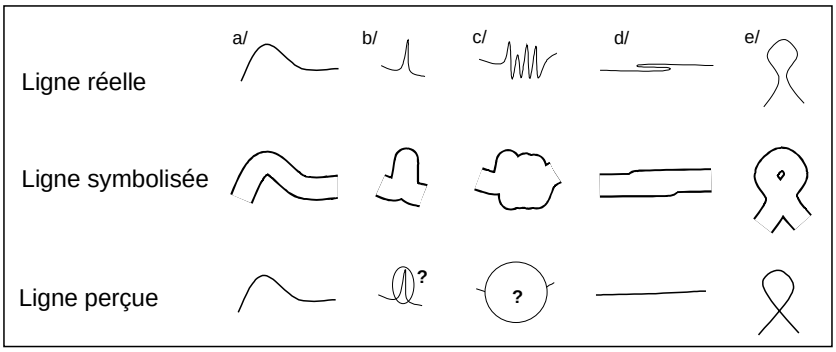


Figure 17. Ligne réelle versus ligne perçue à travers la symbolisation

Lors de la lecture d'une ligne symbolisée, on perçoit essentiellement ses deux bords, au milieu desquels se situe l'axe. Déterminer cet axe est d'autant plus difficile ou imprécis qu'un bord est éloigné de cet axe ou qu'une partie du bord est invisible (i.e. un bord est discontinu). Partant de cette caractéristique de l'empâtement, nous en proposons une deuxième définition plus formelle.

> *Deuxième définition de l'empâtement : une ligne symbolisée est considérée empâtée si un de ses bords s'éloigne trop de son axe ou qu'un de ses bord est discontinu.*

Il existe de nombreuses mesures d'écart entre deux lignes. Un écart adapté à notre besoin peut être défini, de manière intuitive, par la distance maximale entre un point du bord et la partie de l'axe qui lui "correspond". La définition suivante formalise cette notion (cf. Figure 18).

> *Définition de l'écart entre un bord d'un symbole et l'axe de la ligne qu'il représente. Soit P un point du bord, P se projette en un ou plusieurs point sur l'axe. Ces points sont ordonnés dans le sens de parcours de l'axe. Soit L la partie de l'axe comprise entre le premier et le dernier de ces points. Soit d_P la distance maximale entre P et L. L'écart bord-axe est le maximum de d_P pour P parcourant le bord du symbole.*

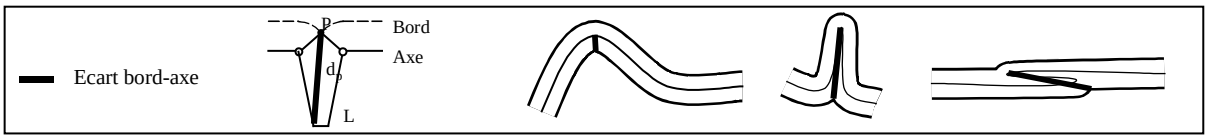


Figure 18. Ecart bord-axe

Par ces définitions, nous supposons que la limite entre ligne empâtée et non empâtée est due à un seuil sur la mesure de "l'écart bord-axe". Si la ligne considérée est une ligne droite, donc non empâtée, cet écart bord-axe est égal à la demi-largeur du symbole. Intuitivement le seuil définissant l'empâtement dépend de la largeur du symbole de la ligne.

B.3.2.2 Evaluation empirique des définitions de l'empâtement

Nous avons réalisé un test pour confirmer cette dépendance entre l'écart bord-axe et la notion d'empâtement, et pour déterminer une loi simple de relation entre le seuil limite et la largeur du symbole.

Nous montrons à plusieurs personnes (des chercheurs du laboratoire COGIT) un ensemble de lignes avec différentes formes et différentes largeurs de symbolisation comme celles de la Figure 19. Nous leur demandons quels virages doivent être considérés empâtés et lesquels ne le sont pas. Les virages sont alors classés en trois groupes : "empâté", "non empâté" et "limite". Un virage est considéré limite si plusieurs personnes ne sont pas d'accord sur sa qualification. Cette limite est certainement due à des critères flous ou esthétiques propres à chaque personne interrogée.

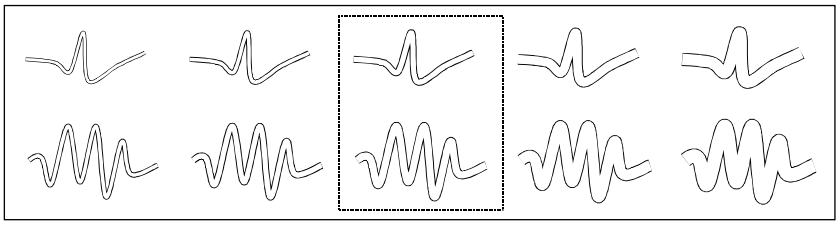


Figure 19. Limite entre zone empâtée et zone non empâtée

La Figure 20 montre les résultats de cette expérience : chaque point (classé empâté, limite ou non empâté) représente un virage d'une ligne donnée en exemple, la demi largeur du symbole de cette ligne est en abscisse, et l'écart bord-axe pour ce virage est en ordonnée. Cette figure montre qu'en première approximation la limite entre virage empâté et virage non empâté peut être définie par un seuil sur la mesure de l'écart bord-axe, et que la valeur de ce seuil est une fonction linéaire de la largeur de symbole. Plus précisément ce seuil se situe approximativement à 1,7 fois la demi-largeur de symbole (nous préférons utiliser la demi-largeur plutôt que la largeur de symbole car l'écart bord-axe est exactement égal à la demi largeur dans le cas des lignes droites).

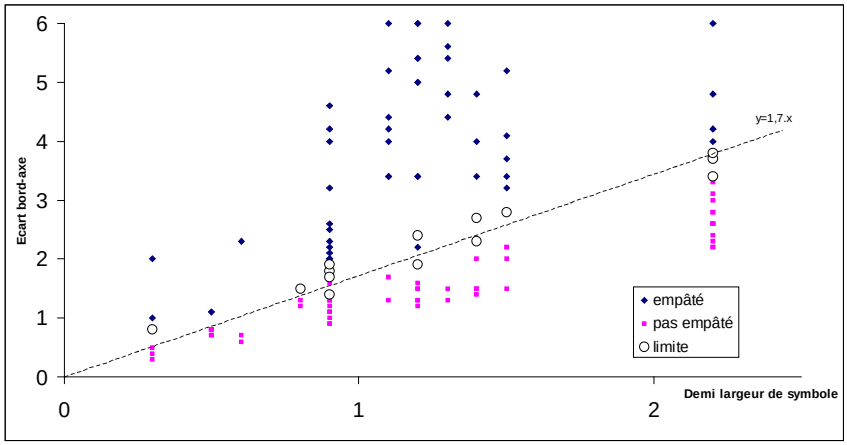


Figure 20. Corrélation entre largeur de symbole et écart bord-axe

Les définitions précédentes ainsi que ces tests nous permettent de définir précisément l'empâtement d'une ligne, et la notion d'empâtement d'un côté ou des deux côtés [Mustière 98a] :

> *Une ligne symbolisée est considérée empâtée si et seulement si l'écart entre un de ces bords et son axe est supérieur ou égal à 1,7 fois la demi-largeur de son symbole, ou qu'un de ses bords est discontinu. Quand la ligne est considérée empâtée, si l'écart bord-axe est supérieur au seuil pour un seul bord ou si un seul bord est discontinu l'empâtement est dit "d'un côté", sinon il est dit "des deux côtés". La valeur 1,7 est une valeur expérimentale moyenne dépendant de critères de lisibilité et d'esthétisme et qui peut être modifiée selon les besoins.*

B.3.2.3 Implémentation et résultats

La définition précédente nous permet de construire un algorithme découpant une ligne en zones empâtées et non empâtées. L'algorithme que nous avons implémenté construit tout d'abord explicitement les bords du symbole de la ligne puis recherche les points de la ligne où l'écart bord-axe dépasse le seuil de 1,7 fois la demi largeur du symbole. La ligne est ensuite découpée autour de ces points. Le détail du fonctionnement de l'algorithme est décrit en annexe (I-1 et I-2). Des variations de cet algorithme peuvent soit servir soit à détecter et qualifier les empâtements d'une ligne ("aucun", "d'un côté", "des deux côtés", "hétérogène"), soit à découper une ligne en parties empâtées ou non empâtées.

La Figure 21 montre un résultat de cet algorithme, sur l'image c/ les zones non empâtées sont représentées en blanc, les zones empâtées d'un seul coté en gris clair, et les zones empâtées des deux côtés en gris foncé. D'autres résultats peuvent être vus en annexe I-2. Les résultats montrent que la détection automatique des empâtements fournit des résultats proches d'une détection interactive, avec néanmoins une légère sur-détection des empâtements sur les virages isolés.

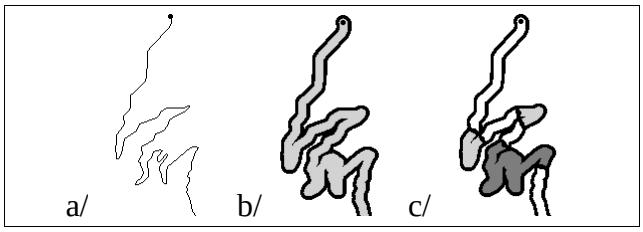


Figure 21. Axe, symbole et zones détectées empâtées d'une ligne symbolisée

B.4 Algorithmes de transformation

Le découpage par empâtement selon l'algorithme défini ci-dessus permet de décomposer une ligne en trois types de parties : les parties non empâtées (que nous identifions par simplicité aux parties éventuellement étendues mais contenant des virages peu serrés et peu importants qu'il faut simplifier), les parties empâtées d'un côté (que nous identifions aux virages importants serrés et isolés qu'il faut amplifier), et les parties empâtées des deux côtés (que nous identifions aux séries de virages en épingle à cheveux qu'il faut caricaturer).

Nous avons utilisé des algorithmes existants ou développé de nouveaux algorithmes pour réaliser chacune de ces opérations : simplification de virages mous, amplification de virages serrés, caricature de séries de virages serrés. La Figure 22 récapitule les différents algorithmes de transformation géométrique utilisés que nous détaillons brièvement par la suite et dont le fonctionnement est décrit plus en détail en annexes I et II.

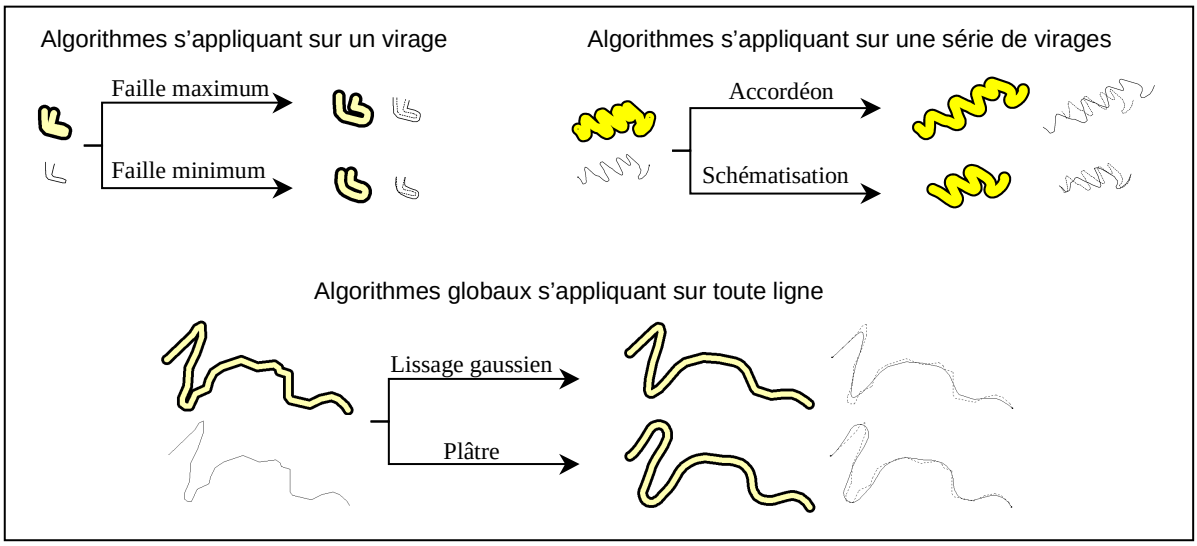


Figure 22. Algorithmes de transformation géométrique utilisés

B.4.1 Algorithmes de caricature d'une série de virages empâtée

Il existe deux façons de caricaturer une série de virages : soit en l'amplifiant, soit en la structurant (ou schématisant). Deux algorithmes ont été développés au COGIT pour effectuer ces opérations. Les versions de ces algorithmes que nous décrivons ci-après correspondent à des versions améliorées par rapport à leurs versions originales[10]. Ces amélioration ont été

[10] Ces améliorations ont été réalisées par F. Lecordix et nous-même et lors de stages réalisés au COGIT par H. Mauffrey et S. Skarbinck.

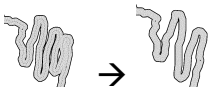
effectuées pour les rendre plus stables, plus efficaces sur les séries hétérogènes, et pour résoudre les problèmes de paramétrage.

Accordéon [Plazanet 96, p.166] 

Le but de cet algorithme (déjà évoqué dans la partie A.3.2.1 et détaillé en annexe II-2) est d'élargir une série de virages, de façon à résoudre les conflits d'empâtement à la base des virages. Il est conçu pour s'appliquer sur une série de virages relativement serrés.

Chaque virage de la série est élargi afin de résoudre les conflits d'empâtement à sa base. Cet élargissement est effectué perpendiculairement à l'axe principal du virage, d'une distance fonction de la largeur du symbole de la ligne. Pour cela il est nécessaire de déterminer la limite de chaque virage à l'intérieur de la série. Comme nous l'avons évoqué, cette opération n'est pas simple. Pour éviter de détecter des virages non pertinents, on détermine les limites des virages sur une version lissée de la ligne à traiter, puis on recherche les homologues de ces points limites sur la ligne à traiter [Plazanet 96, p.104]. Cet algorithme possède un paramètre appelé "exagération". Il s'agit d'un coefficient multiplicateur de la distance d'écartement de chaque virage, normalement égal à 1, mais qui peut être accentué pour forcer la généralisation.

Cet algorithme fonctionne particulièrement bien sur la plupart des séries de virages serrés, sauf si les virages ont des orientations très variées. Ses limites apparaissent sur les longues séries de virages où il peut dégrader la planimétrie en étirant fortement la série. Il peut légèrement modifier les orientations relatives de chaque virage et les différences de forme relative des virages mais cela se perçoit peu.

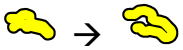
Schématisation [Lecordix, Plazanet et Lagrange 97] 

Le but de cet algorithme (détaillé en annexe II-3) est d'éliminer deux virages consécutifs dans une série de virages afin de libérer de la place pour les autres. Il est conçu pour s'appliquer sur une série de virages relativement serrés, surtout sur les longues séries de virages. Il est ainsi complémentaire à l'*Accordéon*.

L'algorithme détermine la limite de chaque virage par la même méthode que celle utilisée pour l'algorithme *Accordéon*. Il choisit ensuite deux virages consécutifs à éliminer en évitant le premier et le dernier virage, ainsi que les plus grands et ceux avec les formes les plus particulières. La dernière étape consiste à recoller les virages qui précèdent et qui suivent les deux virages éliminés. Le recollage est effectué de manière amortie afin ne pas créer d'angularité visuelle. Cet algorithme n'a pas de paramètre.

B.4.2 Algorithmes de caricature d'un virage empâté

Pour caricaturer un virage il faut l'élargir. Cela peut se faire soit en conservant au mieux sa forme, soit en minimisant son emprise. Un algorithme, nommé "Baudruche", existe pour élargir le sommet des virages d'une série [Lecordix, Plazanet et Lagrange 97], mais celui-ci s'est révélé inadapté pour nos besoins, c'est à dire pour s'appliquer sur un seul virage tel que nous le définissons par l'empâtement. Nous avons donc développé deux nouveaux algorithmes de caricature de virages.

Faille Max [Mustière 98b] 

Le but de cet algorithme (détaillé en annexe I-3) est d'élargir un virage en conservant au mieux sa forme. Il est conçu pour ne s'appliquer que sur un seul virage, mais ce virage peut contenir plusieurs inflexions.

Il effectue une dilatation de la ligne au sens du terme utilisé en morphologie mathématique (ou un "buffer" pour prendre la terminologie SIG). La taille de cette dilatation est fonction de la largeur du symbole. Comme *Accordéon*, cet algorithme possède un seul paramètre appelé "exagération". Il s'agit d'un coefficient multiplicateur de la taille de la dilatation.

Cet algorithme conserve bien la forme d'un virage en en assurant la lisibilité. Mais il est surtout adapté au traitement des virages isolés car, comme il est susceptible de beaucoup écarter le virage, son application peut entraîner des conflits avec les objets entourant ce virage.

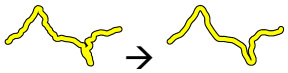
Faille Min [Mustière 98b] 

Le but de cet algorithme (inspiré d'une idée de F. Lecordix et détaillé en annexe I-4) est d'élargir un virage au minimum nécessaire pour qu'il ne soit plus empâté. Il est conçu pour ne s'appliquer que sur un seul virage, mais ce virage peut contenir plusieurs inflexions. Si l'environnement du virage est dense on le préfère à *Faille Max* car il écarte moins le virage.

L'algorithme imite la technique traditionnelle qui consiste à faire se recouvrir le bord intérieur du virage. Il calcule tout d'abord une ligne "squelette" du virage (au sens de la morphologie mathématique), puis effectue une dilatation de ce squelette de manière à ce que le bord intérieur du virage se recouvre. Cet algorithme possède un paramètre régissant le lissage de la ligne squelette, mais ce paramètre peut être relié à la taille du symbole utilisé.

Cet algorithme assure bien la lisibilité d'un virage. Il peut néanmoins en détériorer la forme. En particulier l'éventuelle dissymétrie d'un virage est gommée par cet algorithme puisque l'algorithme reconstruit une forme à partir du squelette du virage.

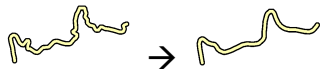
B.4.3 Algorithmes de simplification d'une ligne entière

Plâtre [Fritsch 97] 

Cet algorithme (déjà évoqué dans la partie A.3.2.1 et détaillé en annexe II-4), a pour but de lisser les virages les plus mous et d'écarter les virages les plus serrés.

A l'inverse des autres algorithmes que nous présentons, il ne travaille pas sur la représentation traditionnelle de la ligne en suite de segments mais sur une représentation de la ligne comme la courbure fonction de l'abscisse curviligne. Cet algorithme possède un paramètre régissant à la fois la force du lissage des virages mous et la force d'écartement des virages serrés. Ce paramètre est dépendant de la largeur du symbole considéré. Pour que l'algorithme soit utilisé au mieux, le paramètre doit aussi dépendre de la sinuosité de la ligne considérée.

Cet algorithme est conçu pour s'appliquer sur toute ligne. Il est surtout efficace si la ligne ne contient pas de virages serrés trop proches, sinon il risque d'aggraver les problèmes de lisibilité liés à cette proximité. Il réalise un compromis entre des contraintes de respect d'une courbure minimale, de respect de la planimétrie (en particulier au niveau du sommet des virages serrés) et de diminution de la granularité.

Lissage Gaussien 

Cet algorithme (détaillé en annexe II-1) a pour but d'éliminer les détails sans importance d'une ligne. Il fonctionne selon le principe d'une fenêtre glissante sur la ligne. Chaque point de la ligne est déplacé vers le barycentre des autres points de la fenêtre. Dans le calcul du barycentre, chaque point de la fenêtre est pondéré en fonction (gaussienne) de sa distance curviligne au point traité.

Cet algorithme possède un paramètre σ (écart-type de la gaussienne) représentant la force du lissage. Comme pour *Plâtre*, ce paramètre est dépendant de la largeur du symbole considéré mais il est difficile de le relier directement à cette largeur car il dépend aussi de la sinuosité de la ligne considérée.

Cet algorithme est conçu pour s'appliquer sur toute ligne. Mais il est mal adapté aux virages trop serrés qu'il ressert encore, et dont il diminue l'ampleur. Son but principal est d'éliminer des détails. Il intègre des contraintes secondaires de respect de la planimétrie et de la forme générale de la ligne.

B.4.4 Propagation des déformations

Certains des algorithmes présentés ci-dessus (*Accordéon*, *Faille Max* et *Faille Min*) déplacent les extrémités de la portion de route traitée. Il faut alors propager ces déplacements sur les portions de routes voisines, afin d'assurer la continuité du réseau routier (cf. Figure 23). Pour être efficace, cette propagation doit être amortie dans l'espace : un faible élargissement d'un virage ne doit pas déplacer les routes qui en sont très éloignées et dégrader ainsi inutilement la précision planimétrique de l'ensemble du réseau. Pour effectuer cette propagation amortie, nous utilisons un algorithme, développé par François Lecordix, qui propage les déplacements dans un réseau. Cet algorithme atténue le déplacement de tout point, en fonction de la distance curviligne le long du réseau entre le point déplacé et le point provoquant les déplacements. Cet algorithme est appliqué immédiatement après chaque utilisation d'*Accordéon*, de *Faille Max* ou de *Faille Min*.

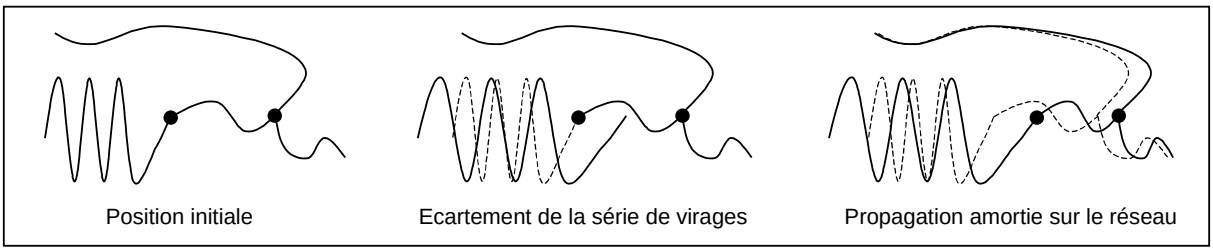


Figure 23. Propagation amortie des déformations

B.5 Processus GALBE

Tous les algorithmes définis ci-dessus ne s'appliquent pas dans les mêmes conditions. Il est donc nécessaire de déterminer des mesures pour guider le choix de l'algorithme à utiliser à un moment donné du processus.

B.5.1 Mesures de description

Les conditions de déclenchement d'un algorithme portent sur la forme de l'objet traité. On doit remplacer "l'œil" du cartographe qui analyse la ligne (est-elle sinueuse, longue,

empâtée…) par des mesures. La conception de mesures pertinentes pour décrire une ligne est un problème difficile qui a déjà fait l'objet de nombreux travaux dont plusieurs thèses [McMaster 83 ; Buttenfield 84 ; Plazanet 96]. Ces travaux permettent de conclure que des mesures pertinentes n'existent que sur des portions homogènes de ligne. Une série de mesures est donc nécessaire pour décrire une ligne, et l'agrégation de ces mesures sous la forme d'une ou plusieurs mesures générales pertinentes est un problème difficile. De plus tous ces travaux se sont confrontés au problème de l'utilisation de ces mesures dans le contexte de la généralisation. Il est difficile de trouver un lien direct entre ces mesures et les opérations de généralisation à réaliser. Par exemple, si de très nombreuses mesures ont été développées pour évaluer la sinuosité[11] d'une ligne (e.g. dimension fractale, étude de la taille et de la forme des virages…), à ce jour on ne sait pas relier ces mesures à la force du lissage nécessaire pour enlever le bruit d'une ligne trop détaillée.

Dans les règles de généralisation des routes, de nombreux critères sur les propriétés de la route doivent être pris en compte et donc mesurés : l'empâtement, la précision planimétrique (écart à la position initiale de la ligne), la forme générale, la sinuosité générale, la granularité, et la sinuosité de parties de ligne.

Déterminer quoi mesurer, comment le mesurer, et comment utiliser ces mesures sont des problèmes difficiles. Dans un premier temps, nous simplifions donc ce problème en n'utilisant que quelques mesures dont nous connaissons la stabilité, et qui sont suffisamment pertinentes, même si elles contiennent trop peu d'information pour être véritablement efficaces dans tous les cas. Nous préférons utiliser des mesures éventuellement trop peu informatives mais faciles à manipuler plutôt que des mesures plus riches mais difficiles à manipuler (comme un arbre de description d'une ligne [Plazanet 96, p.116]).

Nous nous limitons à l'utilisation d'une mesure de l'empâtement et à une mesure de l'écart planimétrique. Nous n'utiliserons pas de mesures pour qualifier des notions de forme d'un objet, en espérant *a priori* que les contraintes de respect de forme intégrées dans la plupart des algorithmes suffisent pour éviter une détérioration de ce caractère.

Nous avons défini la notion d'empâtement (cf. définition p.58). A partir de celle-ci, nous pouvons définir une mesure simple mais stable : l'empâtement d'une ligne qui peut prendre les quatre valeurs "hétérogène", "nul", "empâté d'un seul côté", "empâté des deux cotés".

Pour l'écart planimétrique, la distance de Hausdorff entre deux lignes représente un écart maximal entre ces deux lignes (étudiée en particulier dans [Abbas et Hottier 93]). Cette mesure, souvent étudiée et utilisée dans le domaine du contrôle de la qualité dans les SIG, est stable et répond à notre besoin.

B.5.2 Moteur du processus

Nous disposons de toutes les briques nécessaires à la définition de notre processus : un algorithme de focalisation, des algorithmes de transformation géométrique, et des mesures pour guider ces algorithmes. Ce processus a été défini empiriquement à la suite de nombreux essais. Nous commenterons la manière de définir ce processus par la suite.

Tout d'abord, nous formulons des règles d'utilisation des algorithmes présentés, en partant des règles peu formalisées de la généralisation du routier (cf. B.2) et des connaissances sur les algorithmes :

[11] Nous prenons ici le terme sinuosité au sens très général du terme. Cela recouvre les notions de sinuosité, granularité et complexité que nous différencierons pas la suite (cf. E.1.2)

- Si une ligne est hétérogène vis à vis de l'empâtement il faut la découper par l'algorithme de *Détection des empâtements*.
- Si une partie de ligne ne possède pas de conflit d'empâtement, nous la lissons avec l'algorithme de *Lissage Gaussien* pour en éliminer les virages trop petits. Nous faisons en effet l'hypothèse, souvent vérifiée empiriquement, qu'une ligne non empâtée est trop détaillée, c'est-à-dire visuellement bruitée, et qu'un lissage enlèvera ce bruit.
- Si une partie de ligne possède un conflit d'empâtement d'un seul coté, nous la considérons comme un seul virage (la notion d'empâtement nous permet donc de définir ce qu'est un virage vis-à-vis de nos besoins). Si ce virage est situé au sein d'une série nous l'agrandissons par l'algorithme *Faille Min*, sinon par l'algorithme *Faille Max*.
- Si une partie de ligne possède un conflit d'empâtement des deux côtés à la fois, nous la considérons comme une série de virages à caricaturer (la notion d'empâtement nous permet donc de définir également ce qu'est une série de virages vis-à-vis de nos besoins). Nous lui appliquons donc *Accordéon*, sauf si cela dégrade trop la planimétrie au sens de la *Distance de Hausdorff*, auquel cas nous appliquons l'algorithme *Schématisation* pour faire de la place aux virages avant d'appliquer l'algorithme *Accordéon*.
- Il est nécessaire de traiter un à un les virages d'une série après l'application de *Accordéon* et *Schématisation*, de manière à éliminer l'empâtement résiduel au sommet des virages, ce qui n'est pas traité par ces algorithmes. Il faut donc appliquer à nouveau l'algorithme de *Détection des empâtement* sur une série de virages après application de ces algorithmes.

Ceci nous permet de définir le processus GALBE (pour *Généralisation Adaptative du Linéaire Basée sur l'Empâtement*) décrit dans la Figure 24 (formalisme : diagramme d'activité UML) [Mustière, 98b].

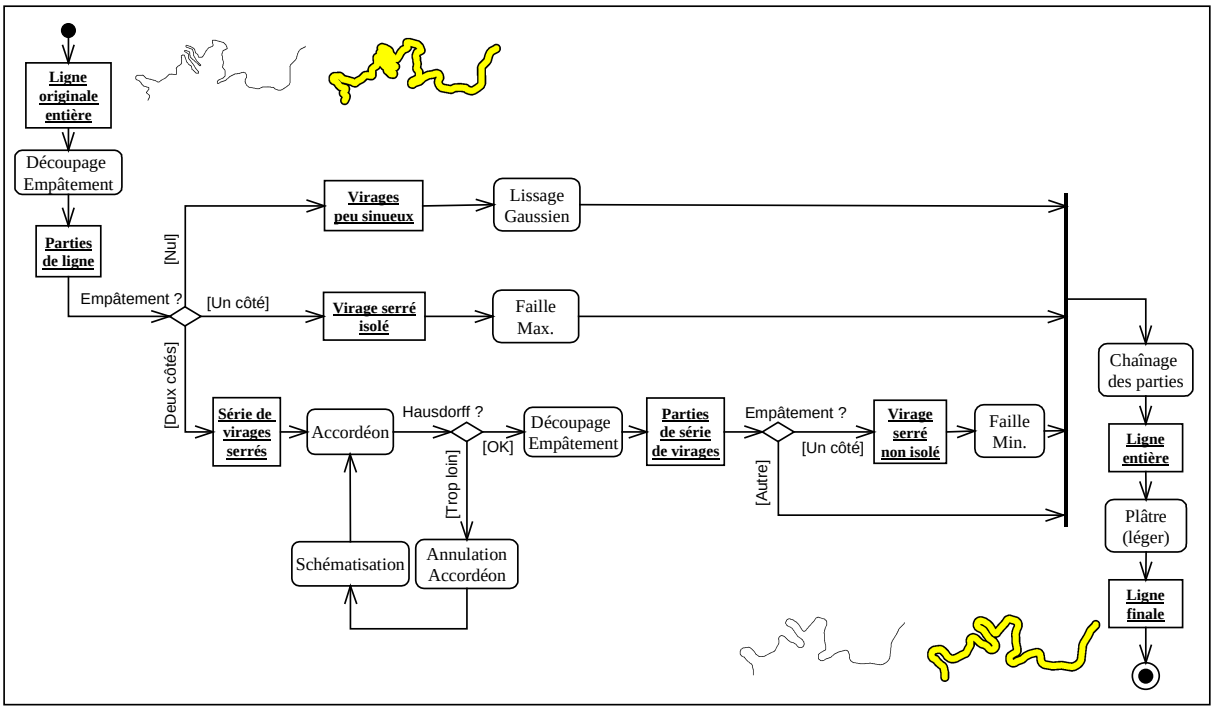


Figure 24. GALBE

Précisons différents points de ce processus :

- Les paramètres des algorithmes sont fixés empiriquement en fonction de l'échelle et sont identiques tout au long du processus et pour toute ligne.
- Le premier test sur l'empâtement est réalisé sur l'ensemble des parties de la ligne. L'ordre dans lequel ces parties sont traitées est le suivant : 1/ traitement des parties empâtées des deux côtés, 2/ traitement des parties empâtées d'un seul côté, 3/ traitement des parties non empâtées.
- Le chaînage des parties consiste à relier les différentes parties traitées séparément afin de considérer à nouveau la ligne dans son ensemble.
- Le test sur la distance de Hausdorff est effectué entre la série de virages avant traitement et la série de virages en cours de traitement. Le seuil déterminant l'écart acceptable est défini en fonction de la précision requise dans les spécifications de la carte à réaliser.
- Après l'application de chaque algorithme, si celui-ci déplace les extrémités de la portion de ligne concernée, ce déplacement est propagé sur l'ensemble de la ligne afin d'assurer la continuité de la ligne.
- *Plâtre* n'est pas utilisé pour ses capacités de caricature, il n'est utilisé que pour lisser légèrement la ligne entière afin de gommer les éventuelles angularités parasites subsistant après le recollage des parties. Cette opération pourrait aussi être réalisée avec le *Lissage Gaussien* sans sensiblement modifier le résultat.
- Enfin, un contrôle final du processus annule tout le traitement si une erreur de topologie (auto-intersection de la l'axe de la ligne) est constatée sur la ligne finale. Ceci est réalisé afin de faciliter une éventuelle retouche interactive après le traitement automatique.

B.6 Evaluation des résultats

D'un point de vue cartographique, les résultats de GALBE sont globalement satisfaisants. Par exemple, la Figure 25 présente en détail les résultats de GALBE sur deux arcs routiers symbolisés au 1:250.000 (ces arcs sont issus de la BDCarto sur les Alpes Maritimes). Sur ces arcs, les résultats sont de qualité proche de celle d'une généralisation interactive. Les figures des pages suivantes présentent les résultats de GALBE sur une zone plus étendue (entre Argelès-Gazost et le Cirque de Gavarnie dans les Pyrénées). Le traitement de cette zone a nécessité environ 5 minutes de calcul (sur une station Alpha 500).

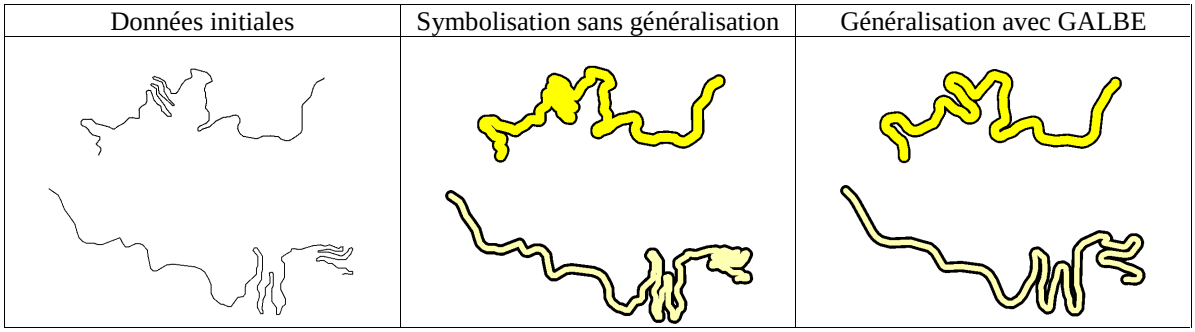


Figure 25. Résultats de GALBE

D'autres résultats peuvent être vus en annexe VII. Une évaluation intensive de GALBE, présentée ci-après, a été réalisée afin d'en déterminer plus précisément les qualités et défauts.

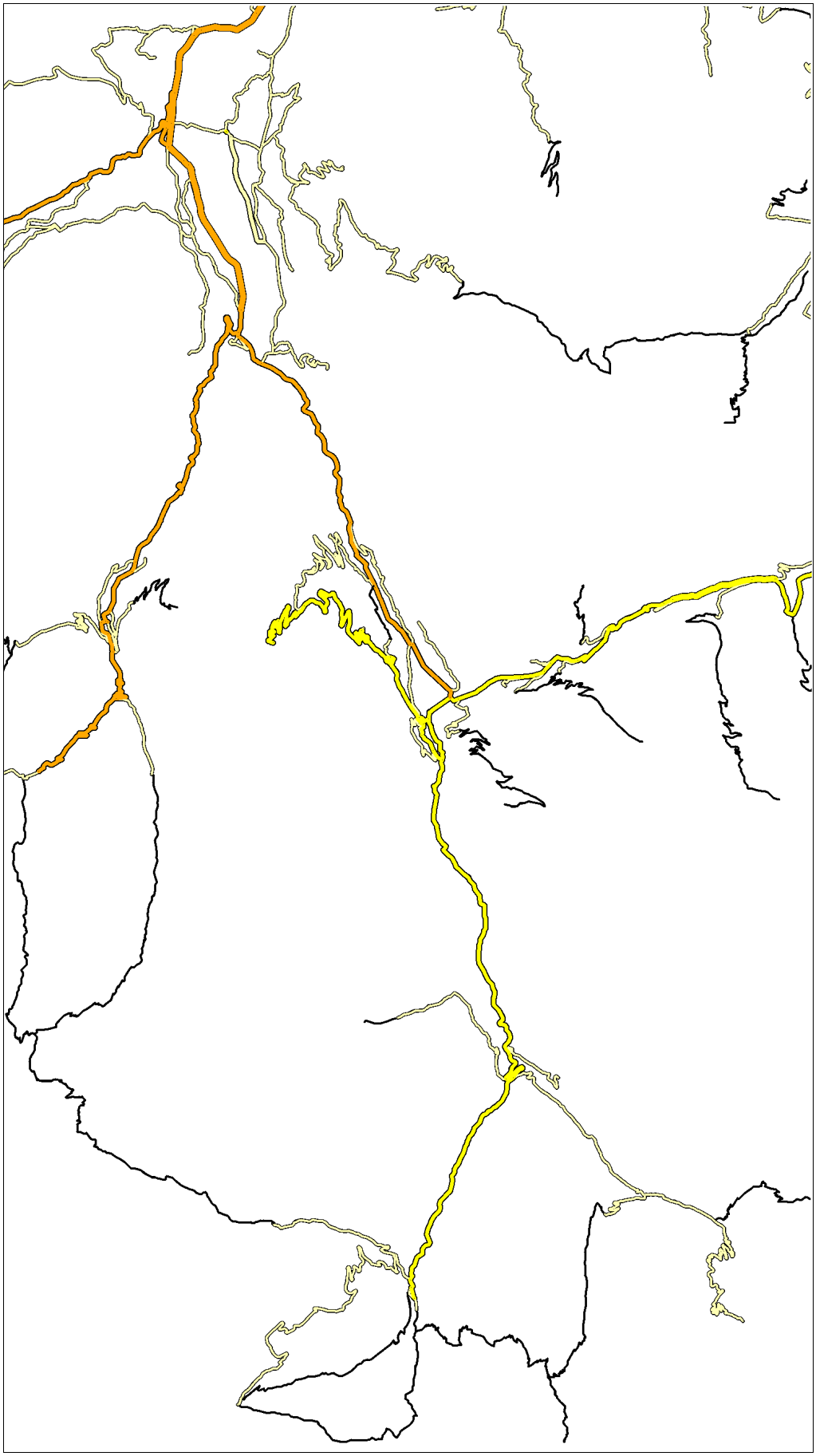


Figure 26. Extrait de la BDCarto, symbolisé au 1:250.000 avant traitement

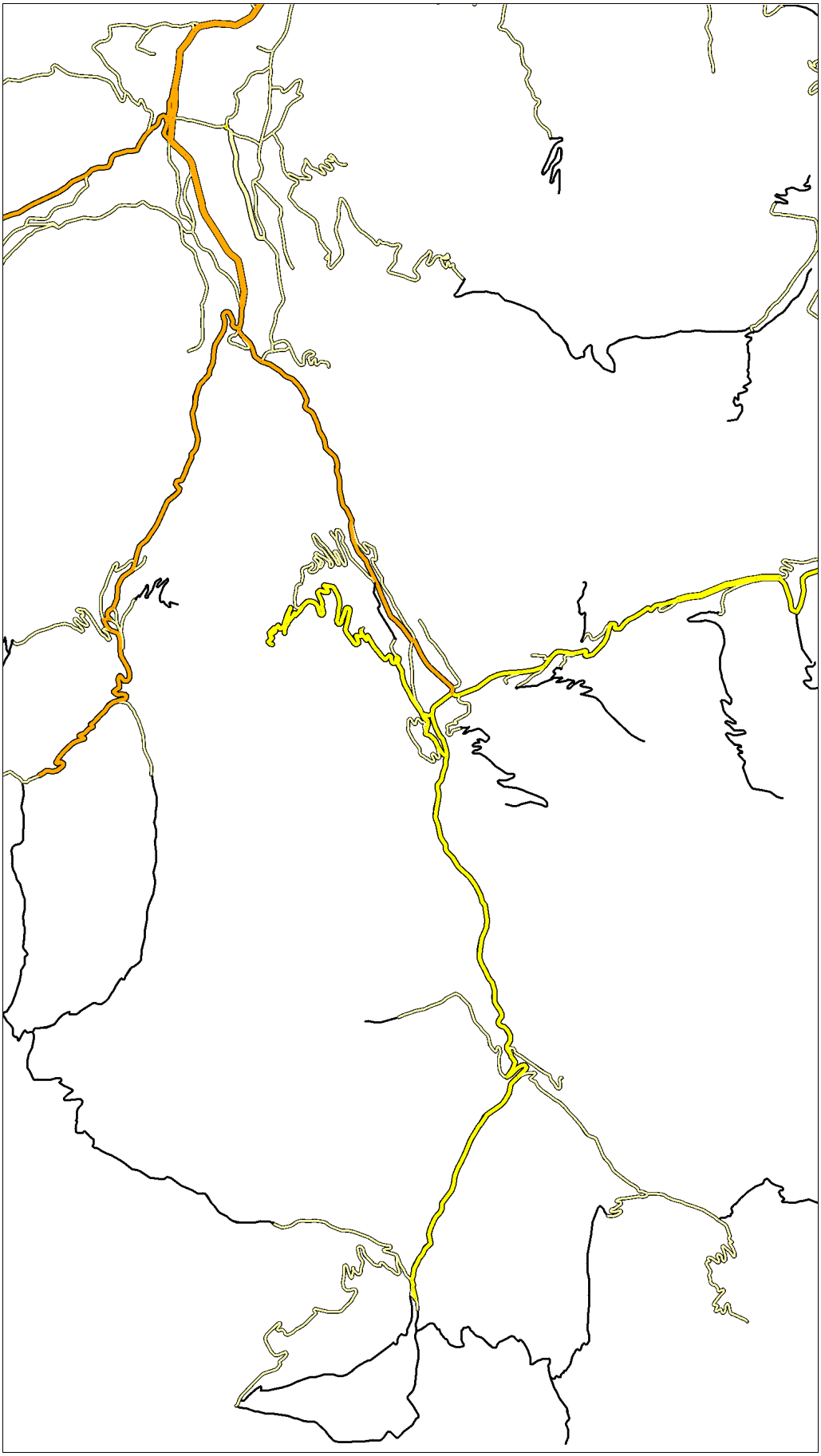


Figure 27. Résultats de GALBE

B.6.1 Analyse par des cartographes

Les résultats de GALBE ont été montrés à différents cartographes pour recueillir leurs commentaires[12]. Ce test a été réalisé sur une zone carrée de 50 km de côté dans la région de Valence, zone qui contient à la fois des routes de plaine (vallée du Rhône) et des routes de montagne (Massif Central et Alpes). Nous avons présenté à ces cartographes quatre sorties graphiques : la zone avant symbolisation, la zone symbolisée non généralisée, la zone symbolisée généralisée avec GALBE, la zone symbolisée généralisée avec *Plâtre*. Nous leur avons présenté les résultats de *Plâtre* car nous considérons qu'il représente le meilleur algorithme existant directement comparable à GALBE. Notons qu'aucune indication n'a été fournie aux cartographes sur le fonctionnement des algorithmes.

Nous relevons ici les commentaires de Gérald Wéger [communication personnelle]. Les commentaires des autres cartographes vont dans le même sens.

- Les généralisations améliorent sensiblement la lecture de la carte, bien que le lissage des petits détails puisse être parfois plus poussé. Globalement la généralisation pourrait être plus poussée.
- GALBE est meilleur que *Plâtre* pour l'échelle considérée. *Plâtre* semble mieux adapté à une généralisation d'un niveau d'abstraction moins élevé, du 1:50.000 au 1:100.000 par exemple, pour lequel l'aspect relativement "topographique" se justifie encore et où la place disponible est encore suffisante.
- En plaine, le résultat de GALBE est satisfaisant. Mais le caractère sinueux de quelques zones a été perdu.
- En montagne, GALBE effectue des déplacements de virages et fait bien ressortir un caractère différent aux accidents principaux et aux sinuosités mineures. Il subsiste cependant des problèmes importants dans la traduction de certaines séries de lacets. Certaines sont encore mal traitées et GALBE n'élimine pas assez de virages en lacets.

Certains de ces commentaires nous ont permis d'améliorer le processus, notamment les reproches liés à la faiblesse de la généralisation. Ces améliorations ont été faites en modifiant quelques paramètres : le paramètre du *Lissage Gaussien* a été accentué et le seuil de l'écart acceptable sur la distance de Hausdorff a été diminué de façon à favoriser les actions de *Schématisation*.

D'autres commentaires sont révélateurs des limites de cette méthode : si dans le cas général le résultat est globalement satisfaisant, des cas particuliers sont mal traités. On peut voir des exemples de ce genre de problèmes au centre de la zone présentée en Figure 27.

B.6.2 Application au réseau routier des cartes au 1:250.000

Le processus GALBE ayant été jugé globalement satisfaisant par les services de production de l'IGN, il a été appliqué pour réaliser l'ensemble des cartes régionales au 1: 250.000 de

[12] Ces évaluations ont été orchestrées par François Lecordix (IGN/COGIT). Les cartographes consultés sont Gérald Weger (professeur de cartographie à l'Ecole Nationale des Sciences Géographiques) pour une étude très détaillée, et d'autres cartographes de l'IGN pour une étude plus générale : Gérard Chappart (professeur de cartographie à l'ENSG, ancien chef d'unité des cartes dérivées), Marie-José Josserand (chef d'unité des cartes dérivées), Josette Chenu, Maryline Mavro, Annie Gostiau et Patrick Petit (dessinateurs cartographes travaillant essentiellement sur les cartes au 1:100.000 et au 1:250.000). Nous leur adressons à nouveau nos remerciements.

l'IGN à partir d'un extrait du réseau routier de la BDCarto. Ceci représente environ 500.000 arcs représentant 500.000 km de routes de la France métropolitaine. Le traitement a nécessité environ 24 heures de calcul pour la France entière (sur une station Alpha 500), ce qui est satisfaisant pour un processus entièrement automatique remplaçant plusieurs mois de travail interactif.

Le fait que GALBE ait été utilisé en production ne signifie pas qu'il est parfait ni qu'aucun post-traitement ne mérite d'être réalisé pour obtenir la qualité des cartes réalisées manuellement. Mais du moins cela montre qu'il est robuste et qu'il est considéré comme performant pour réaliser une partie de la généralisation automatique du réseau routier : le traitement de la forme de chaque route indépendamment des autres objets. Les autres traitements nécessaires, non pris en compte dans GALBE, concernent les déplacements entres routes et leur sélection.

B.6.3 Bilan de GALBE

Les diverses évaluations que nous avons effectuées montrent l'efficacité du processus GALBE. Cela nous permet de valider l'approche générale, à savoir une approche focalisée et pas à pas. Cela confirme également que la focalisation selon l'empâtement est un choix pertinent. Cependant, nous relevons ci-dessous les limites de ce processus, d'une part en termes de résultats obtenus, et d'autre part en termes de mise au point et de capacité d'évolution du processus.

Limites en terme de résultats

Si les résultats de GALBE sont très satisfaisants dans la majorité des cas, certains cas particuliers sont mal traités, comme les virages en lacet très empâtés. On peut envisager diverses raisons à cette limite :

- Même si le choix de l'algorithme à utiliser à un moment du processus est dépendant de la géométrie de la ligne considérée, la séquence d'algorithmes utilisée est trop figée pour traiter certains cas.
- Les mesures utilisées sont trop peu informatives pour qualifier et donc traiter efficacement certains cas. Par exemple, toutes les lignes non empâtées sont lissées alors que certaines ne devraient pas l'être, car nous lissons par défaut toute ligne non empâtée, faute d'utiliser une mesure permettant d'évaluer le niveau de détail d'une ligne.
- Les algorithmes ne sont pas toujours utilisés au mieux. *Plâtre* n'est utilisé que pour réaliser un léger lissage final à la fin du processus, et ses capacités de caricature ne sont pas exploitées.

Limites liées à la mise au point et à l'évolution du processus

Tout d'abord, le travail de conception de GALBE a été long, même si l'on ne considère que la mise au point du processus général et non la mise au point des algorithmes de base utilisés. Nous avons profité des travaux antérieurs réalisés au COGIT d'un point de vue méthodologique et algorithmique, il nous est donc difficile d'évaluer précisément ce temps de mise au point mais notre seule contribution représente un travail de plusieurs mois.

Ensuite, les critiques de trop grande simplicité que nous faisons ci-dessus à GALBE proviennent du fait que nous ne savons pas comment créer et manipuler des descriptions

pertinentes (et complexes) des objets manipulés. Autrement dit, nous avons dû trop simplifier les connaissances cartographiques nécessaires.

Enfin, il est difficile de faire évoluer ce processus en dehors de la modification de paramètres. Plus particulièrement si de nouveaux algorithmes ou de nouvelles mesures sont conçues il est difficile de les intégrer au processus. D'une part, les connaissances utilisées sont réparties dans les tests effectués sur les mesures et dans le moteur du processus lui même. Par exemple, le choix d'utiliser l'algorithme *Faille Max* ou *Faille Min* sur un virage dépend de son appartenance ou non à une série ; ceci n'est pas déterminé en pratique par une mesure, mais par la considération du moment dans le processus où le virage est défini (au premier ou deuxième découpage). D'autre part, les connaissances situées dans les tests relient directement les mesures aux algorithmes à utiliser. Il est donc difficile d'introduire un nouvel algorithme ou une nouvelle mesure sans remettre en cause tout le processus.

Une simplification ou une automatisation de la mise au point du processus permettrait elle de dépasser ces limites ?

B.7 Vers l'utilisation de l'apprentissage automatique

Pour dépasser les limites de GALBE il est nécessaire de recueillir des connaissances plus pertinentes, plus complètes, et mieux organisées. La difficulté d'extraire des connaissances complexes auprès d'experts, du point vue de la qualité des connaissances recueillies et du temps nécessaire au recueil de ces connaissances, est un problème bien connu en Intelligence Artificielle dans le domaine des systèmes experts. Ce problème est désigné par le célèbre terme de "goulot d'étranglement de l'acquisition des connaissances". Ce goulot a été explicitement identifié comme un problème important dans le domaine de la généralisation cartographique [Weibel, Keller et Reichenbacher 95].

En Intelligence Artificielle, de nombreuses recherches ont été effectuées dans deux directions pour élargir ce goulot : l'apprentissage automatique supervisé et l'acquisition des connaissances. L'acquisition des connaissances étudie des outils et des méthodologies pour faciliter le transfert de connaissances de l'homme vers la machine. L'apprentissage supervisé étudie comment construire automatiquement des connaissances manipulables par la machine à partir d'exemples fournis par l'homme. Si au début des années 80 l'apprentissage automatique était considéré comme une forme particulière d'acquisition des connaissances, ces deux domaines se sont peu à peu séparés :

> *"D'un côté, l'acquisition des connaissances s'est principalement attachée à mettre en évidence les différents types de connaissances utiles lors de la construction d'un système à base de connaissances. […] D'un autre côté, les chercheurs en apprentissage cherchent principalement à construire des algorithmes efficaces, requérant le moins possible d'interactions avec l'expert et permettant de construire automatiquement des tels systèmes à base de connaissances"* [Thomas 96, p.3]

La cartographie est essentiellement enseignée par l'exemple. L'apprentissage automatique, et en particulier l'apprentissage automatique supervisé qui a pour but de construire des connaissances à partir d'exemples fournis par un expert du domaine, semble donc une approche bien adaptée à notre problème. La suite de ce mémoire a pour objet de répondre à la question suivante : *peut-on acquérir les connaissances nécessaires au guidage de la généralisation cartographique par apprentissage automatique à partir d'exemples ?*

Avant de répondre dans les chapitre D et E à la question posée, le chapitre C suivant présente l'apprentissage automatique supervisé.

■

C APPRENTISSAGE AUTOMATIQUE SUPERVISE

◆ Dans ce chapitre nous décrivons l'apprentissage automatique supervisé, par l'intermédiaire de la notion de biais d'apprentissage [Mitchell 82]. Nous évoquons les difficultés liées au choix d'un algorithme d'apprentissage adapté à un problème, ainsi que les difficultés liées à la validation des résultats d'un apprentissage.

◆ Nous décrivons aussi les interactions entre l'apprentissage supervisé et la notion de méthode de résolution de problème mise en avant dans le domaine de l'acquisition des connaissances. Ces interactions incitent à définir une telle méthode adaptée au problème traité pour guider l'apprentissage.

C.1 Présentation de l'apprentissage supervisé et définitions

L'apprentissage automatique désigne l'ensemble des changements dans un système qui lui permettent de réaliser une même tâche, ou des tâches similaires, de manière plus efficace ou plus efficiente au cours du temps [Simon 83, p.28]. Cette définition est très large et pourrait même désigner la phase d'introduction interactive de données dans la base de données d'un système d'information, car cela lui permet de répondre à plus de requêtes [Mitchell 97, p.5]. Néanmoins cette définition, si on en conserve l'esprit qui suppose une part d'actions relativement automatiques réalisées par le système lui-même, permet d'établir le contexte général de l'apprentissage.

Il y a deux façons d'apprendre : soit le système se modifie lui-même pour exploiter ses propres connaissances plus efficacement, soit le système acquiert de nouvelles connaissances grâce à des sources externes [Shavlik et Dietterich 90, p.1]. Le premier type d'apprentissage, lorsque le système se modifie de lui-même, est utilisé pour accélérer un système de résolution de problème afin d'aller plus vite et de résoudre plus de problèmes. Dans le cadre de nos travaux, nous nous intéressons au second type d'apprentissage, utilisé pour introduire de nouvelles connaissances dans un système. Pour construire ces nouvelles connaissances, l'apprentissage utilise des sources externes : des *exemples*. Ceux-ci sont souvent comparés entre eux pour y trouver des similarités. C'est pourquoi on qualifie ce type d'apprentissage de *basé sur les similarités*. On y distingue classiquement l'*apprentissage non supervisé* de l'*apprentissage supervisé*.

L'apprentissage non supervisé recherche des régularités parmi un ensemble d'exemples, sans être nécessairement guidé par l'utilisation qui sera faite des connaissances apprises. Par exemple, le *clustering* cherche à grouper des exemples de manière à ce que les exemples au sein d'un même groupe se ressemblent suffisamment, et que les exemples de groupes différents soient suffisamment différents. Il peut être utile comme pré-traitement à l'apprentissage supervisé ou pour simplifier le stockage ou la communication de données [Shavlik et Dietterich 90, p.263-264].

L'apprentissage supervisé, quant à lui, utilise des exemples *étiquetés* ou *classés*. Ces étiquettes ou ces classes peuvent être vues comme fournies par un professeur ou un superviseur, d'où le nom d'apprentissage supervisé. Le but de l'apprentissage est alors de produire une fonction de classification, appelée *hypothèse*, permettant de déterminer la classe d'un exemple. L'apprentissage supervisé a donc pour but de déterminer une représentation en *intension* d'un concept (l'hypothèse) à partir d'un sous ensemble de son *extension* (les exemples). Il réalise un saut *inductif* en passant des exemples *particuliers* à une fonction de classification *générale*.

Autrement dit, étant donné un ensemble d'exemples sous la forme $\{(x_i,y_i)\}$ avec x_i des objets d'un domaine D et y_i les classes associées, le but de l'apprentissage supervisé est de déterminer une hypothèse h tel que $y=h(x)$ pour tout x de D. Les x_i sont appelés les *observables*.

Notons que l'on utilise classiquement le même terme d'*exemples* pour désigner d'une part les *exemples d'apprentissage* qui servent à fabriquer une hypothèse, et d'autre part les objets sur lesquels cette hypothèse sera appliquée pour déterminer leur classe. En cas d'ambiguïté, nous appellerons ces objets des *nouveaux exemples à classer*, pour les différencier des exemples d'apprentissage.

On distingue deux grandes familles d'apprentissage supervisé en fonction du langage des hypothèses manipulé : les approches symboliques et les approches numériques. Les approches

symboliques construisent des hypothèses dans des langages directement compréhensibles par les experts du domaine traité, comme les arbres de décision ou les bases de règles. Les approches numériques utilisent des langages de représentation moins directement interprétables comme les réseaux de neurones ou les réseaux bayesiens.

L'apprentissage supervisé est utile soit pour prédire, soit pour expliquer. Il est utile pour prédire quand l'hypothèse apprise a pour but de servir à classer correctement de nouveaux exemples non encore classés. Il est utile pour expliquer quand on s'intéresse au contenu de l'hypothèse apprise pour comprendre ce qui relie les exemples à leur classe. Pour reprendre le titre d'un livre de René Thom [1993], *prédire n'est pas expliquer*. On peut donc utiliser l'apprentissage, éventuellement différemment, exclusivement pour prédire ou pour expliquer. Mais l'explication peut aussi servir de validation à la prédiction en permettant d'analyser la pertinence des hypothèses apprises, et la prédiction peut servir de validation à l'explication en assurant que les hypothèses sont fondées.

L'apprentissage supervisé a fait, et fait toujours, l'objet de nombreux travaux en Intelligence Artificielle. Ceci est en partie dû au fait qu'il s'agit d'une méthode utile pour élargir le goulot d'étranglement de l'acquisition des connaissances des systèmes experts. Si les experts peuvent plus facilement fournir des exemples illustrant leur savoir que des règles directement utilisables par un système expert, l'apprentissage supervisé permet alors de créer automatiquement ces règles à partir des exemples. Dans un contexte de recueil de connaissances pour les systèmes experts, l'apprentissage symbolique est préféré à l'apprentissage numérique car les hypothèses apprises sont compréhensibles. Certains chercheurs estiment néanmoins que cette compréhensibilité n'est pas si évidente et que, par exemple, les gros arbres de décision sont aussi difficilement interprétables que les réseaux de neurones [Mooney et al. 89].

C.2 Poser un problème d'apprentissage

Nous présentons la trame définie par Mitchell [1997, pp.2-18] pour poser un problème d'apprentissage automatique. Cette trame est définie pour tout type d'apprentissage et s'applique en particulier à l'apprentissage supervisé. Pour cela reprenons sa définition de l'apprentissage qui est une version plus formalisée de celle de Simon [1983] :

> *"Un programme informatique est dit **apprendre** à partir de l'expérience E relativement à une classe de tâches T et à une mesure de performance P, si sa performance à traiter les tâches de T, comme mesurée par P, augmente avec l'expérience E."* [Mitchell 97, p.2]

Pour poser un problème d'apprentissage, il faut donc tout d'abord définir l'ensemble des tâches à traiter. En apprentissage supervisé, on traite des tâches de classification. Par exemple, on veut savoir classer toute ville française selon qu'elle se situe dans le nord-ouest ou le sud-est.

Ensuite, il faut déterminer la source d'expérience. En apprentissage supervisé il s'agit d'un ensemble d'exemples préalablement classés. Par exemple, un ensemble de villes classées par un expert du domaine (ici la géographie) comme faisant partie du nord-ouest ou du sud-est.

La mesure de performance permet quant à elle d'évaluer dans quelle mesure le système s'améliore dans le traitement des tâches posées. En apprentissage supervisé, la mesure de performance est en général le pourcentage de succès de la classification sur l'ensemble des objets à classer. Par exemple, on peut imaginer un système classant au hasard toute ville comme faisant partie du nord-ouest ou du sud-est de la France ; ce système a une chance sur

deux de se tromper et a donc une performance de 50%. Le but de l'apprentissage est de déterminer des règles de classification améliorant la performance du système.

Pour réaliser l'apprentissage il faut déterminer la fonction à apprendre (appelée fonction cible), indépendamment du choix du mode de représentation. On peut par exemple rechercher une fonction déterminant la classe (nord-ouest ou sud-est) en fonction de la localisation de la ville. On peut aussi imaginer rechercher une fonction déterminant la classe en fonction du nom de la ville.

Cette fonction cible est en général impossible à déterminer précisément. Il faut donc choisir comment représenter une approximation de cette fonction, en la définissant dans un langage de représentation. Pour l'apprentissage supervisé, cela signifie qu'il faut définir un langage de représentation des *exemples* et de l'*hypothèse*. Par exemple, dans le cas d'une fonction cible reliant la localisation à la classe, on peut choisir de représenter les exemples sous la forme du couple (latitude, longitude) des coordonnées de son centre (on pourrait aussi choisir de représenter la localisation comme le numéro du département contenant la ville, ou comme le code postal). On peut alors choisir de représenter l'hypothèse sous la forme d'une ligne brisée séparant en deux le plan défini par la latitude et la longitude.

Ce n'est qu'une fois tous ces choix réalisés que la méthode de détermination de cette approximation - c'est-à-dire l'algorithme d'apprentissage - peut être choisie. Pour notre exemple, on peut rechercher de manière exhaustive la ligne brisée la plus courte séparant tous les exemples.

Cette trame nous permet de mettre en valeur que, du point de vue de l'utilisateur des techniques d'apprentissage, le choix de l'algorithme n'est que la dernière étape d'un problème d'apprentissage. Cela même si de nombreux travaux de recherche en apprentissage se concentrent sur la création ou la validation d'algorithmes les plus efficaces et les plus génériques possibles.

C.3 Algorithmes d'apprentissage

C.3.1 L'apprentissage supervisé, un problème de recherche

L'apprentissage supervisé est un mécanisme d'induction, c'est à dire de passage du particulier au général. Du point de vue de la logique déductive, il n'y a aucun moyen de déterminer la classe d'un exemple qui n'a jamais été observé. La solution utilisée par les algorithmes inductifs pour résoudre ce problème est d'introduire des contraintes dans l'algorithme d'apprentissage de telle manière que, étant donné un échantillon d'exemples classés $\{(x_i,y_i)\}$, l'algorithme peut faire des suppositions raisonnables sur la définition de l'hypothèse reliant les exemples et leur classe [Shavlik et Dietterich 90, p.1]. Cette vision de l'apprentissage supervisé est liée à la notion de biais d'apprentissage introduite par Mitchell [1980]. Précisons un point de terminologie source de confusion : le terme *biais* est utilisé en apprentissage car il définit un écart entre l'hypothèse réelle et l'hypothèse estimée. Néanmoins, il ne doit pas être considéré comme une erreur, comme le *biais de mesure*. Si on cherche en général à éliminer les biais de mesure, les biais d'apprentissage sont nécessaires.

Les biais constituent l'ensemble des suppositions qui, associé aux exemples d'apprentissage, permet de justifier par *déduction* les classifications assignées par un algorithme d'apprentissage supervisé à de nouveaux exemples non classés [Mitchell 97, p.43]. Ils sont de trois types :

- Les biais de *représentation*. Ils définissent le langage utilisé pour représenter les exemples et les hypothèses. Par exemple, le langage des exemples peut être un langage attributs/valeurs ou un langage plus expressif comme les langages de la logique du premier ordre. De même le langage des hypothèses peut être celui des réseaux de neurones, des arbres de décision, des règles de production, etc.
- Les biais de *restriction*. Ces biais sont parfois présentés comme un type particulier de biais de représentation. Ils imposent des contraintes sur le langage des hypothèses défini dans les biais de représentation. Par exemple, on peut se contraindre à ne rechercher que des arbres de décision avec au plus k nœuds ou des règles avec au plus k prémisses.
- Les biais de *préférence*. Ces biais définissent un ordre partiel sur l'espace des hypothèses qui permet de choisir entre deux hypothèses valides. Deux biais couramment utilisés sont de préférer les hypothèses les plus simples, et de préférer les hypothèses couvrant le plus d'exemples. Le compromis entre ces deux préférences peut, par exemple, être réalisé par le principe de la longueur de description minimale. Ce principe est de préférer les hypothèses qui minimisent la somme de la longueur de description de l'hypothèse et de la longueur de description des exemples non couverts par l'hypothèse.

La notion de biais permet de voir l'apprentissage comme un problème de recherche : l'apprentissage est la recherche, dans l'espace défini par les biais de langage et de restriction, de la meilleure hypothèse au sens du biais de préférence.

Poser des biais d'apprentissage n'a pas seulement pour but de réduire la taille de l'espace à explorer et donc de faciliter le calcul informatique de l'hypothèse cherchée. Mitchell [1980] a montré que l'induction sans biais est impossible. C'est par leur intermédiaire que les nouveaux exemples à classer sont reliés aux exemples d'apprentissage. Sans biais, il n'existe aucune raison de classer un nouvel exemple dans une classe plutôt qu'une autre. En contraignant l'hypothèse à apprendre, les biais définissent indirectement quels exemples d'apprentissage ressemblent le plus à un nouvel exemple. Cette ressemblance n'est pas universelle mais est définie dans le but précis de classer les exemples. Ce sont donc les biais d'apprentissage qui permettent aux algorithmes d'effectuer une induction.

Les biais ont en particulier pour but d'éviter les problèmes d'*overfitting*. On dit qu'une hypothèse *couvre trop* (*overfit*) les exemples quand elle est trop spécifique aux exemples d'apprentissage, au détriment de sa capacité généralisatrice. C'est à dire qu'elle classe trop bien les exemples ayant servi à déterminer l'hypothèse apprise, au détriment de sa capacité à classer un nouvel exemple inconnu au système jusqu'alors. Dans le cas de données bruitées, l'overfitting se traduit par la création d'hypothèses trop fines, adaptées pour classer les exemples d'apprentissage bruités, alors que ceux-ci devraient être négligés. L'overfitting se traduit à l'extrême par l'*apprentissage par cœur*, où l'hypothèse apprise ne sait classer que les exemples d'apprentissage.

A titre d'illustration de l'overfitting, la Figure 28 montre un problème où les exemples sont décrits par deux attributs (latitude et longitude) et sont classés positifs (si ils sont au nord-ouest) ou négatifs (si ils sont au sud-est). Trois exemples (deux négatifs et un positif) sont mal

classés à cause de problèmes de bruit. On recherche une hypothèse portant sur les deux attributs permettant de différencier les exemples positifs des exemples négatifs. Si on apprend une hypothèse trop complexe (à gauche, la ligne brisée), celle-ci classe bien les exemples d'apprentissage, mais ne correspond pas à la ligne droite plus simple (à droite) que nous aurions naturellement préféré pour séparer les exemples. Cette préférence est une illustration du principe dit du "rasoir d'Occam" énoncé par Guillaume d'Occam (philosophe anglais du XIV^{ème} siècle). Ce principe stipule qu'entre deux explications d'un même phénomène on préfère la plus simple.

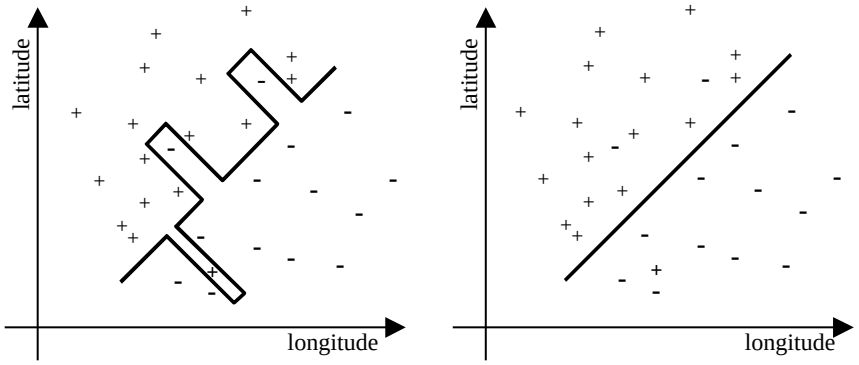


Figure 28. Overfitting et complexité de l'hypothèse

Ce problème apparaît tout particulièrement lorsque les exemples sont bruités, ce qui est le cas de la plupart des problèmes du monde réel. Les exemples sont considérés bruités quand plusieurs exemples proches dans l'espace des observables ont des classes différentes. Ce bruit peut provenir d'erreurs effectuées lors du recueil des exemples, mais un bruit apparent peut aussi provenir de problèmes liés au manque d'information. En effet, lorsque le langage des observables utilisé pour décrire les exemples est trop peu informatif vis à vis de la classe à apprendre, les exemples peuvent avoir des classes différentes alors que leurs observables sont proches. La Figure 29 illustre cela : les attributs "latitude" et "longitude" permettent de distinguer aisément le concept "Nord-Ouest" (+) du concept "Sud-Est" (-). Si on n'utilise que la longitude, certains exemples paraissent bruités et cela est dû au manque d'information. Ce manque d'information ne doit pas être rédhibitoire pour l'apprentissage car l'attribut longitude seul peut, dans notre exemple, permettre d'apprendre une première approximation de la distinction entre sud-est et nord-ouest, en séparant l'est de l'ouest.

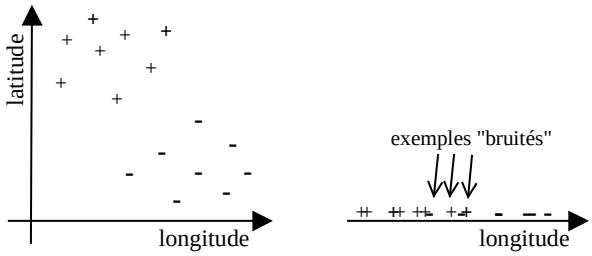


Figure 29. Manque d'information et données apparemment bruitées

C.3.2 Mise en œuvre des biais d'apprentissage

Si on pose des biais d'apprentissage trop forts, c'est à dire définissant un langage des hypothèses peu expressif, il est impossible de trouver une hypothèse qui soit une bonne approximation de la fonction cible à apprendre. Par contre, si on pose des biais trop faibles, c'est à dire autorisant de nombreuses hypothèses complexes, l'espace des hypothèses à

parcourir est gigantesque et les risques d'overfitting sont nombreux. Le choix et l'implémentation des biais est donc au cœur de la conception des algorithmes d'apprentissage.

Les biais de représentation et de restriction s'implémentent relativement directement, ce sont eux qui définissent les langages manipulés. Les biais de préférence sont plus difficiles à implémenter.

Une première façon d'implémenter les biais de préférence est de définir une mesure de comparaison entre deux hypothèses. Pour la même définition théorique d'un biais de préférence, comme le principe de la longueur de description minimale déjà évoqué, plusieurs réalisations informatiques sont possibles. Le choix de mesures est un premier problème auquel est confronté le concepteur d'un algorithme d'apprentissage. Ces mesures sont souvent dérivées de mesures statistiques ou du domaine de la théorie de l'information.

Par ailleurs, si le problème de l'apprentissage supervisé est en théorie parfaitement posé grâce à la notion de biais d'apprentissage, l'espace défini par les biais de représentation et de restriction est en pratique gigantesque. Il est souvent impossible de le parcourir entièrement pour rechercher la meilleure hypothèse au sens des biais de préférence. Les algorithmes d'apprentissage sont donc guidés par des heuristiques.

Ces heuristiques sont en fait un moyen de définir et d'implémenter des biais de préférence [Shavlik et Dietterich 90, p.53]. Par exemple, une heuristique classique des algorithmes construisant des arbres de décision est – dans un premier temps – de partir d'un arbre vide et de le construire en y ajoutant progressivement des branches jusqu'à ce que certaines conditions sur les exemples soient atteintes. Cette heuristique privilégie les petits arbres puisque l'algorithme ne parcours en général qu'une – très petite – partie de l'espace des hypothèses où les arbres sont petits. Ces algorithmes ont donc pour biais de préférer les petits arbres aux grands.

De même, pour éviter le problème de l'overfitting, beaucoup d'algorithmes utilisent une heuristique "croître/élaguer". C'est-à-dire que les exemples sont séparés en deux jeux : un jeu d'*exemples de croissance* et un jeu d'*exemples d'élagage*. Une première passe de l'algorithme utilise les exemples de croissance et crée une hypothèse qui les couvre trop. Une deuxième passe est chargée de simplifier l'hypothèse apprise en se basant sur le jeu d'élagage. Par exemple, dans le cas de la Figure 28, on peut imaginer un algorithme qui, en premier lieu, construit l'hypothèse complexe de la ligne brisée (à gauche) puis, en second lieu, simplifie cette ligne pour arriver à l'hypothèse de la ligne droite (à droite). Cette heuristique est une autre manière d'implémenter un biais de préférence des hypothèses simples sur les complexes, puisqu'à performance similaire on préfère les hypothèses élaguées.

Un algorithme d'apprentissage est donc défini théoriquement par les biais qu'il utilise, et est défini en pratique par les choix réalisés pour les implémenter. Une description détaillée du fonctionnement des algorithmes d'apprentissage dépasse le cadre de cette thèse, dans laquelle nous nous positionnons comme utilisateur des algorithmes d'apprentissage. Pour plus d'information à ce sujet, de nombreuses méthodes algorithmiques sont détaillées par exemple dans le livre *Machine Learning* [Mitchell 97], et l'algorithme RIPPER [Cohen 95] décrit en annexe III illustre la réalisation d'un algorithme d'apprentissage. La partie suivante décrit néanmoins simplement les types d'algorithmes existants.

C.3.3 Types d'algorithmes existants

De nombreux algorithmes ont été développés, chacun possédant ses propres biais. Les premiers et plus nombreux utilisent un biais de représentation des exemples en attribut/valeur

(i.e. un exemple est décrit les valeurs d'un ensemble constant d'attributs) et construisent des hypothèses dans différents langages : par exemple RIPPER [Cohen 95] et C4.5rules [Quinlan 93] construisent des règles de décision, ASSISTANT [Cestnik, Kononenko et Bratko 87] et C4.5 [Quinlan 93] construisent des arbres de décision, CHARADE [Ganascia 87] construit des bases de règles de production, RoC [Ramoni et Sebastiani 99] construit des réseaux bayesiens, et la rétro-propagation [Rumelhart, Hinton et Williams 86] construit des réseaux de neurones. D'autres algorithmes utilisent d'autres biais de représentation des exemples : INDUCE [Michalski 80], ML-SMART [Bergadano et Giordana 88], FOIL [Quinlan 90], PROGOL [Muggleton 95] et REMO [Zucker 96] manipulent des exemples décrits dans des langages de la logique du premier ordre, ENIGME+ [Zucker et Ganascia 96] et RIPPER-MI [Chevaleyre et Zucker 2000] manipulent des langages multi-instances, etc.

En pratique il existe de nombreux algorithmes réalisant de nombreux biais différents. Comment choisir alors un algorithme d'apprentissage plutôt qu'un autre ?

C.3.4 Choisir un algorithme d'apprentissage

Il est communément admis qu'en pratique aucun algorithme d'apprentissage n'est meilleur qu'un autre pour tous les problèmes d'apprentissage [Weiss et Kapouleas 89 ; Aha 92 ; Shavlik, Mooney et Towell 91]. Ceci a été démontré pour les problèmes d'apprentissage de concept utilisant des exemples en langage attribut/valeur [Schaffer 94], et plus généralement par le théorème du "No Free Lunch" [Wolpert et Macready 95] : "pour tout algorithme d'apprentissage il existe un problème pour lequel cet algorithme est mieux adapté que tous les autres". Ces démonstrations s'appuient sur des tâches d'apprentissage théoriques qui n'existent peut-être pas en pratique, mais la pratique ne fait néanmoins pas ressortir d'algorithme universellement meilleur que tous les autres.

Si des connaissances générales sur le champ d'application des algorithmes existent (capacité ou non à traiter les attributs numériques, les attributs manquants, les attributs bruités…), il n'existe pas de règle pour définir *a priori* le meilleur algorithme adapté à un problème donné. Des tests, réalisés par exemple pour comparer les réseaux de neurones et les algorithmes symboliques le confirment [Mooney et al. 89 ; Weiss et Kapouleas 89] : il n'en ressort pas d'algorithme globalement meilleur qu'un autre, ni de règles permettant de choisir un algorithme en fonction d'un problème donné.

Méta-apprentissage

Rechercher *a priori* un algorithme adapté à un problème est une tâche de classification. Il faut déterminer l'algorithme (la classe) adapté à des données (les observables). La classification étant le domaine de prédilection de l'apprentissage supervisé, des travaux, qualifiés de méta-apprentissage, tentent d'apprendre automatiquement à réaliser cette tâche.

Pour déterminer par méta-apprentissage le meilleur algorithme adapté à un jeu d'exemples, on qualifie ce jeu grâce à des méta-attributs comme le nombre d'exemples, le nombre d'attributs symboliques, le pourcentage de données manquantes, des histogrammes de répartition des classes, etc. On applique ensuite plusieurs algorithmes d'apprentissage sur un ensemble de jeux d'exemples (chacun étant un méta-exemple) pour déterminer l'algorithme le plus efficace sur chaque jeu (sa méta-classe). On effectue ensuite un méta-apprentissage sur ces données avec des techniques d'apprentissage classique pour apprendre quel algorithme est le plus efficace sur quel type de données. De nombreuses variantes existent pour effectuer ce méta-

apprentissage. Les premières variantes se situent au niveau du choix de l'algorithme de méta-apprentissage : n'importe quel algorithme d'apprentissage classique peut être utilisé. D'autres variantes se situent au niveau de la classe à apprendre : on peut rechercher le meilleur algorithme, ou rechercher si un algorithme donné est applicable ou non [Brazdil et Henery 94], ou rechercher à classer les algorithmes selon leur efficacité [Gama et Brazdil 95 ; Kalousis et Hilario 2000 ; Soares et Brazdil 2000].

Ces travaux en méta-apprentissage permettent de diminuer légèrement les taux d'erreurs obtenus sur un ensemble de problèmes classiques par rapport à l'utilisation d'un seul algorithme d'apprentissage. Mais à notre connaissance, ils n'ont pas permis d'énoncer des règles simples permettant de choisir un algorithme plutôt qu'un autre pour un problème donné.

Par ailleurs, il nous semble qu'une limite du méta-aprentissage est que l'on recherche le meilleur algorithme adapté à des données, et non le meilleur algorithme adapté à un problème. A notre connaissance, aucune information externe aux données (des méta-données) sur le problème posé n'est utilisée, comme par exemple le domaine d'application (la chimie, la physique…) ou la tâche effectuée (abstraire, raffiner, comparer, calculer…). Le méta-apprentissage suppose donc que l'efficacité d'un algorithme dépend uniquement du jeu d'exemple, et non du problème. Les méta-données décrivant un problème sont certes moins accessibles que les données elles-mêmes, mais le méta-apprentissage se confrontera peut-être, comme en apprentissage, au problème du recueil des (méta) exemples.

De plus, puisque l'on peut "méta-apprendre" de nombreuses façons différentes, on peut se demander s'il ne faudrait pas alors "méta-méta-apprendre" quel méta-apprentissage est le mieux adapté à des données… Le méta-apprentissage étant une approche relativement nouvelle, il est difficile d'en évaluer pleinement les apports possibles à ce jour.

Ces différents travaux montrent qu'il n'existe pour l'instant pas de réponse claire au problème du choix d'un algorithme d'apprentissage pour répondre à un problème donné. En terme de qualité des hypothèses apprises, l'influence du choix d'un algorithme d'apprentissage est faible, relativement à l'influence de la façon de poser le problème (choix de la tâche à apprendre, du langage des observables, des exemples, etc.).

C.3.5 Combiner plusieurs algorithmes

S'il n'est pas possible de choisir *a priori* l'algorithme d'apprentissage adéquat, de nombreux travaux étudient l'opportunité de combiner les résultats de plusieurs algorithmes, ou du même algorithme utilisé de différentes manières. D'après Dietterich [1997], ces études sur les ensembles d'hypothèses sont un des quatre axes de recherche les plus étudiés actuellement en apprentissage (les trois autres sont les méthodes pour augmenter le champ d'action des algorithmes, l'apprentissage par renforcement, et l'apprentissage de modèles stochastiques).

Plusieurs approches existent pour construire différentes hypothèses. On peut utiliser plusieurs algorithmes d'apprentissage sur les mêmes exemples ou utiliser un même algorithme sur différents exemples. Par exemple la technique du *bagging* [Breiman 96] effectue plusieurs apprentissages avec un même algorithme mais avec différents sous-ensembles des exemples. De même, le *boosting* [Freund et Schapire 95 ; Quinlan 96] effectue successivement plusieurs apprentissages en utilisant à chaque étape les résultats des apprentissages précédents ; pour cela on pondère l'importance des exemples à une étape d'apprentissage de manière à donner plus d'importance aux exemples mal classés à l'étape précédente.

Pour ensuite combiner différentes hypothèses, on peut effectuer un simple vote sur la classe prévue par chacune des hypothèses pour déterminer la classe la plus probable [Clemen 89]. On peut également effectuer un vote pondéré en fonction de la confiance que chaque hypothèse met en sa réponse [Ali et Pazzani 96] (cette confiance peut être estimée de diverses manières, cf. C.5 "Evaluation de l'apprentissage" ci-après). Dans cette approche on peut également effectuer du méta-apprentissage : on peut apprendre un meta-classifieur qui choisit entre les prédictions de plusieurs hypothèses pour décider de la classe d'un exemple [Wolpert 92 ; Chan et Stolfo 97], ou encore apprendre le poids à donner au vote de chaque hypothèse [Jordan et Jacobs 94].

Ces approches améliorent parfois les résultats en terme de prédiction par rapport à l'apprentissage classique. Par contre, elles n'en améliorent pas la lisibilité : les hypothèses ainsi obtenues combinent des hypothèses contradictoires et éventuellement exprimées dans plusieurs langages différents. Nous considérons donc que ces hypothèses sont difficilement interprétables.

C.4 Vers des connaissances plus efficaces et mieux structurées

Il existe de nombreux algorithmes d'apprentissage et aucun d'entre eux n'est universel. Pour que l'apprentissage soit efficace sur un problème donné, une première solution, envisagée dans la partie C.3.4, est de rechercher un algorithme adapté *a priori* à un problème. Une deuxième solution est d'utiliser des connaissances du domaine traité pour contraindre l'apprentissage :

> *"Des travaux aussi bien empiriques que théoriques en apprentissage montrent qu'une des seules possibilités pour qu'un apprentissage fonctionne de manière correcte est de lui fournir de nombreuses connaissances sur l'application que l'on doit traiter et le domaine de cette application"* [Thomas 96, p.4]

L'utilisation des connaissances du domaine pour guider l'apprentissage possède un double avantage : elle permet d'apprendre des hypothèses à la fois plus efficaces et plus compréhensibles, comme nous l'expliquons dans cette partie. Pour cela, nous décrivons tout d'abord les systèmes experts dits de première génération, pour mettre en avant les limites dues au manque d'organisation des connaissances qu'ils utilisent. En effet, ces limites apparaissent directement dans les résultats de l'apprentissage supervisé. Nous décrivons ensuite la notion de méthode de résolution de problème (MRP). Cette notion est au centre des méthodologies, regroupées sous le nom d'*acquisition de connaissances basée sur les modèles*, conçues pour mettre au point des systèmes experts dits de deuxième génération [David, Krivine et Simmons 93]. Nous décrivons enfin l'approche mise en œuvre dans le système ENIGME [Ganascia, Thomas et Laublet 93 ; Thomas 96] qui intègre les méthodes d'apprentissage supervisé et d'acquisition des connaissances basée sur les modèles.

Limites des systèmes experts "de première génération"

Les systèmes experts dits de première génération sont uniquement constitués d'une base de règles, d'une base de faits et d'un moteur d'inférence (Figure 30). La description initiale d'un problème est représentée dans la base de faits. Le moteur d'inférence applique les règles sur la base de faits pour l'enrichir de nouveaux faits, jusqu'à obtenir une réponse au problème posé. Le moteur d'inférence est supposé indépendant du domaine traité. Toutes les connaissances du

domaine, qui sont représentées dans la base de règles, sont supposées être de même nature : des règles de production. De plus, ces systèmes considèrent que chaque règle est un élément valide et indépendant des connaissances utilisées. De ce fait, l'expert qui introduit les règles n'a à se préoccuper ni de l'ordre, ni de la manière dont les règles vont être utilisées. Ces systèmes experts ont connu un grand succès et ont été développés en grand nombre. Leur efficacité est donc reconnue, et leurs limites aussi.

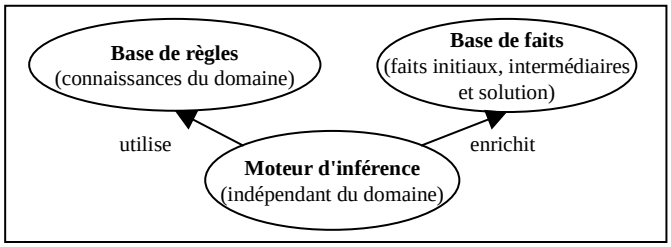


Figure 30. Système expert de première génération

La première limite de ces systèmes réside dans la difficulté qu'il y a à acquérir les connaissances nécessaires sous forme de règles auprès d'experts. Ce problème du "goulot d'étranglement de l'acquisition des connaissances" est, entre autres, à l'origine des travaux en apprentissage symbolique automatique que nous avons évoqué. Il est aussi à l'origine des travaux en acquisition de connaissances qui étudient les outils et méthodologies utiles pour transférer des connaissances de l'homme vers la machine.

Une deuxième limite réside dans la structure même de ces systèmes, et a été mise en valeur dans l'analyse faite par Clancey [1983 ; 1985] du système expert MYCIN [Shortliffe 76], destiné au diagnostic médical. Tout d'abord, l'ordre des prémisses des règles de MYCIN n'est pas anodin : il guide l'ordre dans lequel les règles vont être utilisées car, sans que cela soit explicitement représenté dans la base de règles, le moteur d'inférence va essayer de vérifier les faits dans cet ordre. De plus, on peut clairement distinguer plusieurs types de prémisses dans les règles de MYCIN, comme les prémisses de contexte qui décrivent le contexte d'application de la règle, ou comme les prémisses clés qui sont le cœur de la connaissance représentée dans la règle. Enfin, MYCIN contient une organisation hiérarchique des maladies non explicitement représentée, mais que l'on peut reconstruire à partir de différentes règles. On peut donc identifier dans la base de règles de MYCIN des connaissances sur le domaine implicitement représentées, dont une chaîne d'inférence guidant le raisonnement. Ceci va a l'encontre d'un principe établi en Intelligence Artificielle : toute connaissance utilisée par le système doit être exprimée explicitement. Clancey montre que ces constatations s'appliquent à tous les systèmes experts qu'il a étudié par ailleurs.

Ces limites restreignent considérablement les avantages prêtés à l'origine aux systèmes experts, à savoir l'indépendance et la compréhensibilité des règles utilisées qui permettent de faciliter la conception, la compréhension, la validation, la maintenance et l'évolution du système.

> *"Ces problèmes prennent racine dans les limitations fondamentales de la représentation des règles de MYCIN : la vision que la connaissance experte peut être représentée dans un ensemble uniforme et faiblement structuré d'associations Si/Alors laisse à désirer."* [Clancey 83]

Les critiques de Clancey s'appliquent directement sur les résultats des apprentissages symboliques classiques : ceux-ci produisent un ensemble de règles plus ou moins dépendantes, mais le raisonnement présent dans ces règles n'est pas explicitement représenté.

L'apprentissage automatique produit des règles conçues pour être utilisées par un système expert de première génération [Thomas 96, p.89]

Méthode de résolution de problème

Ces limites des premiers systèmes experts ont conduit à la fin des années 80 à la notion de système expert dit de deuxième génération [David, Krivine et Simmons 93]. Ces systèmes représentent explicitement le raisonnement que doit suivre le système pour parvenir à une solution. Le raisonnement y est considéré lui-même comme une connaissance [Clancey 85, Le Roux 94].

Dans ce contexte, la méthodologie d'acquisition des connaissances nécessaires aux systèmes experts de deuxième génération devient alors un processus très contraint : il faut identifier le raisonnement à suivre avant d'identifier les connaissances nécessaires à chaque étape du raisonnement [Thomas 96]. Ce modèle de raisonnement est appelé la *méthode de résolution de problème*. Dans la méthodologie KADS [Schreiber, Wielinga et Breuket 93 ; Schreiber et al. 2000], qui est une méthodologie d'acquisition des connaissances basée sur les modèles, la méthode de résolution de problème définit :

- une décomposition en tâches et sous-tâches des étapes du raisonnement,
- les rôles donnés aux connaissances du domaine dans ces tâches, c'est-à-dire une spécification des entrées et sorties de ces tâches,
- des connaissances de contrôle permettant d'enchaîner les tâches.

Apprentissage supervisé et acquisition de connaissances basée sur les modèles

Nous décrivons brièvement ci-dessous quelques points du système ENIGME [Ganascia, Thomas et Laublet 93 ; Thomas 96] qui intègre l'apprentissage supervisé et les méthodologies d'acquisition de connaissances basées sur les modèles (qui construisent explicitement une méthode de résolution de problème), dans le but d'en combiner les avantages. Ce système utilise des connaissances du domaine traité, structurées sur le modèle de KADS, et apprend de nouvelles connaissances à partir d'exemples avec l'algorithme CHARADE [Ganascia 87].

Les systèmes d'apprentissage classiques n'ont pour entrée que des exemples et une description de leur langage. ENIGME, quant à lui, utilise une *méthode de résolution de problème*, une couche *domaine*, et des *exemples*. La méthode de résolution de problème décrit les tâches à réaliser et leur enchaînement pour résoudre un problème. Le domaine décrit la nature des rôles impliqués (i.e. les éléments en entrée et sortie des tâches) et divers modèles du domaine comme le fait que tel rôle est nécessaire pour expliquer tel autre rôle dans telle tâche. Un apprentissage est alors effectué pour chaque tâche du raisonnement, c'est-à-dire chaque inférence de la méthode de résolution de problème. Cet apprentissage n'utilise que les rôles nécessaires à chaque étape et les exemples contenant ces rôles, et utilise les modèles du domaine pour contraindre l'algorithme d'apprentissage dans sa recherche d'hypothèses. Par exemple, en fonction des modèle du domaine, ENIGME pourra forcer tel rôle à être utilisé dans une règle pour expliquer un autre rôle.

Pour illustrer cela, reprenons un exemple développé dans [Thomas 96] : la classification des animaux. On désire apprendre, à partir d'exemples, à reconnaître l'espèce d'un animal (e.g. truite, saumon, vache, cheval) à partir d'attributs le décrivant (e.g. couleur, taille, peau, etc.). On peut alors introduire dans ENIGME plusieurs connaissances du domaine. On définira tout d'abord une *méthode de résolution de problème* décrivant le raisonnement à suivre : pour reconnaître l'espèce il faut tout d'abord reconnaître le genre (poisson, mammifère) puis

spécifier ce genre pour reconnaître l'espèce. On introduira ensuite les connaissances suivantes dans le *domaine* : le genre est un élément nécessaire pour déduire l'espèce, la truite et le saumon sont des poissons, le cheval et la vache sont des mammifères. ENIGME utilisera ces connaissances pour guider l'apprentissage : il créera deux bases de règles (une pour déterminer le genre et une pour spécifier l'espèce), il forcera les règles pour spécifier l'espèce à contenir une condition sur le genre, et il utilisera l'organisation hiérarchique des animaux (la vache est un mammifère…) lors de l'apprentissage [Ganascia 87].

On peut voir cette approche, selon le point de vue, soit comme l'introduction de connaissances pour contraindre l'apprentissage, soit comme la mise au point d'un système à base de connaissances dans lequel certaines connaissances sont apprises.

L'approche d'ENIGME présente plusieurs intérêts au niveau de la nature et de la qualité des règles apprises. La méthode de résolution de problème permet de guider l'apprentissage et d'associer chaque règle apprise à une tâche particulière de la méthode. Les règles apprises sont donc beaucoup plus structurées que celles apprises par apprentissage supervisé classique, et le raisonnement suivi par le système expert est ainsi plus compréhensible. De plus, les règles apprises par ENIGME sont supposées être plus efficaces que des règles apprises sans méthode de résolution de problème. En effet, les biais d'apprentissage utilisés pour apprendre chaque inférence sont restreints par rapport à un biais qui s'appliquerait sur l'ensemble des rôles en entrée. Pour résumer simplement, il est plus facile d'apprendre plusieurs fois des connaissances relativement simples que d'apprendre en une seule fois de nombreuses connaissances variées.

Les études mises en œuvre lors de la réalisation d'ENIGME ont montré l'intérêt théorique d'utiliser les modèles du domaine pour améliorer la qualité et la compréhensibilité des règles apprises. A notre connaissance, elles n'ont pas montré empiriquement l'intérêt, du point de vue de la qualité prédictive des règles, de l'utilisation de la méthode de résolution de problème pour améliorer l'apprentissage. Cependant, la compréhensibilité des hypothèses apprises nous semble un objectif primordial qui justifie à lui seul d'utiliser une telle approche.

C.5 Evaluation de l'apprentissage

Puisque l'on ne sait pas assurer a priori la qualité du résultat d'un apprentissage, il est crucial de savoir évaluer les hypothèses apprises. L'exemple de la Figure 28 (page 78) illustre simplement le difficile problème de l'évaluation. L'hypothèse trop complexe classifie correctement les exemples (à gauche sur la figure), ce qui n'est pas le cas de l'hypothèse plus simple et pourtant intuitivement meilleure (à droite). Non seulement l'erreur de classification sur les exemples ayant servi à l'apprentissage ne rend pas compte de la qualité des règles apprises, mais elle ne permet pas plus de définir laquelle des deux hypothèses est la meilleure. Ceci est particulièrement vrai dans les cas où les exemples d'apprentissage sont bruités, et donc où les risques d'overfitting sont multipliés.

L'apprentissage étant un processus inductif, on ne peut pas prouver, au sens de la logique déductive, la validité du résultat d'un apprentissage. On peut néanmoins évaluer de manière approchée la qualité d'une hypothèse. Deux grandes approches sont possibles pour évaluer cette qualité : estimer théoriquement *a priori* un majorant de l'erreur des hypothèses en fonction du nombre d'exemples et des biais utilisés, ou estimer empiriquement *a posteriori* la qualité d'une hypothèse apprise.

C.5.1 Evaluation théorique

Déterminer *a priori* la qualité d'un apprentissage est le domaine de la *théorie de l'apprentissage*. Les travaux dans ce domaine visent à déterminer le nombre d'exemples nécessaires pour qu'un apprentissage soit correct. Plus précisément, la théorie PAC (Probably Approximatly Correct) initiée par Valiant [1984] recherche, pour un biais donné, combien d'exemples sont suffisants pour assurer, avec une probabilité donnée, que le résultat d'un apprentissage ait au plus une erreur donnée. Ce nombre d'exemples est appelé la *complexité d'échantillonnage*. Nous décrivons ici de manière simplifiée les idées très générales guidant ces travaux.

Dans le cadre de l'apprentissage avec un biais de représentation définissant un espace des hypothèses de taille finie, l'idée de départ est la suivante : si l'espace des hypothèses exploré par un algorithme d'apprentissage est petit, il y a peu de chances qu'un algorithme trouve *par hasard* dans cet espace une hypothèse cohérente avec le jeu de d'exemples[13]. Donc, si l'algorithme en trouve une, il y a de grandes chances que le biais d'apprentissage soit adapté au concept à apprendre, et que l'hypothèse apprise soit une bonne approximation de l'hypothèse correcte. De plus, plus les exemples sont nombreux, moins il y a de chance de trouver par hasard une hypothèse cohérente avec ces exemples. Donc, plus l'espace est grand, plus il faut utiliser de nombreux exemples pour s'assurer que l'hypothèse cohérente trouvée ne l'ait pas été du fait du hasard mais du fait que le biais d'apprentissage est adapté au concept à apprendre [Shavlik et Dietterich 90, p.51]. Cette idée peut être étendue au cas des biais de représentation de taille infinie grâce à la notion de dimension de Vapenik-Chervonenkis [Haussler 88].

De nombreuses théories ont été développées à partir de ces notions. Elles permettent de définir des bornes sur le nombre suffisant d'exemples pour assurer une certaine qualité au résultat d'un problème d'apprentissage donné. Cependant ces bornes sont très pessimistes en pratique, elles sont trop exigeantes. Pour assurer statistiquement une certaine qualité du résultat de l'apprentissage, l'évaluation théorique présuppose en particulier de nombreuses indépendances entre les attributs d'un exemple, et entre chaque attribut et la classe d'un exemple. Or, en pratique, cette indépendance est faible. Les bornes données dans ces théories sont donc basées sur l'analyse du problème dans le pire des cas et tendent à ne pas être utilisables en pratique :

> *"Si l'analyse théorique de la complexité d'un échantillon a apporté d'importants éclaircissements sur les relations entre le nombre d'exemples d'apprentissage, le biais de l'algorithme d'apprentissage, et la confiance que l'on peut placer dans l'hypothèse produite par l'algorithme, cela n'a pas fourni de bornes utilisables en pratique. Ainsi, la plupart des travaux appliqués en apprentissage utilisent une technique expérimentale pour déterminer la justesse de l'hypothèse."* [Shavlik et Dietterich 90, p.52]

La théorie de l'apprentissage permet de montrer que plus l'espace des hypothèses est grand, plus l'apprentissage est difficile, et plus il faut utiliser de nombreux exemples. Elle permet également de guider certaines approches de l'apprentissage (l'idée du boosting en est issue par exemple). Mais elle ne permet pas de spécifier en pratique un nombre d'exemples minimal pour un problème ni d'évaluer le résultat d'un l'apprentissage. De notre point de vue utilisateur des outils d'apprentissage, la théorie de l'apprentissage ne nous semble pas exploitable dans nos travaux.

[13] On dit que h est cohérente avec E si elle permet de bien classer les exemples de E

C.5.2 Evaluation empirique

Erreur réelle, apparente et estimée

L'évaluation théorique recherche des conditions sur le problème d'apprentissage pour assurer d'obtenir une hypothèse probablement correcte. L'évaluation empirique cherche, quant à elle, à estimer le taux d'erreur d'une hypothèse apprise h. Ce taux d'erreur, appelé *erreur réelle*, est le pourcentage d'erreur que l'hypothèse h effectue sur l'ensemble E des exemples possibles. Puisque l'on ne connaît pas la classe de tous les exemples de E (d'où le besoin d'apprentissage inductif), cette erreur est en théorie inconnue et on cherche à l'estimer.

Le taux d'erreur sur les exemples ayant servi à la fabrication de h, appelé *erreur apparente*[14], est un estimateur peu fiable[15] et en général très optimiste de l'erreur réelle. Ceci est particulièrement le cas quand l'espace des hypothèses considéré est vaste et que les hypothèses peuvent facilement trop couvrir les exemples d'apprentissage (cf. l'overfitting, p.78). La validation empirique a pour but d'approcher l'erreur réelle par une *erreur estimée* grâce à diverses méthodes décrites ci-après.

Apprentissage et test

La manière la plus simple pour estimer l'erreur réelle d'une hypothèse apprise est de l'appliquer sur un jeu d'*exemples tests* totalement indépendants des *exemples d'apprentissage* ayant servi à fabriquer l'hypothèse. L'indépendance des exemples test et des exemples d'apprentissage est un prérequis nécessaire à la validité de l'approche. Pour cela on partage au hasard avant l'apprentissage le jeu d'exemples en deux jeux : un jeu d'apprentissage à partir duquel l'hypothèse h est apprise, et un jeu test pour évaluer h. On évalue h en comparant sur les exemples test la classification donnée et la classification prédite par h. On sépare typiquement le jeu d'exemples en 2/3 des exemples pour l'apprentissage et 1/3 pour le test [Shavlik et Dietterich 90, p.52].

Quand les exemples disponibles sont nombreux, cette technique d'estimation de l'erreur est simple et fiable. Mais des difficultés apparaissent quand les exemples disponibles sont peu nombreux. Un premier problème réside dans l'apprentissage : si peu d'exemples sont disponibles il peut être dommageable pour la qualité de l'hypothèse apprise d'ignorer 1/3 des exemples. A l'extrême, si une classe est particulièrement peu représentée elle risque d'être complètement absente des données d'apprentissage, et il devient impossible d'apprendre à classer des exemples dans cette classe. Un deuxième problème réside dans l'évaluation de la qualité de l'hypothèse apprise : si peu d'exemples sont utilisés pour estimer l'erreur, celle-ci peut être éloignée de l'erreur réelle. On se retrouve ici dans un problème d'échantillonnage ou la variance de l'erreur estimée est forte lorsque les échantillons (exemples tests) couvrent trop peu l'espace des possibles.

Validation croisée

La validation croisée est une technique souvent utilisée pour remédier aux problèmes dus au manque de données rencontrés avec avec la technique "apprentissage et test". C'est une méthode d'estimation de l'erreur réelle d'une hypothèse h à partir du jeu d'exemples

[14] L'erreur apparente est aussi parfois appelée *erreur d'apprentissage* mais ce terme peut porter à confusion.

[15] Il serait plus précis d'utiliser le terme d'estimateur biaisé. Nous utilisons dans cette partie le terme de fiabilité pour éviter les confusions avec le *biais d'apprentissage* utilisé dans les pages précédentes.

d'apprentissage E qui a permis de créer h. L'hypothèse est apprise sur l'ensemble des exemples et l'évaluation de sa qualité fonctionne comme suit :

- On sépare au hasard les exemples E en k parties $\{P_i\}_{(1\leq i\leq k)}$ (souvent k=10 en pratique).
- Pour chaque partie P_i de E :
  - On apprend, avec le même biais d'apprentissage que celui qui a permis de créer h (i.e. même algorithme avec les mêmes paramètres), une hypothèse h_i sur (k-1) parties de E $\{P_j\}_{(1\leq j\leq k,\ j\neq i)}$.
  - On évalue le taux d'erreur de classification de h_i sur la partie P_i.
- Le taux d'erreur estimé est la moyenne des taux d'erreur des k tests.

Les k évaluations effectuées risquent moins d'être biaisées à cause de l'overfitting puisque, pour chaque partie P_i , les exemples tests sont bien indépendants des hypothèses testées. Mais la validation croisée est une estimation indirecte de la qualité de l'hypothèse évaluée puisque celle-ci n'est jamais prise en compte (elle pourrait ne jamais être explicitement calculée). Cette technique n'évalue en fait pas la qualité de l'hypothèse apprise, mais la capacité de l'algorithme d'apprentissage à traiter des tâches très proches du problème posé (i.e. avec le même langage et un sous-ensemble important des mêmes exemples). On suppose que cette capacité à traiter ces tâches est équivalente à la qualité de l'hypothèse apprise sur l'ensemble des exemples. Cette technique peut ne pas être fiable si l'ajout d'exemples (quand on utilise tous les exemples au lieu de, typiquement, 90% des exemples) introduit une erreur systématique dans l'algorithme. Cependant, si on considère que les algorithmes sont conçus pour fabriquer des hypothèses dont la qualité s'améliore avec le nombre d'exemples utilisés, l'erreur estimée par validation croisée ne peut être que pessimiste, contrairement à l'erreur apparente qui est optimiste.

La validation croisée est donc une évaluation peu biaisée utile dans le cas où les exemples sont peu nombreux. Mais elle est une évaluation indirecte de l'hypothèse apprise ce qui diminue la confiance que l'on peut lui apporter par rapport à la validation "apprentissage et test" effectuée sur un nombre important d'exemples.

Fiabilité de l'évaluation empirique

Que ce soit en validation croisée ou en évaluation "apprentissage et test", le principe pour assurer une certaine fiabilité de la mesure de l'erreur estimée est de séparer les exemples d'apprentissage et les exemples tests. Il existe néanmoins de nombreux risques d'utiliser plus ou moins directement les exemples tests pendant l'apprentissage, et donc de remettre en cause cette fiabilité. En effet il est souvent tentant d'utiliser des informations du jeu test pour aider à prendre certaines décisions durant l'apprentissage : on veut parfois utiliser à l'intérieur de l'algorithme d'apprentissage le jeu test pour choisir entre différentes hypothèses possibles [Shavlik et Dietterich 90, p.52]. Par exemple, lorsque l'on cherche à déterminer le nombre de neurones cachés à utiliser dans un réseau de neurones, on est tenté d'essayer plusieurs configurations et de retenir celle qui minimise l'erreur sur le jeu test. De même lors de la construction d'arbres de décision, on est tenté d'utiliser le jeu test pour guider la phase d'élagage de l'arbre. Dans cette situation, la procédure adéquate est de séparer les exemples en trois jeux : un jeu d'apprentissage, un jeu de réglage, et un troisième jeu test, indépendant des deux premiers.

Ces risques liés à la fiabilité de l'évaluation sont clairement identifiés par les concepteurs d'algorithmes d'apprentissage. Ils le sont moins par les utilisateurs de ces algorithmes qui

effectuent alors des validations peu fiables. En effet, les algorithmes d'apprentissage sont souvent paramétrables, par exemple de manière à fournir des hypothèses plus ou moins détaillées selon les besoins de l'utilisateur. Il est courant qu'un utilisateur utilise alors un algorithme avec différents paramètres pour retenir l'hypothèse qui minimise l'erreur estimée, par exemple par validation croisée. Il n'est alors pas fiable de considérer ensuite que cette erreur minimale est l'estimateur de la qualité de l'apprentissage, puisque les exemples tests ont été utilisés pendant l'apprentissage. Il en est de même si un utilisateur expérimente plusieurs algorithmes différents et utilise le même jeu test pour choisir l'algorithme et valider ses résultats. Un même risque existe enfin si l'utilisateur expérimente différentes représentations des exemples (en éliminant ou ajoutant des attributs par exemple) pour en retenir la plus efficace.

Dans l'absolu, il serait nécessaire de séparer les exemples disponibles en de nombreux jeux indépendants : par exemple un jeu pour choisir un algorithme, un jeu pour régler la représentation des exemples, un jeu pour apprendre et enfin un autre jeu pour évaluer la qualité de l'hypothèse apprise. Chacun de ces jeux devrait être assez conséquent pour couvrir au mieux l'espace des exemples possibles. Ceci est rarement faisable en pratique sauf si les exemples peuvent être recueillis en très grand nombre, et si les exemples tests sont réellement indépendants des exemples d'apprentissage.

De ce fait, en pratique il est très rare que l'erreur d'une hypothèse soit réellement estimée sur des exemples tests entièrement indépendants des exemples ayant servi à construire cette hypothèse. De plus, toutes les techniques d'évaluation empirique de l'apprentissage s'appuient sur l'hypothèse que les exemples sont choisis aléatoirement dans l'espace de tous les exemples possibles, ce qui est rarement le cas en pratique. La fiabilité des estimations est ainsi souvent réduite.

> *"Pour résumer, si on utilise peu de données, aucune procédure pour comparer des méthodes d'apprentissage ne satisfait toutes les contraintes que nous aimerions poser [pour obtenir une estimation précise et fiable]. Néanmoins, elles fournissent des intervalles de confiance approchés qui peuvent être d'une grande aide pour interpréter des comparaisons expérimentales de méthodes d'apprentissage"* [Mitchell 97, p.150]

Il nous faut donc considérer les estimateurs de la qualité comme des indicateurs approchés, et valider les résultats d'un apprentissage en séparant autant que possible la phase d'apprentissage et la phase d'évaluation des résultats.

C.6 Conclusion

Les algorithmes d'apprentissage sont nombreux, en particulier dans le domaine de l'apprentissage symbolique manipulant des attributs représentés en langage attribut/valeur, et il n'existe pas d'algorithme universellement plus efficace que les autres. Les recherches sur le choix d'un algorithme adapté à un problème donné ne fournissent pas à l'heure actuelle de réponse nette à cette question. Par ailleurs, les travaux combinant plusieurs algorithmes posent le problème de la compréhension des hypothèses apprises, et augmentent la difficulté pratique de mise en œuvre d'un apprentissage. Quoi qu'il en soit, le choix d'un algorithme d'apprentissage influence dans une moindre mesure les résultats de l'apprentissage par rapport à la façon de poser un problème d'apprentissage, du moins si on se limite aux algorithmes reconnus dans la communauté de l'apprentissage.

Pour nos travaux, nous préférons nous concentrer sur l'utilisation efficace d'un algorithme particulier pour notre problème, plutôt que sur la recherche d'algorithmes particulièrement adaptés. Nous choisirons donc un algorithme manipulant un biais de représentation adapté à nos besoins et reconnu globalement efficace, sans chercher à optimiser ce choix.

Pour obtenir des hypothèses les plus compréhensibles et efficaces possibles nous utiliserons l'approche du système ENIGME qui intègre les travaux en apprentissage et en acquisition de connaissances basée sur les modèles. C'est à dire qu'avant d'apprendre nous déterminerons une méthode de résolution de problème adaptée à notre contexte et nous utiliserons cette méthode pour guider l'apprentissage.

L'évaluation théorique *a priori* de l'apprentissage est une évaluation pessimiste, utile pour analyser et concevoir des algorithmes d'apprentissage, mais peu utile en pratique pour évaluer la qualité d'un apprentissage pour un problème donné. L'évaluation empirique a posteriori du résultat d'un apprentissage n'est, quant à elle, parfaitement robuste que quand beaucoup d'exemples sont disponibles. La validation croisée est néanmoins une technique utile pour estimer la qualité des hypothèses apprises quand peu de données sont disponibles, ce qui est le cas de nos travaux. Nous évaluerons donc la qualité de nos résultats par validation croisée.

Afin d'obtenir une évaluation la plus fiable et la moins optimiste possible, nous décorrélerons autant que possible la phase d'apprentissage de la phase d'évaluation des résultats. La meilleure évaluation d'un apprentissage étant une évaluation complètement externe, nous évaluerons également nos résultats en en analysant le contenu des hypothèses apprises et en évaluant la qualité de leur application sur de nouvelles données réelles.

Le chapitre D suivant présente en détail la méthode de résolution de problème que nous déterminons dans le cadre de la généralisation cartographique, ainsi que le processus d'apprentissage qui en est dérivé.

Dans le chapitre E nous expérimentons, dans le cadre de la généralisation des routes, l'approche définie dans le chapitre D, afin d'évaluer empiriquement la pertinence de nos travaux à plusieurs niveaux : quelle est la qualité des règles apprises ? Quel est l'apport de l'apprentissage par rapport au processus GALBE présenté au chapitre B ? Quel est l'intérêt de définir la méthode de résolution de problème avant d'apprendre ?

■

D APPRENTISSAGE ET GENERALISATION CARTOGRAPHIQUE

◆ Nous définissons dans ce chapitre une méthodologie pour apprendre, à partir d'exemples, les connaissances utiles au guidage des algorithmes de généralisation cartographique. Nous montrons la complexité de notre problème du point de vue de l'apprentissage supervisé. Afin d'appréhender cette complexité, nous définissons une méthode de résolution de problème adaptée au domaine de la généralisation cartographique.

◆ Nous montrons que la notion d'abstraction est au cœur de la méthode de résolution de problème, et qu'elle permet de faciliter l'apprentissage. Notre approche incite, en particulier, à successivement apprendre à abstraire puis apprendre à représenter les objets manipulés, ceci afin d'apprendre des connaissances plus compréhensibles et plus efficaces par rapport à un apprentissage en une seule étape.

D.1 Introduction

D.1.1 Bref rappel du problème

La généralisation cartographique d'un objet peut être vue comme une succession d'opérations de modification et de focalisation jusqu'à atteindre un état satisfaisant. Etant donné un objet à un moment quelconque du processus de généralisation cartographique, nous désirons apprendre quel prochain algorithme lui appliquer (cf. A.5). Cet algorithme peut être un algorithme de modification géométrique, de focalisation, ou même d'arrêt (pour homogénéiser notre discours nous considérons que l'arrêt est un algorithme comme un autre, qui laisse l'objet inchangé et arrête le processus sur cet objet).

Nous disposons de bibliothèques d'algorithmes de modification et de focalisation ainsi que de mesures adaptées au type d'objet traité. Notre but est d'apprendre à partir d'exemples à relier ces mesures au choix d'un algorithme.

Pour décrire notre problème dans le vocabulaire de l'apprentissage introduit au chapitre C précédent, nous désirons apprendre une *hypothèse* h à partir d'*exemples* constitués d'un ensemble de mesures décrivant un objet à cartographier (les *observables*) et d'un algorithme à appliquer sur cet objet (la *classe*). Le but de ce chapitre est de définir la forme de la *fonction cible* à apprendre (l'*hypothèse*) et le processus d'apprentissage permettant de l'apprendre.

Nous montrons tout d'abord dans la partie D.2 la complexité de notre problème due à la taille relativement importante des exemples (i.e. le nombre d'attributs du langage des exemples) et à leur nombre relativement faible. Ceci nous incite à décomposer la fonction cible à apprendre en un ensemble de fonctions plus simples, à l'instar de l'approche d'ENIGME (cf. C.4). Pour cela, nous montrons tout d'abord dans la partie D.3 le rôle joué par la notion d'abstraction en cartographie ainsi que son utilité en apprentissage. Cette notion est utilisée pour guider la définition d'une méthode de résolution de problème adaptée à la généralisation cartographique, comme cela est expliqué dans la partie D.4. Cette méthode de résolution de problème permet de définir directement une méthode d'apprentissage de la fonction cible que nous présentons et commentons dans la partie D.5.

D.1.2 Contexte : utilisation de la tâche apprise

Avant de définir précisément la forme de la fonction cible à apprendre, nous précisons le contexte d'utilisation de la tâche que cette fonction doit réaliser. En effet, la tâche qui détermine quel algorithme appliquer à un moment donné du processus ne suffit pas à elle seule à réaliser la généralisation cartographique d'un objet géographique. Il faut enchaîner plusieurs fois cette tâche jusqu'à atteindre un résultat satisfaisant.

Nous définissons dans la Figure 31 un moteur pour enchaîner les opérations à réaliser sur des objets géographiques. Ce moteur est minimal et contient les composantes nécessaires pour enchaîner les opérations de généralisation. C'est ce moteur qui sera utilisé dans les expérimentations présentées dans le chapitre E. Un moteur plus complexe tel que celui défini dans le projet AGENT [Ruas 99 ; Lamy et al. 99] pourrait être défini. Nous utilisons néanmoins un moteur très simple, d'une part parce que la définition d'un moteur plus puissant n'est pas l'objet de cette thèse, et d'autre part parce que cela nous permet d'évaluer plus

facilement nos résultats en séparant les effets du moteur et les effets liés à la qualité de la tâche apprise.

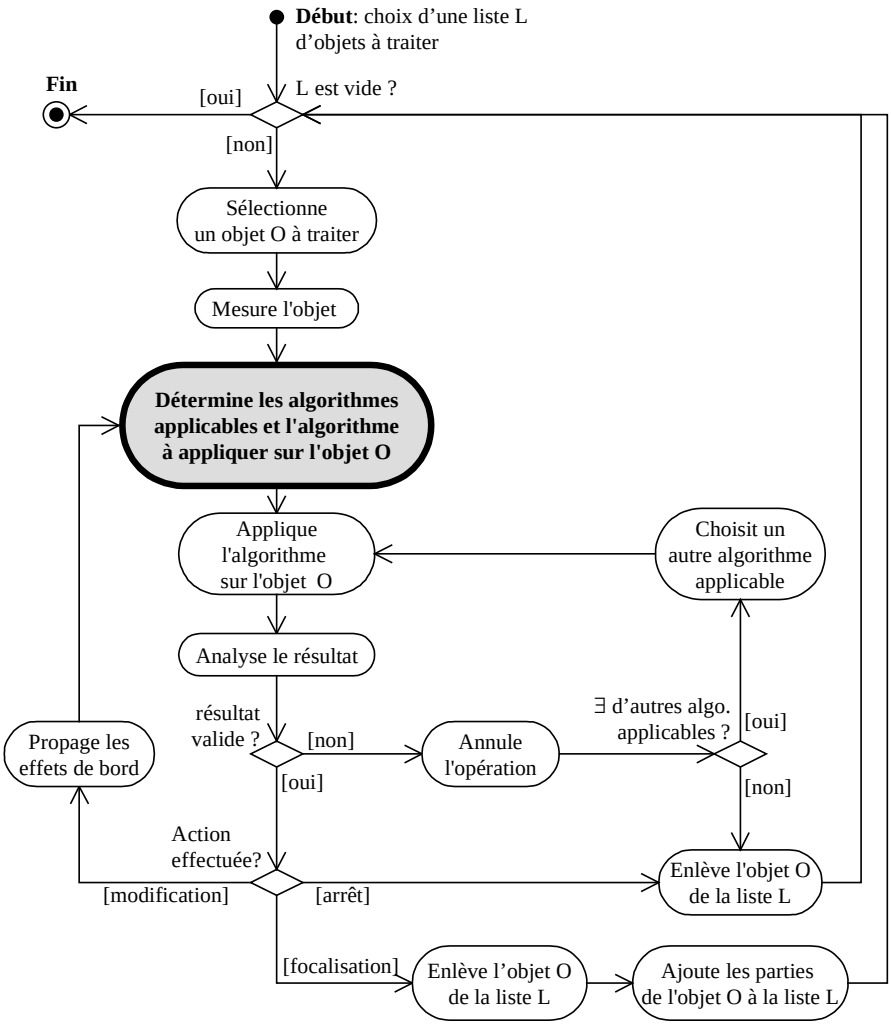


Figure 31. Moteur de la tâche à apprendre

Le moteur prend en entrée une liste d'objets à traiter (éventuellement réduite à un unique objet). L'ordre des objets dans cette liste de départ est aléatoire. Le moteur traite les objets de la liste dans l'ordre de leur classement tant qu'il subsiste des objets dans la liste. La tâche centrale de notre processus, que nous devons apprendre à réaliser, détermine un algorithme à appliquer sur cet objet, à partir d'un ensemble de mesures le décrivant. Cet algorithme est appliqué, et les effets de cette application sont gérés différemment selon le type d'algorithme :

- si l'algorithme utilisé est un algorithme de transformation, l'objet transformé est réinséré en entrée de la tâche de choix d'algorithme.
- si l'algorithme utilisé est un algorithme de focalisation qui découpe l'objet en différentes parties, l'objet est éliminé de la liste et les parties de l'objet y sont ajoutées, sans ordre particulier, à la fin de la liste des objets à traiter. Un nouvel objet à traiter est alors sélectionné dans la liste.
- si l'algorithme utilisé est "l'arrêt", l'objet est simplement enlevé de la liste des objets à traiter et un nouvel objet à traiter est sélectionné dans la liste.

Gestion des erreurs des algorithmes

En cas d'erreur, l'application d'un algorithme sur un objet peut être soit sans effet (e.g. équarrir un bâtiment parfaitement rectangulaire), soit fournir un résultat absurde (e.g. déplacer de plusieurs kilomètres un objet sur une carte à grande échelle). Ceci peut être dû soit à des erreurs dans les algorithmes, soit à l'application par erreur d'un algorithme particulièrement inadapté sur un objet. Même si la validation d'un résultat de généralisation est un problème difficile que nous avons volontairement exclu de notre étude, il est facile de détecter si le résultat d'un algorithme est sans effet ou erroné de manière flagrante. Puisque, comme nous le verrons dans la partie D.4.4, nous disposerons également de la liste des algorithmes applicables sur un objet en sortie de la tâche centrale du processus, nous pouvons à moindre coût raffiner le processus en utilisant cette information. Nous ajoutons donc dans le processus une phase de détection des résultats incontestablement erronés ou sans effet, et une phase de gestion simple de ces cas : si une erreur est détectée un autre algorithme applicable est choisi au hasard et est appliqué. Ce moteur est extrêmement simple puisque tous les choix effectués, hors celui de l'algorithme appliqué qui est au cœur de notre problème, sont simplifiés à l'extrême en étant réalisés aléatoirement (objet à traiter, autre algorithme applicable choisi).

D.2 Spécificité de notre problème vis-à-vis de l'apprentissage

D.2.1 Difficultés du recueil d'exemples

On peut aisément trouver de nombreuses cartes et bases de données géographiques (BDG) d'un même lieu. On peut ainsi trouver des objets d'une BDG et leurs homologues généralisés sur une carte. Ces objets semblent, au premier abord, une source idéale et quasi-inépuisable d'exemples que nous pourrions exploiter pour notre problème. Cependant, exploiter ces données est source de nombreuses difficultés, comme nous l'expliquons ci-dessous.

Difficultés liées à l'utilisation automatique de cartes existantes

Un premier problème réside dans le mode de fabrication des cartes. Comme nous l'avons déjà évoqué dans la partie A.4, aucune des cartes existantes n'a été réalisée par utilisation interactive des algorithmes dont nous cherchons à automatiser l'application. Même si nous disposions d'une trace du processus de fabrication des cartes existantes, les outils employés (du dessin manuel au dessin assisté sur ordinateur) sont beaucoup trop différents des algorithmes de généralisation pour que nous puissions facilement exploiter cette trace. Donc, au mieux, on peut disposer d'exemples sous la forme de couples "objet géographique initial d'une BDG / objet cartographique généralisé". Ces exemples seraient une source indirecte d'exemples pour notre problème, qui est d'associer un objet en cours de traitement au prochain algorithme à lui appliquer (Figure 32).

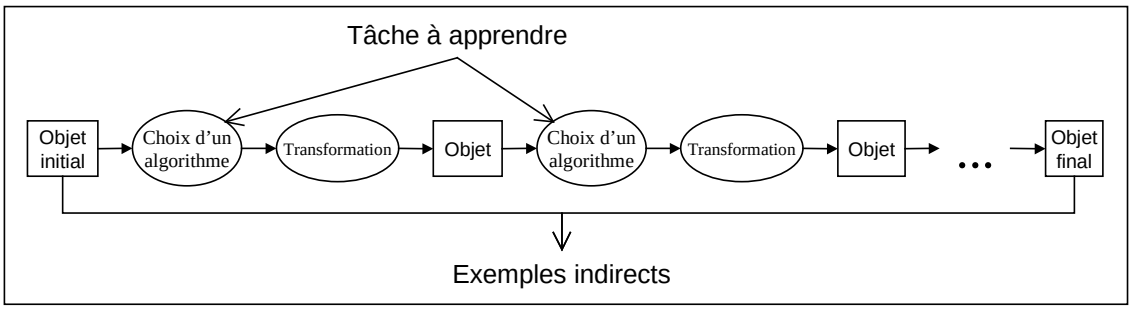


Figure 32. Exemples indirects issus des BDG et cartes existantes

Pour disposer de ces exemples indirects, il faut tout d'abord savoir retrouver des couples homologues "objet géographique / objet cartographique". Or nous ne disposons que de cartes sous forme d'images (i.e. en représentation rasteur, cf. A.1, Figure 8, p.28), et non en représentation vecteur comme la BDG. Ces images forment un tout dans lequel les objets ne sont pas dissociés les uns des autres. Pour retrouver ces objets et, en même temps, les représenter sous forme vecteur comme les objets géographiques initiaux, il faut disposer d'outils de vectorisation de cartes. De tels outils existent [Pierrot Deseilligny, Stamon et Suen 98], mais ce changement de représentation est difficile et ne se fait pas sans perte d'information. En effet, de nombreux objets graphiques se superposent sur une carte (du texte, des symboles) et les isoler les uns des autres est délicat. De plus les axes des objets symbolisés peuvent être déformés lors de la vectorisation.

Néanmoins, après vectorisation nous disposons de deux bases de données en représentation vecteur (une géographique et une cartographique). Pour obtenir nos couples d'objets homologues il faut disposer d'outils d'appariement qui, à un objet d'une base, associent son ou ses correspondants sur l'autre base. Là encore de tels outils existent [Lemarié et Raynal 96]. Mais comme d'une base à l'autre les objets ont subi de nombreuses opérations de généralisation (déplacement, regroupement, élimination…), l'appariement est difficile. Cette difficulté est augmentée du fait que la plupart du temps la carte n'a pas été créée à partir de la base de données géographique en question, mais à partir d'autres données comme des cartes à plus grande échelle. De plus, certains écarts entre la BDG et la carte sont dus à des différences de date entre les données utilisées, et ces différences biaisent les exemples.

Si, après l'appariement, nous disposons de couples d'objets homologues, on peut imaginer une manière d'exploiter cette source indirecte d'exemples. On peut par exemple essayer systématiquement de nombreuses combinaisons d'algorithmes sur un objet géographique, jusqu'à obtenir un résultat similaire à l'objet cartographique généralisé, puis utiliser cette séquence d'algorithmes comme exemples directs pour l'apprentissage. Cela signifie qu'il nous faut alors savoir évaluer cette similarité. On peut en effet rechercher au mieux un résultat similaire à l'exemple de la carte, et non un résultat exactement identique, car il y a peu de chances que les algorithmes fournissent un tel résultat. Cependant, évaluer l'écart entre deux objets est un problème difficile : si les écarts de position posent déjà problème et ont été intensément étudiés [Goodchild 77 ; Chrisman 82 ; Vauglin 97], évaluer les écarts de forme est aussi difficile [Bel Hadj Ali et Vauglin 99].

Notre but n'est d'ailleurs pas de reproduire à l'identique les résultats cartographiques obtenus manuellement sur les cartes. Notre but est d'obtenir des résultats de qualité similaire à celle des cartes réalisées manuellement : il n'existe pas une unique manière de généraliser un objet, mais de nombreuses manières aussi acceptables les unes que les autres. Si on cherche à imiter scrupuleusement les cartes existantes, on complexifie inutilement notre problème en recherchant une solution particulière, peut-être impossible à atteindre avec les algorithmes disponibles, alors que de nombreuses autres solutions sont possibles. Ceci est confirmé par

des expérimentations réalisées pour apprendre à paramètrer un algorithme de généralisation cartographique [Weibel, Keller et Reichenbacher 95]. Ces expérimentations consistaient à déterminer le paramètre d'un algorithme de manière à ce que les résultats soient les plus proches possibles d'exemples dessinés manuellement. Mais les auteurs concluent ces expérimentations en proposant de rechercher une solution possible plutôt que de chercher à atteindre le résultat manuel fourni en exemple.

Pour résumer, il serait envisageable d'utiliser des cartes et BDG existantes pour obtenir automatiquement des exemples indirects à notre problème, à condition d'utiliser des outils de vectorisation, d'appariement, et d'évaluation de l'écart entre deux objets cartographiques généralisés. Certains de ces outils existent, mais leur mise en œuvre, ainsi que la gestion des imprécisions que chacun d'entre eux peut produire, nécessiteraient de surmonter plusieurs difficultés. De plus, ces exemples ne seront pas nécessairement adaptés à notre problème, car il peut être impossible d'imiter exactement les cartes existantes avec les algorithmes de généralisation disponibles.

Construction d'exemples

Pour évaluer l'intérêt de l'apprentissage, il faut prendre en compte le coût de l'explicitation des sources de connaissances utilisées [Thomas 96, p.143]. Or, le coût dû à l'utilisation automatique de cartes et BDG existantes serait trop important pour que l'apprentissage soit réellement utile. Pour notre problème, il faut nous résoudre à fabriquer nous-mêmes les exemples que nous utiliserons pour apprendre. Même si la généralisation interactive est longue, la fabrication de tels exemples sera largement moins difficile que l'exploitation de données existantes, si on se limite à un nombre restreint d'exemples.

Puisque nous devons construire nous-mêmes les exemples, nous pouvons contraindre leur forme de manière à ce qu'ils soient directement utilisables pour notre problème. Ainsi, nous construirons des exemples contenant, au minimum, des mesures décrivant un objet (les observables) et l'algorithme à appliquer à cet objet (la classe). La forme des exemples à recueillir sera définie plus précisément dans la suite de ce chapitre, en fonction de la méthode de résolution de problème définie. Notons que, comme cela est fait en apprentissage, nous appelons expert du domaine la personne chargée de fournir les exemples, même s'il s'agit en l'occurrence de nous-mêmes, ceci afin de distinguer notre rôle de cartographe de notre rôle de cogniticien.

D.2.2 Bruit sur les exemples

Les exemples que nous pouvons utiliser pour l'apprentissage sont nécessairement bruités, à la fois au niveau des observables et au niveau de la classe.

Bruit sur les observables

La forme d'un objet géographique est décrite par sa géométrie. Cette géométrie influence grandement nos choix de généralisation, il faut donc savoir la décrire dans la représentation de nos exemples. Elle est traditionnellement représentée sous la forme d'une liste de couples de coordonnées (cf. partie A.1). Mais si ces coordonnées définissent parfaitement la forme d'un objet, elles sont difficilement exploitables pour constituer directement les observables de notre problème d'apprentissage.

En effet, tout d'abord cette représentation est extrêmement sensible aux translations et aux rotations alors que nos exemples naturels ne le sont pas (la position et l'orientation d'un objet importent peu dans notre problème). Ensuite, cette suite de coordonnées est de taille variable pour chaque objet. Certains algorithmes d'apprentissage sont capables de traiter de tels observables de taille variable en les représentant dans des langages relationnels, mais la complexité de l'apprentissage est alors très grande [Giordana et al. 2000]. Mais surtout, la principale limite de cette représentation est que l'information utile n'y est représentée que très implicitement : un couple de coordonnées n'apporte aucune information sur la forme d'un objet, c'est uniquement l'ensemble de ces couples qui est porteur d'information.

Pour avoir des exemples représentés dans un langage expressif, il faut décrire un objet par des mesures portant sur diverses propriétés (sa taille, sa forme…). En effet, si on applique une même transformation sur deux objets, ils auront sûrement des propriétés proches alors qu'ils peuvent avoir des coordonnées très différentes. Si un ensemble de mesures permet d'obtenir un langage beaucoup plus expressif que la liste des coordonnées, cela ne se fait pas sans perte d'information. Une première raison à cela est que nous ne mesurons que quelques caractères de l'objet. Ce choix des caractères à mesurer est une forme d'abstraction de nos données et cette abstraction est utile à l'apprentissage (cf. D.3.3). Mais les mesures utilisées ne permettent que d'estimer de manière approchée les caractères perçus, car ceux-ci sont peu formalisés et difficiles à mesurer (cf. B.3). Une première difficulté est de déterminer un estimateur du caractère perçu (par exemple comment mesurer la sinuosité d'une ligne ? en comptant le nombre de virages ? en estimant la dimension fractale ?…). Une deuxième difficulté réside dans les problèmes liés aux effets visuels, par exemple la forme ou le contexte d'un objet influence la perception de sa taille. Une mesure mathématique rendra donc difficilement compte de la perception exacte du caractère [Ninio 96].

Ces difficultés pour mesurer un objet, et en particulier les notions liées à sa forme, font que nos mesures ne seront pas complètement informatives, et que nos exemples apparaîtront par essence bruités (car on ne décrira ni complètement ni précisément la forme d'un objet, cf. Figure 29, page 78).

Bruit sur la classe

Par ailleurs, les algorithmes de généralisation peuvent être instables, ce qui entraîne l'apparition de bruit au niveau de la classe des exemples. En effet, un algorithme qui est normalement parfaitement adapté pour s'appliquer dans une configuration donnée pourra, à cause de son instabilité, fournir un résultat aberrant sur un objet particulier de cette configuration (e.g. s'il possède par hasard un certain nombre de points concentriques). Dans ce cas un autre algorithme sera choisi par l'expert construisant les exemples, même si de manière générale il est moins bien adapté pour cette configuration. Pour espérer détecter par apprentissage ces configurations très particulières, il faudrait sûrement utiliser une représentation très détaillée des exemples et utiliser de très nombreux exemples, ce qui en complexifierait énormément le recueil.

Les exemples dus aux erreurs des algorithmes ont donc leur classe bruitée, du point de vue de notre problème d'apprentissage. Néanmoins, si les résultats de l'apprentissage sont de bonne qualité, ils permettront de détecter des cas où les algorithmes font des erreurs. En effet, quand on appliquera les règles apprises, si un algorithme réalise des erreurs flagrantes alors qu'il est utilisé dans un contexte que les règles apprises estiment adapté, on pourra soupçonner une erreur de conception ou de codage de l'algorithme. L'analyse de ces cas pourra aider à la détection et à la correction de telles erreurs.

En résumé, les erreurs des algorithmes introduisent du bruit sur la classe des exemples d'apprentissage, mais les résultats de l'apprentissage aideront à détecter ces erreurs.

D.2.3 Taille des exemples

Taille des observables

Le choix d'un algorithme en fonction d'un objet dépend au moins de la taille de cet objet, de sa forme, de son environnement spatial, et des conflits graphiques dus à sa symbolisation. La plupart de ces caractères ne peuvent être représentés à l'aide d'une seule mesure.

Plusieurs mesures peuvent être nécessaires pour qualifier un caractère perçu [Ruas 99, p.92]. Par exemple, pour évaluer la similarité entre deux bâtiments dans un contexte de généralisation cartographique, Regnauld [1998, pp.69-79] envisage, entre autres, six mesures de forme (trois pour qualifier l'allongement et trois pour qualifier la concavité), même s'il n'en conserve que deux pour simplifier son problème. En effet, d'une part, une mesure peut être plus ou moins adaptée à un type d'objet et, d'autre part, différentes mesures peuvent se concentrer sur différents aspect d'un caractère. Ceci est illustré dans la Figure 33 : on peut imaginer de calculer l'orientation générale d'un bâtiment à partir de l'orientation de ses murs ou à partir de sa plus grande diagonale [Regnauld 98]. La première mesure est mieux adaptée aux bâtiments relativement rectangulaires qu'aux bâtiments en escalier, alors que c'est l'inverse pour la deuxième. De plus, les deux mesures n'ont pas le même sens : elles peuvent être toutes deux pertinentes mais mesurent différents aspects de l'orientation.

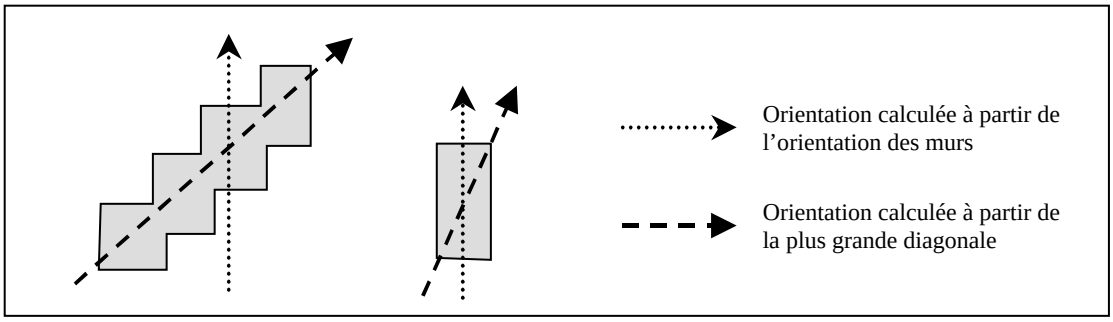


Figure 33. Mesures de l'orientation d'un bâtiment

Les exemples de généralisation cartographique doivent donc être décrits par plusieurs caractères, chacun éventuellement représenté par plusieurs mesures. Une représentation utile des observables des exemples naturels de généralisation peut donc contenir de nombreux attributs, en général numériques.

Valeurs possibles de la classe

De nombreux algorithmes ont été conçus pour s'appliquer sur un type d'objet donné. Par exemple, la plate-forme PlaGe [Lecordix, Plazanet et Lagrange 97] possède une vingtaine d'algorithmes potentiellement applicables sur tout objet linéaire. Une première analyse de ces algorithmes vis-à-vis de besoins donnés permet d'en faire une première sélection : par exemple certains algorithmes sont mieux adaptés aux objets artificiels de forme relativement régulière (comme les voies ferrées) qu'aux objets naturels de forme complexe (comme les cours d'eau). Néanmoins, même après cette sélection, il peut rester de nombreux algorithmes pertinents pour traiter différentes configurations d'un objet de type donné. Ainsi, pour notre problème d'apprentissage, la classe des exemples (i.e. l'algorithme à appliquer) peut prendre de nombreuses valeurs possibles.

D.2.4 Bilan : caractéristiques des exemples

Notre problème, d'un point de vue apprentissage, se caractérise donc par l'utilisation de relativement peu d'exemples, sensiblement bruités, et de taille relativement importante. Dans les expérimentations que nous avons réalisées (cf. chapitre E), nous avons utilisé entre 100 et 200 exemples, décrits par 10 à 25 attributs numériques, avec une classe pouvant prendre une dizaine de valeurs, et que nous avons estimé contenir de 5 à 10% de bruit.

Etant donné leur taille relativement importante, le nombre d'exemples disponibles est donc très faible par rapport aux problèmes couramment traités en apprentissage. Les premiers tests que nous avons réalisés avec différents algorithmes d'apprentissage montrent que l'apprentissage direct d'hypothèses à partir de ces exemples est extrêmement inefficace, avec des taux d'erreur estimée pouvant atteindre les 50% et dépasser l'erreur par défaut des exemples[16].

Ceci justifie la nécessité de contraindre fortement le processus d'apprentissage grâce à des connaissances du domaine, et donc de définir une méthode de résolution de problème préalablement à l'apprentissage (cf. C.4). Pour cela, nous montrons dans la partie D.3 suivante le rôle central joué par la notion d'abstraction en cartographie, ainsi que son intérêt en apprentissage. Ceci nous permet de définir, dans la partie D.4, une méthode de résolution de problème adaptée à l'apprentissage pour la généralisation cartographique.

D.3 Abstraire

D.3.1 Modèle théorique d'abstraction

La représentation et la manipulation des connaissances sont l'un des principaux sujets de recherche en Intelligence Artificielle [Davis, Shrobe et Szolovits 93]. Depuis une cinquantaine d'années de nombreux langages formels ont été utilisés en IA, et chacun d'eux est plus ou moins adapté pour représenter différents domaines de connaissances [Ginsberg 97]. Bien que de nombreux domaines de connaissances puissent être formulés dans *un seul* langage donné, certains, comme le processus cartographique, requièrent *plusieurs* langages de représentation pour représenter les différentes données manipulées, des données brutes du monde à leur représentation finale par une carte.

En IA, la prise en compte de plusieurs niveaux de détail au sein d'un même langage de représentation et la capacité à passer d'un niveau de détail à un autre fait depuis longtemps référence à la capacité humaine dite d'*abstraction.*

Saitta et Zucker [Saitta et Zucker 98 ; Zucker 98] ont proposé un modèle général d'abstraction. Partant de l'analyse du rôle de l'abstraction dans la représentation et l'apprentissage de concept, leur étude les a conduit à différencier deux processus fondamentaux : le changement de langage de représentation et l'abstraction au sein d'un langage de représentation. Nous résumons ici quelques grandes lignes de ce modèle, afin de mettre en évidence la différence entre les notions de changement de langage et d'abstraction. Ces notions sont souvent confondues, à tort, en généralisation cartographique :

[16] Erreur de l'hypothèse qui à tout exemple attribue la classe majoritaire dans les exemples.

"L'abstraction de l'information géographique est un moyen de mieux appréhender la réalité géographique présente dans les données. […] Des études supplémentaires pour clarifier la nature des relations entre abstraction de bases de données géographiques et généralisation cartographique seraient utiles. Ceci n'a pas été suffisamment fait à cause de la confusion liée à l'utilisation d'un même terme dans deux contextes [informatique et cartographique] : la généralisation." [Nyerges 91]

Des langages de représentation pour raisonner sur le monde

Sans rentrer dans des considérations philosophiques qui dépassent le cadre de cette thèse, à la base de toute expérience se trouve le *monde* (*W*) qui contient les entités réelles. Ce monde n'est jamais directement connu car nous n'y avons accès que par une *perception* de celui-ci, *P(W)*. A ce niveau, ne sont perçus que les stimuli physiques que le monde crée sur un vecteur de notre perception. Ces stimuli représentent une vision instantanée du monde. Afin de pouvoir utiliser ces stimuli, il est nécessaire de les stocker dans une *structure* organisée (*S*), c'est-à-dire une représentation en extension [Van Dalen 83] de la perception du monde. Cette structure peut être par exemple organisée sous la forme de tables relationnelles, sur lesquelles une algèbre relationnelle peut être appliquée. Ensuite, pour pouvoir décrire symboliquement le monde, et pour communiquer avec d'autres agents, un *langage* (*L*) est requis. Enfin, une *théorie* (*T*) est nécessaire pour raisonner sur le monde. Cette théorie contient des connaissances générales ou spécifiques à un domaine qui permettent la réalisation d'inférences sur le langage, et s'enrichit ainsi grâce à l'expérience exprimée dans le langage. Saitta et Zucker appellent *contexte de raisonnement* (R) l'ensemble de ces quatre langages de représentation : <*P(W),S,L,T*> (Figure 34).

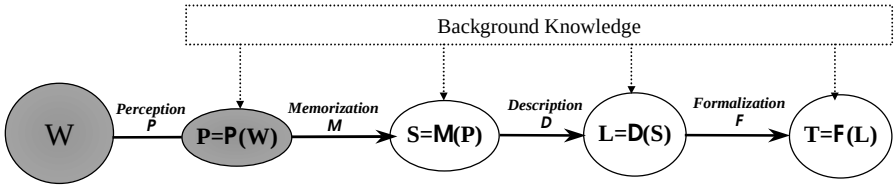


Figure 34. Contexte de raisonnement [Saitta et Zucker 98]

Abstraction sur la perception du monde

Le monde peut être perçu d'une infinité de manières différentes par un agent intelligent en fonction des outils d'observation, du but de l'observation, de la culture de l'agent, etc. Il existe donc, pour un même monde, une infinité de perceptions possibles de celui-ci. C'est à ce niveau que sont déterminés le type et la quantité d'information que l'agent va stocker, décrire et analyser. Plus la perception est détaillée, moins elle est abstraite. L'agent peut parfois contrôler le contenu de sa perception du monde, s'il contrôle les outils d'observation et s'il connaît à ce niveau le but de l'observation. Cependant, le plus souvent il ne peut pas contrôler cette perception : c'est le cas lorsqu'il collecte plus d'information qu'il ne lui est nécessaire, ou lorsqu'il a à effectuer plusieurs tâches, chacune nécessitant différentes informations qu'il est par ailleurs facile de collecter simultanément. Ceci montre qu'il est utile d'avoir des méthodes pour concrètement, ou virtuellement, transformer une perception en une autre plus abstraite. La définition suivante met en valeur ces considérations.

> *Etant donné un monde W, soit R_r=<$P_r(W)$,S_r,L_r,T_r> et R_a=<$P_a(W)$,S_a,L_a,T_a> deux contextes de raisonnement. Une abstraction est une fonction A :$R_r(W) \rightarrow R_a(W)$ telle que la perception $P_a(W)$ dite abstraite est plus simple que la perception $P_r(W)$ dite de référence.* [Saitta et Zucker 98]

Cette définition mérite quelques commentaires. Dans [Saitta et Zucker 98] la relation "plus simple que" est définie formellement en termes de diminution de quantité d'information contenue dans la perception du monde. La perception la moins abstraite est conventionnellement appelée de référence, mais la relation d'abstraction n'est que relative et toute perception peut être considérée de référence. En outre, le processus d'abstraction peut être réitéré plusieurs fois, la relation d'abstraction étant une relation transitive [Iwasaki 90 ; Yoshida et Motoda 90]. Il est également important de noter que, dans cette définition, l'abstraction est définie au niveau de la perception du monde. Les abstractions au niveau de la structure, du langage, de la théorie en sont ensuite dérivées (Figure 35).

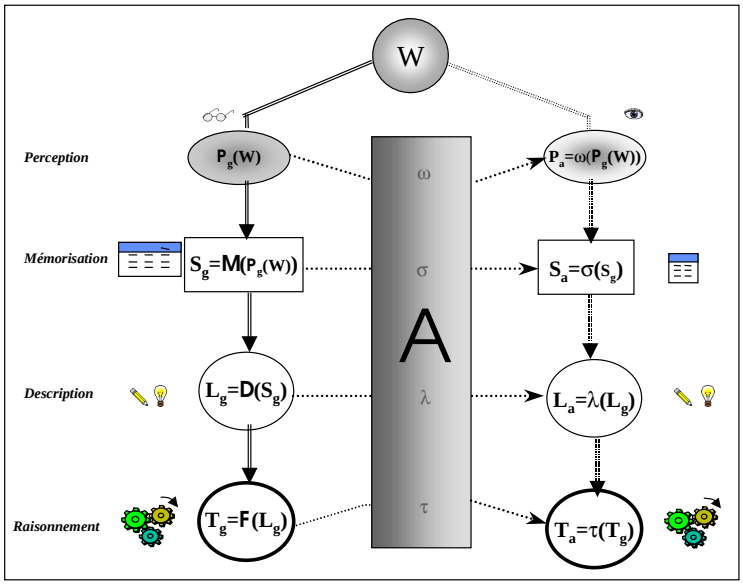


Figure 35. L'abstraction et ses dérivés [Saitta et Zucker 98]

D.3.2 Abstraction et cartographie

Dans cette partie nous présentons les analogies entre le processus de création d'une carte et le modèle théorique introduit ci-dessus [Mustière, Zucker et Saitta 2000a ; 2000b]. Nous présentons les différents langages de représentation utilisés en cartographie et le processus d'abstraction qui y est réalisé. Ceci nous permet de définir la généralisation cartographique en y mettant en valeur les notions de changement de représentation et d'abstraction.

Le processus cartographique

Considérons tout d'abord les différents langages de représentation utilisés en cartographie. Pour illustrer notre propos nous nous concentrons sur les cartes dites topographiques, c'est-à-dire de la surface de la terre. Les différents langages de représentation et le processus cartographique décrits ci-dessous correspondent, de manière simplifiée, à ceux utilisés dans les principaux organismes producteurs de données topographiques.

En topographie, le monde *W* étudié est une partie du monde physique, et plus précisément la surface de la terre (son relief, l'occupation du sol, le réseau routier…). Selon le but de l'observation, nous avons une certaine vision *P(W)* du monde, appelée terrain nominal en terminologie SIG. Dans un but de recueil de données topographiques par un institut de cartographie comme l'IGN, cette vision contient un ensemble d'éléments topographiques relativement stables dans le temps. En topographie, la source principale de la perception *P(W)* est constituée de photographies aériennes ou d'images satellites couvrant l'ensemble de la

zone étudiée, ainsi que d'enquêtes sur le terrain. Ces images sont le moyen le plus rapide d'obtenir une grande quantité de données localisables dans l'espace. Les éléments contenus dans ces images, et correspondant à la perception *P(W)*, peuvent être identifiés puis stockés dans une base de données géographique. Même si les objets de cette BDG sont nommés (route, maison…), celle-ci représente plus un stockage de données, i.e. une structure *S*, qu'un langage de description à partir duquel il est possible de communiquer. En effet, la BDG est difficilement analysable directement, une liste de coordonnées ne permettant pas d'appréhender directement la forme d'un objet. Ensuite, le langage *L* communément utilisé pour représenter et analyser des données géographiques est la carte, qui est réalisée à partir de la BDG. Sur la carte chaque objet géographique est représenté par un symbole. En ce sens on peut considérer la cartographie comme un langage iconique. Enfin, une théorie *T* est nécessaire pour raisonner sur la carte. Celle-ci contient des connaissances géographiques et cartographiques qui permettent d'analyser les différentes configurations géographiques rencontrées, que ce soit dans le cadre d'une recherche d'itinéraire, de la reconstruction mentale du paysage, de la recherche d'un modèle de développement des villes, etc. La conception de la carte est guidée pour satisfaire au mieux la théorie qui y sera utilisée.

Généralisation cartographique et abstraction

Le but de la généralisation cartographique est de créer des cartes à partir d'une base de données géographique trop détaillée. C'est-à-dire créer une carte représentant une autre perception du monde que celle qui a guidé la création de la BDG qui sert de *référence*. En effet, la BDG de référence a été créée avec une certaine perception du monde (par exemple une vision très locale, à l'échelle d'un randonneur) et la généralisation cartographique a pour but de créer une carte avec une autre perception correspondant à un nouveau besoin (par exemple celle moins locale d'un automobiliste). En terminologie SIG, la carte et la BDG de référence sont issus de deux terrains nominaux différents.

La généralisation cartographique consiste donc tout d'abord à changer de *langage de représentation* : fabriquer une carte à partir d'une base de données géographique. Les objets, qui sont stockées dans la BDG comme une suite de coordonnées {(x,y)}, doivent être représentés graphiquement par des symboles. Mais il s'agit également d'*abstraire* des données puisque l'on doit réaliser une carte à partir de données trop détaillées. **On peut donc définir la généralisation comme la *représentation* cartographique d'une *abstraction* du monde géographique.**

On peut en effet voir toutes les opérations de généralisation cartographique comme la conséquence d'une abstraction définie au niveau de la perception du monde, et réalisée pendant le changement de langage pour mettre en valeur un des caractères d'un objet géographique au détriment d'autres caractères.

Illustrons ceci par deux exemples : la simplification et le déplacement d'objets, deux opérations courantes en cartographie. D'une part, si un objet est trop détaillé sa forme est difficilement identifiable sur la carte, il faut donc la simplifier. D'autre part, si deux objets sont trop proches, on ne peut plus les distinguer sur la carte, il faut donc les éloigner l'un de l'autre. La simplification est aisément perçue comme une abstraction car la quantité d'information (en terme de nombre de coordonnées) nécessaire pour représenter l'objet dans les bases de données est diminuée durant cette opération. Par contre, la quantité d'information présente avant et après déplacement reste constante dans les bases de données, même si la précision de localisation des objets est dégradée. Le déplacement n'est donc pas toujours aisément perçu comme une abstraction, si on se limite à une analyse au niveau des bases de données.

Mais si on analyse ces deux opérations au niveau de la carte, elles sont très semblables. Ces deux transformations sont provoquées par la même cause : les objets ne sont pas lisibles si on les affiche tels quels. Elles ont aussi les mêmes effets : rendre les objets lisibles en mettant en avant un de leur caractère, c'est à dire en les abstrayant. Dans un cas l'abstraction retient essentiellement la forme générale de l'objet. Dans l'autre cas, l'abstraction retient la séparation entre les objets ou, autrement dit, l'abstraction retient la forme générale du groupe constitué des deux objets. Ces deux abstractions ont été choisies au niveau de la perception du monde. Dans le cas du déplacement par exemple, on est passé d'une perception du monde où la localisation précise et la séparation entre les bâtiments sont importantes à une perception où la séparation est plus importante que la localisation précise. C'est cette abstraction de la perception du monde, choisie pour satisfaire les besoins de la théorie à laquelle est dédiée la carte, qui permet de sélectionner une opération efficace de transformation (le déplacement ou la simplification) pour passer de la BD à la carte.

D.3.3 Abstraction et apprentissage

Le modèle théorique d'abstraction de Saitta et Zucker [1998] décrit dans la partie D.3.1 a été défini pour être utilisé en particulier dans le cadre de l'apprentissage supervisé. En effet, l'abstraction y est définie au niveau de la perception du monde, c'est à dire au niveau des exemples naturels. Or l'apprentissage est la recherche d'hypothèses dans un espace défini par les biais d'apprentissage et, plus l'espace est grand, plus l'apprentissage est difficile. Abstraire les exemples permet de réduire la taille de l'espace des hypothèses, et donc de faciliter l'apprentissage [Giordana et Saitta 90]. Par exemple, si on abstrait les exemples en diminuant le nombre d'attributs utilisés pour les décrire, on diminue le nombre d'hypothèses possibles pour expliquer la classe en fonction des observables : seules les hypothèses portant sur les attributs conservés sont envisageables.

Afin de diminuer l'espace des hypothèses à parcourir, une première façon d'abstraire les exemples est d'abstraire les observables. La technique certainement la plus étudiée pour cela est la discrétisation des attributs numériques : un attribut pouvant prendre une infinité de valeurs est remplacé par un attribut ayant seulement quelques valeurs possibles. Cette discrétisation peut être dirigée par la seule distribution des valeurs de l'attribut (e.g. méthode des quantiles). Elle peut aussi être dirigée par cette distribution et par la valeur de la classe des exemples (e.g. utilisation de tests du chi−2 pour déterminer les meilleures discrétisation vis à vis de la classe [Kerber 92 ; Richeldi et Rossoto 95]). Elle peut enfin être dirigée par la qualité de l'apprentissage (e.g. réitération de la discrétisation pour améliorer le résultat de l'apprentissage [Kohavi et Sahami 96]). Une autre manière d'abstraire les observables est de n'utiliser que certains attributs des exemples pour apprendre, ces attributs pouvant être choisis sur les conseils d'un expert [Cherkauer 96] ou par des méthodes statistiques telles que les analyses en composantes principales [Popelinski et Brazdil 2000]. A l'instar de l'algorithme CHARADE [Ganascia 87 ; 91], une autre approche possible est de modifier les observables en utilisant des connaissances du domaine traité, comme des classifications hiérarchiques des valeurs possibles des attributs[17]. Par exemple on peut remplacer un attribut *espèce* avec de nombreuses valeurs possibles {*truite, saumon, vache, cheval*} par un attribut *genre* avec moins de valeurs possibles {*mammifère, poisson*}.

[17] CHARADE abstrait ainsi les attributs, mais il n'élimine pas les valeurs concrètes des attributs pour l'apprentissage : il utilise à la fois les valeurs abstraites et les valeurs concrètes des attributs dans une recherche exhaustive de règles expliquant la classe en fonction des observables.

Une deuxième façon d'abstraire les exemples est d'abstraire leur classe au lieu d'abstraire leurs observables. Par exemple, dans le cas où la classe peut prendre de nombreuses valeurs, Dietterich et Bakiri [1995] proposent de grouper au hasard les valeurs des classes (les groupes de valeurs deviennent des classes abstraites). Ils effectuent alors de nombreux apprentissages avec différents regroupement, et combinent les différents classifieurs appris sur les classes abstraites pour déterminer la classe concrète d'un nouvel exemple : si un exemple est classé par un classifieur comme étant de la classe "Valeur1 ou Valeur2" et de la classe "Valeur2 ou Valeur3" par un autre classifieur, alors il est classé "Valeur2". Cette méthode améliore sensiblement les résultats de l'apprentissage [Ricci et Aha 97].

L'abstraction des exemples, au moyen d'une abstraction de leurs observables ou de leur classe, permet donc de réduire l'espace des hypothèses à explorer et ainsi de faciliter l'apprentissage, à condition que l'information utile contenue dans les exemples ne soit pas trop diminuée durant la phase d'abstraction.

D.4 Construction de la méthode de résolution de problème

Le rôle central joué par l'abstraction en généralisation cartographique, comme en apprentissage, nous permet de définir une méthode de résolution adaptée à notre problème. Comme dans la méthodologie GDM (Generalised Directive Models) [Le Roux et al. 93], développée pour guider la construction de méthodes de résolution de problème, nous présentons la méthode proposée comme le raffinement progressif d'une méthode élémentaire.

Le raffinement de notre méthode de résolution de problème a été guidé, d'une part, par l'analyse théorique de la généralisation cartographique (présentée dans la partie D.3.2) et, d'autre part, par l'analyse de résultats obtenus de manière empirique [Mustière, Zucker et Saitta 2000b ; Zucker, Mustière et Saitta 2000]. Pour obtenir ces résultats empiriques, nous avons réalisé des tests sur la généralisation des bâtiments[18]. Les résultats de ces premiers tests sont présentés en annexe VI. L'analyse des résultats obtenus a été facilitée grâce à la compréhensibilité des règles apprises, avantage certain de l'approche symbolique de l'apprentissage, que nous utilisons pour nos travaux. En effet, pour raffiner le processus, nous n'avons pas seulement étudié la qualité prédictive des hypothèses apprises, nous avons également étudié le contenu des hypothèses pour comprendre comment les rendre plus efficaces et plus compréhensibles. Cette approche cyclique (définition de la méthode de résolution de problème, apprentissage, raffinement de la méthode, apprentissage…) est naturelle dans une approche intégrant acquisition des connaissances et apprentissage symbolique [Nédellec et Causse 92 ; Bisson 94 ; Thomas 96, p.87].

D.4.1 Méthode initiale de résolution de problème

En première approche, notre problème se présente sous la forme élémentaire de la Figure 36. A un moment donné du processus, un objet est décrit par un ensemble de mesures et nous

[18] Les exemples pour réaliser ces expérimentations nous ont été fourni par Nicolas Regnauld de l'université d'Edimbourg. Ces exemples ont utilisé des mesures et des algorithmes issus de divers travaux [Ruas 88 ; Regnauld 98 ; Regnauld, Edwardes et Barrault 99] et implémentés sur le SIG Lamps2 de Laser-Scan lors du projet européen AGENT (ESPRIT/LTR/24939). Nous remercions vivement Nicolas Regnauld pour son aide.

connaissons la transformation[19] qu'il a subi à l'étape précédente. Nous désirons ensuite déterminer un nouvel algorithme à lui appliquer. Les parties suivantes (D.4.2 à D.4.5) expliquent comment raffiner peu à peu cette méthode de résolution de problème élémentaire.

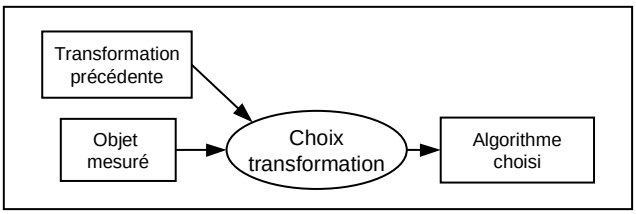


Figure 36. Méthode de résolution de problème élémentaire

Il peut paraître surprenant de prendre en compte la dernière transformation effectuée sur l'objet pour déterminer l'algorithme à appliquer à un moment donné du processus. En effet, on peut penser qu'un objet devrait être traité uniquement en fonction de ses propriétés, et non de la manière dont il a été créé. Mais plusieurs raisons nous poussent à utiliser cette information. Tout d'abord elle est directement disponible, son utilisation ne complique donc pas le processus. De plus, si cette information est inutile, elle ne sera pas prise en compte dans les règles apprises. Ensuite, cette information compense en partie le manque éventuel de mesures de description de l'objet. En effet, le choix d'une transformation dépend en théorie non seulement des propriétés intrinsèques de l'objet mais aussi de l'écart à ses propriétés initiales. Si nous ne disposons pas de mesures efficaces pour décrire cet écart, la description de la transformation précédente est un moyen de décrire (en petite partie) cet écart. Enfin, cette information peut être utile pour découvrir que certaines transformations sont systématiquement réalisées à la suite l'une de l'autre, ou que le processus est systématiquement arrêté après certaines transformations, ou encore que certaines transformations sont systématiquement réalisées au début du processus.

Nos expérimentations, présentées dans le chapitre E, confirment cela et montrent que la connaissance de la dernière transformation effectuée est particulièrement utile pour décider quand arrêter le processus.

D.4.2 Abstraire les mesures

Principe

L'inférence élémentaire de choix de transformation décrite ci-dessus est difficile à apprendre si beaucoup de mesures sont nécessaires pour décrire un objet et si peu d'exemples sont utilisés (cf. D.2.4). Comme l'abstraction des observables permet de simplifier l'apprentissage (cf. D.3.3), cette inférence élémentaire peut être raffinée en introduisant tout d'abord une étape d'abstraction de l'objet géographique mesuré. Cette approche se justifie d'autant plus que la généralisation cartographique peut être vue comme la représentation d'une abstraction du monde géographique (cf. D.3.2) : pour choisir comment représenter une abstraction du monde, on peut raisonner sur une abstraction de celui-ci.

[19] Nous utilisons le terme *Transformation* pour désigner l'ensemble des éléments qui permettent de décrire la dernière action effectuée sur l'objet, à savoir l'opération, l'algorithme et le paramètre comme nous le préciserons par la suite.

La généralisation cartographique contient des opérations d'abstraction pour mettre en avant certains caractères de l'information géographique (la taille, la forme, la séparation…). On fait naturellement appel à ces caractères pour décrire l'information géographique, dans un but de généralisation cartographique (e.g. "Si un bâtiment est *petit* et en *forme de L*, alors il faut mettre en avant sa *forme* au détriment de sa *taille*"). Pour abstraire un objet mesuré, et en même temps le décrire par des concepts naturellement utilisés dans le domaine de la généralisation, nous décrivons cet objet sous la forme d'un ensemble d'attributs symboliques. Ces attributs, ainsi que leurs valeurs possibles, doivent être choisis par un expert du domaine.

Ceci nous permet de raffiner la méthode de résolution de problème de la Figure 36 en découpant l'inférence du choix d'un algorithme en deux phases (Figure 37). Une inférence d'*abstraction* décrit symboliquement et plus simplement l'objet manipulé. Une inférence de *représentation* choisit ensuite l'algorithme à appliquer en fonction de la description abstraite de l'objet et éventuellement en fonction de la transformation qu'il a subit à l'étape précédente. Nous appelons cette inférence *représentation* car c'est elle qui détermine indirectement comment représenter l'objet de la BDG sur la carte, en déterminant quelle transformation géométrique appliquer sur l'objet pour le cartographier correctement.

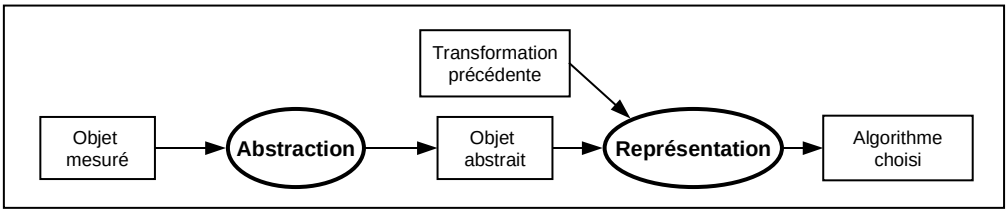


Figure 37. Premier raffinement : abstraire les mesures

La phase d'abstraction définie ici diffère des techniques de discrétisation des attributs numériques. Tout d'abord, les techniques de discrétisation définissent un attribut symbolique en fonction d'un seul attribut numérique, alors que nous proposons de définir un attribut symbolique en fonction de plusieurs attributs numériques. Ensuite, les techniques de discrétisation se basent sur une analyse de la répartition des valeurs d'un attribut numérique et étudient *a posteriori* si l'abstraction ainsi définie est adaptée au domaine traité. A l'inverse, nous proposons de définir *a priori* la forme de l'abstraction, puis d'apprendre *ensuite* comment déterminer cette abstraction grâce à des exemples. Notre approche utilise donc des connaissances du domaine, non contenues dans les exemples, pour définir l'abstraction.

Pour que l'introduction de la phase d'abstraction soit efficace, il faut que deux conditions soient satisfaites :

- l'information nécessaire et suffisante au choix d'un algorithme de généralisation d'un objet doit être contenue dans la représentation abstraite de cet objet.
- la représentation abstraite doit être déterminée efficacement en fonction de la représentation concrète de l'objet.

Pour que ces deux conditions soient satisfaites au mieux, il peut être utile de simplifier la description abstraite d'un objet, peut-être au détriment de la première condition, mais au bénéfice de la seconde. Plus concrètement, cela signifie qu'il peut être utile de réduire le nombre d'attributs abstraits utilisés, si ceux-ci sont mal déterminés à partir des mesures.

Intérêt vis-à-vis de l'apprentissage

Dans l'étape d'abstraction des mesures, l'abstraction est dirigée par les connaissances du domaine. La forme de l'hypothèse à apprendre est ainsi fortement contrainte : l'hypothèse doit déterminer des abstractions adaptées au problème, avant de choisir un algorithme uniquement en fonction de ces abstractions. Cette approche, qui contraint les hypothèses grâce aux connaissances du domaine, est celle du système ENIGME [Ganascia, Thomas et Laublet 93 ; Thomas 96] (cf.C.4). Elle permet d'améliorer la lisibilité des règles apprises, elle peut aussi permettre d'améliorer les résultats de l'apprentissage par rapport à un apprentissage non contraint.

L'introduction de l'étape d'abstraction permet de réduire la taille des espaces des hypothèses manipulés. Tout d'abord, la phase de *représentation* utilise des observables abstraites définies dans un espace beaucoup plus petit que celui des mesures : l'espace des observables avant abstraction est défini par un ensemble d'attributs numériques (les mesures), alors que l'espace des observables après abstraction est défini par un ensemble plus petit d'attributs symboliques.

Ensuite les attributs abstraits prennent en général moins de valeurs que le nombre d'algorithmes de généralisation possibles, étant donné que ceux-ci sont nombreux. Pendant la phase *d'abstraction*, l'espace des hypothèses reliant les mesures à un attribut abstrait est ainsi moins grand que celui des hypothèses reliant les mesures à un algorithme de généralisation. Mais surtout, lors de la phase d'*abstraction*, nous pouvons n'utiliser qu'un sous-ensemble des mesures pour apprendre chaque attribut abstrait : celles identifiées comme pertinentes (par les connaissances du domaine) pour apprendre chacun des attributs abstraits. Ceci est illustré dans la Figure 38.

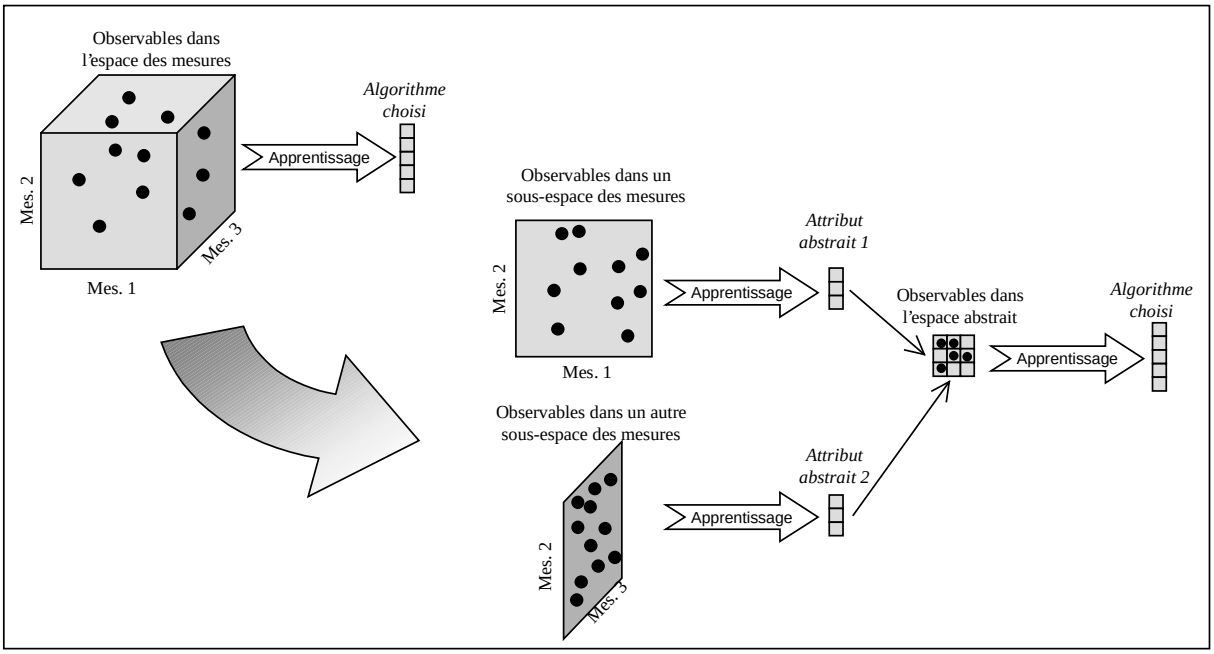


Figure 38. Reformulation de l'apprentissage grâce à l'abstraction

On peut donc espérer, en analysant les biais de représentation manipulés, améliorer les résultats de l'apprentissage grâce à l'abstraction. Ceci à condition que la majeure partie de l'information utile pour décrire un objet soit représentable dans le langage abstrait, et que la combinaison des résultats des apprentissages des différentes phases (aux erreurs supposées peu fréquentes) produise moins d'erreurs que l'apprentissage direct (aux erreurs supposées plus fréquentes).

D'un point de vue plus cognitif, on peut aussi espérer améliorer les résultats de l'apprentissage lors de l'abstraction. En effet, l'apprentissage est plus difficile à partir d'attributs numériques qu'à partir d'attributs symboliques (porteurs de plus de sens et exprimés dans un espace plus restreint). On peut donc estimer que la plus grande source d'erreurs de l'apprentissage en plusieurs phases réside dans les apprentissages à partir d'attributs numériques (i.e. la phase d'*abstraction*). Or, les descripteurs abstraits sont conceptuellement beaucoup plus proches des mesures que ne l'est le concept de l'algorithme à appliquer (les descripteurs abstraits et les mesures décrivent les propriétés de l'objet). Dans le cas de l'apprentissage direct, l'apprentissage à partir de données numériques doit relier des concepts éloignés (mesures et algorithme), alors que dans l'apprentissage en plusieurs phases, l'apprentissage à partir de données numériques doit relier des concepts proches (mesures et attributs abstraits). L'apprentissage en plusieurs phases permet donc de transposer la tâche d'apprentissage la plus difficile (numérique) sur l'apprentissage de concepts proches.

D.4.3 Déterminer et spécifier : opération, algorithme

Principe

Afin de faciliter encore l'apprentissage, nous pouvons également abstraire la classe à apprendre (i.e. l'algorithme de généralisation cartographique), comme expliqué en D.3.3. C'est à dire que nous apprenons tout d'abord une abstraction de ces classes pour ensuite raffiner le choix effectué.

Les algorithmes réalisent certaines opérations de généralisation (simplification, harmonisation, etc., cf. A.2). Nous pouvons donc d'abord rechercher quelle opération appliquer sur un objet, puis spécifier ensuite quel algorithme utiliser pour réaliser cette opération. Cette étape de spécification s'effectue en connaissant l'opération qui a été choisie, la description abstraite de l'objet traité et la transformation qui y a été appliquée précédemment (cf. Figure 39).

Les opérations réalisées peuvent souvent être organisées hiérarchiquement. En particulier, si nous considérons l'arrêt, les algorithmes de focalisation et les algorithmes de modification comme trois grands groupes d'algorithmes, nous pouvons les hiérarchiser assez naturellement. Le plus haut niveau de la hiérarchie choisit entre arrêter ou continuer le processus. Ensuite, s'il s'agit de continuer le processus, un deuxième niveau de la hiérarchie spécifie l'action à réaliser en choisissant entre focaliser et modifier l'objet dans son ensemble. Dans ce dernier cas, un troisième niveau peut spécifier l'opération de modification en une opération de simplification, d'exagération ou d'harmonisation, etc.

Une classification hiérarchique des opérations de généralisation est parfois naturelle. Mais dans certains cas elle est difficile à définir, car une opération de bas niveau peut réaliser plusieurs opérations de plus haut niveau, ou un algorithme peut réaliser plusieurs opérations à la fois. Dans ce cas les relations entre différents niveaux d'opérations et des algorithmes peuvent être représentées sous la forme d'un treillis, comme cela est illustré sur la Figure 39.

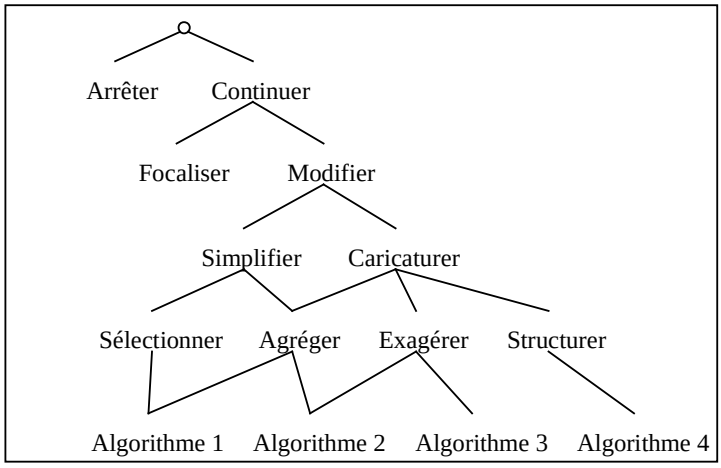


Figure 39. Un treillis de relations entre opérations et algorithmes

Un tel treillis permet de raffiner notre méthode de résolution de problème, en introduisant une inférence pour déterminer chaque niveau du treillis, comme en Figure 40. Dans cette figure nous n'avons représenté qu'un niveau d'opération. Si plusieurs niveaux sont nécessaires on peut ajouter autant d'inférences dans notre méthode de résolution de problème : une inférence pour déterminer le premier niveau du treillis en fonction de la description abstraite de l'objet, une autre pour déterminer le deuxième niveau en fonction du choix fait au premier niveau et de la description de l'objet, etc. Cette spécification progressive d'un choix est en général désigné en acquisition de connaissances sous le terme "approche déterminer et spécifier".

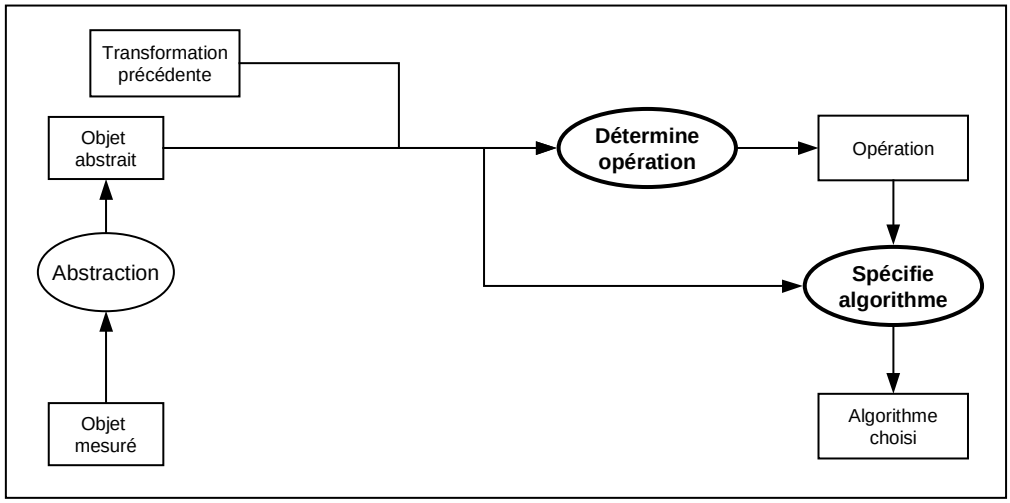


Figure 40. Deuxième raffinement: abstraire les algorithmes

Intérêt vis-à-vis de l'apprentissage

Comme pour l'abstraction des mesures, ce découpage de l'inférence de représentation en plusieurs inférences permet de transformer une tâche d'apprentissage en deux sous-tâches plus faciles. Tout d'abord, il est plus facile d'apprendre à partir de l'ensemble des exemples l'opération à réaliser que l'algorithme à utiliser, puisque les opérations sont moins nombreuses que les algorithmes. Ensuite, une fois l'opération choisie, il est plus facile de déterminer l'algorithme à utiliser. En effet le choix est à faire uniquement entre les algorithmes réalisant cette opération, moins nombreux que l'ensemble des algorithmes.

Ceci est illustré sur la Figure 41. Il nous faut, par exemple, faire le choix entre quatre algorithmes, dont les deux premiers réalisent une opération et les deux derniers une autre

opération. Apprendre directement l'algorithme à appliquer est alors un problème avec quatre valeurs possibles de classe. Mais apprendre l'opération à effectuer est un problème avec seulement deux valeurs de classe, donc plus facile. Puis, pour une opération donnée, spécifier quel algorithme utiliser est aussi un problème à deux valeurs de classe. Pour ce dernier problème, deux fois moins d'exemples sont utilisés par rapport à l'apprentissage direct : ceux qui réalisent l'opération considérée. Mais ceux-ci se situent uniquement dans une moitié de l'espace des observables : la moitié de l'espace associée à l'opération donnée. Par rapport à l'apprentissage direct, cette phase de spécification utilise une classe avec deux fois moins de valeurs et utilise deux fois moins d'exemples, mais répartis uniquement sur la moitié de l'espace des observables. Ces exemples couvrent donc en proportion autant l'espace des observables que tous les exemples couvrent l'espace entier.

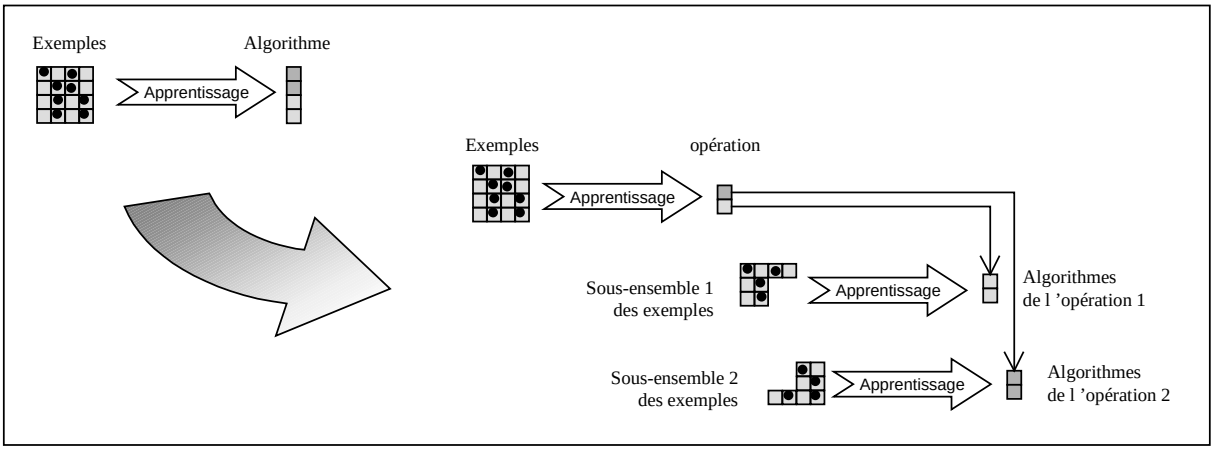


Figure 41. Reformulation de l'apprentissage grâce aux opérations

Cette approche déterminer et spécifier se justifie d'autant plus que certains algorithmes d'apprentissage traitent les problèmes à n classes en cherchant d'abord à dissocier la première classe de toutes les autres, puis ensuite la deuxième de toutes les autres sauf la première, etc. (e.g. RIPPER [Cohen 95]). Or dans notre cas, certains algorithmes de généralisation sont conceptuellement proches (ceux réalisant une même opération), et différencier un algorithme de tous les autres est difficile. Il paraît plus facile de dissocier d'abord les groupes d'algorithmes proches, puis de dissocier les algorithmes au sein de chaque groupe. Prenons un exemple plus intuitif : nous devons apprendre à différencier une truite, un saumon, un chien et un chat. Il est alors plus facile de chercher d'abord des critères permettant de distinguer les poissons des mammifères, puis ensuite de chercher des critères permettant de distinguer chaque espèce au sein de son genre (l'introduction de cet exemple dans ENIGME est détaillée dans [Thomas 96, chap.3]).

D.4.4 Couvrir et différencier : algorithmes applicables, algorithme choisi

Principe

Puisqu'il y a différentes façons de généraliser un objet, pour obtenir des résultats différents mais de qualité équivalente, plusieurs experts pourront choisir différents algorithmes à appliquer sur le même objet. De plus, si deux algorithmes fournissent des résultats proches, certains experts choisiront d'appliquer le premier algorithme alors que d'autres choisiront le deuxième. Ceci complique la tâche de rechercher quel algorithme appliquer sur un objet donné : il n'y a pas de réponse absolue, ou autrement dit en terminologie de l'apprentissage, les classes d'algorithmes ne sont pas mutuellement exclusives. Néanmoins deux experts seront

souvent d'accord sur la détermination des algorithmes *applicables* sur un objet, même si ils ne choisissent pas le même algorithme *à appliquer*.

Nous raffinons ainsi notre méthode de résolution de problème en introduisant une phase de détermination des algorithmes applicables avant de déterminer l'algorithme à appliquer (Figure 43). Cette approche, qui consiste à déterminer les solutions possibles avant d'en choisir une, est en général désignée en acquisition de connaissances sous le terme de "couvrir et différencier".

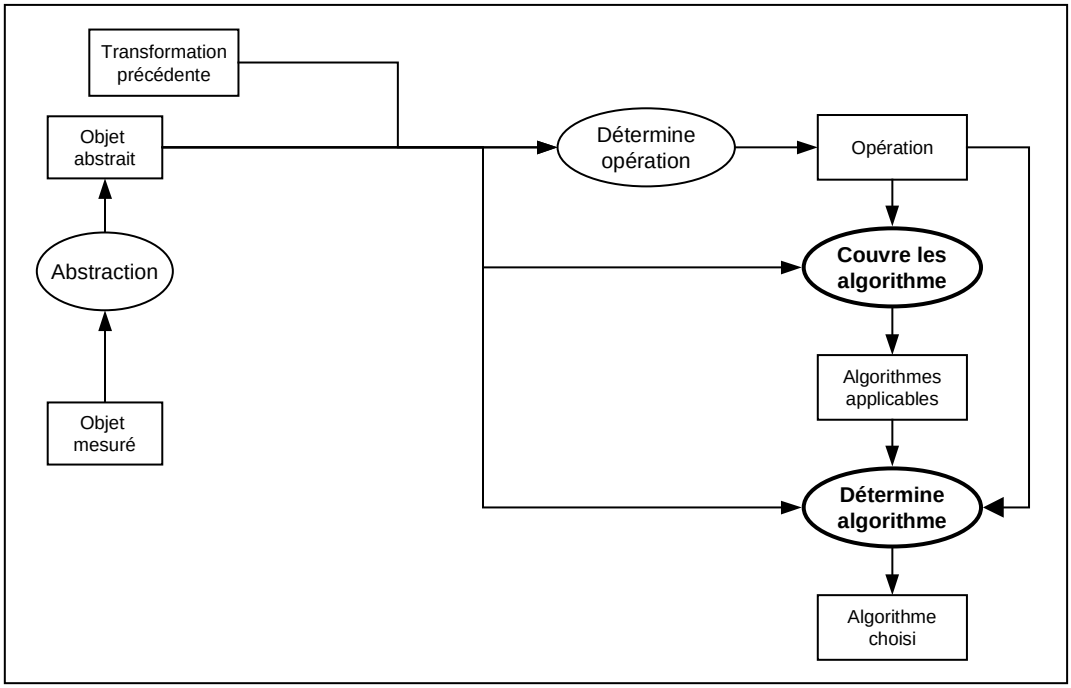


Figure 42. Troisième raffinement : déterminer les algorithmes applicables

Intérêt vis-à-vis de l'apprentissage

L'introduction des phases d'abstraction des mesures et d'abstraction des algorithmes permettait de décomposer l'apprentissage en plusieurs phases plus simples, du point de vue de l'espace des hypothèses manipulé. Ce n'est pas le cas de l'introduction d'une phase de détermination des algorithmes applicables, mais cette phase peut néanmoins simplifier l'apprentissage. Ceci est illustré sur la Figure 43 où le choix doit être fait entre trois algorithmes. La détermination de l'applicabilité des algorithmes est supposée être plus simple que le choix direct d'un algorithme, et son apprentissage doit donc produire moins d'erreurs. Cette phase permet d'enrichir les exemples en leur ajoutant la liste des algorithmes qui y sont applicables. Cette information supplémentaire augmente la taille des observables, donc élargit l'espace des hypothèses, mais est une information pertinente pour choisir un algorithme : si un algorithme n'est pas applicable, il ne peut pas être choisi comme l'algorithme à appliquer. Ainsi, sur une partie donnée de l'espace des observables, seuls les algorithmes applicables peuvent être choisis. Sur notre exemple, sur certaines parties de l'espace des observables un seul algorithme est applicable, et la phase de choix de l'algorithme peut concentrer ses efforts d'induction sur le reste de l'espace.

Plus généralement, ceci se traduit par le fait que, pour apprendre des connaissances complexes, il vaut mieux tout d'abord apprendre facilement des connaissances plus simples, et s'appuyer au besoin sur ces connaissances simples pour ensuite apprendre plus facilement les connaissances complexes.

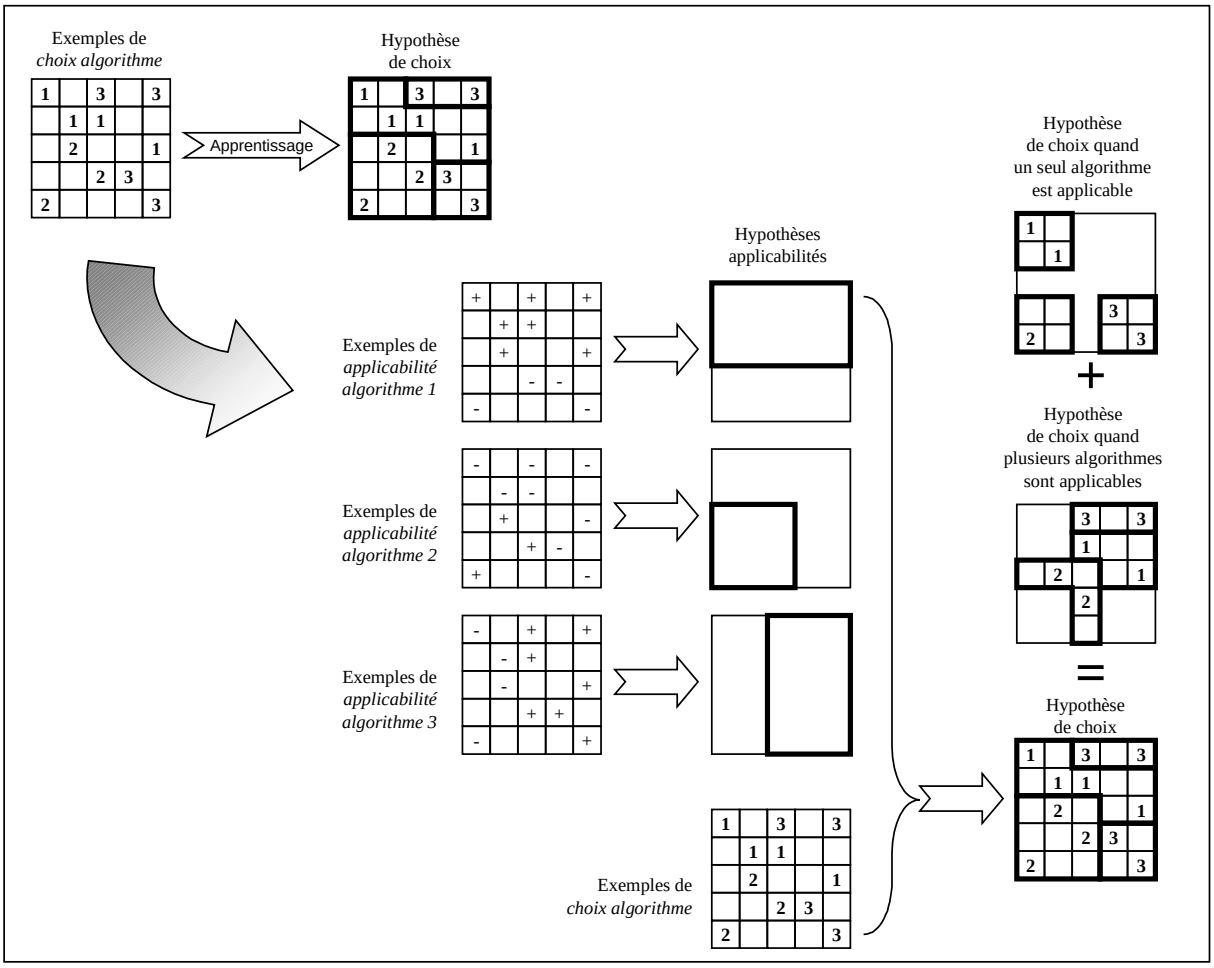


Figure 43. Reformulation de l'apprentissage grâce aux applicabilités des algorithmes

D.4.5 Paramétrage des algorithmes

Nous avons jusqu'à présent éludé le problème du paramétrage des algorithmes de généralisation. Non pas que ce paramétrage soit un problème facile, mais la plupart des algorithmes que nous utilisons n'ont pas de paramètre. Plus précisément une grande partie des algorithmes que nous utilisons ont un seul paramètre, directement relié aux spécifications de la carte (largeur de symbole, taille minimale perceptible…), et dans ce cas la question du choix du paramètre ne se pose pas. Néanmoins certains algorithmes ont un paramètre qu'il faut déterminer non seulement en fonction des spécifications de la carte, mais aussi en fonction de la forme de l'objet traité.

Il y a deux façons de déterminer ce paramétrage : soit on choisit un algorithme puis ensuite son paramètre, soit on considère chaque algorithme associé à une valeur paramétrique comme un algorithme particulier. Si on utilise la deuxième solution, alors notre méthode de résolution de problème définie ci-dessus en Figure 42 est directement utilisable, mais le nombre d'algorithmes possibles se trouve fortement multiplié. Si on utilise la première solution, il nous faut raffiner encore la méthode en introduisant une phase de détermination des paramètres (cf. Figure 44).

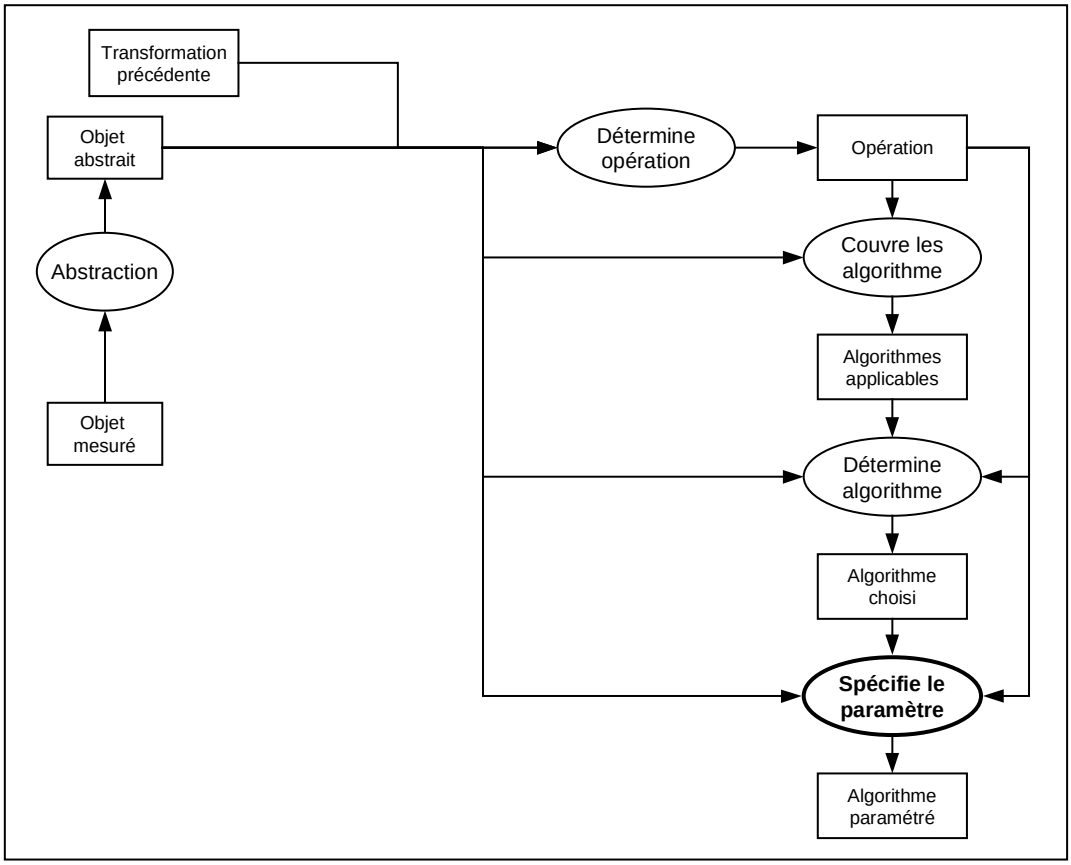


Figure 44. Quatrième raffinement : spécifier le paramètre

D.5 Bilan : processus d'apprentissage

D.5.1 Méthode de définition du processus d'apprentissage

On peut définir directement un processus d'apprentissage en s'appuyant sur la méthode de résolution de problème définie ci-dessus. Nous résumons les grands principes de cette méthode et ses conséquences sur l'apprentissage. Pour cela, nous précisons les tâches à résoudre, la fonction cible, une représentation de cette fonction, la source d'expérience, et enfin la méthode d'apprentissage (cf. C.2).

Tâches à résoudre

Les tâches à résoudre dans le cadre de nos travaux désignent la généralisation cartographique des objets d'un type donné. Pour chaque type d'objet considéré il faut réitérer le processus d'apprentissage défini ci-dessous.

Fonction cible

La fonction cible à apprendre relie l'état d'un objet géographique ainsi que la dernière transformation qu'il a subie au choix de la prochaine transformation à lui appliquer. Cette transformation peut être une opération d'arrêt, de focalisation ou de modification géométrique de l'objet. Elle est spécifiée par un algorithme paramétré permettant de la réaliser.

Un moteur enchaînant les transformations doit être défini pour réaliser la tâche de généralisation. Ce moteur peut prendre la forme de celui de la Figure 31 (p. 95). Selon les outils disponibles et les objets traités, il peut être plus complexe, incluant par exemple une phase de validation des résultats de chaque étape ou une phase de validation finale du résultat. Il peut aussi être plus simple si aucune opération de focalisation n'est nécessaire sur le type d'objet traité, ou si on décide de ne pas utiliser la notion d'algorithme applicable sur un objet (afin de simplifier le recueil des exemples).

Représentation de la fonction cible

Une étape nécessaire à la définition de la fonction cible est la décomposition de la méthode de résolution de problème en plusieurs inférences spécialisées. On peut identifier les inférences générales suivantes :

- Abstraire les mesures décrivant un objet en un ensemble d'attributs symboliques.
- Déterminer l'opération à réaliser. Si les opérations sont organisées en plusieurs niveaux, une inférence peut être réalisée pour déterminer chaque niveau.
- Déterminer les algorithmes applicables.
- Déterminer l'algorithme à appliquer.
- Déterminer les paramètres.

Une autre étape, réalisée parallèlement à la première, est la détermination du langage utilisé en entrée et sortie de chaque étape. Il faut définir :

- Le langage concret, soit un ensemble de mesures décrivant l'état d'un objet. L'apprentissage sera facilité si cet ensemble est de taille fixe pour tous les objets du type considéré. Ces mesures peuvent décrire les propriétés intrinsèques de l'objet (e.g. sa longueur, son degré de lisibilité), décrire son environnement (e.g. la distance à son plus proche voisin), décrire des écarts de propriétés entre l'objet géographique initial et son état cartographique courant (e.g. un écart de position).
- Le langage abstrait, soit un ensemble d'attributs symboliques décrivant l'état d'un objet, ainsi que les valeurs que chaque attribut peut prendre. Ces attributs doivent être choisis parallèlement aux mesures précédentes : chacun des attributs abstraits doit pouvoir être déterminé à partir des mesures, et il est inutile d'utiliser une mesure indépendante des attributs abstraits. L'apprentissage sera facilité par la connaissance des mesures pertinentes vis-à-vis d'un attribut abstrait.
- Une organisation en treillis des algorithmes et des opérations qu'ils réalisent. Chaque opération doit pouvoir être réalisée par au moins un algorithme. Un algorithme peut réaliser plusieurs opérations à la fois.
- Une spécification des paramètres des algorithmes. Plus ces paramètres sont reliés aux spécifications de la carte, plus il sera facile de les déterminer.

Source d'expérience

Si les connaissances nécessaires à la réalisation de certaines inférences élémentaires de la fonction cible sont connues, il est inutile de les apprendre. Sinon, on peut les apprendre à partir d'exemples. Les exemples instanciés doivent contenir :

- Les valeurs des mesures sur l'objet, qui peuvent être calculées automatiquement.

- Les valeurs des attributs abstraits, qui doivent être déterminées interactivement par un expert du domaine.
- La transformation précédente effectuée sur l'objet, qui peut être décrite sous la forme d'un ensemble d'attributs : l'opération effectuée (éventuellement à différents niveaux), l'algorithme utilisé, et son paramètre. Ces attributs peuvent être stockés automatiquement par traçage des actions réalisées.
- L'opération à effectuer, représentée par plusieurs attributs si elle est organisée selon plusieurs niveaux hiérarchiques. Cette opération est choisie interactivement par un expert du domaine. L'existence d'un treillis à plusieurs niveaux sur les opérations doit guider le recueil des exemples : si l'expert choisit de réaliser une opération de haut niveau, il est inutile de lui demander de choisir entre toutes les opérations de bas niveau possibles. Il faut restreindre le choix aux opérations de bas niveau compatibles avec celle de haut niveau choisie.
- Les algorithmes applicables sur l'objet. Pour recueillir cette information, il faut demander à un expert de déterminer l'applicabilité de chaque algorithme pouvant réaliser l'opération choisie. Si l'expert possède une grande expertise des algorithmes, il peut déterminer cela en analysant uniquement l'objet à traiter. Il faut néanmoins lui fournir des outils permettant de tester et d'annuler les effets de chaque algorithme, en cas de doute sur leur applicabilité. Si l'expert possède une grande expertise cartographique mais peu d'expertise des algorithmes, on peut lui présenter simultanément les états de l'objet après application de chaque algorithme pouvant réaliser l'opération choisie. Il déterminera alors l'applicabilité des algorithmes en fonction des résultats de leur application.
- L'algorithme choisi, ainsi que ses paramètres. Il peut aussi être choisi par l'expert au regard des résultats des algorithmes. Ces informations peuvent être stockées par traçage des actions réalisées.

Méthode d'apprentissage

On peut ensuite réaliser l'apprentissage de chaque étape. Pour cela il faut choisir un algorithme d'apprentissage pour chaque inférence. Si l'on sait que certains algorithmes d'apprentissage sont plus adaptés que d'autres à certaines inférences, on peut choisir différents algorithmes pour apprendre les différentes inférences. Néanmoins, à notre connaissance, peu de règles existent pour choisir *a priori* un algorithme d'apprentissage plutôt qu'un autre pour un problème donné (cf. C.3.4). On peut toutefois fonder ce choix sur des critères de capacité à manipuler des exemples bruités, des attributs manquants, des attributs numériques ou symboliques. Cependant, utiliser différents algorithmes manipulant différents langages de représentation des hypothèses peut nuire à la lisibilité de l'ensemble des hypothèses apprises pour un expert du domaine.

Pour apprendre chaque inférence, il faut extraire des exemples les observables utiles ainsi que la classe à déterminer dans le cadre de l'inférence. On peut alors réaliser un apprentissage supervisé pour chacun des rôles en sortie de l'inférence (e.g. pour chaque attribut abstrait en sortie de l'inférence d'abstraction).

D.5.2 Intérêt de l'approche

Notre approche consiste à découper l'apprentissage en une série d'apprentissages élémentaires, en introduisant en particulier des inférences d'abstraction des mesures décrivant un objet et d'abstraction de l'algorithme à choisir.

Coût de l'approche

Cette approche par découpage a un coût non négligeable : il faut définir précisément la méthode de résolution de problème, définir le langage des rôles intermédiaires introduits (les attributs abstraits et les opérations), et le recueil des exemples y est plus difficile que pour un apprentissage direct puisqu'il faut instancier les rôles intermédiaires de la méthode de résolution de problème (e.g. les attributs abstraits). Ces limites ne sont pas inhérentes à notre problème particulier, mais à une approche intégrant l'apprentissage symbolique et l'acquisition de connaissances basée sur les modèles [Thomas 96, p.142].

Compréhensibilité du système

En contrepartie, cette approche permet d'obtenir des règles mieux organisées et plus indépendantes, et de ce fait plus compréhensibles que celles obtenues par apprentissage direct. Les règles sont mieux organisées car chacune est associée à une inférence particulière de la méthode de résolution de problème. Elles sont ainsi plus indépendantes. Clancey [83 ; 85] a montré que les règles d'un système expert ne sont pas aussi indépendantes qu'elles le semblent au premier abord. Grâce à l'approche proposée, les règles associées à une inférence sont au moins indépendantes des règles associées aux autres inférences. Mieux encore, une inférence peut contenir plusieurs bases de règles indépendantes, une par rôle en sortie de cette inférence. Par exemple l'inférence d'abstraction des mesures contient une base de règles pour chaque attribut abstrait à déterminer.

Efficacité du système

Par ailleurs, la décomposition du problème doit permettre d'obtenir des règles plus efficaces qu'un apprentissage direct, puisque cette décomposition est une façon de contraindre l'apprentissage par des connaissances du domaine (cf. partie C.4). Ceci est particulièrement le cas dans notre approche, grâce aux inférences d'abstraction qui permettent de réduire la taille de l'espace des hypothèses manipulés lors de l'apprentissage.

Maintenance du système

L'organisation des connaissances du système en un ensemble de bases de règles indépendantes et organisées apporte de nombreux avantages au niveau de la maintenance du système. En effet, si les règles apprises sont inefficaces dans certaines configurations, l'analyse interactive du raisonnement effectué pour parvenir à une solution non correcte permet de détecter aisément quelle inférence du raisonnement a été inefficace (détermination des abstractions, des opérations, des applicabilités, ou de l'algorithme). Ainsi, si cela est possible, on peut corriger manuellement les règles de cette inférence, sans risquer d'influencer les règles associées à une autre inférence. S'il n'est pas possible de corriger manuellement les erreurs, on peut recueillir des exemples supplémentaires contenant uniquement les valeurs des rôles liés à l'inférence défectueuse, et réaliser un nouvel apprentissage à partir de ces exemples pour modifier les règles. Par exemple, si l'abstraction des mesures est défectueuse pour apprendre un des attributs abstraits, il n'est nécessaire de recueillir que les mesures supposées pertinentes pour cet attribut ainsi que la valeur de cet attribut sur un ensemble de nouveaux exemples pour affiner les règles apprises.

Evolution du système

L'organisation des règles apprises présente de nombreux intérêts pour l'évolution du système, en particulier parce que les relations entre les mesures et le choix de l'algorithme passent par

l'intermédiaire des abstractions. Les règles utilisant les mesures et les règles choisissant l'algorithme sont donc relativement indépendantes.

Tout d'abord, si une nouvelle mesure est introduite pour qualifier un objet, celle-ci peut aisément être introduite dans le système. Il suffit d'identifier les caractères abstraits vis-à-vis desquels la mesure peut-être pertinente, de calculer cette mesure sur l'ensemble des exemples ayant servi à la création des règles, puis de relancer l'apprentissage de chacun des caractères abstraits concernés. Aucun recueil interactif d'exemples n'est donc nécessaire, à condition d'avoir conservé les exemples d'origine. Si ces exemples n'ont pas été conservés, il faut effectuer une nouvelle phase de recueil d'exemples. On n'y recueillera pour chaque objet que la valeur des caractères abstraits concernés par la nouvelle mesure, et on calculera automatiquement l'ensemble des mesures descriptives de l'objet.

Ensuite, si un nouvel algorithme est introduit, celui-ci peut aussi être introduit dans le système, moins facilement toutefois qu'une nouvelle mesure. Deux cas peuvent se présenter : soit l'algorithme réalise une ou plusieurs opérations déjà prises en compte par le système, soit il réalise une nouvelle opération.

Le premier cas, où l'algorithme réalise des opérations déjà considérées, nécessite l'intervention de l'expert pour modifier les exemples originels. Il faut présenter à un expert tous les exemples où ces opérations ont été choisies, lui demander sur quels exemples l'algorithme est applicable, et dans quels cas l'algorithme doit être choisi à la place de celui qui avait été choisi dans les exemples d'origine. Un nouveau rôle est alors introduit dans le système : l'applicabilité du nouvel algorithme. Une nouvelle base de règles est alors apprise pour déterminer l'applicabilité du nouvel algorithme en fonction des attributs abstraits, puis la base de règles choisissant l'algorithme à appliquer est réapprise.

Le second cas, où l'algorithme réalise une nouvelle opération, nécessite aussi l'intervention de l'expert, mais dans une moindre mesure. Il faut considérer chacun des exemples originels puis déterminer sur quels exemples la nouvelle opération doit être effectuée au lieu de celle choisie originellement. Le nouvel algorithme est alors considéré applicable et choisi dans tous ces cas (puisqu'il est le seul algorithme réalisant cette opération). La base de règles déterminant le choix de l'opération est alors réapprise. La base de règles déterminant l'applicabilité des algorithmes pour cette opération est alors très simple : elle contient une seule règle rendant applicable dans tous les cas le nouvel algorithme. De même une seule nouvelle règle est ajoutée à la base de règle régissant le choix de l'algorithme : "Si *Opération = Nouvelle Opération* Alors *Choix = Nouvel algorithme*".

Enfin, si un algorithme est retiré du système, il faut considérer chaque exemple où cet algorithme a été choisi. Il faut ensuite demander à l'expert quel autre algorithme appliquer, parmi ceux applicables dans le cadre de l'opération choisie pour l'exemple. La base de règles régissant le choix de l'algorithme est alors réapprise. De même si une mesure est retirée du système, il suffit de réapprendre chacune des inférences d'abstraction dans lesquelles elle intervient.

L'évolution du système, par introduction de nouveaux algorithmes ou de nouvelles mesures, peut être néanmoins rendue difficile si les règles ont été corrigées interactivement. En effet si on réapprend certaines bases de règles, les corrections qui y ont été faites sont perdues. Pour

que le système soit réellement évolutif il faut donc conserver une trace des corrections interactives réalisées pour proposer à nouveau ces corrections en cas de nouvel apprentissage de bases de règles.

■

E EXPERIMENTATION DE L'APPRENTISSAGE SUR LES ROUTES

◆ Dans ce chapitre nous appliquons l'approche définie dans le chapitre précédent au cas de la généralisation cartographique des routes. Nous spécifions les connaissances de ce domaine nécessaires à cette approche : les mesures, le langage abstrait, les opérations et les algorithmes manipulés dans la méthode de résolution de problème. Nous décrivons ensuite le recueil des exemples nécessaires à l'apprentissage de chaque inférence du processus.

◆ Nous commentons les règles apprises et comparons le processus utilisant ces règles, d'une part au processus GALBE décrit au chapitre B, et d'autre part à un processus utilisant des règles apprises sans décomposer la méthode de résolution de problème. Ceci nous permet de mettre en évidence l'intérêt de l'approche, du point de vue de la compréhensibilité et de l'efficacité des règles apprises.

E.1 Présentation des tests

E.1.1 Objets étudiés

La mise au point de notre approche de l'apprentissage par décomposition des inférences à apprendre a été guidée par les résultats de tests empiriques sur la généralisation des bâtiments isolés[20]. Ces tests sont présentés en annexe VI et dans [Mustière, Zucker et Saitta 2000b ; Zucker, Mustière et Saitta 2000]. Nous ne détaillons pas dans ce chapitre les résultats sur les bâtiments pour deux raisons principales :

- La première raison est pratique. Ces résultats sont trop incomplets pour pouvoir être utilisés en pratique dans un processus complet de généralisation. Nous aimerions modifier la forme des exemples qui nous ont aidé à mettre au point l'approche, et aussi appliquer les hypothèses apprises sur de nouveaux bâtiments à généraliser. Or, les exemples sur les bâtiments nous ont été fournis et nous connaissons mal la plate-forme utilisée pour les créer. Ainsi nous ne disposons pas de façon immédiate des outils et des données qui seraient utiles pour effectuer tous les tests que nous aimerions réaliser.
- La deuxième raison est méthodologique. Evaluer des hypothèses apprises sur les données qui ont servi à mettre au point l'apprentissage risque d'introduire un biais optimiste sur l'évaluation (cf. C.5.2). Pour restreindre autant que possible ce biais nous préférons donc appliquer et évaluer l'approche dans le cadre d'un autre domaine.

Dans ce chapitre nous concentrons notre étude sur un autre cas : la généralisation cartographique des routes. Ceci nous permet d'utiliser les travaux sur ce domaine qui ont été réalisés auparavant au laboratoire COGIT, ainsi que ceux que nous avons réalisés lors de la mise au point du processus GALBE (chapitre B). De plus, tester l'approche proposée sur les routes nous permet d'en comparer directement les résultats avec ceux de GALBE. Nous pouvons ainsi comparer la mise au point empirique à l'apprentissage automatique de processus aux mêmes buts.

Dans cette partie, nous définissons tout d'abord les différents rôles manipulés dans le cas des routes : le langage abstrait, les mesures utilisées, les opérations et algorithmes manipulés. Ceci nous permet de préciser la méthode de résolution de problème adaptée aux routes, sur laquelle s'appuient le recueil des exemples et les apprentissages réalisés. Par la suite, nous présentons et commentons les résultats obtenus (E.2 et E.3), en les comparant en particulier à un apprentissage direct (E.4).

E.1.2 Langage abstrait utilisé

Choix des attributs

Selon l'approche que nous proposons au chapitre D, il nous faut en premier lieu définir un langage abstrait de description des routes. Une analyse des règles de cartographie, telles que celles décrites dans la partie B.2 (p.53), permet d'identifier un ensemble de concepts

[20] Ces tests portaient sur la généralisation de la BDTopo (base de données géographique de l'IGN de précision métrique et d'échelle de l'ordre du 1:15.000) pour réaliser des cartes au 1:50.000

pertinents pour décrire une route. On peut identifier en particulier les notions relatives à la forme de la route, à sa lisibilité graphique et à son environnement. Ces notions se recoupent car la lisibilité graphique est liée à la forme et à l'environnement.

Pour décrire la forme d'une route, le terme de sinuosité est souvent utilisé, sans véritable définition formelle. Pour formaliser ce concept dans un but de généralisation cartographique, Plazanet [1996, pp.69,73] distingue les notions de sinuosité et de complexité (Figure 45). La sinuosité est définie relativement "aux détours ou plis que fait une ligne, à opposer à la rectitude". La complexité est définie comme "l'imbrication d'ondulations dans les ondulations de la ligne", notion relativement liée à la notion de dimension fractale. D'autres notions peuvent aussi être identifiées pour décrire la forme d'une route : sa taille, sa granularité (qui fait référence à la taille des plus petits détails de forme, cf. Figure 46), et la forme générale (pour distinguer les lignes droites des virages seuls et des séries de virages).

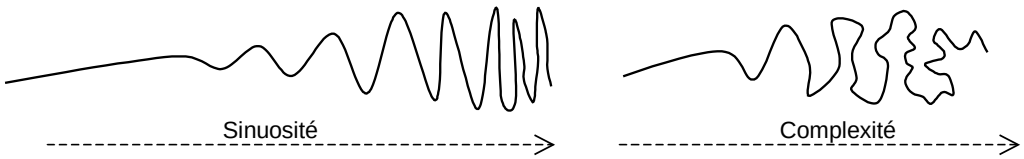


Figure 45. Sinuosité et complexité, d'après [Plazanet 96]

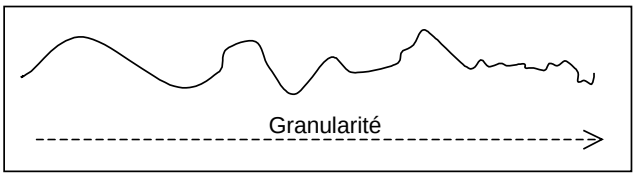


Figure 46. Granularité

Pour décrire le niveau de lisibilité d'une route, on peut identifier les notions d'empâtement (défini en détail dans la partie B.3.2), de granularité, et de conflit externe (conflit de superposition des symboles d'une route et de ses voisines). Enfin, la densité de l'environnement d'une route apparaît également comme un concept guidant la généralisation.

Nous décidons ainsi de décrire une route à l'aide de huit attributs abstraits : la taille, la sinuosité, la complexité, la forme générale, la granularité, l'empâtement, les conflits externes, et la densité de l'environnement.

Choix des valeurs des attributs

Pour définir plus précisément le langage abstrait de description des routes, il nous faut ensuite définir les valeurs possibles de chacun des attributs descriptifs choisis. Il est difficile de définir ces valeurs sans ambiguïté : quelle est la limite entre une route peu sinueuse et très sinueuse ? quelle est la limite entre une route faiblement et fortement empâtée ? etc. Différentes personnes pourront classer différemment les objets analysés, et une même personne pourra les classer différemment au cours du temps. Heureusement, les objets traités sont des objets géographiques et ne sont pas de forme quelconque, ce qui aide à leur classification. Par exemple, la notion de sinuosité d'une route est moins ambiguë que la notion de sinuosité d'une ligne quelconque. On distingue trois grands degrés de sinuosité des routes : les lignes droites ou très peu courbées construites pour se déplacer facilement (typiquement les formes des voies rapides), les routes peu sinueuses construites pour éviter les obstacles ou suivre le relief (typiquement les formes des petites routes de campagne), ou les routes très sinueuses de montagne construites pour gravir le relief.

Pour définir avec le moins d'ambiguïté possible le langage abstrait utilisé, nous l'avons défini par essais successifs en essayant de classer interactivement un ensemble de routes. De plus, nous avons essayé de nous donner des règles générales pour décider comment classer chaque route, à l'instar de l'exemple de la sinuosité décrit ci-dessus.

Ceci nous permet de définir un ensemble d'attributs symboliques pour décrire une route, ainsi que les valeurs possibles de ces attributs (cf. Tableau 2). Notons que ces descripteurs se situent au niveau de la carte et non du monde géographique : nous décrivons les objets d'un point de vue cartographique.

attribut	*valeurs possibles*
Taille	petit | moyen | grand
Sinuosité	nulle | virages peu sinueux | épingles à cheveux | hétérogène
Complexité	zéro niveau | un niveau | plusieurs niveaux
Granularité	faible | moyenne | forte
Forme générale	ligne droite | virage | courte série de virages | longue série de virages
Empâtement	nul | faible | fort | hétérogène
Environnement	libre | dense
Conflit externe	nul | peu | nombreux

Tableau 2. Descripteurs abstraits d'une route cartographiée

E.1.3 Mesures utilisées

Pour mesurer une route, nous avons utilisé des mesures rendant compte, au moins en partie, des différents descripteurs abstraits définis ci-dessus. Ces mesures sont listées dans le Tableau 3. Elles sont pour la plupart inspirées des travaux de Plazanet [1996] pour ce qui concerne les notions de forme d'une ligne, de ceux de Nickerson [1988] pour les notions de conflit externe (i.e de superposition des symboles de plusieurs lignes), et de notre mesure de l'empâtement (cf. B.3.2). Afin que les règles apprises s'adaptent autant que possible à différentes symbolisations et à différentes échelles cartographiques, les mesures sont définies dans l'espace de la carte et non dans l'espace géographique, à l'instar des attributs abstraits. Ainsi nous ne mesurons pas la longueur en kilomètres d'une route, mais sa longueur en millimètres sur le papier, et les mesures de conflit graphique sont dépendantes des largeurs de symbole utilisées.

De plus, chaque caractère abstrait est relié à l'ensemble des mesures potentiellement pertinentes pour le décrire (Tableau 4). Un des rôles de l'apprentissage est de déterminer quelles mesures sont réellement pertinentes.

mesure	*unité*	*valeurs possibles*	*Description succincte*
Longueur	mm	]0,∞[	Longueur de la ligne
Base sur longueur		[0,1]	Distance entre les extrémités de la ligne divisée par sa longueur
Nombre virages		[1, ∞[	Nombre de virages de la ligne (définis selon les points d'inflexion)
Nombre grands virages		[1, ∞[	Nombre de virages de la ligne après lissage (de force fonction de la largeur du symbole), d'après [Plazanet 96]
Résistance lissage		]0,1]	Nombre de grands virages / nombre de virages, d'après [Plazanet 96]
Fréquence des virages	mm^{-1}	[0,∞[	Nombre de virages divisé par la longueur de la ligne.
Homogénéité de complexité	mm	[0,∞[	Ecart type des largeurs des virages, d'après [Plazanet 96]
Taille max virages	mm	[0,∞[	Hauteur du plus grand virage de la ligne, d'après [Plazanet 96]
Taille min virages	mm	[0,∞[	Hauteur du plus petit virage de la ligne, d'après [Plazanet 96]
Taille moyenne virages	mm	[0,∞[	Moyenne des hauteurs des virages de la ligne, d'après [Plazanet 96]
Taille virages moyenne sur max		[0,1]	Taille moyenne virages / taille max virages, d'après [Plazanet 96]
Force empâtement		[0,1[	1 –surface du symbole / (longueur de la ligne × largeur du symbole). Cette mesure compare la surface du symbole avec la surface de la même ligne "étirée" et donc non empâtée
Type empâtement		aucun monolatéral bilatéral	Aucun si la ligne n'est pas empâtée, monolatéral si la ligne est empâtée d'un seul côté, bilatéral sinon (cf. chap. B).
Longueur empâtement	mm	[0,∞[	Longueur de la ligne en empâtement
Pourcentage empâtement		[0,1]	Longueur d'empâtement / longueur de la ligne
Homogénéité empâtement		[1/2,1]	Max(h,1-h) avec h = pourcentage d'empâtement
Force du bruit	mm	[0,∞[	Surface de la fermeture mathématique (de la taille du symbole) de la ligne divisée par la longueur de la ligne, inspirée de [Perkal 58].
Granularité des virages	$mm^{½}$	]0,∞[	Mesure fonction des hauteurs et longueurs des virages (d'après X. Barillot, communication personnelle)
Nombre d'arcs proches		ℕ	Nombre de lignes dont le symbole intersecte celui de la ligne
Nombre d'arcs en conflit		ℕ	Nombre d'arcs en conflit de proximité avec la ligne. La notion de conflit est définie au sens de [Nickerson 88].
Surface de conflit proche	mm^2	[0,∞[	Somme des surfaces d'intersection entre le symbole de la ligne considérée et le symbole des lignes en conflit avec cette ligne, au sens de [Nickerson 88].
Surface de conflit loin	mm^2	[0,∞[	Somme des surfaces d'intersection entre le symbole de la ligne et le symbole des lignes en conflit avec cette ligne, en doublant la taille des symboles des lignes, au sens de [Nickerson 88].
Max conflit externe		[1,10]	Plus fort conflit de la ligne avec d'autres lignes, au sens de [Nickerson 88]
Nb arcs proches par longueur	mm^{-1}	[0,∞[	Nombre d'arcs proches divisé par la longueur de la ligne
Nb arcs proches par largeur	mm^{-1}	[0,∞[	Nombre d'arcs proches divisé par la largeur du symbole

Tableau 3. Mesures utilisées pour décrire une route

Caractère abstrait	*Mesures potentiellement utiles*
Taille	Longueur
Sinuosité	base sur longueur, nombre de virages, fréquence des virage, taille moyenne des virages, granularité des virages, taille virages moyenne sur max, pourcentage empâtement
Complexité	longueur, base sur longueur, nombre de virages, nombre de grands virages, taille moyenne des virage, taille max des virages, taille virages moyenne sur max, résistance lissage, homogénéité de complexité
Granularité	résistance lissage, taille moyenne des virages, taille min des virages, force empâtement, force du bruit, granularité des virages, taille virages moyenne sur max
Forme générale	longueur, base sur longueur, nombre de virages, nombre de grands virages, type empâtement
Empâtement	force empâtement, type empâtement, longueur empâtement, pourcentage empâtement, homogénéité empâtement,
Environnement	nombre d'arcs proches, nombre d'arcs loin, nombre d'arcs en conflit, surface conflit proche, surface conflit loin, nb arcs proches par longueur, nb arcs proches par largeur
Conflit Externe	nombre d'arcs proches, nombre d'arcs loin, nombre d'arcs en conflit, surface conflit proche, surface conflit loin, max conflits externes

Tableau 4. Mesures potentiellement utiles pour décrire un caractère des routes

E.1.4 Opérations et algorithmes géométriques utilisés

Après avoir défini comment décrire une route par des mesures et des attributs abstraits, nous définissons des algorithmes potentiellement utiles pour leur traitement, et organisons ces algorithmes selon les opérations qu'ils réalisent. Le treillis des relations entre les *opérations* et *algorithmes* utilisés pour les routes, ainsi que leurs *paramètres*, est résumé en Figure 47 (les algorithmes sont décrits en détail dans les annexes I et II)

Un premier niveau d'organisation (que nous appelons *Action)* des opérations réalisées consiste à définir si le traitement doit s'arrêter ou continuer sur un objet donné. Si l'on choisit de continuer, comme le montrent les résultats du processus GALBE, il est parfois utile de découper une route pour focaliser sur différentes parties et y appliquer différents traitements. On identifie donc deux grands *types d'opérations* pour les routes qui constituent le deuxième niveau d'organisation des opérations : focaliser ou modifier la géométrie.

Le processus GALBE montre aussi la pertinence du découpage selon l'empâtement, pour focaliser nous pouvons utiliser l'algorithme de focalisation selon l'empâtement décrit dans la partie B.3.2.3. Plazanet [1996, p.135] propose par ailleurs un algorithme pour découper selon un critère d'homogénéité de sinuosité. Nous l'avons introduit un premier temps dans nos tests, mais nous l'avons ensuite éliminé à cause de la difficulté que nous avions à déterminer ses paramètres en interactif.

Par ailleurs, en cas de choix de modification de la géométrie, trois *opérations* sont communément réalisées sur les routes : la simplification pour éliminer les petits détails, le déplacement pour écarter les routes trop proches, et l'exagération pour mettre en valeur certaines parties. Pour réaliser ces opérations, nous utilisons les mêmes *algorithmes* que ceux utilisés dans GALBE, plus un algorithme de déplacement : l'algorithme de *Nickerson* [Nickerson 88]. Notons que dans un premier temps nous avions introduit l'algorithme de *Lowe* [Lowe 88] en tant qu'algorithme de mise en valeurs de virages peu sinueux. Mais là encore notre difficulté à le paramètrer et sa trop grande sensibilité au bruit nous a fait renoncer à utiliser cet algorithme.

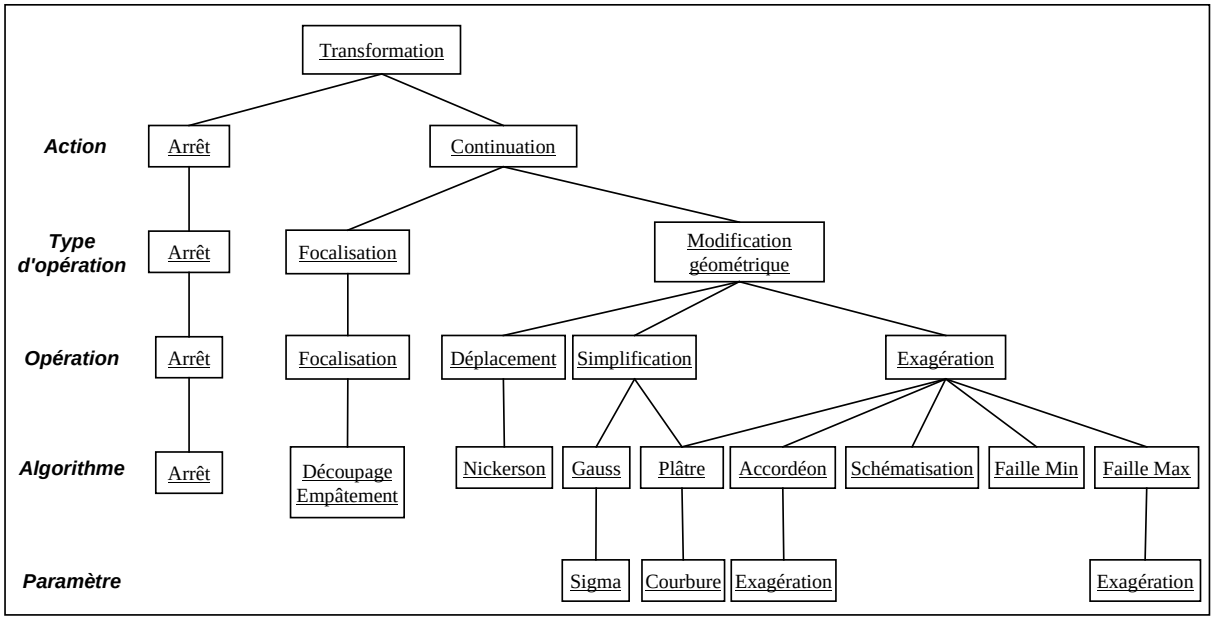


Figure 47. Organisation des opérations et algorithmes utilisés pour les routes

Ce treillis est un choix arbitraire. Une autre organisation possible des relations entre algorithmes et opérations aurait été de considérer la focalisation comme une opération

d'exagération. C'est-à-dire que si le processus doit continuer on choisit d'abord entre les opérations de déplacement, de simplification et d'exagération, et qu'en cas d'exagération on choisit ensuite entre modifier la géométrie ou focaliser. Cependant, la classification que nous proposons nous paraît plus naturelle, peut-être parce que nous sommes influencés par le processus GALBE qui met en avant les actions de focalisation.

E.1.5 Méthode de résolution de problème choisie

Les différents niveaux du treillis des opérations et algorithmes décrits ci-dessus nous permettent de définir précisément la tâche de détermination des opérations, en la décomposant en plusieurs inférences. La méthode de résolution de problème qui en découle dans le cas des routes est décrite en Figure 48.

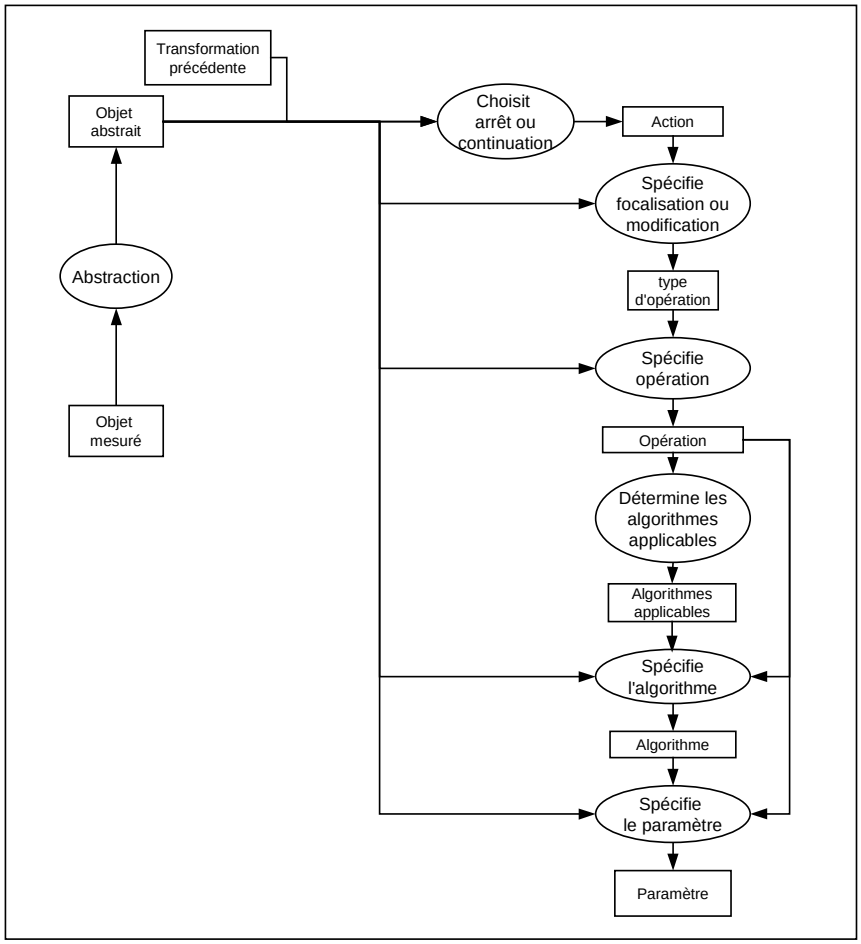


Figure 48. Méthode de résolution de problème adaptée aux routes

E.1.6 Recueil des exemples

Les exemples des routes ont été recueillis avec la plate-forme PlaGe [Lecordix, Plazanet et Lagrange 97]. Nous avons recueilli les exemples sur une petite zone de montagne afin de manipuler des routes de différentes formes. Cette zone, présentée en Figure 49, est située dans les Alpes-Maritimes et contient 22 routes (une route = un arc du graphe routier initial). Les exemples ont été recueillis dans le cadre de la généralisation de la BDCarto pour effectuer une généralisation cartographique à l'échelle du 1:250.000 (ce cadre est aussi celui dans lequel GALBE a été défini).

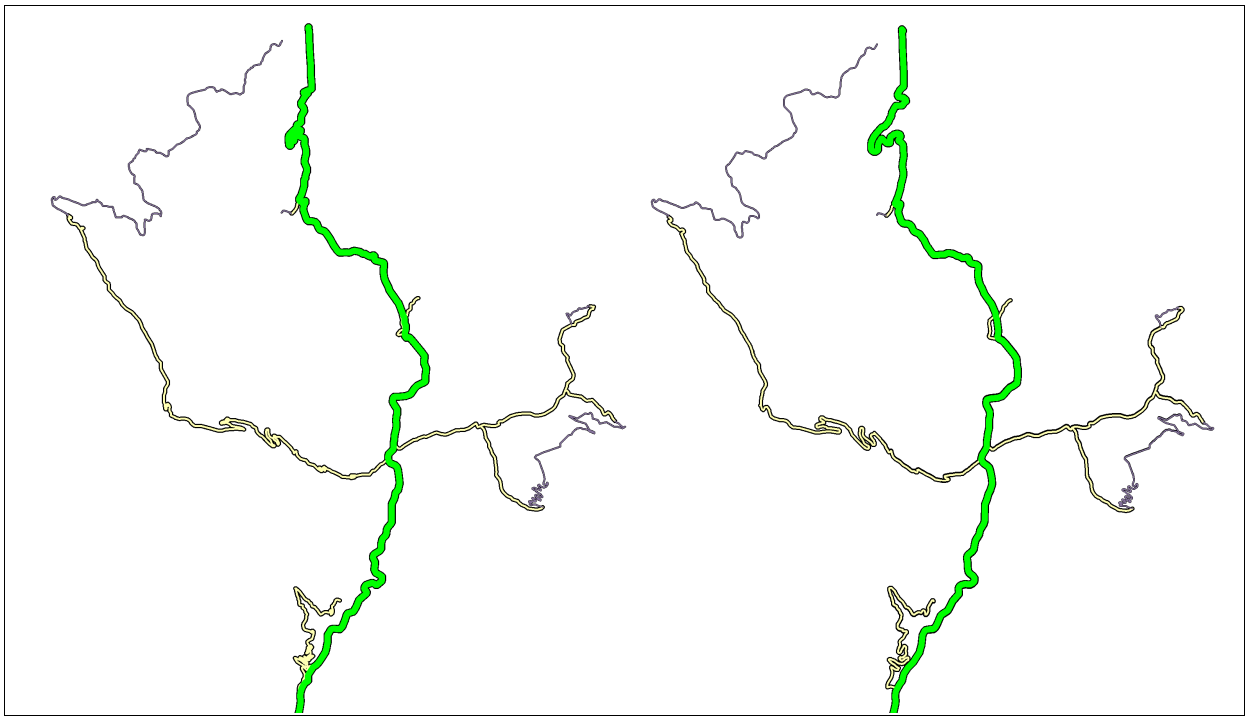


Figure 49. Zone d'exemples avant et après généralisation interactive.

Nous avons traité les routes de cette zone une à une. Tout d'abord, les mesures définies ci-dessus ont été calculées sur les objets avant chaque transformation. Puis nous avons interactivement décrit les objets par les attributs abstraits définis. Nous avons ensuite interactivement choisi l'action à réaliser (arrêter ou continuer), puis au besoin le type d'opération (focaliser ou modifier) et l'opération (simplifier, exagérer, déplacer). Ensuite, en testant les résultats des différents algorithmes potentiellement utiles pour réaliser l'opération choisie, nous avons déterminé ceux qui étaient applicables sur l'objet donné. Enfin, nous avons choisi un des algorithmes applicables et éventuellement déterminé par essai son paramètre.

La zone d'exemples contient initialement 22 routes, mais certaines routes ont été découpées en plusieurs parties par les opérations de focalisation, et ce de manière recursive. Chaque partie de route après découpage étant considérée comme une route à part entière, nos exemples contiennent en fait une cinquantaine de routes.

Pour traiter cette zone nous avons effectué environ 120 transformations au sens large du terme, c'est à dire y compris l'arrêt. Certains algorithmes étaient peu utilisés pour traiter cette zone d'exemples (*Schématisation*, *Faille Max* et *Nickerson* étaient utilisés moins de cinq fois). Nous avons alors recueilli quelques exemples supplémentaires pour obtenir d'autres cas d'utilisation de ces algorithmes. Ces nouveaux exemples ont aussi été recueillis dans les Alpes-Maritimes. Nous avons ainsi effectué de l'apprentissage actif, c'est à dire un apprentissage où le recueil des exemples n'est pas complètement aléatoire mais guidé par l'analyse des exemples précédemment recueillis. Après ce nouveau recueil, nous disposons de 153 exemples, représentant le traitement d'une soixantaine d'objets différents. Deux à trois transformations (dont l'arrêt ou la focalisation) sont réalisées sur chaque objet.

E.1.7 Algorithme d'apprentissage utilisé : RIPPER

Choix de l'algorithme

Nous utilisons volontairement un unique algorithme d'apprentissage pour apprendre les hypothèses liées à chaque inférence de la méthode de résolution de problème. En effet, nous désirons évaluer l'intérêt de découper l'apprentissage en plusieurs étapes. En utilisant un même algorithme pour tous les apprentissages que nous réalisons, nous étudions l'effet de la décomposition de l'apprentissage en plusieurs étapes indépendamment de l'effet d'un changement d'algorithme d'apprentissage. Par ailleurs, si nous essayons plusieurs algorithmes pour retenir le meilleur pour chaque inférence, nous biaisons les résultats de l'évaluation en utilisant lors de l'apprentissage les données d'évaluation (cf. C.5.2).

Nous avons choisi d'utiliser l'algorithme RIPPER [Cohen 95], détaillé en annexe III. Cet algorithme est réputé globalement efficace, il est notamment comparable à C4.5 [Quinlan 93] qui est particulièrement reconnu. Il est de plus bien adapté aux attributs numériques, comme le sont les mesures que nous manipulons. RIPPER possède aussi un avantage pratique : il produit des hypothèses sous la forme de règles de décision. Ces règles sont directement traduisibles en langage de programmation procédural classique sous la forme d'une suite de tests "Si … Alors … Sinon …". Ceci n'est pas le cas des règles de production qui nécessitent un moteur d'inférence pour être utilisées. Comme la plupart des logiciels SIG, et la plate-forme PlaGe, possèdent un langage de programmation procédural, nos résultats sont ainsi facilement transposables dans un SIG (à condition qu'il contienne les mesures et les algorithmes utilisés).

Néanmoins, en Annexe V-5, nous montrons les résultats d'apprentissage avec un autre algorithme : C4.5 [Quinlan 93] qui produit des arbres de décision. Ceci nous permet d'illustrer les différences qui peuvent apparaître entre les résultats de différents apprentissages, réalisés avec différentes méthodes et différents langages de représentation des hypothèses.

Mode d'utilisation de l'algorithme RIPPER

De même que nous n'utilisons qu'un seul algorithme, nous ne cherchons pas à en optimiser les paramètres. Nous utilisons donc les paramètres par défaut de RIPPER. Nous montrons également en annexe V-5 les résultats de l'apprentissage en changeant un paramètre de RIPPER de manière à obtenir des règles plus détaillées. Notons que ces règles plus détaillées ont été évaluées comme légèrement plus efficaces que celles apprises avec les paramètres par défaut, mais que la différence est peu sensible.

Le hasard intervient dans le fonctionnement de RIPPER pour séparer les exemples en deux groupes, avant la création de chaque règle. Ces deux groupes sont utilisés respectivement pour créer et élaguer une règle. Du fait de cette utilisation du hasard, on peut obtenir différentes bases de règles en exécutant RIPPER plusieurs fois sur le même jeu d'exemples. Une option, proposée par défaut, permet néanmoins de séparer ces groupes au hasard, mais de la même manière à chaque lancement de RIPPER. Les bases de règles présentées dans ce mémoire ont été calculées avec cette option. En pratique, nous avons constaté que, sans utiliser cette option, les différents résultats obtenus ont des taux d'erreur estimés équivalents.

Il existe néanmoins un paramètre de RIPPER dont nous n'avons pas utilisé la valeur par défaut : l'ordre dans lequel chaque classe est séparée des autres. Lors d'un problème d'apprentissage où la classe peut prendre plus de deux valeurs (e.g. *petit*, *moyen*, *grand*), RIPPER cherche d'abord des règles pour séparer une valeur de classe de toutes les autres (e.g. comment différencier *petit* de "*moyen* ou *grand*"). Il cherche ensuite des règles pour séparer

une seconde valeur de toutes les autres non encore déterminées (e.g. *moyen* de *grand*), et ainsi de suite au besoin. Par défaut, RIPPER détermine des règles pour chaque valeur de la classe dans l'ordre croissant de leur fréquence dans les exemples. Cependant, quand les valeurs de classes sont ordonnées (comme *petit*, *moyen* et *grand*), les règles apprises sont peu compréhensibles si elles ne séparent pas les valeurs de classe dans cet ordre. Par exemple, les deux bases de règles de décision présentées ci-dessous dans le Tableau 5 sont logiquement équivalentes : elles classent toutes deux les objets de longueur inférieure à 10 comme *petits*, ceux de longueur entre 10 et 100 comme *moyens*, et ceux de longueur supérieure à 100 comme *grands*. Cependant, la deuxième base de règles de décision est beaucoup plus compréhensible.

Base de Règles 1	*Base de Règles 2*
si longueur ≥ 10 **et** longueur ≤ 100 **alors** *Taille = moyen* **sinon si** longueur ≥ 55 **alors** *Taille = grand* **sinon** *Taille = petit*	**si** longueur ≥ 100 **alors** *Taille = grand* **sinon si** longueur ≥ 10 **alors** *Taille = moyen* **sinon** *Taille = petit*

Tableau 5. Deux bases de règles logiquement équivalentes

Durant nos expérimentations, lorsque les valeurs de classe étaient naturellement ordonnées, nous avons donc forcé RIPPER à les apprendre dans cet ordre naturel, de manière croissante (*petit*, *moyen*, *grand*) ou décroissante (*grand*, *moyen*, *petit*). Nous avons effectué le choix entre l'ordre croissant ou décroissant de manière à ce que les valeurs apparaissant le plus souvent dans la classe des exemples apparaissent en dernier dans les règles, car cela semble l'option la mieux adaptée pour RIPPER.

E.1.8 Expérimentations réalisées

Processus évalués

Nous avons réalisé plusieurs expérimentations à partir des données décrites ci-dessus. En particulier nous avons appris plusieurs processus de généralisation en simplifiant plus ou moins la structure d'inférence utilisée pour guider l'apprentissage. Par souci de simplicité, nous appelons *apprentissage par les abstractions* ou simplement *apprentissage* le processus issu de l'apprentissage guidé par la structure d'inférence complète décrite en Figure 48 (p. 129). Par ailleurs, nous appelons *apprentissage direct* le processus issu de l'apprentissage sans décomposition de méthode de résolution de problème (Figure 50).

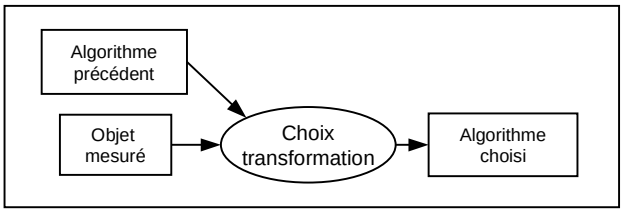


Figure 50. MRP pour l'apprentissage direct

Méthode d'évaluation

Les taux d'erreur des hypothèses liées à chaque inférence du processus ont été estimés par validation croisée (avec un nombre de passes k=10, cf. C.5.2). De même, nous avons évalué

par validation croisée l'enchaînement des inférences : pour chaque étape de la validation croisée, toutes les inférences sont apprises avec les mêmes 90% des exemples, et le résultat de leur enchaînement jusqu'au choix de l'algorithme est évalué sur les 10% restants.

Les résultats de l'apprentissage direct et de l'apprentissage par les abstractions ont aussi été appliqués concrètement sur de nouvelles routes à généraliser pour évaluer, d'un point de vue cartographique, la qualité des résultats obtenus par application de ces processus.

Les résultats de ces expérimentations sont commentés en détail dans la suite de ce chapitre. L'implémentation des expérimentations est présentée dans l'annexe IV.

Paramétrage des algorithmes de généralisation

Nous avons concentré nos tests sur l'évaluation de la qualité de la détermination de l'algorithme. Mais nous avons effectué très peu d'expérimentations sur la détermination des paramètres à partir des exemples recueillis. La première raison à cela est que nous avons considéré ce problème comme secondaire par rapport au choix de l'algorithme dans le cadre nos travaux. De plus, les paramètres de la plupart des algorithmes que nous utilisons peuvent être directement reliés aux spécifications de la carte, ce qui rend leur détermination immédiate.

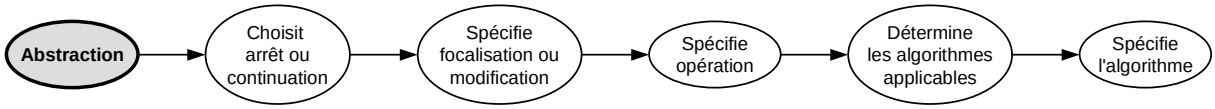
Quand les paramètres peuvent prendre une infinité de valeurs, leur apprentissage réclame de nombreux exemples ainsi que, peut-être, une approche numérique comme celle des réseaux de neurones. De telles expérimentations d'apprentissage de paramètre avec des réseaux de neurones ont été menées pour des algorithmes de lissage ou filtrage [Werschlein et Weibel 94 ; Weibel, Keller et Reichenbacher 95 ; Reichenbacher 95 ; Lagrange et Landras 99 ; Lagrange, Landras et Mustière 2000]. Elles concluent sur la nécessité de mieux mesurer la forme des objets sur lesquels sont appliqués les algorithmes, et sur la nécessité d'utiliser de nombreux exemples.

Pour déterminer les paramètres de quelques algorithmes dans le cadre de nos expérimentations sur les routes, nous avons utilisé des règles empiriques. L'algorithme RIPPER nous a néanmoins servi à identifier, à partir de nos exemples, des attributs pertinents pour déterminer ces paramètres. Le nombre d'exemples utilisés étant très faible, on ne peut pas évaluer ces règles empiriques avec des techniques comme la validation croisée. L'application pratique de ces règles de choix fournit néanmoins des résultats suffisamment satisfaisants d'un point de vue cartographique. Ce problème nécessiterait tout de même des études supplémentaires dans les domaines où le paramétrage des algorithmes est critique (e.g. pour utiliser l'algorithme de *Lowe* [Lowe 88] très difficile à paramètrer).

E.2 Résultats : règles apprises

Dans cette partie, nous détaillons et commentons les règles apprises dans le cas des routes. L'annexe V regroupe les résultats de l'application de l'apprentissage sur les routes, c'est à dire les informations réparties tout au long de cette partie : les règles apprises, les langages utilisés, etc.

E.2.1 Détermination des attributs descriptifs abstraits



Règles apprises

Les règles apprises pour déterminer chaque attribut abstrait sont présentées ci-dessous (Règles 1). Une base de règles indépendante est associée à chaque attribut abstrait. Ces bases de règles contiennent des règles de décision, c'est-à-dire que les règles sont ordonnées et que la première règle rencontrée qui est respectée est appliquée. Autrement dit, elles peuvent se lire "Si .. alors … / **sinon** si … alors … / **sinon** …". Par exemple, si la longueur est de 10mm, alors la taille est classée comme *moyen* car la deuxième règle "**si** longueur ≥ 3,4 **alors** *Taille = moyen*" est la première à être respectée.

TAILLE
si longueur ≥ 13,5 **alors** *Taille = grand*
si longueur ≥ 3,4 **alors** *Taille = moyen*
sinon *Taille = petit*

COMPLEXITE
si taille max virages ≤ 0,09 **et** nombre grands virages = 1 **alors** *Complexité = zéro niveau*
si nombre grands virages ≤ 6 **alors** *Complexité = un niveau*
si base sur longueur ≥ 0,64 **alors** *Complexité = un niveau*
sinon *Complexité = plusieurs niveaux*

SINUOSITE
si nombre virages ≥ 16 **et** base sur longueur ≤ 0,61 **alors** S*inuosité = hétérogène*
si base sur longueur ≥ 0,97 **alors** *Sinuosité = nulle*
si base sur longueur ≥ 0,73 **alors** *Sinuosité = virages peu sinueux*
sinon *Sinuosité = épingles à cheveux*

GRANULARITE
si nombre virages ≥ 6 **et** force empâtement ≥ 0.019 **et** base sur longueur ≥ 0.30) **alors** *Granularité = forte*
si force empâtement ≥ 0.287 **et** force empâtement ≤ 0.296 **alors** *Granularité = forte*
si nombre virages ≥ 3 **et** taille moyenne virages ≤ 0.17 **et** Granularité ≥ 0.04 **alors** *Granularité = moyenne*
si taille des virages moyenne sur max ≤ 0.82 **et** taille des virages moyenne sur max ≥ 0.66 **et** force bruit ≤ 0.07 **alors** *Granularité = moyenne*
sinon Granularité = faible

FORME GENERALE
si base sur longueur ≥ 0,97 **alors** *Forme = ligne droite*
si nombre virages ≤ 2 **et** base sur longueur ≤ 0,32 **alors** *Forme = virage*
si nombre virages = 1 **alors** *Forme = virage*
si nombre grands virages ≥ 6 **alors** *Forme = longue série de virages*
sinon *Forme = courte série de virages*

EMPATEMENT
si homogénéité empâtement ≤ 0,81 **et** longueur empâtement ≥ 2,9 **alors** *Empâtement = hétérogène*
si longueur empâtement ≥ 0.9 **et** force empâtement ≥ 0,013 **alors** *Empâtement = fort*
si type empâtement ≠ aucun et force empâtement ≥ 0.008 **alors** *Empâtement = faible*
sinon *Empâtement = nul*

ENVIRONNEMENT
si surface conflit proche ≤ 0,1 **et** nb arcs proches par longueur ≤ 1,2 **alors** *Environnement = libre*
si surface conflit proche ≤ 0,5 **et** nombre arcs proches ≤ 2 **alors** *Environnement = libre*
sinon *Environnement = dense*

CONFLIT EXTERNE
Conflit externe = aucun (i.e. classe par défaut attribuée à tous les exemples)

Règles 1. Règles apprises pour déterminer les abstractions

Nous présentons dans le Tableau 6 les taux d'erreur de ces règles estimés par validation croisée (cf. C.5.2). Nous comparons ces taux d'erreur à l'erreur par défaut, c'est-à-dire l'erreur de l'hypothèse qui à tout exemple attribue la classe majoritaire dans les exemples. Par exemple, si on doit apprendre à classer des exemples dans deux classes *positif* ou *négatif*, et si 60% des exemples sont classés *positifs* et 40% *négatifs*, l'erreur par défaut correspond à celle de l'hypothèse qui classe tous les exemples comme *positifs*, soit 40%. Dans le Tableau 6, les valeurs entre parenthèses après les erreurs par défaut correspondent à la classe majoritaire dans les exemples. Enfin, le gain relatif dans ce tableau représente l'apport de l'apprentissage relativement à l'erreur par défaut, il est défini par :

$$\text{Gain Relatif} = \frac{\text{Erreur par défaut} - \text{Erreur estimée}}{\text{Erreur par défaut}}$$

Caractère abstrait	***Erreur estimée par validation croisée***	*Erreur par défaut*	*Gain relatif*
Taille	**4%**	40% (petit)	90%
Complexité	**12%**	33% (un niveau)	64%
Sinuosité	**14%**	56% (épingles à cheveux)	75%
Forme générale	**14%**	64% (courte série de virages)	78%
Empâtement	**16%**	41% (nul)	61%
Environnement	**20%**	47% (libre)	57%
Granularité	**34%**	38% (faible)	11%
Conflits externe	**29%**	28% (aucun)	-3%

Tableau 6. Erreurs estimées sur les abstractions

Analyse des taux d'erreur

Les règles et leurs erreurs estimées montrent tout d'abord que RIPPER n'a pas ou peu réussi à apprendre à déterminer les concepts de *granularité* et de *conflit externe*. Pour ces attributs, les erreurs estimées sont proches des erreurs par défaut. Cela signifie que les règles apprises ne sont pas plus efficaces que l'hypothèse la plus simple, et évidemment non pertinente, qui attribue la classe par défaut à tout exemple.

Pour apprendre ces attributs problématiques nous avons alors essayé de modifier les paramètres de RIPPER, d'utiliser C4.5 à la place de RIPPER, d'utiliser plus de mesures, de créer de nouvelles mesures en combinant les mesures choisies à l'origine, ou de regrouper les classes (e.g. regrouper les *granularité* "faible" et "moyenne" en une seule classe). Aucun de ces tests n'a permis d'apprendre à déterminer ces concepts. Plus précisément, l'évaluation par validation croisée des différentes hypothèses apprises fournissait des taux d'erreurs très élevés, parfois plus élevés que l'erreur par défaut. De plus, une analyse visuelle des hypothèses apprises confirmait que ces règles étaient non pertinentes, par exemple parce qu'elles utilisaient des mesures de manière non logique (e.g. si *Surface de conflit* < X alors *Conflit externe* = fort).

Ceci confirme une des conclusions des tests effectués sur la généralisation des routes (indépendamment de leur contexte) durant le projet AGENT. Le système AGENT utilise sur les routes des mesures similaires à celles que nous utilisons [Duchêne 2001], et il part du principe que toute action doit être déclenchée par un conflit cartographique [Ruas 99]. Il ne déclenche pas d'opération de lissage (qui a pour but de diminuer la granularité), faute de mesure de granularité assez pertinente. Par contre, le système AGENT utilise d'autres mesures plus pertinentes pour identifier les conflits entre routes [Bader et Barrault 2000], ce qui permet de déclencher des opérations de déplacement entre routes.

Nous en concluons que les mesures utilisées ne sont pas assez pertinentes pour déterminer ces attributs *granularité* et *conflit externe* pendant l'inférence d'*abstraction*. Ceci a de nombreuses conséquences sur les inférences de *représentation* qui permettent de choisir l'algorithme à appliquer sur un objet. Ces conséquences seront discutées par la suite.

Par ailleurs, l'apprentissage a permis de déterminer des règles pour déterminer les caractères abstraits autres que *granularité* et *conflit externe*. Notons que l'hypothèse qui fournit le plus d'erreurs est celle déterminant le caractère abstrait *environnement*. Ceci s'explique par le fait que nous n'avons pas utilisé de mesures très pertinentes pour décrire ce caractère. Les mesures utilisées qualifient de manière globale l'environnement d'un objet (e.g. le nombre d'arcs proches de la route). Il serait plus pertinent de mesurer l'environnement de manière plus localisée. Par exemple, il faudrait distinguer l'environnement le long de la route et l'environnement dans son prolongement qui influencent différemment la perception de l'environnement d'une route, et qui influent différemment sur le choix des transformations cartographiques à y appliquer.

Analyse du contenu des règles

Outre la qualité en terme de prédiction des règles, nous analysons leur contenu. L'intérêt de l'apprentissage a été de déterminer les mesures pertinentes vis-à-vis des attributs abstraits, ainsi que des seuils sur ces mesures et la manière de combiner les tests sur les mesures. Les bases de règles apprises pour chaque descripteur abstrait sont simples. Chacune contient au maximum cinq règles et trois conditions par règle. Elles sont ainsi relativement faciles à interpréter pour un expert du domaine, du moins lorsque les mesures sont elles-mêmes facilement interprétables. Nous analysons ci-dessous quelques aspects de ces règles.

Le plus surprenant dans ces règles est la fréquente utilisation de la mesure *base sur longueur* (distance entre les deux extrémités de la ligne divisée par sa longueur). Cette mesure est extrêmement simple et semble, au premier abord, contenir peu d'information par rapport à d'autres mesures introduites pour qualifier la forme d'une ligne. En effet, toutes les lignes de la Figure 51 ont des valeurs similaires pour cette mesure, tout en ayant des tailles et des formes très différentes.

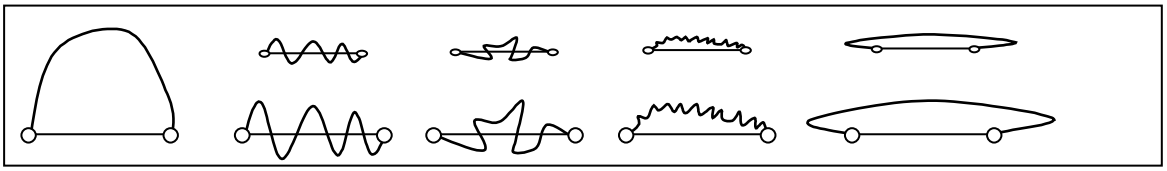


Figure 51. Différentes lignes de même base sur longueur

On peut avancer deux explications à l'utilisation intensive de cette mesure, et donc à sa pertinence estimée par l'apprentissage.

D'une part, les routes sont pas des objets géométriques quelconques. Leur forme est contrainte par le relief sur lequel elles se trouvent et par leur rôle de support des véhicules qui leur interdit une courbure trop forte. L'ensemble des formes possibles d'une route est donc restreint. Ainsi, si la mesure de base sur longueur n'est pas pertinente pour une ligne quelconque, elle permet néanmoins de relativement différencier les formes que peuvent prendre les routes.

D'autre part, la mesure *base sur longueur* est très stable au bruit : elle est stable vis-à-vis des petites modifications de formes, et elle est stable vis-à-vis du nombre de points utilisés pour décrire une même ligne. Ceci est moins le cas pour certaines des autres mesures de forme utilisées. Ces mesures ont été intensément testées pour qualifier des routes d'une BDG [Plazanet 96], par exemple pour les regrouper selon leur forme (e.g. par clustering). Mais elles ont moins été testées pour comparer des arcs avant et après un traitement de généralisation. Ceci peut expliquer leur relative instabilité dans notre contexte. En effet, nous utilisons des mesures sur des arcs initiaux de la BDG, mais aussi sur des arcs en cours de traitement, qui ont déjà subi plusieurs transformations par les algorithmes. Or, certains algorithmes ont tendance à densifier le nombre de points utilisés pour décrire une ligne ou à privilégier des formes particulières (e.g. les algorithmes *Faille Min*, *Faille Max* et *Plâtre* arrondissent les formes). Si certaines mesures sont très sensibles à cette densification ou à ces formes particulières, elles qualifient différemment les arcs initiaux et les arcs en cours de traitement. Comme nous posons pour *a priori* dans notre approche de la généralisation que la prise de décision est identique au début et en cours de traitement, l'apprentissage a privilégié l'utilisation de mesures stables au cours du traitement, au détriment de mesures éventuellement plus pertinentes sur l'ensemble des arcs avant traitement mais biaisées sur les arcs en cours de traitement.

Dans notre contexte, le fait que les mesures soient stables au cours des traitements est au moins aussi important que la quantité d'information qu'elles véhiculent.

Par ailleurs, certaines règles peuvent être aisément analysées. C'est le cas par exemple des règles définissant ce qu'est un virage dans notre contexte. Les lignes ne contenant pas de points d'inflexion (mesure *nombre virages* = 1) sont considérées comme ne contenant qu'un seul virage, ce qui est naturel. Mais de plus, les lignes contenant un seul point d'inflexion (mesure *nombre virages* = 2) et assez "serrées" (*base sur longueur* < 0.32) sont aussi considérées comme ne contenant qu'un seul virage. Ces règles peuvent être illustrées par les exemples de la Figure 52.

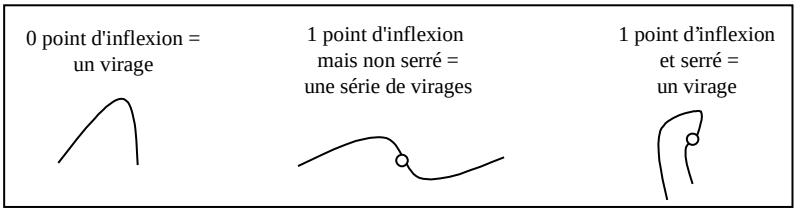


Figure 52. Définition apprise d'un virage

D'autres règles utilisant des mesures moins directement interprétables sont moins faciles à analyser en détail. En effet, les mesures ne correspondent pas toujours à des concepts aussi facilement visualisables que le nombre de points d'inflexions. On peut néanmoins analyser une règle en regardant le sens de l'inégalité d'une condition. Analysons ainsi par exemple les règles déterminant l'empâtement. La règle "**si** type empâtement ≠ aucun et force empâtement ≥ 0.008 **alors** *Empâtement = faible* **sinon** *Empâtement = nul* " peut s'exprimer en termes un peu plus compréhensibles comme : si la mesure de détection des empâtements en a détecté d'un seul côté, et que l'empâtement est suffisamment important, alors l'empâtement est considéré comme faible et non nul. En d'autres termes encore, la mesure de détection des empâtements ne suffit pas à elle seule pour déterminer si il y a ou non empâtement, il faut que l'empâtement soit suffisamment important. Ceci confirme que notre algorithme de détection des empâtements effectue parfois une sur-détection de ceux-ci (cf. B.3.2.3).

Enfin, le fait que les règles soient lisibles nous permet d'estimer au moins certains aspects de leur cohérence. Par exemple s'il est difficile d'interpréter en détail la règle "**si** surface conflit proche ≤ 0,1 **et** nb arcs proches par longueur ≤ 1,2 **alors** *Environnement = libre*", on peut au moins constater qu'elle utilise de manière cohérente les mesures qu'elle utilise : si la surface de conflit est inférieure à un seuil, et le nombre d'arcs proches également, alors l'environnement est considéré comme libre, ce qui est cohérent.

La lisibilité des règles nous permet de comprendre aisément certaines d'entre elles et ainsi d'enrichir notre connaissance du domaine. Elle permet aussi de vérifier qu'elles possèdent une certaine cohérence, et ainsi d'en fournir une première validation.

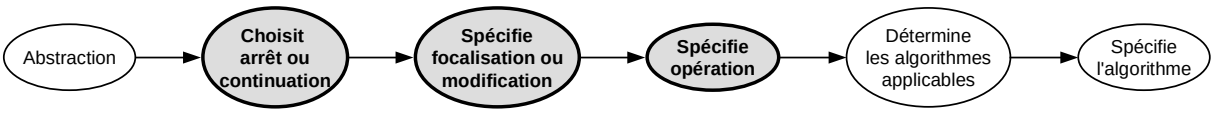
Mesures pertinentes

L'analyse des règles apprises nous permet d'identifier les mesures considérées par l'apprentissage comme pertinentes vis-à-vis de notre problème. L'ensemble de ces mesures apparaissant dans les hypothèses est listé dans le Tableau 7 suivant, avec leur nombre d'occurrences dans les hypothèses apprises (en excluant les hypothèses non efficaces déterminant la *granularité* et le *conflit externe*).

Mesure	*Nombre d'occurrences*
base sur longueur	6
nombre virages	3
nombre grands virages	3
longueur	2
force empâtement	2
longueur empâtement	2
surface de conflit proche	2
taille max virages	1
type empâtement	1
homogénéité empâtement	1
nombre d'arcs proches	1
nb arcs proches par longueur	1

Tableau 7. Mesures pertinentes

E.2.2 Détermination de l'opération



Influence de l'abstraction mal apprise *conflit externe*

A l'étape précédente, l'apprentissage de l'abstraction *conflit externe* a été particulièrement inefficace, faute de mesure pertinente. L'hypothèse apprise est la règle par défaut, qui attribue à tout exemple la classe *aucun* : tous les nouveaux exemples seront classés comme n'ayant pas de conflit graphique avec les objet de son environnement. Or, les opérations de déplacement sont déclenchées pour résoudre de tels conflits. Il devient donc impossible d'apprendre quand réaliser des transformations de déplacement.

Nous devons donc nous résoudre à ne plus considérer les opérations de déplacement dans nos expérimentations, faute de mesure assez pertinente pour déterminer l'attribut conflit externe.

Nous éliminons donc de nos tests les exemples où un déplacement est réalisé (une fois ces exemples éliminés nous disposons de 140 exemples sur les 153 exemples initiaux).

Influence de l'abstraction mal apprise *granularité*

De même, l'apprentissage de l'abstraction *granularité* est peu efficace (gain relatif = 21%). Si on utilise cet attribut lors de l'apprentissage des inférences suivantes de représentation (choix de *Action*, *Type opération et Opération*), celui-ci est utilisé dans les règles apprises. Or, comme cet attribut sera souvent mal déterminé sur un nouvel objet à traiter, son utilisation dans les inférences de représentation sera trompeuse, et l'enchaînement des inférences sera inefficace. Nous avons confirmé cela par l'expérience. L'enchaînement des inférences d'abstraction puis de représentation produit plus d'erreur en conservant l'attribut *granularité* qu'en l'ignorant, même si l'utilisation de cet attribut permet de mieux apprendre les inférences de représentation, quand on s'intéresse seulement à celles-ci (Tableau 8). En particulier, l'hypothèse apprise pour choisir l'*action* (arrêter ou continuer) est efficace quand l'attribut *granularité* est utilisé (erreur = 4%), mais l'enchaînement des étapes d'*abstraction* et de *choix de l'action* est alors inefficace (erreur = 18%). Si on ignore cet attribut problématique, l'hypothèse pour choisir l'*action* est moins efficace (erreur = 10%), mais l'enchaînement des étapes d'*abstraction* et de *choix de l'action* est plus efficace (erreur = 11%). Si nous possédions des mesures plus pertinentes pour déterminer la *granularité* les résultats seraient de meilleure qualité (erreur sur le choix de l'arrêt = 2%). Faute de telles mesures, il faut ne pas prendre en compte cet attribut pour améliorer les résultats.

		En utilisant *granularité*		**SANS utiliser *granularité***	
		Erreur estimée de l'étape seule	*Erreur estimée de l'enchaînement des inférences jusqu'à l'étape*	*Erreur estimée de l'étape seule*	*Erreur estimée de l'enchaînement des inférences jusqu'à l'étape*
Etape	Action	2%	18%	7%	9%
	Type opération	2%	20%	2%	11%
	Opération	1%	21%	1%	14%

Tableau 8. Influence des abstractions mal apprises sur les inférences de représentation

Du point de vue de l'apprentissage, ceci confirme qu'il est plus efficace d'ignorer les attributs trop bruités que de les prendre en compte pour l'apprentissage [Quinlan 86b]. En effet, puisque l'attribut *granularité* est mal appris durant l'inférence d'abstraction, on peut le considérer comme bruité pour les inférences suivantes.

Pour que l'enchaînement des inférences soit efficace, il est plus efficace d'éliminer un attribut abstrait trop mal déterminé que de le prendre en compte lors de l'apprentissage des inférences de représentation. La qualité de l'apprentissage d'une étape guide donc le choix des observables prises en compte aux étapes suivantes.

Ainsi, les attributs *granularité* et *conflit externe* seront ignorés dans nos expérimentations.

Règles apprises

Une fois les attributs *granularité* et *conflit externe* éliminés des observables, et les exemples de déplacement éliminés des jeux d'exemples, nous apprenons les bases de règles nécessaires à la détermination de l'opération à réaliser (Règles 2, également évaluées par validation croisée). Les taux d'erreur évalués de ces règles sont présentés dans le Tableau 9.

CHOIX de L'OPERATION

ACTION
si Type opération précédente = modification géométrique **et** Empâtement = nul **alors** *Action = Arrêt*
si Opération précédente = simplification **et** Empâtement = faible **alors** *Action = Arrêt*
si Algorithme précédent = plâtre **et** Empâtement = faible **alors** *Action = Arrêt*
si Complexité = zéro niveau **alors** *Action = Arrêt*
sinon *Action = Continuer*

TYPE OPERATION
si Action = arrêt **alors** *Type opération = Arrêt*
si Empâtement = hétérogène **alors** *Type opération = Focalisation*
sinon *Type opération = Modification géométrique*

OPERATION
si Action = arrêt **alors** *Opération = Arrêt*
si Type opération = focalisation **alors** *Opération = Focalisation*
si Empâtement = nul **alors** *Opération = Simplification*
sinon *Opération = Exagération*

Règles 2. Règles pour le choix de l'opération

	Erreur estimée	*Erreur par défaut*	*Gain relatif*
Action	**7%**	45% (continuer)	84%
Type opération	**2%**	51% (modification)	96%
Opération	**1%**	55% (arrêt)	98%

Tableau 9. Erreurs estimées des inférences de représentation

Analyse

L'information de la précédente transformation réalisée est très largement prise en compte dans le choix de l'arrêt. Ceci peut être expliqué grâce aux tests suivants, réalisés en modifiant les observables prises en compte pour apprendre l'*action* (Tableau 10). Si on ignore pendant l'apprentissage l'information sur la transformation précédente, l'apprentissage est incapable d'apprendre une hypothèse pour déterminer l'arrêt : il apprend l'hypothèse "**Si** empâtement = nul **alors** *Action = Arrêt*, **sinon** *Action = Continuer*", avec une erreur estimée de 27%. Cette information est donc très importante. Ceci n'est pas le cas pour une hypothèse apprise en utilisant l'attribut *granularité*. En effet, si on ignore la transformation précédente effectuée, mais que l'on introduit cet attribut pour déterminer l'action, RIPPER apprend l'hypothèse suivante :
si Granularité = faible **et** Empâtement = nul **alors** *Action = arrêt*
si Empâtement = faible **et** Forme = Courte série de virages **alors** *Action = arrêt*
si Granularité = moyenne **et** Complexité = Plusieurs niveaux **et** Empâtement = Nul **alors** *Action = arrêt*
sinon *Action = continuer*

Cette hypothèse possède un taux d'erreur estimé à 8%, ce qui est proche de l'erreur de l'hypothèse apprise en ignorant les abstractions problématiques mais en introduisant l'information de la transformation précédente effectuée (7%).

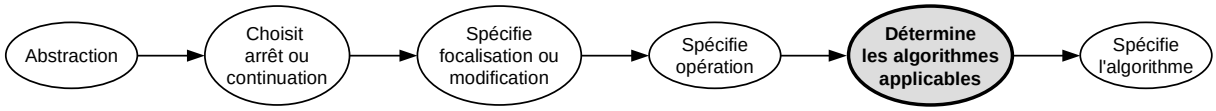
	En utilisant la *Transformation précédente*	Sans utiliser la *Transformation précédente*
En utilisant *Granularité*	2%	8%
Sans utiliser *Granularité*	7%	27%

Tableau 10. Erreurs de l'inférence Action *en fonction des observables utilisées*

L'information de la transformation précédente effectuée permet donc de compenser l'incapacité à déterminer la *granularité.*

On peut noter qu'une fois l'action déterminée, la transformation précédente n'est plus utilisée dans les règles pour le choix du type d'opération et de l'opération. Ces règles sont simples et efficaces. On peut noter leur grande similarité avec le processus GALBE : quand l'empâtement est hétérogène la route est découpée, quand l'empâtement est nul elle est simplifiée, sinon elle est caricaturée. Ce sont ces mêmes principes qui régissent GALBE, même si la détermination de l'empâtement est plus complexe dans les règles apprises que dans GALBE qui n'utilise qu'une seule mesure. On ne peut néanmoins pas en conclure résolument que les résultats de l'apprentissage confirment la pertinence de l'approche proposée dans GALBE. En effet, nous avons à la fois conçu GALBE et recueilli les exemples pour l'apprentissage. Ainsi, nous avons indéniablement biaisé le recueil des exemples, même involontairement, en généralisant interactivement avec une logique proche de celle de GALBE. Cette similarité pourrait plutôt confirmer que l'apprentissage a bien fait ressortir notre approche de la généralisation des routes.

E.2.3 Applicabilité des algorithmes



Règles apprises

Les règles déterminant l'applicabilité des algorithmes (Règles 3) sont, elles aussi, apprises sans considérer les attributs de *granularité* et de *conflit externe*, et sans utiliser les exemples réalisant l'opération de *déplacement* (Règles 3). Leurs taux d'erreur estimés sont présentés dans le Tableau 11.

APPLICABILITÉ DES ALGORITHMES

LISSAGE GAUSSIEN
si Opération = Simplification **et** Complexité ≠ plusieurs niveaux **alors** *Lissage Gaussien = applicable*
sinon *Lissage Gaussien = pas applicable*

PLATRE
si Opération = Simplification **et** Taille ≠ petit **alors** *Plâtre = applicable*
si Type opération = Modification **et** Empâtement ≠ fort **et** Empatement ≠ nul **alors** *Plâtre = applicable*
sinon *Plâtre = pas applicable*

ACCORDEON
si Empâtement = fort **et** Forme = courte série de virages **alors** *Accordéon = applicable*
sinon *Accordéon = pas applicable*

SCHEMATISATION
si Empâtement = fort **et** Complexité = plusieurs niveaux **alors** *Schématisation = applicable*
sinon *Schématisation = pas applicable*

FAILLE MIN
si Forme = virage **et** Empâtement ≠ nul **alors** *Faille Min = applicable*
sinon *Faille Min = pas applicable*

FAILLE MAX
si Forme = virage **et** Environnement = libre **et** Empâtement ≠ nul **alors** *Faille Max = applicable*
sinon *Faille Max = pas applicable*

Règles 3. Règles pour le choix des applicabilités

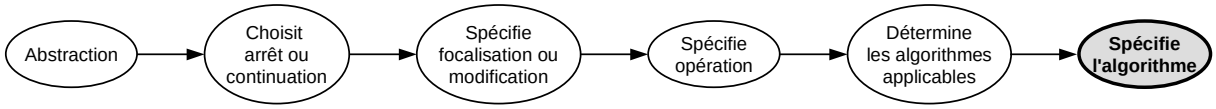
		Erreur estimée	*Erreur par défaut*	*Gain relatif*
Applicabilités	Lissage Gaussien	**1%**	17% (pas applicable)	94%
	Plâtre	**9%**	20% (pas applicable)	55%
	Accordéon	**1%**	9% (pas applicable)	89%
	Schématisation	**4%**	10% (pas applicable)	60%
	Faille Min	**1%**	10% (pas applicable)	90%
	Faille Max	**1%**	7% (pas applicable)	86%

Tableau 11. Erreurs sur le choix des applicabilités

Analyse

Les règles apprises sont efficaces et leur contenu est cohérent. Aucune des hypothèses apprises n'utilise l'information sur la transformation précédente effectuée. Les difficultés les plus grandes résident au niveau de l'applicabilité de l'algorithme *Plâtre*, pour lequel le gain relatif est plus faible que celui des autres algorithmes. Ceci est dû au champ d'action très large de *Plâtre* : celui-ci étant conçu pour pouvoir s'appliquer sur de nombreuses routes (cf. A.3.2.1), les critères permettant de spécifier quand il est applicable sont moins nets que pour les autres algorithmes.

E.2.4 Choix de l'algorithme



Les règles apprises pour la détermination finale de l'algorithme sont présentées ci-dessous (Règles 4). Ces règles ont une erreur estimée de 11%, soit un gain relatif de 80% par rapport à l'erreur par défaut de 55% (erreur quand l'*arrêt* est systématiquement choisi).

CHOIX ALGORITHME

si Action = arrêt **alors** *Choix algo = Arrêt*

si Type opération = focalisation **alors** *Choix algo = Découpage conflit*

si Forme = virage **et** Empâtement = fort **alors** *Choix algo = Faille Min*
si Forme = virage **et** Environnement = dense **alors** *Choix algo = Faille Min*

si Faille Max = applicable **alors** *Choix algo = Faille Max*

si Schématisation = applicable **alors** *Choix algo = Schématisation*

si Empâtement = fort **alors** *Choix algo = Accordéon*

si Lissage Gaussien = pas applicable **alors** *Choix algo = Plâtre*

sinon *Choix algo = Lissage Gaussien*

Règles 4. Règles pour le choix final de l'algorithme

Analyse

Ces règles n'utilisent pas l'information sur la transformation précédente effectuée. Plus généralement, seules les règles pour décider ou non de l'arrêt (cf. Règles 2 p.140) utilisent cette information, pour compenser l'absence de certains attributs abstraits mal déterminés.

Cela signifie que l'on peut voir la généralisation comme un processus markovien, où le choix de la transformation à effectuer à un moment donné du processus ne dépend pas des opérations précédemment réalisées.

Les deux premières règles apprises sont triviales. L'algorithme *arrêt* est choisi quand l'action d'*arrêt* a été choisie, et l'algorithme de découpage par les conflits d'empâtement est choisi en cas de focalisation, puisque c'est le seul algorithme utilisé permettant de focaliser.

Dans le cas où la ligne est un virage, l'algorithme *Faille Min* est préféré à *Faille Max* quand l'environnement est dense ou quand l'empâtement est fort. En effet, si l'environnement est dense *Faille Min* permet de minimiser les risques de superposition entre le virage et les objets de son environnement, et si l'empâtement est fort cela signifie que le virage est très serré et l'application de *Faille Max* en dégraderait trop la forme serrée.

Les règles suivantes, concernant *Schématisation, Accordéon, le Lissage Gaussien et Plâtre*, sont aussi cohérentes quand on analyse les raisons de l'applicabilité de *Schématisation* et du *Lissage Gaussien*. Cependant, la combinaison de tests portant sur les descripteurs abstraits et sur les applicabilités ne facilite pas la lecture de ces règles : pour les comprendre, il est nécessaire d'analyser les raisons de l'applicabilité de chaque algorithme, expliquées dans un autre jeu de règles. Par exemple, pour comprendre quand *Schématisation* est choisi plutôt que *Accordéon*, *Plâtre* ou le *Lissage Gaussien*, il faut savoir que *Schématisation* n'est applicable que si l'empâtement de la route est fort et si la route est une longue série de virages (cf. Règles 3).

Nous présentons ci-dessous les règles apprises pour choisir l'algorithme *sans* utiliser les applicabilités des algorithmes. Ces règles sont différentes des précédentes, et elles ont un taux d'erreur estimé supérieur (14% au lieu de 11% précédemment). Mais elles sont plus faciles à analyser.

CHOIX ALGORITHME

si Action = arrêt **alors** *Choix algo = Arrêt*

si Type opération = focalisation **alors** *Choix algo = Découpage conflit*

si Forme = virage **et** Empâtement = fort **alors** *Choix algo = Faille Min*
si Forme = virage **et** Environnement = dense **alors** *Choix algo = Faille Min*

si Forme = virage **alors** *Choix algo = Faille Max*

si Empâtement = fort **et** Forme = Longue série de virages **alors** *Choix algo = Schématisation*

si Empâtement = fort **alors** *Choix algo = Accordéon*

si Taille ≠ petit **alors** *Choix algo = Plâtre*
si Opération = Exagération **alors** *Choix algo = Plâtre*

sinon *Choix algo = Lissage Gaussien*

Règles 5. Règles pour le choix final de l'algorithme, sans utiliser les applicabilités

La prise en compte des applicabilités des algorithmes applicables permet d'améliorer le pouvoir de prédiction des règles apprises pour choisir l'algorithme. Cependant elle en dégrade la lisibilité.

E.2.5 Paramétrage

Nous avons adopté les règles suivantes (Règles 6) pour déterminer le paramètre de certains algorithmes, afin de pouvoir appliquer les règles apprises durant nos expérimentations. Comme expliqué dans la partie E.1.8 (p.124), ces règles sont purement empiriques, même si nous avons utilisé l'algorithme d'apprentissage RIPPER pour déterminer les attributs pertinents pour déterminer ce paramétrage. Elles ont été établies sur très peu d'exemples, pour un problème d'apprentissage numérique, et il est donc difficile de les évaluer. Nous ne les commentons donc pas. Notons simplement que ces règles utilisent les mesures numériques en plus des attributs abstraits, ce qui paraît logique étant donné que les paramètres prennent des valeurs numériques quantifiant l'effet des algorithmes.

PARAMETRAGE

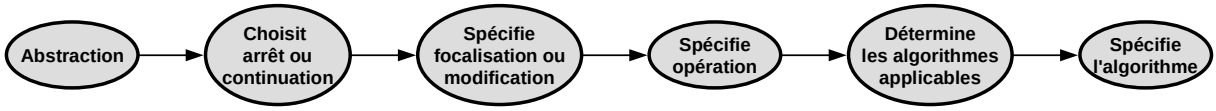
PLATRE
Si Nombre virages < 4 **alors** Courbure = 150 * Largeur de symbole
Si Complexité = un niveau **alors** Courbure = 270 * Largeur de symbole
Sinon Courbure = 210 * Largeur de Symbole

LISSAGE GAUSSIEN
Si Largeur de symbole < 0.4 **alors** sigma = 150 * Largeur de symbole
Sinon sigma = 215 * Largeur de symbole

ACCORDEON
Si Complexité = plusieurs niveaux **alors** Exagération = 1.0
Si Environnement = dense **alors** Exagération = 1.0
Sinon Exagération = 1.1

Règles 6. Paramétrage des algorithmes

E.2.6 Enchaînement des inférences



L'enchaînement de toutes les inférences de la méthode de résolution de problème produit un taux d'erreur estimé à 29 % sur le choix final de l'algorithme à appliquer. Le Tableau 12 suivant présente la matrice de confusion estimée par validation croisée sur le choix final de l'algorithme. Les confusions obtenues seront comparées à celles des règles apprises par apprentissage direct dans la partie E.4.

		Classes des exemples fournies par l'expert								
		Arrêt	Découpage Empâtement	Lissage Gaussien	Plâtre	Accordéon	Schématisation	Faille Max	Faille Min	Total
Classes apprises	Arrêt	**55**	1	2	1	1				60
	Découp. Empât.	2	**6**		1		1			10
	Lissage Gaussien	6		**15**	8	2				31
	Plâtre			1	**5**					6
	Accordéon		1		1	**5**	3		1	11
	Schématisation					1	**6**			7
	Faille Max							**5**	4	9
	Faille Min					1		2	**3**	6
	Total	63	8	18	16	10	10	7	8	**140**

Tableau 12. Matrice de confusion estimée du choix final de l'algorithme

Les plus grandes confusions se font entre les algorithmes conceptuellement les plus proches (*Faille Min* et *Faille Max*, *Accordéon* et *Schématisation*, *Lissage Gaussien* et *Plâtre*). Les erreurs les plus fréquentes se situent lors de l'utilisation de l'algorithme du *Lissage Gaussien*, qui est le choix par défaut des règles de détermination finale de l'algorithme. En effet celui-ci est choisi 31 fois par les règles, alors qu'il n'est utilisé que 18 fois dans les exemples. Il est choisi la plupart du temps par erreur à la place de l'algorithme *Plâtre* ou à la place de l'*Arrêt*. Quand il est choisi à la place de *Plâtre*, ceci a souvent peu d'importance car ces algorithmes sont relativement proches, du moins sur les routes non empâtées. Quand il est choisi à la place de l'*Arrêt*, la route traitée sera trop simplifiée, mais cela aura souvent un effet peu visible, du moins dans le cas des routes peu sinueuses.

Plus généralement, on ne peut pas mettre sur le même plan toutes les confusions réalisées. Tout d'abord, lorsque l'algorithme choisi est un algorithme non applicable, cela est une erreur beaucoup plus importante que lorsque l'algorithme choisi est néanmoins applicable. Le Tableau 13 détaille cela : il présente pour chaque algorithme, si celui-ci a été choisi à juste titre (i.e. comme dans les exemples), ou si il a été choisi par erreur mais qu'il était applicable, ou enfin si il a été choisi sans être applicable. Au total, un algorithme est choisi sans être applicable dans 17% des cas.

	Arrêt	Découpage Empâtement	Lissage Gaussien	Plâtre	Accordéon	Schématisation	Faille Max	Faille Min	Total
Algorithme choisi a juste titre	55	6	15	5	5	6	5	3	100 (71%)
Algorithme choisi par erreur mais applicable	0	0	7	1	3	1	2	3	17 (12%)
Algorithme choisi par erreur et non applicable	5	4	9	0	3	0	1	1	23 (17%)

Tableau 13. Détail des confusions réalisées.

Par ailleurs, certaines erreurs vont beaucoup influencer la qualité du résultat final, comme par exemple choisir *Faille Min* à la place de *Accordéon* ce qui peut énormément dégrader la forme de l'objet traité (ceci peut arriver si l'attribut *forme générale* a mal été déterminé, cf. E.3.2). Par contre d'autres erreurs auront des effets limités, comme par exemple choisir le *Lissage Gaussien* au lieu de l'*Arrêt*, ce qui aura pour effet de trop simplifier une route mais qui n'en dégradera souvent pas la lisibilité.

E.3 Analyse cartographique de l'application des règles apprises

E.3.1 Qualité des résultats

Nous avons appliqué les règles apprises sur de nouvelles routes à traiter. D'autres résultats de cette application peuvent être vus en annexe VII. Les résultats cartographiques obtenus sont globalement satisfaisants, compte tenu des outils utilisés (cf. E.3.2 sur l'analyse des erreurs). La Figure 53 présente les résultats sur deux arcs routiers bien traités (issus des Alpes Maritimes). Les figures des pages 148 et 149 suivantes présentent les résultats sur une zone plus étendue. Ces résultats peuvent être comparés à ceux de GALBE sur la même zone (présentés dans la Figure 25 page 65). Globalement, les résultats du processus appris sont de qualité similaire à ceux de GALBE, avec un avantage au processus appris pour le traitement des zones complexes.

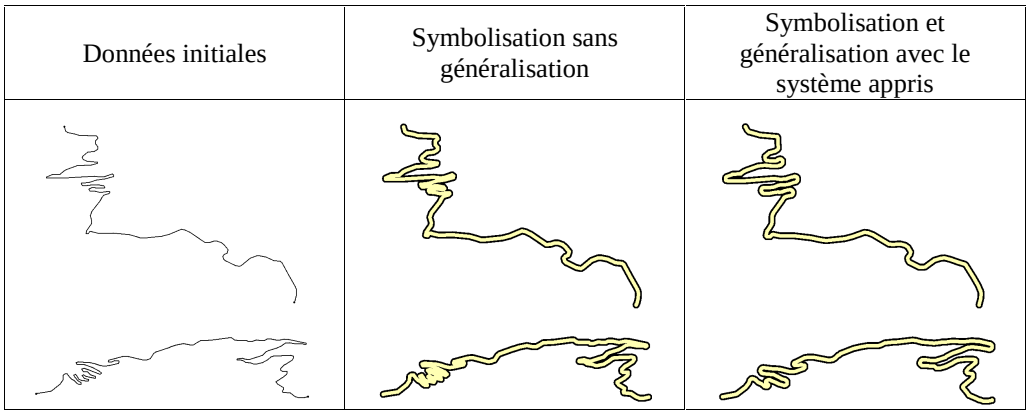


Figure 53. Résultats de l'apprentissage

E.3.2 Analyse des erreurs

Principales erreurs rencontrées

La Figure 54 présente les résultats sur une zone difficile à traiter. En particulier au centre de la zone, les routes sont très sinueuses, très proches les unes des autres, et une série d'épingles à cheveux est exceptionnellement longue.

Compte tenu de la difficulté de la zone, les résultats sont satisfaisants. Ils sont meilleurs que ceux de GALBE qui n'arrive pas du tout à traiter le centre de la zone et laisse inchangée la très longue série de virages. Cependant, une étude critique de ces résultats permet de mettre en évidence les problèmes typiques rencontrés lors de l'application des règles de généralisation cartographique apprises. Nous avons isolé sur la Figure 54 les endroits problématiques et nous avons représenté dans les cartons numérotés de 1 à 13 l'axe des données généralisées correspondantes. Tout d'abord, il existe quelques cas de trop forte généralisation : une série de virages serrés est un peu trop simplifiée (cas 8), ou des virages peu sinueux ont été trop lissés (cas 13). Ces deux cas peuvent cependant être considérés acceptables. Ensuite, il peut arriver que l'axe d'une route fasse un nœud (cas 5), ce qui crée un problème de lisibilité et surtout une erreur de topologie dans la base de données cartographique. Ce problème provient de la propagation des effets d'un algorithme sur le réseau qui peut être parfois inefficace (cf. B.4.4). Tous les autres problèmes sont des manques de généralisation. Les erreurs les plus fréquentes rencontrées sont des manques de caricature

qui ont pour conséquence que des virages empâtés subsistent (cas 1 et 6). En particulier, le symbole d'une route peut se chevaucher, créant ainsi l'impression trompeuse de l'existence d'un carrefour (cas 11). Les autres erreurs rencontrées sont des manques de déplacement et concernent les couples de routes proches. Soit deux routes non connectées se chevauchent ou se rapprochent trop l'une de l'autre, faisant ainsi croire à tort à la présence d'un carrefour (cas 4, 9 et 10). Soit l'intersection entre deux routes n'est pas clairement lisible (cas 2, 7 et 12), ce qui trompe la perception de l'emplacement et de la forme de l'intersection.

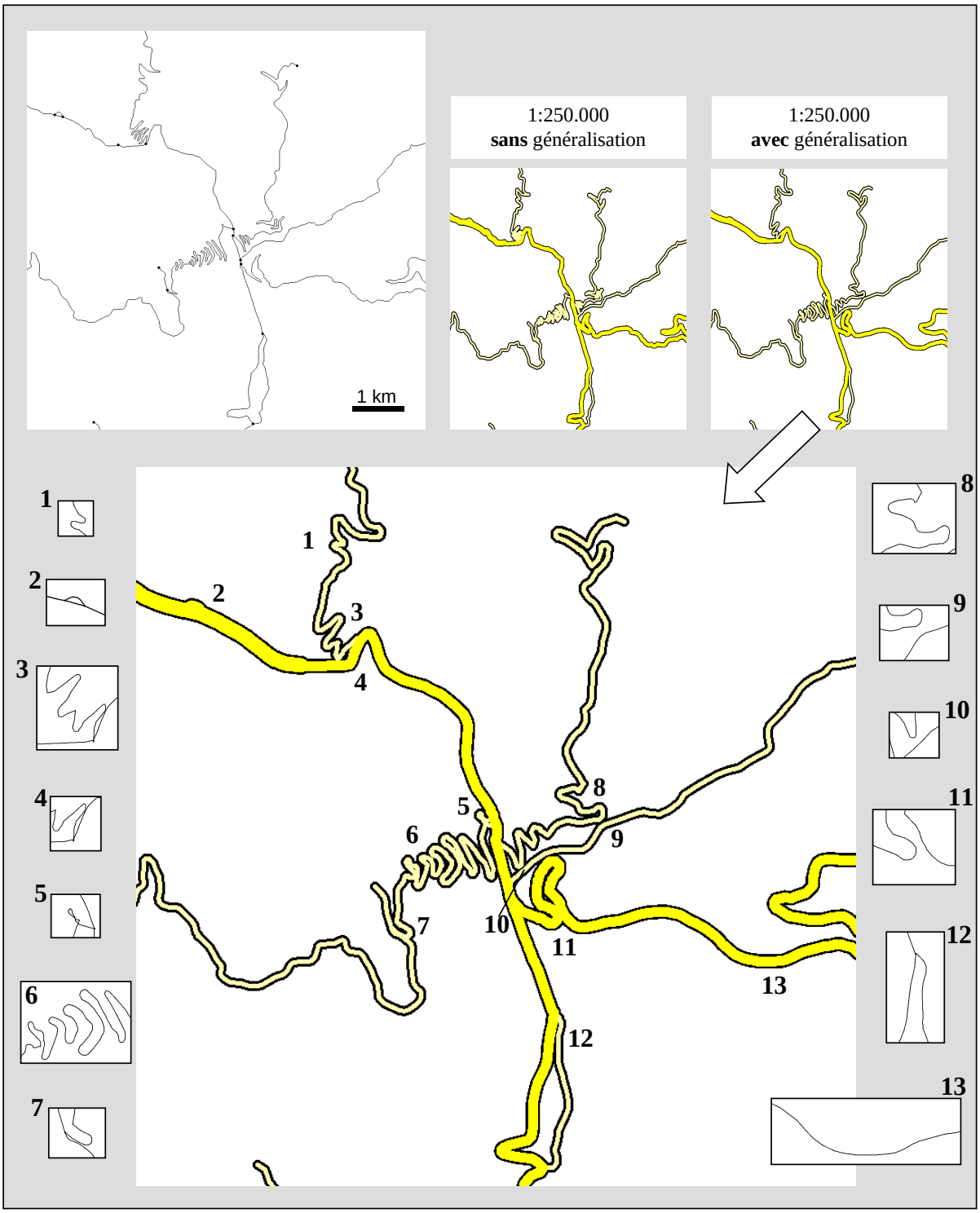


Figure 54. Problèmes cartographiques rencontrés lors de l'application des règles apprises

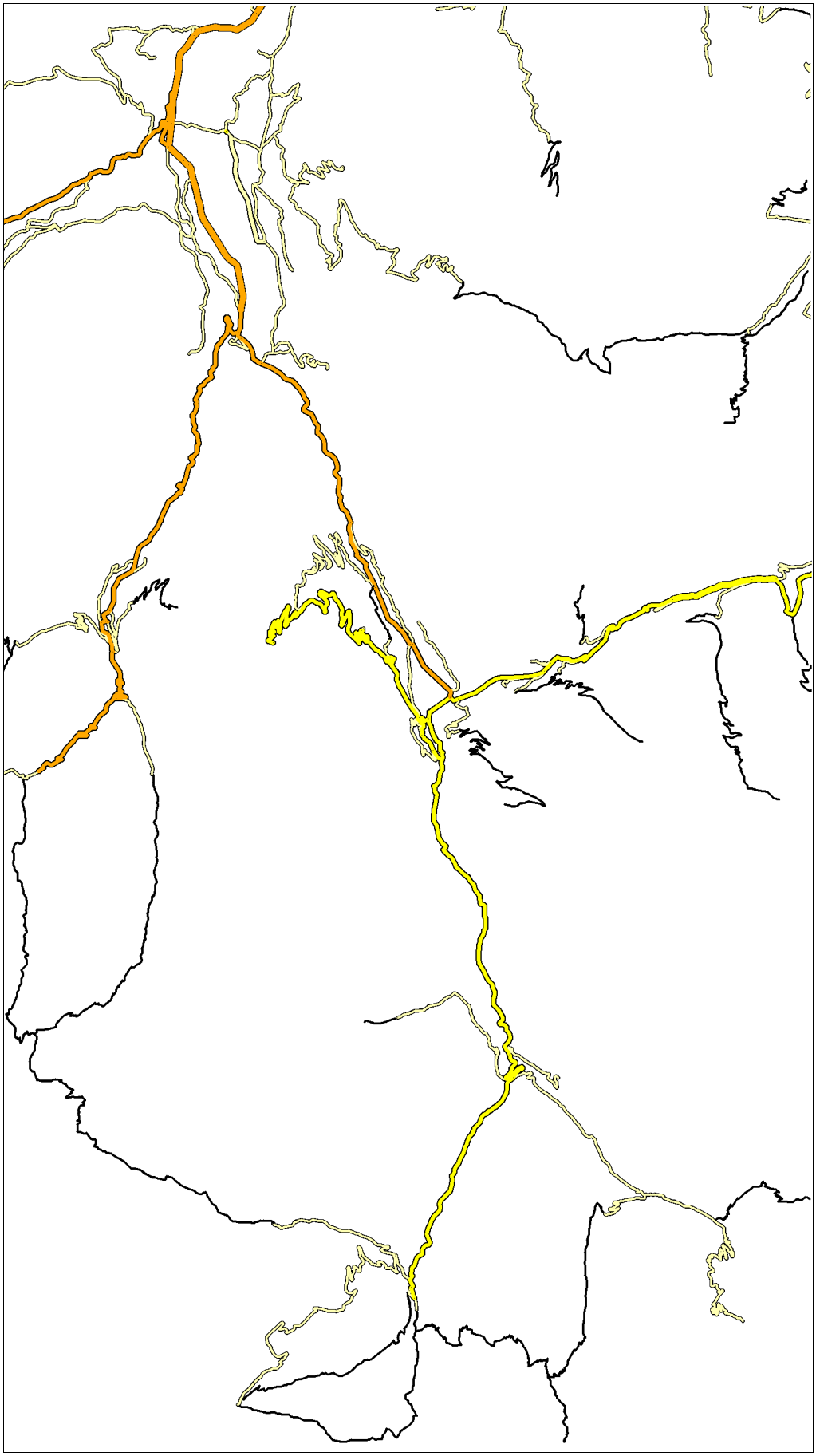


Figure 55. Extrait de la BDCarto, symbolisé au 1:250.000 avant traitement

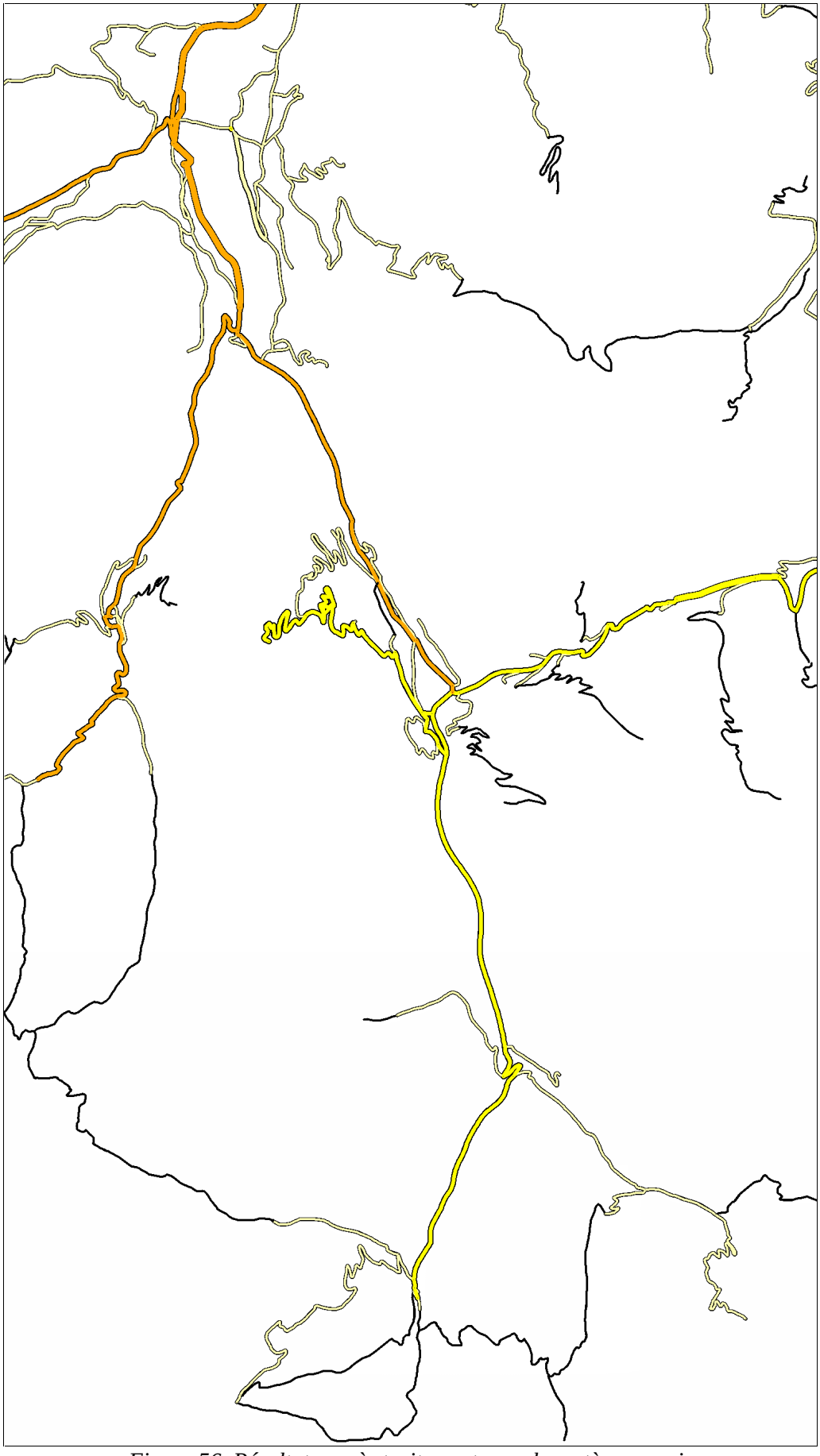


Figure 56. Résultats après traitement avec le système appris

Source des erreurs rencontrées

Nous avons analysé les décisions prises par le processus qui ont conduit à ces erreurs. La séparation des règles d'abstraction et de représentation est d'une grande utilité pour cette analyse. Il est en effet aisé de comprendre si le système s'est trompé lors de la phase d'abstraction ou de représentation, en comparant l'abstraction automatiquement déterminée et l'objet considéré.

L'analyse des résultats cartographiques et du raisonnement suivi par le système pour y parvenir permet d'identifier les sources des erreurs.

Ainsi, par exemple, l'analyse du cas présenté dans la Figure 57 a permis de remettre en cause la règle utilisée pour déterminer ce qu'est *un virage*. Le traitement automatique fournissait un résultat cartographique de mauvaise qualité (au centre de la figure). Nous avons alors analysé le raisonnement suivi par le système pour aboutir à ce résultat. La série de deux petits virages très serrés était considérée par le système, à tort, comme un seul virage ; l'algorithme *Faille Min* y a alors été appliqué par erreur. En effet, la règle appliquée dans le cas présent "**si** nombre virages ≤ 2 **et** base sur longueur ≤ 0,32 **alors** *Forme = virage*" est fausse. Puisque les règles sont compréhensibles, il est aisé de les corriger de manière interactive. Nous avons donc corrigé la règle défectueuse en y ajoutant la condition "**et** type empâtement = monolatéral". Après correction, le traitement automatique fournit un résultat de meilleure qualité (à droite sur la figure).

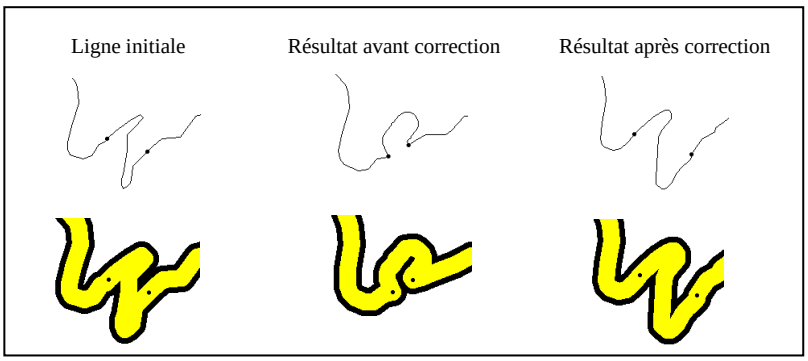


Figure 57. Avant et après correction de la règle de détermination d'un virage

En analysant ainsi quelques erreurs rencontrées, nous avons déterminé les causes les plus fréquentes des erreurs. Celles-ci sont présentées dans le Tableau 14 et commentées par la suite. Nous distinguons les erreurs provenant de l'apprentissage des règles, les erreurs dues aux outils utilisés (mesures ou algorithmes), et les erreurs dues à une mauvaise gestion des propagations (propagation sur le graphe de la déformation d'une partie de route).

		Sources des erreurs cartographiques		
		Erreur des règles	*Erreur dues aux outils*	*Mauvaise gestion des propagations*
Erreurs cartographiques rencontrées	Trop grande simplification	Le processus ne s'arrête pas assez vite. Choix d'une simplification au lieu de l'exagération.	*Schématisation* élimine parfois trop de virages si ceux-ci sont petits. Les mesures et algorithmes utilisés ne savent pas détecter ni traiter les virages peu sinueux.	
	Empâtement résiduel	Le processus s'arrête trop vite, ou choisit de simplifier au lieu de caricaturer	Certains algorithmes ne sont pas optimaux sur les séries de petits virages ou sur les séries très complexes. Un algorithme est annulé car il a crée des intersections internes, mais aucun autre algorithme est applicable ; aucune action n'est donc réalisée.	Un empâtement éliminé re-apparaît lors de la propagation de l'écartement des objets voisins
	Intersection topologique interne			Propagation de l'écartement des objets voisins inefficace
	Superposition entre les routes	L'environnement est mal déterminé	Manque de mesures et d'outils de résolution des conflits entre objets	Propagation de l'écartement des objets voisins inefficace
	Superposition interne du symbole	L'environnement est mal déterminé. Un algorithme demandant trop d'espace est choisi à tort	Manque d'algorithmes pour traiter ces cas.	Propagation de l'écartement des objets voisins inefficace

Tableau 14. Causes des erreurs cartographiques rencontrées

Compte tenu des outils utilisés, les règles sont globalement satisfaisantes, elles seraient néanmoins améliorées si des mesures plus pertinentes pour qualifier la granularité et l'environnement d'un objet étaient utilisées. Les algorithmes utilisés font relativement peu d'erreurs graves. Les cas sur lesquels les algorithmes sont les moins efficaces sont les séries de virages très complexes où les orientations des virages sont très disparates.

Les erreurs les plus fréquentes sont dues à une mauvaise gestion des propagations des effets de bord des algorithmes, qui est un problème peu traité dans notre système : dans un premier temps une partie d'un objet est correctement traitée, mais par la suite la propagation du traitement de parties voisines dégrade cette partie (Figure 58).

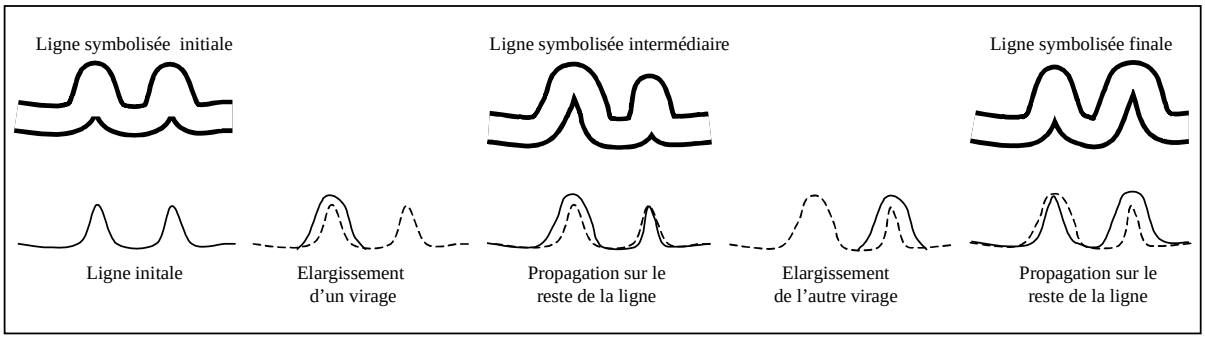


Figure 58. Empâtement résiduel dû à la propagation

Deux pistes sont envisageables pour améliorer les erreurs dues aux propagations. D'une part, il faudrait améliorer l'algorithme de propagation des déformations d'un objet sur le reste du réseau (par exemple en intégrant les travaux réalisés sur ce sujet à l'université de Zurich [Barrault, Bader et Weibel 2000 ; Bader et Barrault 2000]). D'autre part, il faudrait modifier le

moteur très simple utilisé pour enchaîner les algorithmes. En effet ces erreurs de propagation sont révélatrices des limites de ce moteur :

- Une fois qu'un objet a été correctement traité il est éliminé de la liste des objets à traiter, même s'il est ensuite déformé à cause de la propagation de traitements sur les objets voisins.
- Les objets sont traités dans un ordre aléatoire. Il serait sûrement plus judicieux de traiter les objets qui risquent de déformer leur voisinage en premier, voire d'apprendre en fonction des attributs abstraits un ordre de traitement plus optimal.
- L'approche est totalement descendante : une fois les objets découpés ils ne sont jamais traités à nouveau dans leur ensemble.

Un moteur plus complexe a été développé durant le projet AGENT, en particulier pour la gestion des routes [Duchêne 2001]. Il serait utile de coupler ces travaux avec nos résultats, mais ceci n'est pas immédiat : certaines des règles apprises pourrait être directement insérées dans le prototype AGENT, mais d'autres règles sont moins compatibles avec le modèle utilisé dans AGENT. Ceci est le cas en particulier des règles prenant en compte la transformation qu'un objet a subi à l'étape précédente [Mustière et Duchêne 2001].

Conséquence des erreurs sur une utilisation en production

En dehors des problèmes de superposition entre objets qui n'ont pas été traités, les erreurs les plus fréquentes sont les manques de caricature qui entraînent des restes d'empâtement. Les résultats sont beaucoup plus souvent sous-généralisés que sur-généralisés. Ceci un avantage important dans le cadre d'une utilisation en production du processus. En effet, comme nous avons défini une mesure de détection des empâtements, nous pouvons automatiquement détecter ces erreurs qui demandent une retouche interactive des résultats. Cette identification des zones à retoucher permet un gain de temps considérable lors de la retouche, par rapport à un contrôle exhaustif des résultats. Notons que GALBE possède le même avantage d'effectuer beaucoup plus d'erreurs par défaut de généralisation que d'erreurs par excès de simplification des formes.

E.3.3 Convergence et temps de calcul

L'arrêt du processus étant déterminé par les règles apprises, il existe un risque de non convergence si les règles sont défectueuses. En pratique nous avons contrôlé cela en autorisant un nombre maximum de transformations sur un même objet. Ce nombre était fixé à 15 dans nos tests. Néanmoins, si un objet est découpé en plusieurs parties durant le processus, nous considérons ses parties comme de nouveaux objets à traiter, qui pourront à leur tour subir plusieurs transformation. Ainsi une ligne initiale, si elle est découpée en cours de traitement, peut subir plus de 15 transformations au total. En pratique, ce nombre maximal de 15 transformations a rarement été atteint par le processus appris.

En ce qui concerne les temps de calcul, nous avons moins intensément testé les règles apprises que le processus GALBE. Nous ne fournissons donc pas de temps de calcul précis. Ce temps de calcul paraît néanmoins environ 10 fois plus lent que celui de GALBE. Ceci est dû à plusieurs raisons :

- Un ensemble de mesures est calculé avant chaque opération dans le cas du processus appris, alors que GALBE ne calcule qu'une seule mesure d'empâtement.
- Le processus appris peut effectuer plus de transformations que GALBE sur un même objet, puisque le nombre de transformations réalisées sur une même route n'est pas contraint dans le processus appris.
- Le code du processus appris n'a pas été optimisé contrairement à celui de GALBE. En particulier différentes mesures calculées sur un objet peuvent effectuer chacune un même calcul intermédiaire coûteux en temps (par exemple, le bord du symbole d'une ligne est calculé plusieurs fois).

Nous estimons qu'une optimisation du code pourrait au moins diminuer de moitié les temps de calcul. Ce temps de calcul reste raisonnable dans le cadre d'une utilisation complètement automatique du processus. Mais, actuellement, la relative lenteur des calculs interdit l'utilisation du processus appris en tant qu'outil interactif, à déclencher et valider interactivement.

E.3.4 Généricité de lieu et d'échelle

Les exemples ont été recueillis dans les Alpes, pour créer des cartes au 1:250.000. Afin de juger l'influence de ce cadre sur les règles apprises, les règles ont tout d'abord été appliquées sur différentes zones du réseau routier français (Alpes, Pyrénées, Massif Central, et différentes zone de plaines). Les résultats sont de qualités similaires sur les différentes zone de montagne étudiées, ils sont de meilleure qualité en plaine (mais la généralisation des arcs routiers en plaine est un problème simple). La restriction géographique du recueil des exemples n'a donc pas biaisé le résultat.

De même les règles apprises ont été appliquées pour créer des cartes à différentes échelles (du 1:100.000 au 1:1M). Les résultats sont de qualités similaires pour des échelles allant environ du 1:100.000 au 1:500.000. Par contre les résultats sont de moins bonne qualité pour des échelles plus petites. A de telles échelles, très peu de virages doivent être conservés, et une approche différente de la généralisation est sûrement nécessaire : les algorithmes que nous utilisons cherchent globalement à éliminer le surplus de détail, alors qu'aux petites échelles il faudrait prendre l'approche complémentaire de ne représenter que les informations de première importance. La Figure 59 présente les résultats sur deux routes de montagne à différentes échelles. Le résultat de la deuxième route à l'échelle du 1:1M est particulièrement de mauvaise qualité, à cause de la très forte déformation de la route traitée.

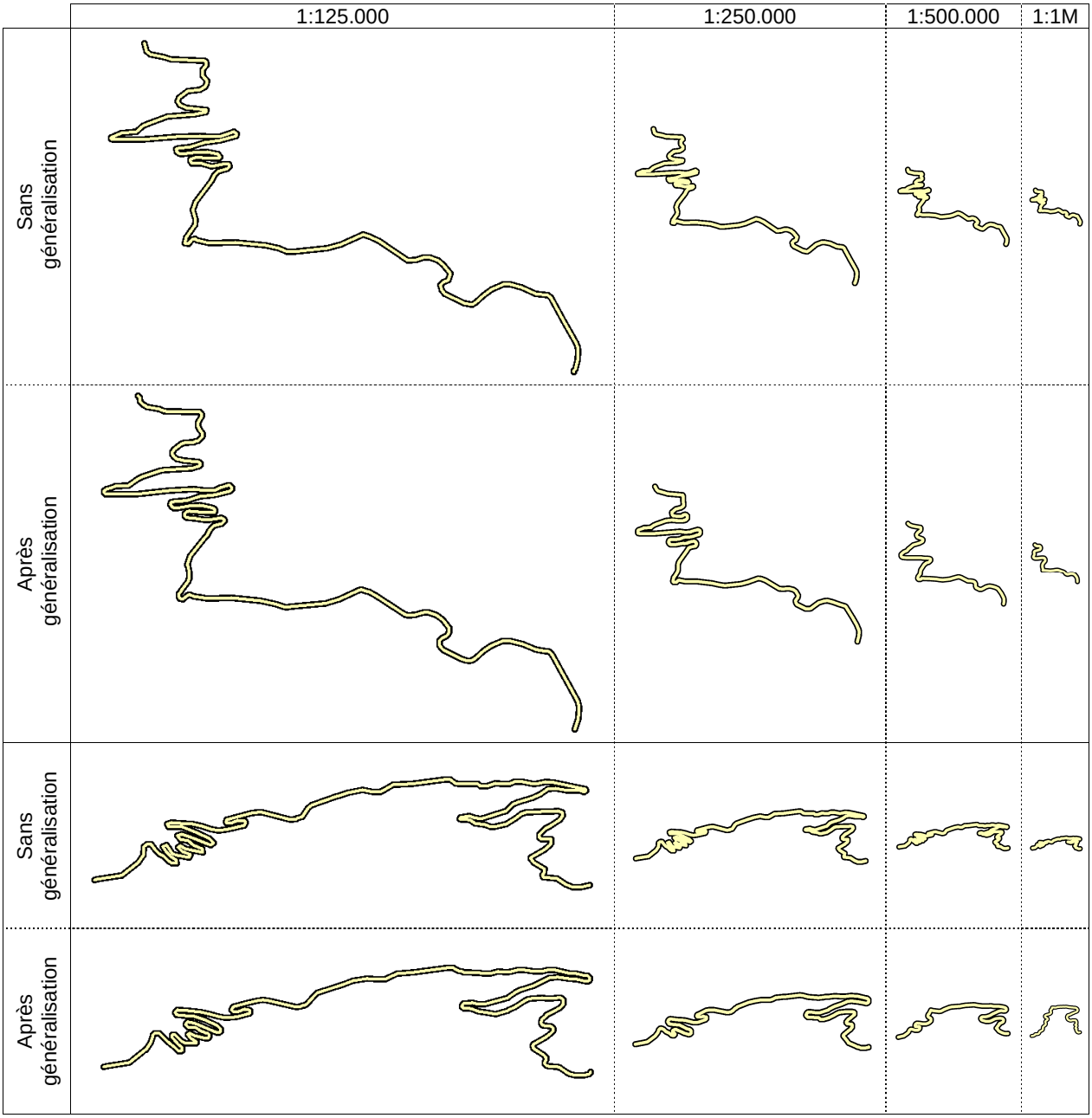


Figure 59. Résultats à différentes échelles

E.4 Intérêt de la méthode de résolution de problème

L'apprentissage en plusieurs étapes complexifie le recueil des exemples mais il permet d'obtenir des règles bien organisées. Dans cette partie, nous évaluons l'intérêt d'avoir décomposé la méthode de résolution de problème, vis-à-vis de la qualité des règles apprises.

E.4.1 Comparaison à l'apprentissage direct

Pour évaluer l'intérêt de la décomposition de l'apprentissage en plusieurs étapes, nous comparons ses résultats avec ceux d'un apprentissage direct, reliant directement le choix de l'algorithme de généralisation aux mesures utilisées pour qualifier un objet. Lors d'un apprentissage direct, RIPPER crée les règles présentées ci-dessous (Règles 7).

CHOIX DE L'ALGORITHME

si Type opération précédente = modification géométrique **et** longueur empâtement ≤ 0,6 **alors** *Choix Algo = Arrêt*
si base sur longueur ≥ 0,97 **alors** *Choix Algo = Arrêt*
si Algorithme précédent = Plâtre **alors** *Choix Algo = Arrêt*
si taille max virages ≤ 0,06 **alors** *Choix Algo = Arrêt*
si homogénéité empâtement ≤ 0,76 **et** nb arcs proches par longueur ≤ 0,3 **alors** *Choix Algo = Découpage empâtement*
si longueur base ≤ 0,29 **alors** *Choix Algo = Faille Min*
si taille moyenne virages ≥ 0,63 **alors** *Choix Algo = Faille Max*
si longueur empâtement ≥ 11,7 **alors** *Choix Algo = Schématisation*
si longueur empâtement ≥ 2,7 **et** résistance lissage ≤ 0,67 **alors** *Choix Algo = Schématisation*
si pourcentage empâtement ≥ 0,96 **et** résistance lissage ≥ 0,75 **alors** *Choix Algo = Accordéon*
si taille virages moyenne sur max ≤ 0,53 **et** nombre arcs proches par largeur ≥ 6,7 **alors** *Choix Algo = Plâtre*
si granularité ≥ 0,33 **et** longueur ≥ 2,2 **alors** *Choix Algo = Plâtre*
sinon *Choix Algo = Lissage Gaussien*

Règles 7. Résultat de l'apprentissage direct

Ces règles sont moins nombreuses que celles apprises en plusieurs phases, mais elles sont néanmoins beaucoup moins faciles à interpréter et à modifier. Par exemple, il est difficile d'interpréter précisément la règle "**si** longueur empâtement ≥ 2,7 **et** résistance lissage ≤ 0,67 **alors** *Choix Algo = Schématisation"*. Cette règle sous-entend que l'algorithme *Schématisation* est choisi quand l'empâtement est important (longueur empâtement ≥ 2,7) et quand la forme est relativement complexe (résistance lissage ≤ 0,67 signifie que le nombre de virage après lissage de la ligne est grandement diminué), mais ceci n'est pas immédiat à comprendre.

Mais surtout, ces règles sont beaucoup plus difficiles à faire évoluer. Si une nouvelle mesure de description d'une route est conçue, il est difficile de l'introduire dans cette base de règles qui relie directement les algorithmes aux mesures. Alors que dans les règles apprises par les abstractions, l'introduction d'une nouvelle mesure n'influence que les règles d'abstraction, indépendamment des règles choisissant l'algorithme (cf. D.5.2).

Qualité des règles

Les règles issues de l'apprentissage direct ont un taux d'erreur estimé à 41% (dont 30% d'erreurs importantes où un algorithme non applicable est choisi). Ces règles sont donc moins efficaces que celles de l'apprentissage décomposé en plusieurs inférences : 29% d'erreurs dont 17% d'erreurs importantes.

Même si chaque inférence apprise dans l'apprentissage par les abstractions produit des erreurs, l'enchaînement de ces inférences produit moins d'erreurs que les hypothèses issues d'un apprentissage direct.

Le Tableau 15 suivant présente la matrice de confusion estimée par validation croisée sur le choix de l'algorithme. Les chiffres entre parenthèses rappellent les confusions obtenues par un apprentissage en plusieurs phases.

		Classes des exemples fournies par l'expert								
		Arrêt	Découp. Empât.	Lissage Gauss.	Plâtre	Accordéon	Schémat.	Faille Max	Faille Min	Total
Classes Apprises	Arrêt	**54 (55)**	1 (1)	4 (2)	4 (1)	1 (1)				64 (60)
	Découp. Empât.	1 (2)	**3 (6)**		1 (1)		1 (1)			6 (10)
	Lissage Gaussien	5 (6)	3 (0)	**9 (15)**	5 (8)	1 (2)				23 (28)
	Plâtre	2 (0)		4 (1)	**3 (5)**					9 (9)
	Accordéon		0 (1)		1 (1)	**3 (5)**	4 (3)	2 (0)	0 (1)	10 (11)
	Schématisation		1 (0)			5 (1)	**5 (6)**			11 (7)
	Faille Max							**1 (5)**	3 (4)	4 (9)
	Faille Min	1 (0)		1 (0)	2 (0)	0 (1)		4 (2)	**5 (3)**	13 (6)
	Total	63	8	18	16	10	10	7	8	**140**

Tableau 15. Matrice de confusion estimée du choix de l'algorithme par apprentissage direct

Application des règles

Les différences de qualité des règles apprises avec ou sans décomposition de la méthode de résolution de problème sont confirmées par une évaluation cartographique de celles-ci. L'application des règles apprises par apprentissage direct fournit de nettement moins bons résultats cartographiques que ceux de l'approche par apprentissage séparé des différentes étapes du raisonnement (cf. Figure 60, d'autres résultats peuvent être vus en annexe VII-2).

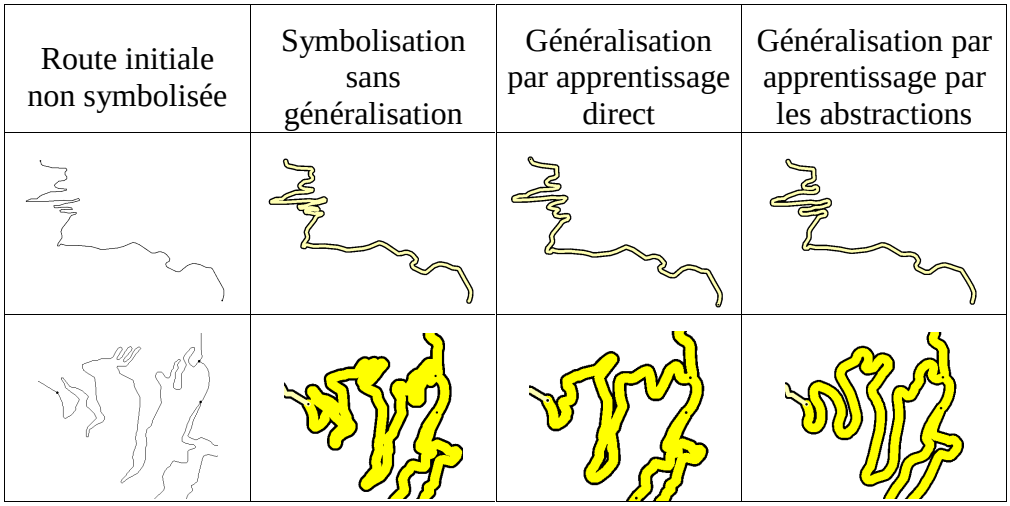


Figure 60. Résultats avec ou sans décomposition de la méthode de résolution de problème

E.4.2 Influence de chaque étape

Le découpage de la méthode de résolution de problème en plusieurs étapes permet d'améliorer le résultat de l'apprentissage. Pour évaluer l'intérêt de chaque étape en termes de qualité des résultats, nous effectuons deux séries de tests. Nous évaluons l'intérêt de chaque étape en étudiant, d'une part, ce que son ajout apporte par comparaison avec un apprentissage direct et, d'autre part, ce que son élimination fait perdre par comparaison avec la méthode complète. Les résultats de ces tests sont présentés en Figure 61. Par exemple, quand on introduit seulement l'inférence d'abstraction des mesures dans la méthode de résolution de problème, l'erreur estimée est de 31%. L'apport de cette inférence est alors estimé à 9% par rapport à l'apprentissage direct (erreur = 40%). De même, quand on élimine uniquement cette inférence d'abstraction par rapport à la méthode complète, l'erreur estimée est de 39%. L'apport de cette étape est alors estimée à 10 % dans l'apprentissage avec la méthode complète (erreur = 29%).

Pour simplifier l'étude, nous considérons comme une seule étape les différentes inférences de la détermination de l'opération (i.e. choix de l'action, du type d'opération et de l'opération).

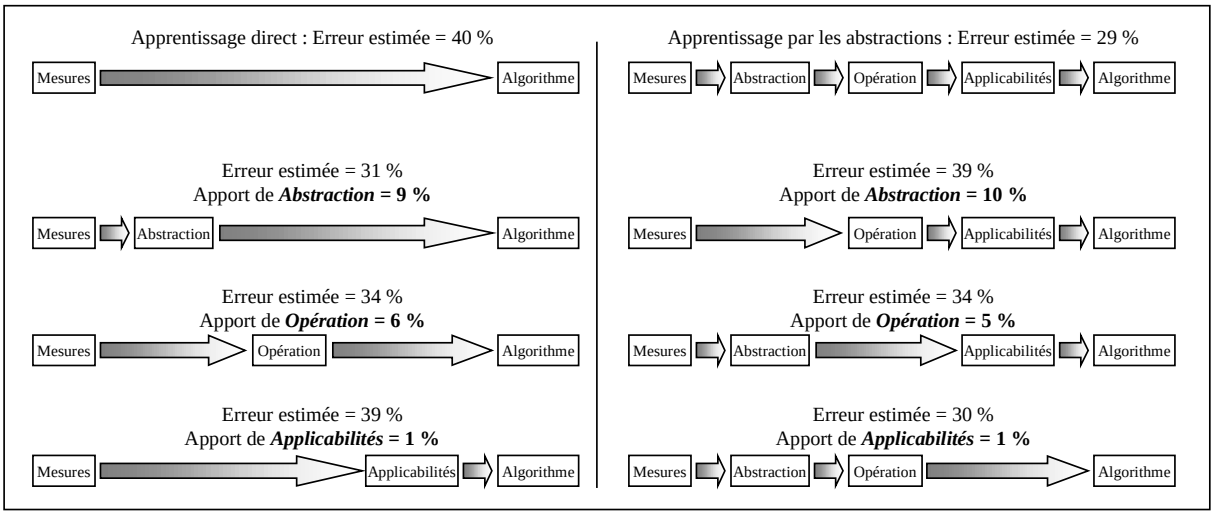


Figure 61. Apport des différentes étapes

Les deux séries de tests sont cohérentes : les apports de chaque étape évalués, d'une part comme une perte par rapport à l'apprentissage par les abstractions et, d'autre part comme un gain par rapport à l'apprentissage direct, sont équivalents. L'étape présentant le plus d'intérêt est l'étape de d'abstraction des mesures. Par contre l'étape de détermination des applicabilités apporte très peu, en terme de qualité des règles apprises. Comme par ailleurs cette étape n'aide pas non plus énormément à la compréhensibilité des hypothèses, elle pourrait être éliminée de la méthode de résolution de problème. Cependant, posséder la liste des algorithmes applicables sur un objet est utile lors de l'application des règles : si jamais l'algorithme appliqué produit des erreurs, un autre algorithme applicable peut être essayé (cf. D.1.2). Ceci pourrait être encore plus utile si une véritable phase de validation de l'algorithme appliqué était introduite dans le moteur d'enchaînement des règles apprises.

E.4.3 Intérêt de l'étape d'abstraction des mesures

Origine de l'intérêt de l'étape d'abstraction

L'étape d'abstraction est particulièrement utile, nous l'évaluons donc un peu plus en détail. La décomposition de l'apprentissage en deux étapes d'abstraction et de représentation permet d'introduire des connaissances du domaine à deux niveaux lors de l'apprentissage. D'une part, elle contraint l'apprentissage en forçant les hypothèses à décrire de manière abstraite les objets manipulés avant de choisir un algorithme à appliquer. D'autre part, elle contraint l'apprentissage en forçant à n'utiliser qu'un sous-ensemble des mesures disponibles (celles jugées pertinentes) pour apprendre chaque attribut abstrait. Pour évaluer plus précisément l'intérêt de l'introduction de l'étape d'abstraction, nous avons appris à nouveau l'ensemble des inférences sans spécifier les mesures potentiellement pertinentes pour déterminer chaque attribut abstrait. L'enchaînement des hypothèses apprises possède un taux d'erreur estimé à 31%, c'est-à-dire très proche de celui des hypothèses apprises en spécifiant les mesures pertinentes pour chaque abstraction (29%). L'intérêt de l'introduction de l'étape d'abstraction s'explique donc peu par le fait de spécifier les mesures potentiellement pertinentes pour chaque abstraction.

On peut donc en déduire que l'intérêt principal de l'introduction de l'étape d'abstraction est de contraindre les hypothèses à décrire de manière abstraite les observables manipulées avant de déterminer la classe. Cette description abstraite est définie par les connaissances du domaine et est une manière d'introduire un biais de représentation des hypothèses.

Qualité de l'abstraction

On peut évaluer la qualité de l'étape d'abstraction à deux niveaux : l'abstraction est-elle bien apprise à partir des mesures ? l'abstraction permet-elle de bien choisir l'algorithme de généralisation à appliquer ?

La réponse à la première question a déjà été donnée dans la partie E.2.1. Les abstractions *granularité* et *conflit externe* sont mal déterminés, et les autres attributs abstraits sont déterminés avec des taux d'erreur estimés entre 4 et 20% selon les attributs.

Pour évaluer la qualité de l'abstraction pour le choix de l'algorithme, nous estimons par validation croisée le taux d'erreur produit par l'enchaînement des étapes de représentation (détermination de l'opération, des applicabilités et de l'algorithme). Autrement dit, nous estimons le taux d'erreur des règles apprises, en supposant qu'aucune erreur n'est réalisée pendant l'étape d'abstraction.

Si nous n'utilisons pas l'attribut abstrait *granularité*, l'enchaînement des étapes de *représentation* a un taux d'erreur estimé à 18%, se décomposant en 10% d'erreurs importantes quand un algorithme non applicable est choisi et à 8% d'erreurs bénignes où un algorithme néanmoins applicable est choisi par erreur. Si on utilise l'attribut abstrait *granularité*, le taux d'erreurs descend à 14%, dont seulement 5% d'erreurs importantes où un algorithme non applicable est choisi. Ces résultats peuvent être comparés au résultat de l'apprentissage complet, y compris les abstractions, dont nous rappelons le taux d'erreur estimé : 29% d'erreurs dont 17% d'erreurs importantes où un algorithme non applicable est choisi (cf. E.2.6 page 144)

On peut déduire de ces expérimentations que sur les 29% d'erreurs réalisées sur le choix de l'algorithme (17% d'erreurs importantes), 18% sont dues à la phase de *représentation* (10% d'erreurs importantes). On peut aussi en déduire que si des mesures pertinentes étaient utilisées pour mesurer la *granularité*, et donc si celle-ci était utilisée dans l'étape de *représentation*, cette étape ne ferait que 11% d'erreurs (5% d'erreurs importantes).

E.5 Bilan des expérimentations

La Figure 62 résume sur une route les résultats cartographiques des expérimentations réalisées, en présentant sur une route les résultats à différentes échelles de GALBE, de l'apprentissage direct, et de l'apprentissage par les abstractions. Les meilleurs résultats obtenus sont en général ceux de l'apprentissage par les abstractions. Celui-ci est parfois moins efficace que GALBE mais l'inverse est plus courant, en particulier sur les objets les plus difficiles à traiter. Il est par contre rare que l'apprentissage direct fournisse des résultats meilleurs que ceux de GALBE ou de l'apprentissage par les abstractions.

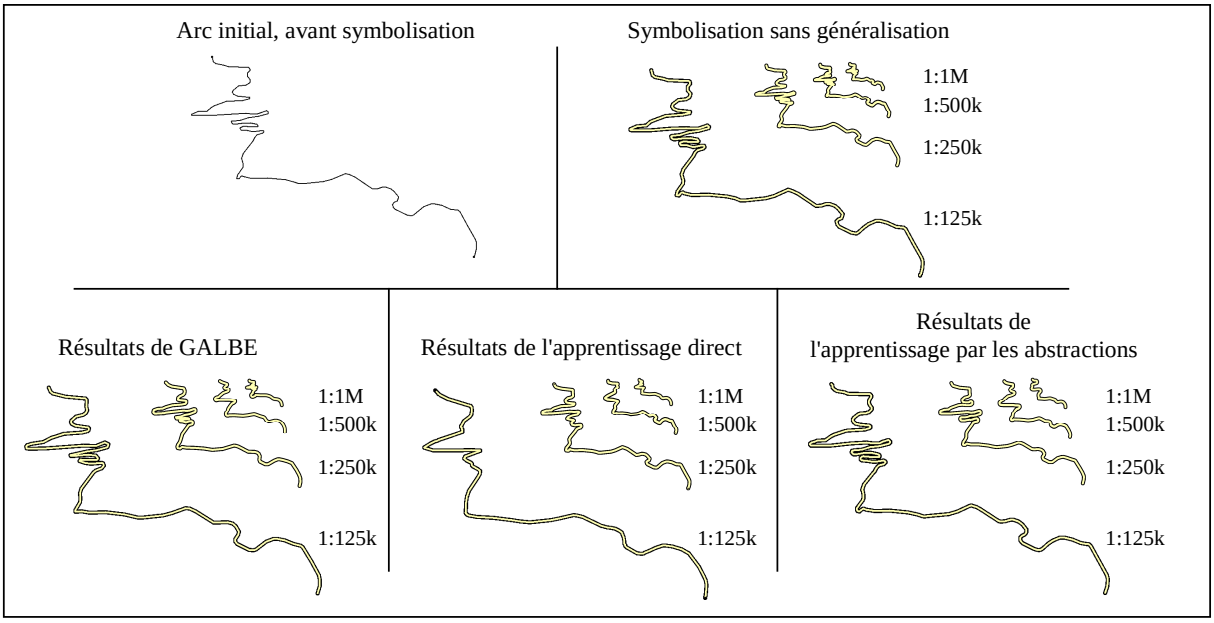


Figure 62. Généralisation à différentes échelles et avec différentes techniques

L'apprentissage par les abstractions a permis d'identifier des mesures pertinentes vis-à-vis de notre problème. Pour cela, il a privilégié les mesures les plus stables. Il a également mis en évidence l'absence dans nos expérimentations de mesures pertinentes pour qualifier la granularité d'une route et les conflits graphiques d'une route avec les routes voisines. En conséquence, il a été impossible d'apprendre quand réaliser des opérations de déplacement. Dans une moindre mesure, il a également mis en évidence le manque de mesure pour qualifier la densité de l'environnement d'une route.

La définition d'une méthode de résolution de problème a permis d'améliorer l'organisation et la compréhensibilité des règles apprises par rapport à un apprentissage direct. Elle a aussi permis d'améliorer sensiblement la qualité des règles apprises. Cette méthode de résolution de problème a été un moyen de contraindre l'apprentissage par les connaissances du domaine. Cela a permis de simplifier la résolution de notre problème d'apprentissage rendu difficile par le faible nombre d'exemples utilisés. En contrepartie, cette phase d'abstraction a alourdi le recueil des exemples.

Par rapport à GALBE, les connaissances manipulées par le système appris sont beaucoup mieux organisées et compréhensibles. L'identification des limites du système et ses capacités d'évolution sont aussi plus faciles. De plus, l'apprentissage a permis l'introduction de mesures de description des objets plus pertinentes que celles utilisées par GALBE. Les résultats de l'application du système appris sont en conséquence sensiblement plus efficaces que ceux de GALBE.

■

CONCLUSIONS ET PERSPECTIVES

◆ Dans le cadre de la généralisation cartographique, l'apprentissage à partir d'exemples peu nombreux et de taille importante est possible à condition de le contraindre fortement par les connaissances du domaine. Pour cela, nous avons mis au point une méthodologie pour guider le recueil des connaissances, basée sur la définition d'une méthode de résolution de problème contenant en particulier une étape d'abstraction de la description des objets manipulés.

◆ Pour que cette méthodologie soit utilisée au mieux, nous évoquons des pistes pour mieux qualifier les objets manipulés et pour mieux utiliser les règles apprises. Nous définissons également quelques directions de recherche pour améliorer la méthodologie proposée.

Rappelons la question que nous nous posions au début de cette thèse : est-il possible d'apprendre automatiquement, à partir de peu d'exemples, des connaissances efficaces et compréhensibles pour guider l'utilisation des algorithmes de généralisation cartographique ?

La réponse que nous donnons à cette question tout au long de cette thèse peut être résumée ainsi : l'apprentissage est possible, à condition de contraindre fortement le processus d'apprentissage supervisé. Ceci se réalise en contraignant le raisonnement à suivre pour résoudre le problème traité, et plus précisément en contraignant ce raisonnement à s'appuyer sur une description abstraite des objets manipulés.

Nous détaillons cette réponse en décrivant ci-dessous les apports de notre thèse ainsi que les enseignements que l'on peut en tirer. Nous concluons ensuite en envisageant les directions de recherche qui permettraient d'enrichir ce travail.

i. Apports

Outils de généralisation pour le traitement des routes

Un premier apport de notre travail est l'enrichissement de la boite à outils des algorithmes de généralisation cartographique.

Nous avons tout d'abord défini un outil de détection et de qualification des empâtements d'une ligne symbolisée (cf. B.3.2). Cet outil est basé sur la comparaison de la ligne stockée dans la base de données (son axe) avec sa représentation graphique (son symbole). Plus précisément, une ligne est considérée empâtée quand son axe est trop éloigné des bords de son symbole. Cet outil nous paraît particulièrement utile pour guider la généralisation. Nous l'avons conçu dans le but de focaliser sur des espaces de travail adaptés aux algorithmes dont nous disposons. Il est au centre des processus de généralisation des routes définis dans nos travaux (GALBE et le processus appris), et la qualité cartographique des résultats obtenus nous conforte dans l'idée que l'empâtement est une notion centrale pour guider la généralisation des objets linéaires symbolisés.

Cet outil de détection des empâtements est également utile pour évaluer la qualité des résultats d'une généralisation cartographique : il permet d'évaluer la lisibilité d'une ligne symbolisée. Cette lisibilité est un des critères d'évaluation d'une carte, avec l'écart à la réalité (la carte rend elle bien compte de la réalité géographique ?) et la pertinence du contenu (la carte contient elle les informations suffisantes et nécessaires au besoin ?).

Nous avons également conçu deux algorithmes pour caricaturer un virage : *Faille Max* et *Faille Min* (cf. B.4.2). Ces algorithmes sont très ciblés, puisqu'ils ne s'appliquent que sur un seul virage. Cette focalisation extrême en limite le champ d'application, et ces algorithmes ne sont utiles que si ils sont utilisés conjointement à de nombreux autres algorithmes. En contrepartie, ces outils sont simples et leurs effets sont faciles à appréhender.

Enfin, nous avons mis au point deux processus complets de généralisation des routes isolées : GALBE (cf. chapitre B) et le processus issu des règles apprises par les abstractions (cf. chapitre E). GALBE est opérationnel et a été utilisé dans les services de production de l'IGN. Le processus appris est plus efficace que GALBE mais mériterait d'être optimisé pour être

utilisé dans un contexte de production. A travers ces processus, nous avons proposé divers moyens d'enchaîner différentes briques réalisées jusqu'alors pour la généralisation des routes, en particulier lors des recherches antérieures sur ce sujet au laboratoire COGIT [Plazanet 96 ; Fritsch 97 ; Lecordix, Plazanet et Lagrange 97]. Nous nous sommes concentrés sur la tâche de savoir choisir, à un moment donné du processus de généralisation, l'algorithme de généralisation à appliquer sur un objet donné. Les résultats cartographiques sont très satisfaisants. Ils trouvent leurs limites, d'une part dans le manque de mesures d'analyse spatiale et, d'autre part, dans le besoin d'enrichir le moteur enchaînant les algorithmes de généralisation.

Méthodologie de recueil de connaissances

Nous considérons l'automatisation de la généralisation cartographique comme l'application successive de différents algorithmes à différents niveaux d'analyse [Brassel et Weibel 88 ; Ruas et Plazanet 96 ; Ruas 99] (cf. A.3.3). Dans ce contexte, nous proposons une méthodologie pour recueillir les connaissances nécessaires au guidage des algorithmes de généralisation cartographique pour un type d'objet donné (cf. chapitre D). Cette méthodologie, résumée ci-après, est basée sur la rencontre de plusieurs idées :

- L'*abstraction des exemples* permet de faciliter l'apprentissage automatique supervisé, [Giordana et Saitta 90 ; Saitta et Zucker 98] (cf. D.3.3).
- Contraindre l'apprentissage par les connaissances du domaine manipulé permet d'en améliorer les résultats, en termes de lisibilité et de pouvoir de prédiction. Pour cela, on peut décomposer un problème d'apprentissage en plusieurs sous-problèmes plus simples par l'intermédiaire d'une *méthode de résolution de problème* adaptée au domaine traité [Ganascia, Thomas et Laublet 93 ; Thomas 96] (cf. C.4).
- Décomposer l'apprentissage en y introduisant une étape d'abstraction dirigée par les connaissances du domaine doit donc améliorer à la fois la lisibilité et la qualité prédictive des résultats de l'apprentissage.

La méthodologie proposée est basée sur la définition, préalablement à l'apprentissage, d'une méthode de résolution de problème. Nous proposons que cette méthode de résolution prenne la forme générale présentée dans la partie D.4 et rappelée en Figure 63.

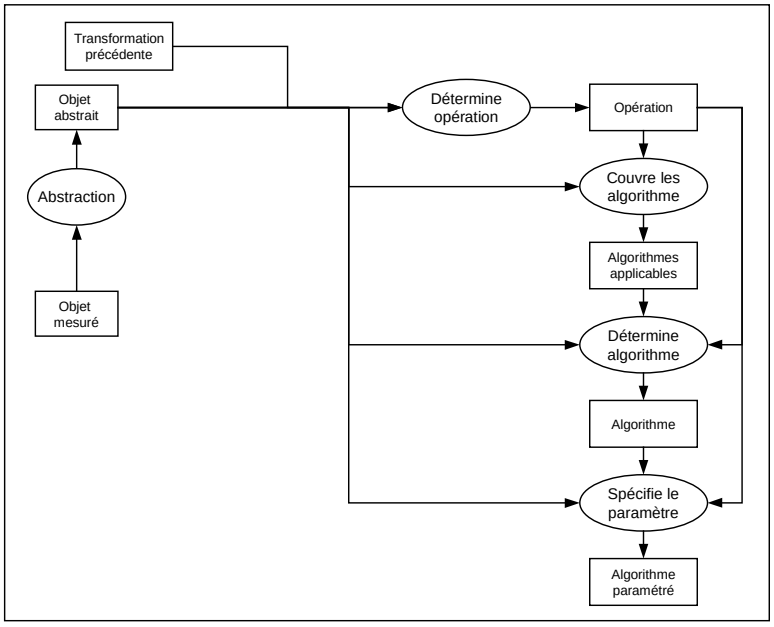


Figure 63. Méthode de résolution de problème adaptée à la généralisation cartographique

Pour définir précisément chaque étape de la méthode de résolution de problème, un ensemble de choix doivent être faits :

- Un expert choisit un ensemble d'attributs abstraits permettant de décrire l'objet dans le but de sa généralisation (taille, forme, lisibilité, etc.) ainsi que les valeurs possibles de ces attributs.
- Un expert choisit un ensemble de mesures permettant de qualifier l'objet, en les associant aux attributs abstraits pour lesquels ces mesures sont estimées potentiellement pertinentes.
- Un expert choisit un ensemble d'algorithmes adaptés au traitement du type d'objet considéré. Il organise ces algorithmes selon les opérations qu'ils réalisent. Cette organisation peut prendre la forme d'un treillis à plusieurs niveaux.

Une fois cette méthode de résolution de problème définie, nous proposons la démarche suivante pour recueillir les connaissances nécessaires à chaque étape de la méthode de résolution :

- Un expert choisit une ou plusieurs zones d'exemples représentatives des objets traités.
- Un expert effectue une généralisation interactive de la zone d'exemples en utilisant les algorithmes choisis. Durant le traitement, l'expert qualifie les objets traités à l'aide des attributs abstraits, décrit les opérations qu'il réalise, et détermine l'ensemble des algorithmes applicables sur un objet avant d'en choisir un. Parallèlement, les mesures sont automatiquement calculées sur chaque objet traité.
- Une analyse automatique des exemples recueillis est effectuée : si certains algorithmes sont trop peu utilisés durant la généralisation interactive de la zone d'exemples, on demande à l'expert de fournir de nouveaux exemples où ces algorithmes sont utilisés.
- Une phase d'apprentissage supervisé symbolique est effectuée automatiquement pour créer des hypothèses permettant de réaliser chaque étape de la méthode de résolution de problème.
- Une première phase semi-automatique de validation des règles apprises peut être effectuée en recherchant les attributs abstraits mal appris.
    - La recherche des attributs abstraits mal appris peut être faite à deux niveaux. On recherche d'abord les attributs abstraits dont les hypothèses de détermination associées pendant l'étape d'abstraction possèdent une erreur estimée importante. On recherche ensuite les attributs abstraits dont l'élimination fait diminuer l'erreur estimée de l'enchaînement des inférences jusqu'au choix de l'algorithme.
    - Si des attributs abstraits sont mal appris on peut demander à un expert de fournir de nouvelles mesures pour déterminer cet attribut. Ces mesures sont alors calculées sur les exemples et on revient à l'étape précédente d'apprentissage automatique. Si aucune nouvelle mesure n'est proposée, les attributs abstraits défectueux sont éliminés de la méthode de résolution de problème.
    - Notons que si on utilise ainsi les taux d'erreur estimés pour affiner les règles, il serait judicieux d'utiliser de nouveaux exemples pour évaluer la qualité des règles apprises au final (cf. C.5.2)
- Une deuxième phase de validation et de raffinement des règles apprises peut être réalisée par un expert en analysant leur contenu. Cette analyse du contenu des règles est rendue possible grâce à l'utilisation de techniques d'apprentissage symbolique.

- Les hypothèses apprises sont appliquées sur de nouveaux objets à traiter. L'analyse des éventuels résultats cartographiques non satisfaisants, ainsi que du raisonnement suivi pour y arriver, permet à un expert de raffiner à nouveau les règles apprises.

En théorie, les experts désignés ci-dessus peuvent être différentes personnes : on peut imaginer faire appel à des experts des mesures, des experts des algorithmes, des experts du dessin cartographique. En pratique, la sollicitation de nombreux experts est difficilement envisageable : elle nécessiterait de nombreux efforts pour que ceux-ci accordent leurs points de vue. Pour qu'elle soit applicable en pratique, la méthodologie proposée nécessite de ne solliciter, au maximum, que deux experts : le géomaticien spécialiste de la manipulation des données géographiques numériques qui fournit les algorithmes, et le cartographe spécialiste de la généralisation qui fournit les exemples et décrit de manière abstraite les objets manipulés.

La méthodologie proposée a été implémentée et intensément testée dans le cadre de la généralisation cartographique des routes (cf. chapitre E). Plus globalement, nous avons implémenté un algorithme permettant d'enchaîner automatiquement les différentes étapes de l'apprentissage supervisé, de les évaluer, et de traduire les hypothèses apprises dans un langage procédural directement exploitable dans un logiciel SIG.

Nous avons montré de manière empirique, sur l'exemple des routes, l'intérêt de l'étape d'abstraction des mesures dans la méthode de résolution de problème proposée. Elle permet d'apprendre des règles organisées et compréhensibles, facilitant ainsi la validation, la maintenance, et l'évolution du système. Elle contraint fortement l'apprentissage par les connaissances du domaine traité, permettant ainsi d'apprendre efficacement à partir de peu d'exemples. Elle permet d'améliorer grandement la qualité des règles apprises, par rapport à un apprentissage direct des mesures aux algorithmes de généralisation, surtout du point de vue de la qualité cartographique des résultats de l'application des règles apprises. Elle permet aussi d'évaluer la qualité et la complétude d'un ensemble de mesures d'analyse spatiale dans un but de généralisation cartographique. Elle est donc au cœur de la méthodologie proposée.

Notre approche de l'abstraction des exemples se caractérise par une définition *a priori* du langage abstrait, en fonction du domaine manipulé. Les relations entre les mesures utilisées pour décrire un objet et sa description abstraite sont ensuite automatiquement apprises à partir d'exemples. Notre approche diffère donc d'une discrétisation des mesures, technique plus couramment utilisée en apprentissage automatique, où l'abstraction des attributs numériques est définie en fonction des exemples manipulés et n'est pas contrainte par les connaissances du domaine. Cette abstraction possède l'avantage d'abstraire fortement les observables et de définir des concepts abstraits aisément compréhensibles et manipulables. En contrepartie, elle alourdit considérablement le recueil des exemples pour l'apprentissage : lors de ce recueil il faut qualifier de manière abstraite les objets manipulés.

ii. Enseignements

Approche de la généralisation cartographique

Durant nos travaux, nous nous sommes fortement inspirés des thèses de Corinne Plazanet sur la caractérisation du linéaire routier [Plazanet 96] et d'Anne Ruas sur les stratégies de généralisation [Ruas 99]. La qualité des résultats cartographiques obtenus, ainsi que l'étude des règles apprises, appuient certaines des idées soutenues dans l'une ou l'autre de ces thèses :

- La généralisation cartographique peut être automatisée en appliquant successivement de nombreux algorithmes, applicables dans différents contextes et dans différents buts.
- Il est alors important de savoir définir et calculer un espace de travail adapté à chacune des opérations réalisées. Dans le cas des routes, ceci s'est traduit par l'importance des étapes de focalisation dans nos processus.
- Le traitement doit être guidé par les conflits cartographiques, c'est à dire par l'étude des violations des contraintes cartographiques. En effet, dans notre cas, la notion d'empâtement (i.e. violation de la lisibilité du symbole) est au centre de GALBE comme des règles apprises. Mais il est aussi utile d'analyser certaines propriétés des objets, autres que les conflits cartographiques, pour guider la généralisation.
- L'automatisation de la généralisation peut être vue en grande partie sous la forme d'un processus markovien, où l'action à réaliser à une étape du processus ne dépend pas des opérations précédemment réalisées. En effet, pour choisir l'algorithme à appliquer à un moment donné du processus, les règles apprises n'utilisent que des informations sur l'état courant de l'objet traité. Cependant dans nos expérimentations, faute d'avoir utilisé des mesures de validation du résultat assez pertinentes, la connaissance des opérations précédemment réalisées est utile à la détermination du moment où arrêter le processus (cf. E.2).

Approche de l'apprentissage

Nos travaux fournissent une confirmation expérimentale d'une partie des théories développées lors de la mise au point du système ENIGME [Ganascia, Thomas et Laublet 93 ; Thomas 96]. L'utilisation de connaissances du domaine permet de faciliter l'apprentissage. En particulier, la définition d'une méthode de résolution adaptée au problème permet de contraindre l'apprentissage en le décomposant en plusieurs étapes plus simples, et de le rendre plus efficace, en termes de lisibilité et de pouvoir de prédiction des règles apprises (cf. partie C.4).

Nous avons montré que l'abstraction des mesures est une étape clé dans la méthode de résolution de problème que nous proposons : elle permet de diminuer fortement la taille des exemples manipulés et ainsi de diminuer l'espace des hypothèses à parcourir, et elle permet d'améliorer la compréhensibilité des hypothèses apprises (cf. E.4.2). Nous appuyons ainsi les études sur l'intérêt de l'abstraction des exemples pour l'apprentissage [Giordana et Saitta 90 ; Saitta et Zucker 98].

Si la méthodologie de recueil de connaissances proposée doit être réitérée sur de nouveaux types objets à cartographier, la mise au point du langage abstrait est certainement l'étape qui mérite le plus de temps et d'attention. Ce temps est néanmoins beaucoup moins long que celui de la mise au point empirique d'un processus complet comme GALBE.

Forces et faiblesses des outils géométriques utilisés

L'apprentissage nous permet à la fois d'acquérir des connaissances de guidage des algorithmes et des informations au niveau de la pertinence des mesures utilisées. En termes de mesures, nous avons mis en évidence la nécessité de concevoir des mesures stables pour guider la généralisation cartographique (cf. E.2.1). Toute mesure utilisée doit être stable pendant le traitement : par exemple, si un algorithme peut densifier le nombre de points utilisés pour décrire une ligne, toute mesure utilisée doit être stable vis-à-vis de ce nombre de points. L'apprentissage montre cela en favorisant les mesures les plus stables pour la construction des règles. De plus, notre approche permet d'analyser la qualité et le contenu des règles apprises. Ceci permet de déterminer les mesures pertinentes vis à vis du problème traité, et de déterminer les attributs abstraits pour lesquels il n'existe pas de mesure assez pertinente. L'apprentissage nous a ainsi permis de mettre en évidence des propriétés des routes pour lesquelles les mesures utilisées ne sont pas assez pertinentes : l'environnement et le niveau de granularité (cf. E.2.1 et E.2.2).

Forces et faiblesses des outils d'apprentissage utilisés

Durant nos expérimentations, nous avons essentiellement manipulé des arbres de décision et des bases de règles de décision. Nous pouvons relever les avantages respectifs, en terme de lisibilité, de ces deux langages dans notre contexte :

- De manière générale, en présentant nos travaux à différentes personnes, les arbres de décision sont apparus plus naturellement compréhensibles que les règles de décision. Les règles de décision sont particulièrement difficiles à comprendre dans le cas où la classe peut prendre plusieurs valeurs non ordonnées. Quand ces valeurs sont ordonnées (e.g. taille = *petit*, *moyen*, *grand*), les règles de décision sont compréhensibles, à condition que les valeurs de la classe apparaissent dans les règles selon cet ordre naturel (i.e. règles sur *petit*, puis règles sur *moyen*, puis sur *grand* ; cf. E.1.7).
- Pour l'inférence d'abstraction, la lisibilité des arbres de décision et des règles de décision est relativement équivalente, car la plupart des abstractions ont des valeurs relativement ordonnées.
- Les règles de décision semblent bien adaptées pour la détermination de l'applicabilité des algorithmes. Dans ce cas, seulement deux classifications sont possibles : applicable ou non applicable. Les règles de décisions énoncent alors l'ensemble des conditions pour qu'un algorithme soit applicable, ce qui est facilement compréhensible. Notons que dans ce cas l'ordre des règles importe peu, et chacune peut être lue indépendamment des autres ce qui en facilite grandement la lecture. Les règles de décisions sont alors proches de règles de production (sauf la dernière règle "sinon l'algorithme n'est pas applicable").
- Les arbres de décisions sont plus compréhensibles pour l'inférence de détermination finale de l'algorithme à appliquer. Cela provient du fait que dans ce cas les classes sont nombreuses et non ordonnées.

Nous avons eu une approche "utilisateur" de l'apprentissage. C'est-à-dire que nous nous sommes concentrés sur la manipulation d'algorithmes d'apprentissage plutôt que sur leur mise au point. Dans ce contexte, en cas d'échec d'un apprentissage, nous avons toujours été confrontés au problème de l'explication de l'échec : n'y a-t-il rien à apprendre à partir des exemples utilisés ? ou l'algorithme d'apprentissage est il incapable d'apprendre ce qu'il y a à apprendre ? Ce problème reste une question ouverte. D'une part, dans la plupart de nos tests à

partir d'un même jeu d'exemples, soit aucun des algorithmes essayés (C4.5, RIPPER…) n'arrive à apprendre efficacement, soit tous les algorithmes réussissent à apprendre des hypothèses de qualité similaire. On peut donc penser que lorsqu'il y a "quelque chose" à apprendre les algorithmes classiques arrivent à l'apprendre. Mais d'autre part, nous avons également montré que décomposer l'apprentissage en plusieurs étapes, et donc le contraindre par des connaissances du domaine, permet d'en améliorer le résultat. On peut en déduire que l'apprentissage peut être inefficace parce que l'algorithme non contraint a été inefficace, et non parce que les exemples n'étaient pas assez pertinents.

Par ailleurs, d'un point de vue utilisateur des algorithmes d'apprentissage symbolique, il nous a paru que les hypothèses apprises par ces algorithmes ne sont pas toujours optimales en terme de lisibilité. Un ensemble de transformations très simples permettent parfois d'en améliorer la lisibilité, sans en modifier le contenu logique. Les algorithmes d'apprentissage gagnerait à intégrer une phase de mise en forme, simple et automatique, des hypothèses apprises. Par exemple, RIPPER apprend parfois des règles du type "si Att1 < 15 et Att1 < 10 alors…". Le premier test de cette règle peut évidemment être éliminé, ce qui en faciliterait la lecture. De même, changer simplement l'ordre des prémisses d'une règle en facile parfois la lisibilité.

iii. Perspectives

Amélioration des règles apprises

Les expérimentations que nous avons effectuées sur les routes méritent d'être complétées pour en améliorer le résultat. Dans le futur, il faudra certainement enrichir la qualification des objet manipulés.

Tout d'abord, comme nous l'avons mis en évidence dans nos tests, il faudra utiliser des mesures plus pertinentes pour qualifier la granularité et l'environnement d'un objet. Plus généralement, des recherches doivent être poursuivies dans le domaine de l'analyse spatiale pour mieux qualifier les données géographiques manipulées, dans un but de généralisation cartographique.

Ensuite, nous n'avons qualifié les objets que de manière intrinsèque. Il sera utile de qualifier les objets en terme d'écart à leur état initial. Plus précisément, il sera utile de les qualifier en terme d'écart de position et d'écart de forme. Ceci permettra, par exemple, d'apprendre des règles rejetant les algorithmes déplaçant trop un objet quand sa position planimétrique est déjà dégradée aux limites de ce qui est autorisé dans les spécifications de la carte.

De plus, nous avons confondu dans nos expérimentations les notions d'espace d'analyse et d'espace de travail [Ruas 99, p.51]. L'espace de travail est l'espace sur lequel un algorithme est appliqué, l'espace d'analyse est l'espace sur lequel les propriétés d'un objet sont étudiées. Par exemple, on peut appliquer un algorithme sur un virage (espace de travail) faisant partie d'une portion de route sinueuse (espace d'analyse). Comme nous avons introduit une qualification de l'environnement d'un objet, on pourra introduire une qualification de l'espace d'analyse contenant un objet. Ceci requiert de savoir identifier et qualifier des espaces d'analyse pertinents. Des outils pour réaliser cela existent, par exemple pour les routes, comme le découpage selon la sinuosité [Plazanet 96, p.135]. Nous avons jugé cet algorithme trop instable pour focaliser sur un espace de travail adéquat (cf. E.1.4), mais cette instabilité

est peu pénalisante pour identifier les espaces d'analyse dont les limites précises sont sans importance : il nous importe seulement de savoir de quel espace d'analyse un objet fait partie.

Enfin, nous considérons dans notre approche tous les attributs abstraits de même type. Pour faciliter le recueil des connaissances et l'analyse des règles apprises, il sera certainement utile de différencier au moins deux grands types de descripteurs. Des attributs décrivent l'écart au besoin : par exemple quand nous qualifions une route d'*empâtée*, nous sous-entendons *trop empâtée par rapport au besoins de lisibilité*. D'autres attributs décrivent les propriétés de l'objet qui aident à choisir un algorithme mais ne représentent pas un écart au besoin : c'est le cas par exemple de l'attribut décrivant l'*environnement* d'une route qui permet de ne pas utiliser les algorithmes nécessitant beaucoup d'espace. Un étude sur le sens des attributs abstraits utilisés et l'influence de ce sens sur les traitements mériterait d'être engagée.

Par ailleurs, nous n'avons pas utilisé tout le potentiel des règles apprises. Les algorithmes d'apprentissage peuvent en général associer un taux de confiance à chaque règle apprise, estimé par exemple en fonction du nombre d'exemples d'apprentissage couverts par la règle. Nous n'avons pas utilisé cette information dans l'utilisation que nous faisons des hypothèses apprises. Cette information peut être utilisée pour mieux choisir l'algorithme de généralisation cartographique à appliquer, en privilégiant par exemple ceux dont l'applicabilité est déterminée avec une forte confiance. Ce taux de confiance peut aussi être utilisé, dans une approche stochastique, pour choisir au hasard l'algorithme à appliquer parmi les algorithmes applicables.

Elargissement aux objets structurés

Notre approche suppose de savoir décrire l'objet cartographique traité par un ensemble fixe de mesures et d'attributs abstraits. Ceci est possible pour les objets peu structurés, c'est à dire des objets que l'on peut décrire comme un tout, sans avoir besoin de décrire chacune de leurs parties ni les relations entre ces parties. Par exemple, même si une route est constituée de plusieurs parties, nous la décrivons comme un tout dans nos expérimentations. Cette description s'est révélée suffisante pour nos besoins. Mais il aurait pu être nécessaire de décrire chacune de ses parties et leurs relations, comme par exemple "cette route est constituée d'une série de virage serrés, suivie d'une ligne droite, suivie d'un petit virage, suivi d'une longue courbe qui se rapproche de la première série de virages, etc.".

Certains objets géographiques sont beaucoup plus structurés que les routes considérées isolément, et méritent peut-être une description plus complexe dans un but de généralisation. C'est le cas par exemple des échangeurs routiers que l'on peut difficilement décrire sans décrire les relations entres les différentes voies d'accès le composant.

Il existe des algorithmes d'apprentissage manipulant des langages structurés que l'on pourrait alors utiliser, mais ceux-ci ont beaucoup moins prouvé leur efficacité que les algorithmes d'apprentissage propositionnels tels que RIPPER que nous avons utilisé. Des recherches sont nécessaires pour étudier si notre approche peut être étendue au cas des objets décrits dans des langages structurés. En particulier, il faut étudier comment abstraire les objets structurés, et comment représenter et recueillir ces exemples structurés.

Mais, l'espace peut et doit être analysé à différents niveaux. Si le carrefour routier mérite d'être décrit dans un langage complexe, il doit peut-être être analysé dans un langage plus simple. Si nous ressentons le besoin de décrire de manière complexe les objets structurés, c'est

sûrement que nous espérons que l'apprentissage automatique compensera notre incapacité à qualifier ces objets géographiques de manière plus simple dans leur globalité. La solution pour manipuler de tels objets réside sans doute avant tout dans l'amélioration des outils d'analyse spatiale.

Amélioration du moteur

La faiblesse du moteur utilisé pour enchaîner les algorithmes de généralisation cartographique explique une grande partie des erreurs identifiées dans les résultats cartographiques obtenus avec le processus appris par les abstractions (cf. E.3.2).

Un moteur plus élaboré a été développé lors du projet AGENT (ESPRIT/LTR/24939). Ce moteur contient en particulier une phase de validation après chaque application d'algorithme et permet d'effectuer de nombreux retours en arrière en cas d'invalidation des résultats. Ce moteur pourrait être utilisé pour enchaîner les algorithmes choisis par les règles apprises, mais celles-ci ne s'intègrent pas directement dans le modèle utilisé par le projet AGENT. Dans AGENT une action est choisie explicitement dans le but de satisfaire une contrainte cartographique d'un objet, et la validation de l'action prend en compte la connaissance de la contrainte traitée : une des conditions pour qu'une action soit validée est qu'elle ait diminué le degré de violation de la contrainte traitée. Par contre, dans le processus appris, une action est choisie en réponse à des conditions données (les propriétés de l'objet et la dernière action effectuée), sans identifier explicitement le conflit à résoudre. Pour les routes, le mécanisme de choix d'un algorithme issu des règles apprises est plus efficace que celui de AGENT. Mais si on remplaçait directement le mécanisme de choix de AGENT par les règles apprises, on ne pourrait pas évaluer le résultat de l'action proposée, faute d'identifier explicitement une contrainte traitée. Des études supplémentaires doivent être envisagées pour étudier comment les règles apprises peuvent être introduites dans le modèle AGENT [Mustière et Duchêne 2001]. Ceci nécessite soit d'enrichir certains aspects du modèle AGENT, en modifiant le mécanisme de choix de la transformation à appliquer sur un objet, soit d'affiner la méthode de résolution de problème proposée pour guider l'apprentissage, en introduisant par exemple une étape de détermination de la contrainte principale traitée.

La combinaison de la méthodologie exposée dans cette thèse et des principes utilisés dans le système AGENT présenterait néanmoins de nombreux avantages. Dans le système AGENT, des connaissances doivent être introduites au niveau de chaque type d'objet géographique considéré. La méthodologie que nous proposons permet de recueillir ces connaissances. Par ailleurs, l'utilisation des principes utilisés dans AGENT améliorera les résultats cartographiques que nous avons obtenus.

Cela permettra tout d'abord de réaliser des retours en arrière (annulation au cours du processus des opérations dégradant trop l'état de l'objet traité). La liste apprise des algorithmes applicables à un moment du processus sera alors très utile. En cas d'invalidation du résultat d'un algorithme, on peut essayer un autre algorithme applicable ou, si aucun autre algorithme n'est applicable, on peut annuler la transformation précédente, et ceci de manière récursive [Regnauld 2001].

Cela permettra de dépasser notre approche purement descendante : au cours du traitement nous ne faisons que focaliser sans jamais regrouper plusieurs objets pour avoir une vision plus

globale des objets traités. Cette approche descendante est limitée car il serait utile d'alterner des traitements à un niveau local et à un niveau plus général. Ceci est formalisé dans le modèle AGENT avec les notions de niveaux micro et méso [Ruas 99]. Deux solutions sont envisageables pour dépasser l'approche descendante.

La solution la plus facile à mettre en œuvre est de réitérer plusieurs fois l'approche descendante. C'est-à-dire de regrouper à la fin du traitement d'un objet toutes ses parties issues de la focalisation, puis de réintégrer le nouvel objet reconstruit dans la liste des objets à traiter, si celui-ci n'est pas dans un état satisfaisant. Cette approche est celle utilisée dans le modèle de généralisation AGENT sur les routes [Duchêne 2001], et est en partie utilisée dans GALBE : à la fin du traitement d'une route, ses parties sont regroupées et un lissage sur l'ensemble de la route est effectué.

Une autre solution consiste à réaliser à la fois des opérations descendantes (de focalisation) et des opérations ascendantes (de regroupement). A l'instar de la focalisation, qui découpe un objet en plusieurs parties sur un critère d'homogénéité, on pourrait réaliser des opérations de regroupement qui regrouperaient un objet avec ses objets voisins selon des critères de ressemblance.

Ces deux solutions nécessitent l'utilisation de méthodes de validation robustes pour savoir arrêter le traitement, car elles présentent beaucoup plus de risques de non convergence qu'une approche essentiellement descendante. En effet, dans ces approches on risque de focaliser puis regrouper indéfiniment, alors qu'une approche purement descendante s'arrête quand il est impossible de traiter une partie ou de focaliser plus avant.

L'utilisation des principes utilisés dans AGENT permettra aussi d'introduire des mécanismes de gestion des propagations. En effet, des objets bien traités à un moment du processus peuvent être dégradés à cause des effets de bord dus à la propagation des modifications de leurs voisins (cf. Figure 58 p.151). Pour corriger cela, il sera utile de traiter à nouveau les objets dégradés après cette propagation. Ceci est aussi effectué dans le modèle AGENT pour les routes [Duchêne 2001], mais là encore, cela augmente les risques de non convergence du processus, et nécessite des méthodes de validation robustes.

Enrichissement de la méthodologie

La méthodologie proposée pour recueillir les connaissances de généralisation cartographique peut aussi être enrichie, au niveau du recueil des exemples, de l'apprentissage, et de la modification des connaissances apprises.

La principale limite de la méthodologie proposée repose dans la lourdeur du recueil des exemples. Il serait utile d'automatiser autant que possible ce recueil. Nous avons expliqué dans la partie D.2.1 qu'il est difficile et peut être vain d'utiliser des cartes et BD existantes. Nous avons également expliqué dans la partie A.4 que les algorithmes de généralisation ne sont pas utilisé en production actuellement, ce qui empêche de recueillir des exemples par traçage des opérations interactives réalisées par les cartographes. Pour que ce traçage devienne possible il est nécessaire de revoir le but des algorithmes développés en recherche. Actuellement, les algorithmes de généralisation cartographique sont conçus pour être utilisés entièrement automatiquement. Il faudrait revoir cette approche et concevoir des algorithmes dans l'optique de les utiliser interactivement. Cela suppose de concevoir des algorithmes rapides et faciles à utiliser. Par exemple, il n'est pas réaliste de penser que les algorithmes

Faille Min et *Faille Max* soient utilisés, en l'état, interactivement : ils nécessitent de découper une ligne autour d'un virage à traiter, d'appliquer l'algorithme, de valider ou non le résultat, puis de réintégrer le virage dans la ligne, ce qui demande trop d'actions interactives pour ne traiter qu'un seul virage. Par contre, si des outils sont mis en œuvre pour que toutes ses opérations soient automatisées lorsque l'on pointe interactivement sur un virage, ces outils deviennent utilisables en interactif, et il devient alors possible de recueillir des exemples par traçage automatique.

Le traçage, s'il devient possible, permettra de recueillir automatiquement des exemples d'utilisation des algorithmes. Il ne permettra néanmoins pas de recueillir des exemples d'abstraction. Mais l'abstraction peut être apprise indépendamment des autres étapes de la méthode de résolution de problème, grâce à des exemples recueillis uniquement pour apprendre à abstraire. De plus, si on dispose de très nombreux exemples, l'abstraction n'est peut-être plus nécessaire à l'apprentissage, du moins si on ne recherche pas des règles très compréhensibles.

Le traçage automatique ne permettra pas non plus de recueillir des exemples où un algorithme n'est pas applicable : lorsque l'on utilise un algorithme en interactif, cela signifie qu'il est applicable sur l'objet traité, mais cela ne signifie pas que les autres algorithmes ne sont pas applicables. De ce fait, nous ne disposerons que d'exemples positifs pour apprendre les applicabilités, ce qui est insuffisant pour l'apprentissage. Les travaux relativement récent en apprentissage à partir d'exemples positifs et d'exemples non étiquetés permettraient peut-être de dépasser ce problème [Blum et Mitchell 98].

La méthodologie que nous proposons suppose l'utilisation d'un algorithme d'apprentissage pour apprendre les règles liées à chaque inférence de la méthode de résolution de problème. Dans nos expérimentations nous avons utilisé le même algorithme pour réaliser tous nos apprentissages. Il serait peut-être utile d'utiliser différents algorithmes. Mais existe-t-il des biais d'apprentissage plus adaptés que d'autres à l'expression et à l'acquisition des connaissances liées à un type d'inférence particulière ? Plus généralement, peut on relier le choix optimal d'une méthode d'apprentissage au type de problème traité (abstraction, spécification…) ? Ces questions restent ouvertes.

Enfin, la méthodologie pourrait être complétée en aval. Une fois les bases de connaissances apprises, il serait utile de mettre en place des procédures pour les améliorer. Deux grandes approches sont possibles pour cela : le "speed-up learning" et la révision.

La première approche serait d'optimiser les règles apprises en les appliquant sur de nombreux nouveaux exemples à cartographier [Ruas 99, p.216]. On peut par exemple rechercher des "macro-opérateurs" qui regrouperaient deux algorithmes si on constate qu'un algorithme est systématiquement utilisé après un autre. Si on dispose de fonction d'évaluation du résultat, on peut aussi apprendre à évaluer la qualité des règles apprises pour, dans le futur, privilégier les règles les plus efficaces. Ces techniques, qui cherchent à réorganiser des bases de connaissances afin de les utiliser plus efficacement sont regroupées sous le terme de "speed-up learning" (cf. [Shavlik et Dietterich 90, p.429-434] pour une introduction à ce domaine).

Une autre approche serait d'utiliser des informations externes pour réviser les bases de connaissances. Si le système appris est appliqué dans un contexte de production, ses résultats seront sûrement retouchés en interactif pour corriger les imperfections qui subsistent. Il serait utile d'utiliser ces corrections pour réviser automatiquement les bases de connaissances du système. Ceci nécessite au moins de conserver la trace de toutes les règles qui ont été utilisées

pour traiter un objet, afin d'identifier le ou les règles défectueuses en cas de retouche. Mais lorsqu'un objet est retouché interactivement, plusieurs algorithmes y ont été appliqués sur différentes parties. Il est donc difficile d'identifier l'algorithme qui est à la source des imperfections, et de là les règles défectueuses qui ont conduit à l'utilisation de cet algorithme. Une telle révision a été envisagée dans le cas théorique où un seul algorithme suffit pour traiter un objet [Bard 2000 ; Bard et Mustière 2001], mais des études supplémentaires doivent être envisagées dans un cadre plus complexe.

Enfin, durant nos travaux nous avons alternativement coiffé les casquettes du cogniticien, de l'expert cartographe et du géomaticien chargé d'automatiser la généralisation cartographique. La compréhension entre les experts et le cogniticien a ainsi été fortement facilitée ! En tant que cartographe et géomaticien, nous avons réalisé le processus GALBE qui nous a fait étudier intensivement le cas de l'automatisation de la généralisation cartographique des routes, avant de devenir l'expert fournissant les exemples au cogniticien. Nous avons ainsi indéniablement biaisé la qualité des résultats obtenus.

Pour évaluer plus intensément l'approche que nous proposons pour recueillir les connaissances de généralisation, il serait opportun de l'expérimenter en séparant le cogniticien de l'expert.

■

ANNEXES

I. Algorithmes de traitement des routes développés

1. DILATATION D'UNE LIGNE [MUSTIERE 98A]

Cet algorithme est utilisé par les autres algorithmes que nous décrivons par la suite. Il a pour but de déterminer la dilatation d'une ligne représentée en mode vecteur sous la forme d'une suite de segments. Un cas particulier d'utilisation est la détermination en vecteur des bords du symbole d'une ligne. Il s'applique à n'importe quelle ligne (y compris une ligne fermée).

Définitions

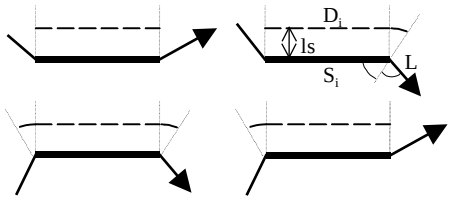
La dilatation est l'opérateur de base en morphologie mathématique. La dilatation de X par un élément structurant B est l'ensemble des positions u pour lesquelles B_u, translaté de u en B, coupe X. Dans notre cas nous nous situons dans le plan euclidien et notre élément structurant est une boule de rayon r quelconque. La dilatation de X est alors plus simplement le lieu des points à distance au plus égale à r de X. Plus précisément nous cherchons à calculer la frontière de la dilatation de X, soit le lieu des points à distance exactement égale à r de X. Puisque dans notre cas X est une ligne orientée, nous différencions les parties de la frontière à gauche et à droite de X. Définissons intuitivement ces notions de côté comme : un point P est *à gauche* de X si la projection de P *arrive* sur X par la gauche.

En SIG, l'opérateur de dilatation est plus connu sous le nom de buffer.

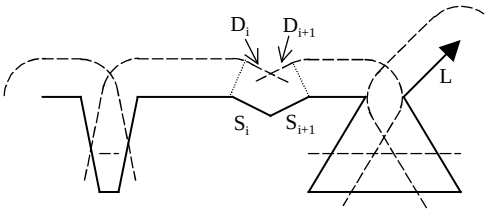
Description

Considérons une ligne orientée L constituée d'un ensemble de segments $\{S_i\}$ et que nous voulons dilater d'une largeur ls. Pour simplifier nous considérons ici un seul côté : le gauche.

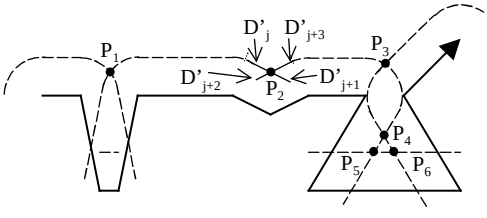
a/ On attribue à un segment donné S_i de L, une ligne décalée D_i. Celle-ci est constitué d'un segment de droite (S_i translaté à gauche de ls) plus 0, 1 ou 2 arcs de cercle selon la forme des angles entre ce segment et les segments suivant et précédant.



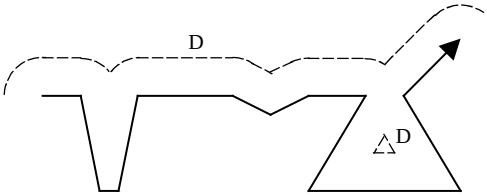
b/ L'étape a/ est effectuée sur chaque segment indépendamment les uns des autres. Nous obtenons ainsi un ensemble de lignes $\{D_i\}$ (ordonnées dans le sens de parcours de L). Dans le cas particulier des extrémités, la ligne décalée contient un quart de cercle.



c/ Les intersections $\{P_k\}$ entre toutes les lignes décalées D_i sont calculées (les P_k sont ordonnés dans le sens de parcours de L). Les lignes décalées sont découpées en plusieurs parties en fonction de ces intersections. On obtient ainsi un nouvel ensemble $\{D'_j\}$de lignes décalées.



d/ Afin d'obtenir la dilatation D de L, toutes les lignes D'_j dont la distance à L est inférieure à ls sont éliminées. Notons que $D=\{D'_j\}$ n'est pas nécessairement connexe.



L'algorithme est en $O(n^2)$ avec n le nombre de segments de la ligne considérée.

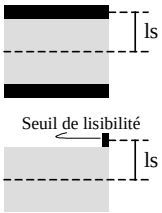
Notons que cette méthode de construction permet de conserver un lien entre chaque partie D'_j de D et le segment S_i de L qui l'a générée en a/. Ce lien et la liste des points P_k seront utilisées par l'algorithme de détection des empâtements décrit ci-après.

Application au calcul du bord du symbole

Dans les utilisations que nous faisons de cet algorithme, ls correspond à la demi-largeur du symbole cartographique appliqué sur cette ligne. Les dilatations à droite et à gauche correspondent alors aux bords gauche et droit du symbole.

Les symboles cartographiques peuvent être bordés ou non d'un trait noir. Pour être précis, notons quelle largeur ls nous prenons en compte :

- Si le symbole est bordé, alors ls est la demi largeur du symbole sans le bord plus la demi-largeur du bord.
- Si le symbole n'est pas bordé, alors ls est la demi-largeur du symbole plus la moitié du seuil de lisibilité (en général considéré à 0.1 mm).



Plus qu'un détail, cela est important en particulier pour que les algorithmes de détection d'empâtement et *Faille Min* décrit ci-après soient efficaces.

Précision des calculs

La précision des calculs intermédiaires est extrêmement importante pour cet algorithme. Dans l'étape d/ on élimine toutes les lignes D'_j qui sont à une distance inférieure à ls de la ligne L et on conserve les autres. Cette opération est très sensible aux erreurs de calcul :

- La limite entre ligne à éliminer et ligne à conserver est très fine. En effet, d'une part, les lignes que nous voulons conserver sont en théorie à une distance exactement égale à ls de L et, d'autre part, dans le cas très courant où un angle entre deux segments est très ouvert (i.e. les segments sont presque alignés) cette distance est très proche de ls pour les parties à éliminer.
- Les lignes décrivant la dilatation de L sont issues d'opérations de translation mais surtout de rotations et d'intersections qui sont particulièrement sources d'erreurs d'arrondis.

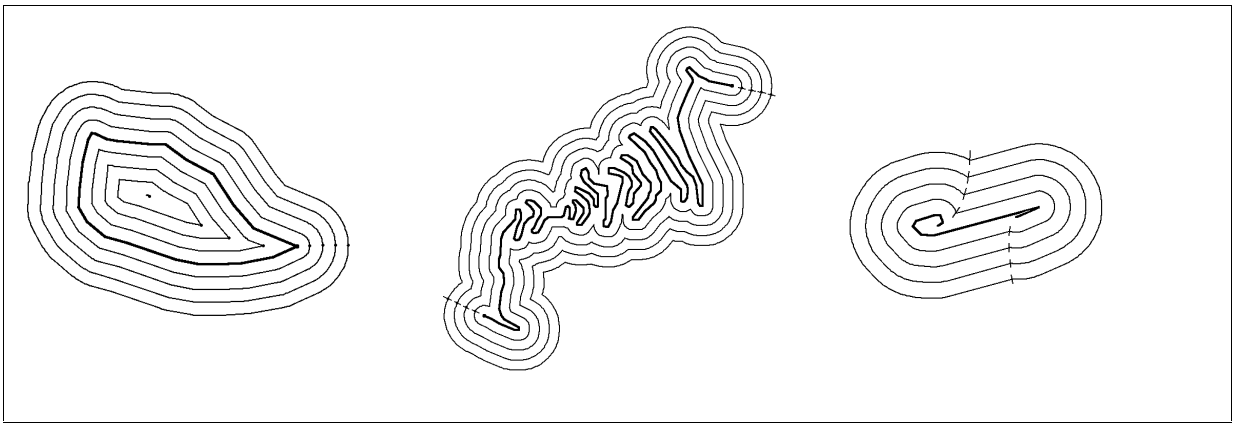
De plus, une première version de l'algorithme considérait pendant les calculs toute ligne comme une suite de segments, objets faciles à manipuler. Les arcs de cercles étaient alors approchés par une suite de courts segments. Les problèmes de précision étaient accentués à cause de cette modélisation, car la suite de segments n'était pas exactement à la distance ls du centre du cercle.

Ces erreurs étaient pénalisantes pour l'utilisation que nous faisons de l'algorithme mais encore plus pour d'autres utilisations qui en ont été faites. En particulier pour des opérations de fermeture, c'est à dire réalisant plusieurs dilatations successives.

Pour corriger cela, représenter les coordonnées avec une grande précision n'était pas suffisant. La meilleure solution que nous ayons trouvée est de représenter explicitement les arcs de cercle comme tels (centre + rayon + angles limites) dans les calculs. Cela complique l'implémentation de l'algorithme, mais résout beaucoup de problèmes d'arrondis. Par ailleurs cela accélère sensiblement le calcul.

Résultats

Les images suivantes montrent pour trois lignes (en gras), dont une fermée, les résultats de la dilatation à droite et à gauche à différentes distances. Pour la ligne fermée, l'un des cotés est en fait l'intérieur de la ligne et l'autre l'extérieur. Pour les autres lignes, les pointillés délimitent la gauche et la droite de la dilatation.



Résultats de l'algorithme de dilatation

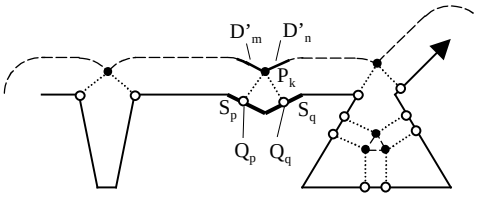
2. DETECTION DES EMPATEMENTS [MUSTIERE 98A]

Cet algorithme détecte les empâtements perçus au sein d'une ligne symbolisée quelconque.

Description

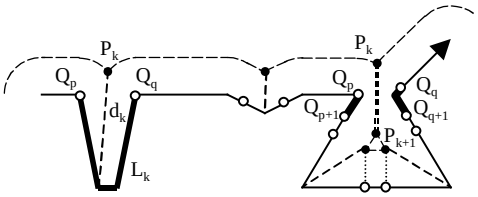
Considérons une ligne orientée L constituée d'un ensemble de segments $\{S_i\}$ et symbolisée avec un symbole de largeur ls. L'algorithme est en trois étapes : 1/ détermination de la dilatation D de la ligne comme décrit auparavant, 2/ comparaison entre L et D pour déterminer les empâtements de chaque côté, 3/ agrégation des résultats de chaque côté.

2a/ Chaque intersection P_k obtenue pendant la dilatation (étape c) sépare deux parties de D D'_m et D'_n, issues respectivement de deux segment S_p et S_q de L. Chaque P_k est projeté sur S_p et S_q, respectivement en Q_p et Q_q.



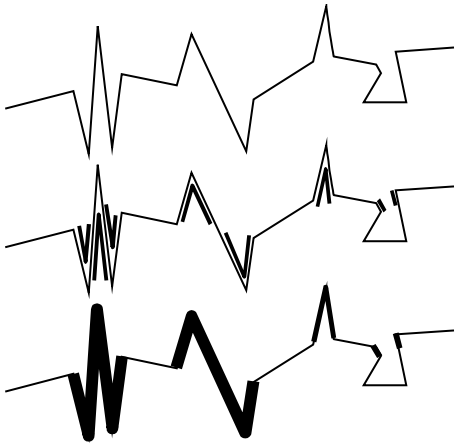
2b/ Si deux points successifs P_k et P_{k+1} sont sur deux composantes connexes distinctes de D, alors les parties de L comprises entre Q_p et Q_{p+1} d'une part et Q_{q+1} et Q_q d'autre part sont dites en conflit de "trou dans le symbole" (cf. à droite sur la figure).

Sinon, soit d_k la distance maximale entre P_k et la partie L_k de L comprise entre Q_p et Q_q. Si d_k est supérieure à 1,7×ls, alors L_k est dit en conflit d'empâtement (cf. à gauche sur la figure). NB: cf. p.57 pour une justification du 1,7.



3/ Après les étapes 2a et 2b de chaque côté, la ligne se trouve constituée d'une suite de parties soit sans empâtement, soit avec de l'empâtement d'un seul côté, soit de l'empâtement des deux cotés.

Les séries de parties empâtées adjacentes, ou très peu séparées (nous utilisons là un seuil aussi égal à 1.7×ls), sont fusionnées. Ce qui nous donne le découpage final de la ligne. A la sortie de l'algorithme la ligne est donc découpée et chaque partie est simplement qualifiée comme non-empâtée (trait fin), empâtée d'un côté (trait épais), ou empâtée des deux cotés (trait très épais).



Les calculs sont en $O(n^2)$ avec n le nombre de points de la ligne, ceci est dû à la dilatation.

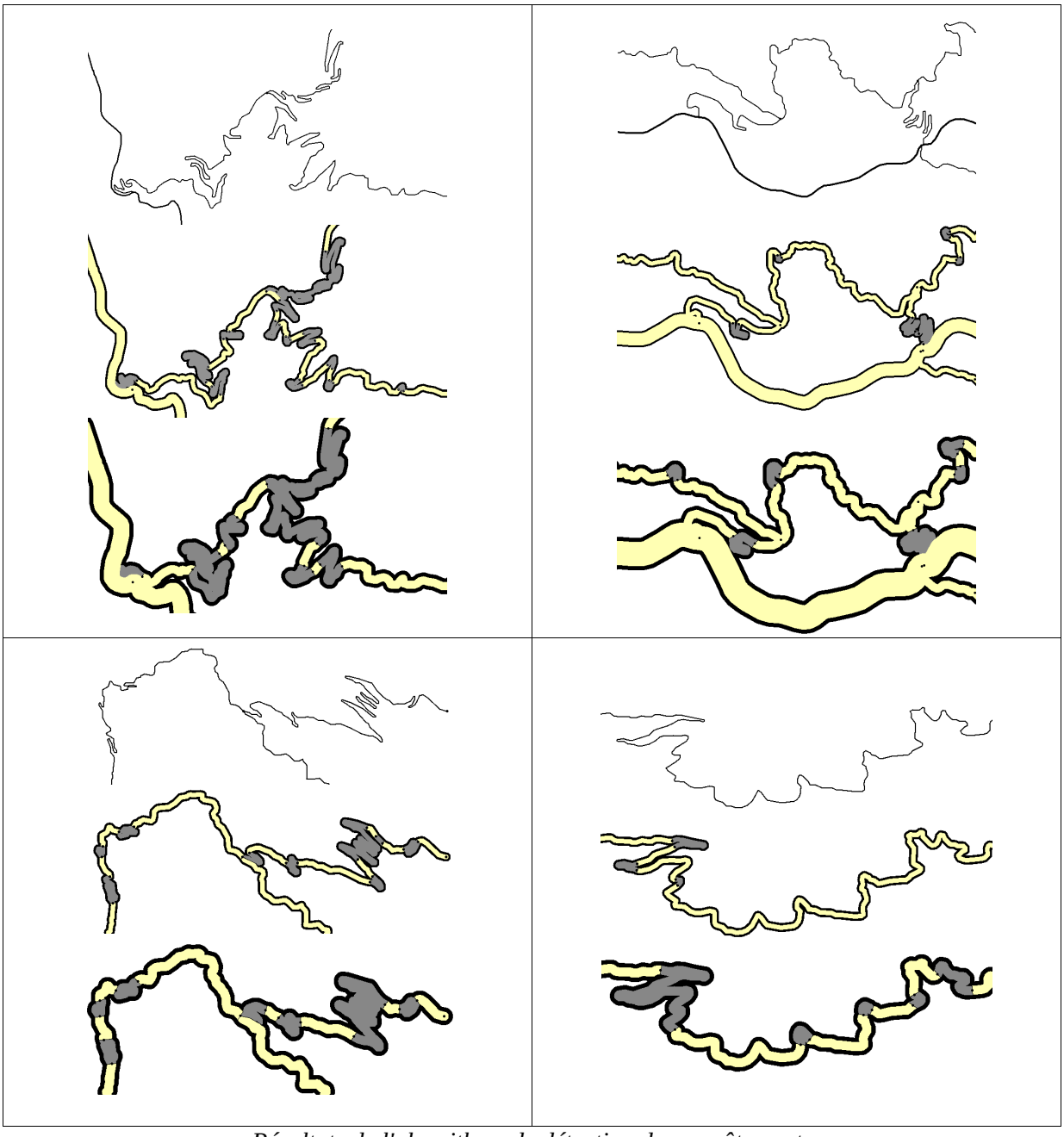
Limites

L'algorithme détecte tous les empâtements détectés visuellement. Le seul type d'empâtement non détecté correspond au cas d'une chicane en S très courte et très serrée (cf. ci-contre). Mais cela signifie une courbure très forte de la ligne et nous n'avons jamais rencontré ce cas dans tous nos tests sur les routes qui ont naturellement une courbure limitée.

L'algorithme fait une légère sur-détection des empâtements dans les cas limites sur les virages isolés.

Résultats

Les résultats ci-dessous montrent des extraits d'un réseau de routes avant symbolisation, puis symbolisées avec différentes largeurs de symboles, avec en grisé les zones détectées empâtées. Rappelons que nous recherchons ici les empâtements au sein d'une ligne et non les problèmes de proximité entre plusieurs lignes.



Résultats de l'algorithme de détection des empâtements

3. FAILLE MAX [MUSTIERE 98B]

Cet algorithme a pour but d'écarter un virage tout en conservant sa forme. Il s'applique à un virage (avec le flou que cela comporte cf. p.54), mais ce virage peut être contenir plusieurs inflexions.

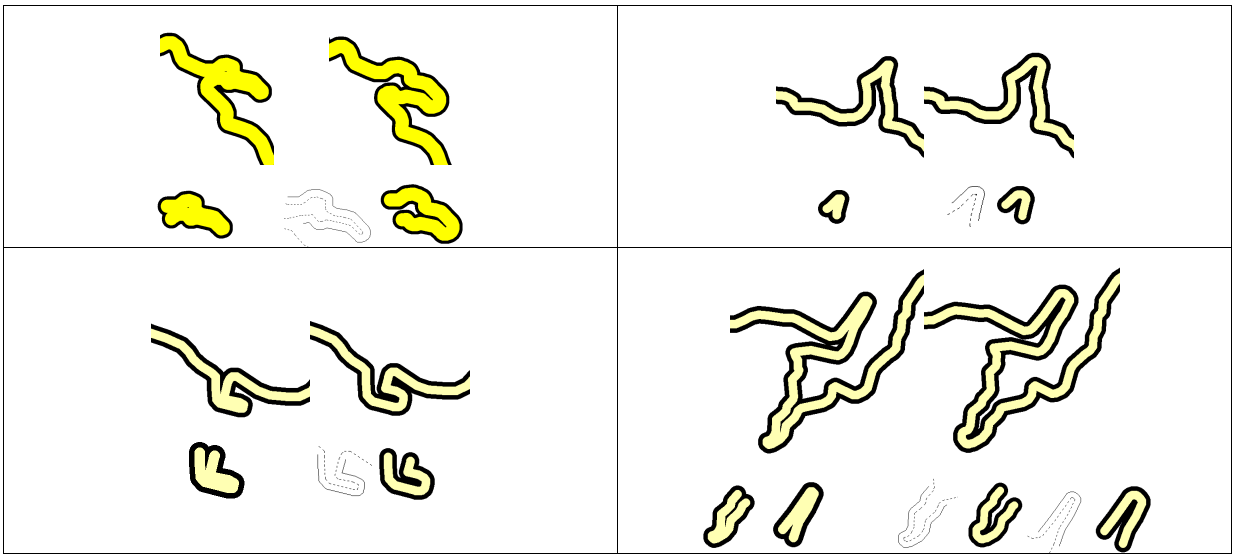
Description

Le fonctionnement est simple : une fois déterminé de quel côté de la ligne est l'extérieur du virage, il effectue une dilatation de ce côté, d'une distance égale à la demi-largeur du symbole de la ligne. Autrement dit l'axe de la ligne traitée se superpose au bord de la ligne initiale. La seule différence avec l'algorithme de dilatation décrit ci-avant est la gestion des extrémités. Dans l'étape b/ de la dilatation on ne rajoute pas ici de quart de cercle aux extrémités de la dilatation.

Un paramètre *exagération*, coefficient multiplicateur de ls, permet d'amplifier ou de réduire la dilatation.

Résultats

Les résultats ci-dessous montrent des routes où l'algorithme *Faille Max* est appliqué sur un ou plusieurs virages, puis où le déplacement des extrémités de chaque virage est propagé sur la ligne. Dans nos exemples la limite des virages a été déterminée automatiquement par application de l'algorithme de détection des empâtements décrit auparavant.



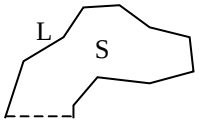
Résultats de l'algorithme Faille Max

4. FAILLE MIN [MUSTIERE 98B]

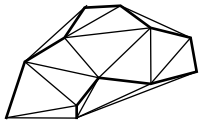
Comme *Faille Max*, cet algorithme a pour but d'écarter un virage. Mais ici le but est de minimiser l'emprise du virage. Pour cela cet algorithme imite l'astuce de cartographie manuelle de faire se superposer le bord noir interne du virage.

Description

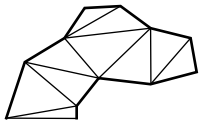
a/ La ligne L considérée est fermée pour en faire une surface S, en joignant les deux extrémités par un segment.



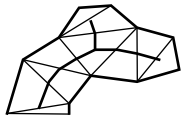
b/ Une triangulation de Delaunay est effectuée sur l'ensemble des points de la surface S (par la méthode de [Devijver et Dekesel 82]).



c/ Les triangles extérieurs à la surface S sont éliminés.



d/ L'arbre squelette de S est construit comme suit (le squelette est un arbre car la surface considérée n'a pas de trou):
- si un triangle à 2 voisins, les milieux des côtés communs sont reliés
- si un triangle à 3 voisins, le barycentre du triangle est relié au milieu de chaque côté.



e/ Le plus long chemin de l'arbre squelette est considéré comme la ligne squelette, que l'on relie au milieu des deux extrémités de la ligne L. Cette ligne squelette est légèrement lissée.

f/ Cette ligne squelette, est dilatée de chaque côté, sans ajouter d'arc de cercle à son extrémité proche de la base du virage.

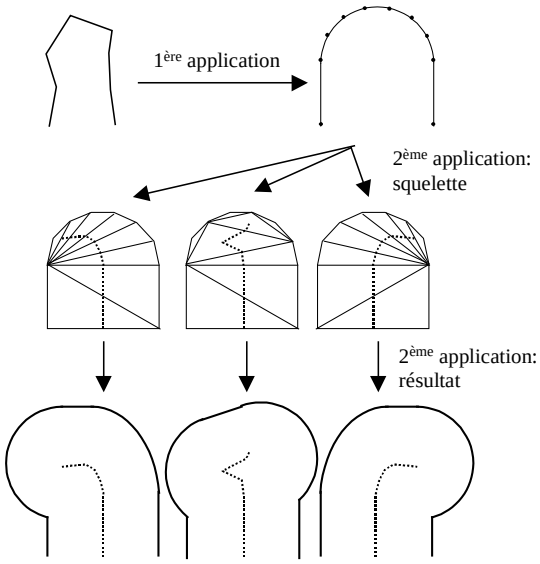
Cette ligne dilatée est la ligne en sortie de l'algorithme

Faille Min est en O(n.log n) avec n le nombre de points de la ligne, ceci est dû au calcul de la triangulation de Delaunay.

Limites

L'algorithme résout bien les empâtement mais il ne peut pas être appliqué deux fois de suite.

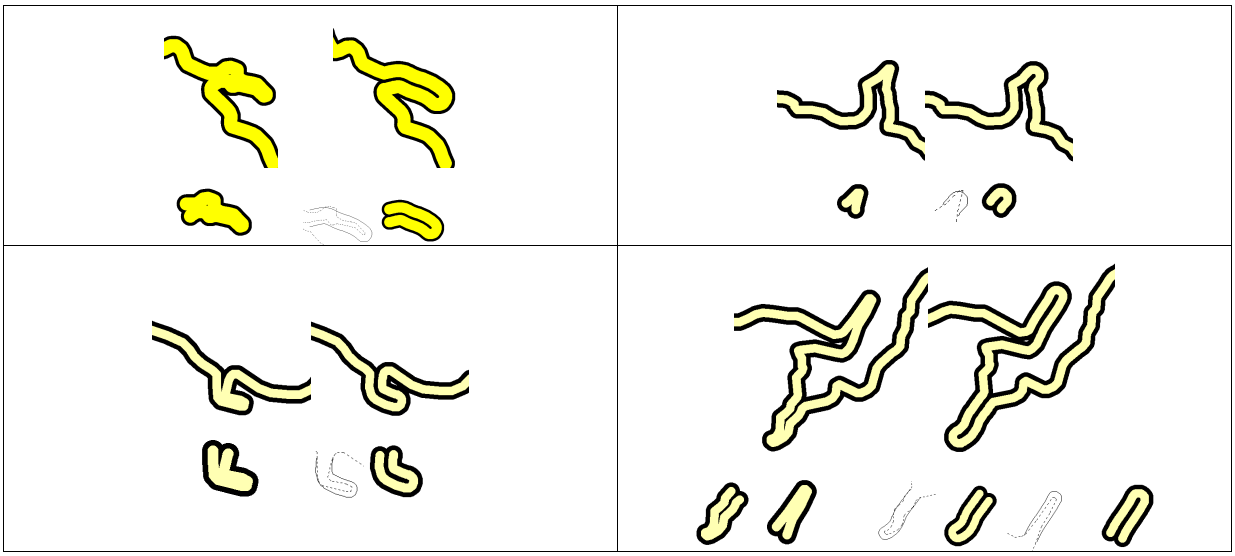
La triangulation de Delaunay sur un ensemble de points est définie de façon unique sauf si au moins quatre points sont exactement concentriques. Cela ne se rencontre qu'extrêmement rarement pour des virages. Cependant lorsque l'on applique *Faille Min* sur un virage l'extrémité de ce virage devient exactement un arc de cercle. La triangulation de Delaunay sur un tel arc est très instable, et le résultat de *Faille Min* est alors très instable. A titre d'illustration, la figure suivante montre, sur un arc déjà traité par *Faille Min*, différentes triangulations possibles, et donc différents résultats possible de *Faille Min* appliqué une seconde fois.



Il est possible de corriger simplement ce défaut en recherchant avant le calcul de la triangulation de Delaunay les points concentriques, et en ne conservant que les points aux extrémités et au milieu de l'arc de cercle détecté. Cela n'a pas été implémenté.

Résultats

Les résultats ci-dessous montrent des routes où l'algorithme *Faille Min* est appliqué sur un ou plusieurs virages puis où le déplacement des extrémités de chaque virage est propagé sur la ligne. Ces résultats sont directement comparables à ceux de *Faille Max* présentés avant, car ils sont appliqués sur les mêmes virages.



Résultats de l'algorithme Faille Min

II. Autres algorithmes géométriques utilisés

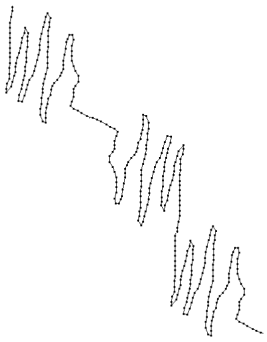
1. LISSAGE GAUSSIEN

Le but de cet algorithme est de lisser une ligne afin d'en atténuer les inflexions. Le *Lissage Gaussien* a notamment été étudié par Badaud et al. [1986] dans le domaine du traitement d'image et par Affholder [1993] et Plazanet [1996] dans le domaine de la généralisation cartographique du linéaire. La force du lissage est contrôlée par un paramètre σ.

Description

Soit L une ligne définie par l'ensemble des points P_i.

1. La ligne est décomposée en segments de longueur constante et relativement faible. Elle devient définie par une suite Q_i de points



2. Cette ligne est prolongée à ses extrémités par symétrie de la ligne initiale.

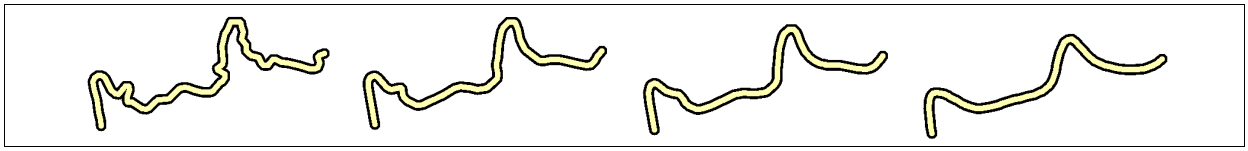
3. Chaque point est Q de la ligne est déplacé vers le barycentre de tous les points Q_i de la ligne pondéré par $G_\sigma(d(Q,Q_i))$, avec $d(Q,Q_i)$ la distance curviligne Q à Q_i, σ est le paramètre de l'algorithme, et :

$$G_\sigma(t) = \frac{e^{-t^2/\sigma^2}}{\sigma\sqrt{\pi}}$$

4. Les homologues sur L de chaque point Q_i sont extraits de la ligne lissée.

Résultats

La figure suivante montre les résultats du *Lissage Gaussien* avec différents paramètres σ.



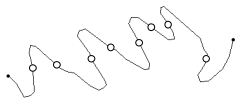
Résultats du lissage gaussien

2. ACCORDEON [PLAZANET 96]

Le but de cet algorithme est d'écarter une série de virages afin de résoudre les conflits d'empâtement entre les virages, en élargissant chaque virage perpendiculairement à leur direction principale (amélioration apportée par H. Mauffrey par rapport à la version initiale de l'algorithme qui élargissait tous les virages dans la direction principale de la série de virages).

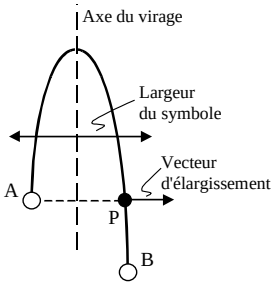
Description

1. Détermine les limites des virages de la ligne traitée
- Lisse la ligne avec le *Lissage Gaussien* décrit auparavant
- Détermine les points d'inflexion de la ligne lissée
- Les limites des virages de la ligne sont les homologues des points d'inflexion de la ligne lissée

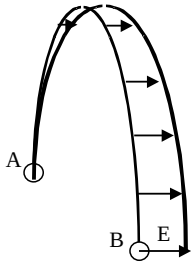


2. Elargissement de chaque virage

Pour chaque virage délimité par A et B détermine l'orientation principale du virage, puis détermine le vecteur d'élargissement E à appliquer au virage :
- Détermine, en partant de B et parcourant à rebours le virage, le premier point P du virage sur la droite passant par A et orthogonale à l'axe du virage.
- Le vecteur d'élargissement est porté par la droite AP, dans la direction AP, et de longueur la largeur du symbole moins |AP|.
- Si aucun point P n'a été trouvé, effectue la même opération dans le sens opposé, i.e. en partant de A.

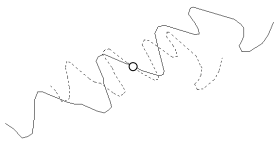


Chaque point Q du virage déplacé de $\frac{d(A,Q)}{d(A,B)}E$, avec d(A,Q) la distance curviligne entre A et Q



3. Repositionnement

Reconnecte l'ensemble des virages, et translate la série afin que le point d'inflexion central de la ligne élargit ne soit pas déplacé par rapport à son homologue sur la ligne initiale



Paramétrage du lissage

La détermination du paramètre σ utilisé lors du lissage pour la détermination des points d'inflexion peut être un paramètre de l'algorithme ou peut être réalisée automatiquement. En automatique, la ligne est lissée en faisant augmenter σ peu à peu jusqu'à ce que tous les angles formés par le sommet et les limites de chaque virage soient assez fermés (angle limite de 100^o). Cette méthode automatique est généralement efficace vis à vis de l'utilisation des points d'inflexions faite dans cet algorithme. Elle peut échouer à déterminer un paramètre σ si la série de virages est trop complexe, auquel cas une valeur par défaut est utilisée.

Résultats

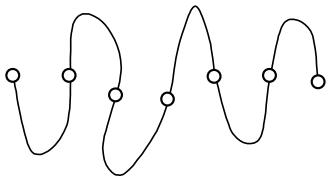
Résultats de l'algorithme Accordéon

3. SCHEMATISATION [LECORDIX, PLAZANET ET LAGRANGE 97]

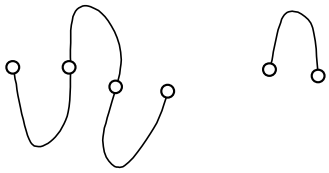
Le but de cet algorithme est d'éliminer deux virages consécutifs dans une série de virages afin de libérer de la place pour les autres. L'algorithme évite d'éliminer les premier et dernier virages, ainsi que les plus grands et ceux avec les formes les plus particuliers (amélioration apportée par S. Skarbinck à l'algorithme initial par rapport à la version initiale de l'algorithme qui élimine systématiquement les deux derniers virages).

Description

1. Détermine les limites des virages de la ligne traitée comme pour l'algorithme *Accordéon*.

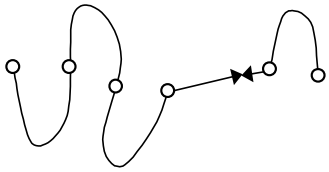


2. Choisit les virages à éliminer
- Interdit l'élimination des premier et dernier virages, et ceux dont la taille est beaucoup plus grand que la moyenne dans la série.
- Choisit deux virages consécutifs à éliminer grâce à une fonction de coût intégrant 4 mesures : la hauteur, la largeur, la longueur, la symétrie de chaque virage.

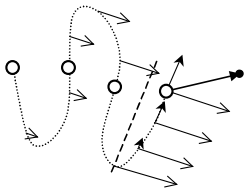


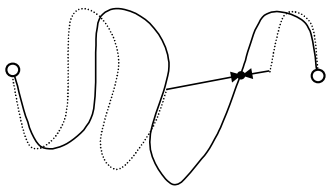
3. Raccordement.

Les deux parties restantes de la série AP et QB sont recollées au barycentre de P et Q pondérés respectivement par la longueur curviligne de QB et de AP.

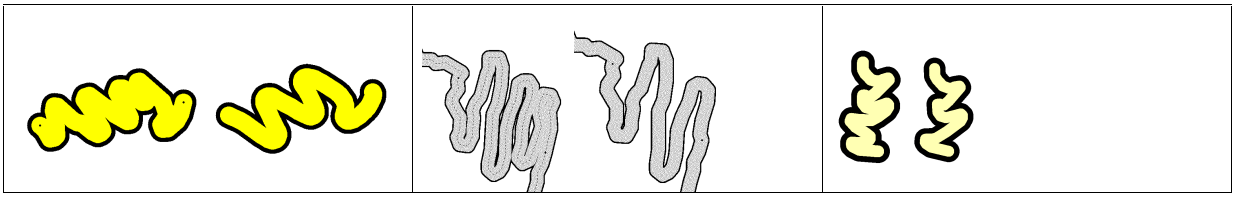


Le raccordement est amorti, pour chaque partie à raccorder:
- La direction principale du virage avant le raccordement est calculée.
- Dans la direction perpendiculaire à la direction principale, le déplacement est amorti sur l'intégralité de la partie
- Dans la direction principale, le déplacement est entièrement amorti sur le demi-virage avant la rupture.





Résultats



Résultats de l'algorithme Schématisation

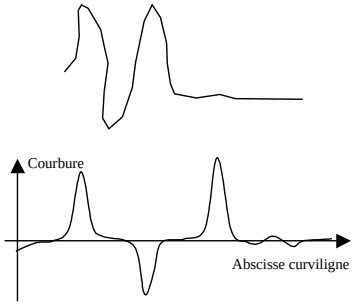
4. PLATRE [FRITSCH 97]

Cet algorithme a pour but d'atténuer les virages les moins prononcés et d'écarter les virages les plus serrés.

Description

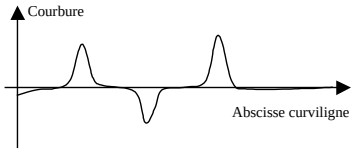
1. Changement de représentation.

La ligne qui est représentée comme une suite de segments (polyligne) est transformée en une fonction "courbure en fonction de l'abscisse curviligne" (cf. [Fritsch 97] pour la réalisation de cette transformation).



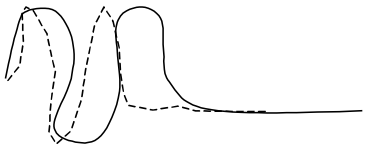
2. Lissage de la courbure

La fonction courbure est lissée par un *Lissage Gaussien*. La force de ce lissage est le paramètre de l'algorithme.



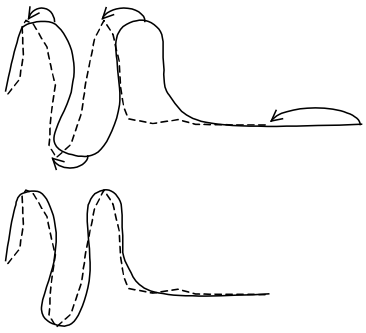
3. Retour à la représentation en polyligne.

Par transformation inverse de celle réalisée en 1, une polyligne lissée est calculée à partir de la fonction courbure lissée.

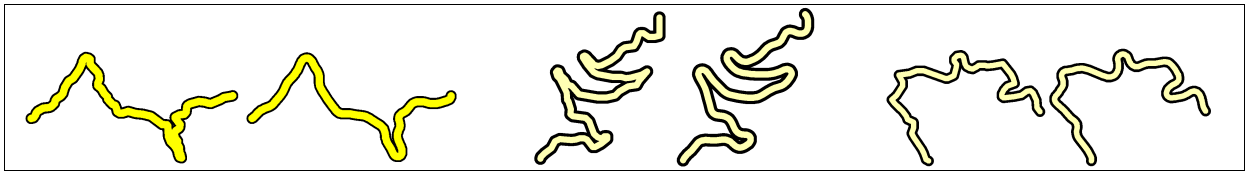


4. Repositionnement des virages serrés

Les extrémités et les virages les plus serrés de la ligne sont ramenés à leur position initiale. Ces virages serrés sont déterminés par un seuil sur la courbure également fonction du paramètre du lissage. Les parties de lignes entre les virages serrés "suivent" les virages serrés par interpolation des déplacements.



Résultats



Résultats de Plâtre

■

III. Algorithme d'apprentissage RIPPER [Cohen 95]

Inspiré de l'algorithme IREP (Incremental Reduced Error Pruning) [Fürnkranz et Widmer 94], RIPPER (Repeated Incremental Pruning to Produce Error Reduction) est un algorithme d'apprentissage supervisé de règles de décision. RIPPER est connu pour être extrêmement rapide et en particulier bien adapté aux attributs numériques.

Langage des attributs et des hypothèses

RIPPER utilise des exemples représentés en langage attributs/valeurs de la forme $(\{x_j\}_{1\leq j\leq N}, c)$. Chaque attribut x_j peut être soit symbolique, soit numérique, soit un ensemble de valeurs symboliques. RIPPER gère l'éventuelle présence d'attributs non renseignés. La classe c peut prendre un nombre quelconque de valeurs symboliques.

RIPPER est un algorithme d'apprentissage de règles de décision. Il construit un ensemble de règles de longueur quelconque sous la forme "$test_1 \wedge test_2 \wedge \ldots \wedge test_l \Rightarrow c_i$", où chaque test est soit de la forme "x_j=valeur" ou "$x_j \neq$valeur" si x_j est un attribut symbolique, soit de la forme "$x_j \leq$valeur" ou "$x_j \geq$valeur" si x_j est un attribut numérique, soit de la forme "valeur$\in x_j$" ou "valeur$\notin x_j$" si x_j est un ensemble de valeurs symboliques.

Fonctionnement de l'algorithme

Nous décrivons ici l'algorithme pour le cas d'une classe binaire, où les exemples sont soit positifs, soit négatifs. Le pseudo-code de RIPPER pour un nombre quelconque de valeurs de classe, ainsi que les mesures utilisées et les paramètres sont décrits dans les pages suivantes.

Notons que, pour gérér les attributs manquants, RIPPER considère que tout test fait sur un attribut manquant est faux. Ceci encourage RIPPER à séparer les exemples positifs des exemples négatifs en utilisant les attributs les plus présents.

RIPPER fonctionne en plusieurs étapes: création d'un jeu de règles initial (création et élagage itératif des règles une à une), élagage du jeu de règles, puis optimisation du jeu de règles.

1. Création du jeu de règle initial

RIPPER crée un jeu de règles initial en partant d'un jeu de règles vide et en construisant les règles une à une. Il élague chaque règle juste après sa création avant de l'ajouter au jeu de règles.

Pour cela, avant la création de chaque règle, il sépare l'ensemble des exemples en un ensemble d'exemples de croissance (contenant 2/3 des exemples choisis au hasard) et un ensemble d'exemples d'élagage (le 1/3 restant).

En ne considérant que le jeu d'exemples de croissance, et en partant d'une règle vide, il ajoute itérativement des tests à la règle grâce à l'heuristique de gain d'information de FOIL [Quinlan 90]. Il arrête d'ajouter des tests quand la règle ne couvre plus d'exemples négatifs (on dit qu'une règle couvre un exemple si l'exemple satisfait tous les tests de la règle, indépendamment de la valeur de sa classe).

RIPPER élague chaque règle immédiatement après sa création. Il élimine une série finale de tests en considérant le jeu d'exemples d'élagage, et en se basant sur une mesure définie dans IREP.

Une fois une règle créée et élaguée RIPPER l'ajoute au jeu de règles, et élimine de l'ensemble des exemples ceux couverts par la règle.

RIPPER arrête d'ajouter des règles sur un principe de longueur de description minimale (selon le principe de [Quinlan 95]) : quand la longueur de description augmente de plus de 64 bits lors du dernier ajout de règles.

2. Elagage du jeu de règles

Pour élaguer l'ensemble du jeu de règles, il considère chacune des règles dans l'ordre inverse de leur création et élimine toute règle qui permet de diminuer la longueur de description.

3. Optimisation du jeu de règles

Pour optimiser le jeu de règles, RIPPER considère une à une les règles. Il les élimine du jeu de règles puis crée et élague deux nouvelles règles de remplacement potentielles. La première est créée en partant d'une règle vide, la deuxième en partant de la règle éliminée. La meilleure de ces deux règles, au sens de la longueur de description, est choisie. Cette phase d'optimisation est répétée deux fois.

Pseudo code, mesures utilisées, paramètres.

```
procédure RIPPER(Exemples)
// apprend itérativement à séparer une classe de toutes les autres non encore séparées, puis optimise le jeu de règles
    JeuDeRègles ← ∅
    pour c_i parcourant l'ensemble des valeurs de la classe c dans l'ordre inverse de leur fréquence d'apparition
        P ← ensemble des exemples ayant la classe c_i
        N ← ensemble des exemples n'ayant pas la classe c_i
        JeuDeRègles_i ← Crée_JeuDeRègles(P,N)
        Ajoute JeuDeRègles_i à JeuDeRègles
        enlève de {Exemples} les éléments couverts par JeuDeRègles
    fin pour
    répète 2 fois
        JeuDeRègles ←Optimise_JeuDeRègles(JeuDeRègles,P,N)
    fin rèpète
    renvoie JeuDeRègles
fin RIPPER

procédure Crée_JeuDeRègles(P,N)
// Crée et élague chaque nouvelle règle, puis elague l'ensemble du jeu de règles
    JeuDeRègles ← ∅
    LD ← LongueurDeDescription(JeuDeRègles,P,N)
    tant que P ≠ ∅
        sépare (P,N) en (CroitPos,CroitNeg) et (ElaguePos, ElagueNeg) au hasard dans un rapport (2/3,1/3)
        // crée une nouvelle règle
        Règle ← Crée_Règle(∅,CroitPos,CroitNeg)
        // élague la nouvelle règle
        enlève de Règle la série finale de test qui maximise Métrique_élagage(Règle,ElaguePos,ElagueNeg)
        Ajoute Règle à JeuDeRègles
        si LongueurDeDescription(JeuDeRègles,P,N) > LD + 64 alors
            // élague l'ensemble du jeu de règles
            JeuDeRègles ← Elague_JeuDeRègles(JeuDeRègles,P,N)
            renvoie JeuDeRègles
        fin si
        LD ← LongueurDeDescription(JeuDeRègles,P,N)
        enlève de P et N tous les exemples couverts par Règle
    fin tant que
fin Crée_JeuDeRègles
```

```
Procédure Crée_Règle(Règle,P,N)
// construit une règle en partant d'une règle vide ou non selon le principe de FOIL
    tant que Règle couvre au moins un exemple de N
        Ajoute à Règle le Test qui maximise Gain_FOIL(Règle,Test,P,N)
        Enlève de P et N les exemples couverts par Règle
    fin tant que
    renvoie Règle
fin Crée_Règle

Procédure Elague_JeuDeRègles(P,N,JeuDeRègles)
    pour chaque règle R de JeuDeRègles parcouru dans l'ordre inverse
        si LongueurDeDescription(JeuDeRègles-R,P,N) < LD alors
            enlève R de JeuDeRègles
            LD ← LongueurDeDescription(JeuDeRègles,P,N)
        fin si
    fin pour
    renvoie JeuDeRègles
fin Elague_JeuDeRègles
                                                                                    ...
Procédure Optimise_JeuDeRègles(JeuDeRègles,P,N)
    pour chaque Règle dans JeuDeRègles
        enlève Règle de JeuDeRègles
        UPos ← exemples de P non couverts par JeuDeRègles
        UNeg ← exemples de N non couverts par JeuDeRègles
        sépare (UPos,UNeg) en (CroitPos,CroitNeg) et (ElaguePos, ElagueNeg) au hasard dans un rapport (2/3,1/3)
        RemplaceRègle1 ← CréeRègle(∅,CroitPos,CroitNeg)
        RemplaceRègle1 ← ElagueRègle(RemplaceRègle1,ElaguePos,ElagueNeg)
        RemplaceRègle2 ← CréeRègle(Règle,CroitPos,CroitNeg)
        RemplaceRègle2 ← ElagueRègle(RemplaceRègle2,ElaguePos,ElagueNeg)
        RemplaceRègle ← la meilleure entre RemplaceRègle1 et RemplaceRègle2 au sens de la LD minimum
        Ajoute RemplaceRègle à JeuDeRègles
    fin pour
    renvoie JeuDeRègles
fin Optimise_JeuDeRègles
```

Pseudo-code de RIPPER

$$Gain\_FOIL(Règle, Test, Positifs, Négatifs) \equiv (p_1 + n_1).(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0})$$

avec p_0 et n_0 le nombre d'éléments respectivement de Positifs et Négatifs couverts par Règle, p_1 et n_1 le nombre d'éléments respectivement de Positifs et Négatifs couverts par la règle créée par ajout de Test à Règle.

$$Metrique\_élagage(Règle, Positifs, Négatifs) \equiv \frac{p - n}{p + n}$$

avec p et n le nombre d'éléments respectivement de Positifs et Négatifs couverts par Règle

LongueurDescription(JeuDeRègle,Exemples) ≡ cf [Quinlan 95] pour une définition précise

Mesures utilisées par RIPPER

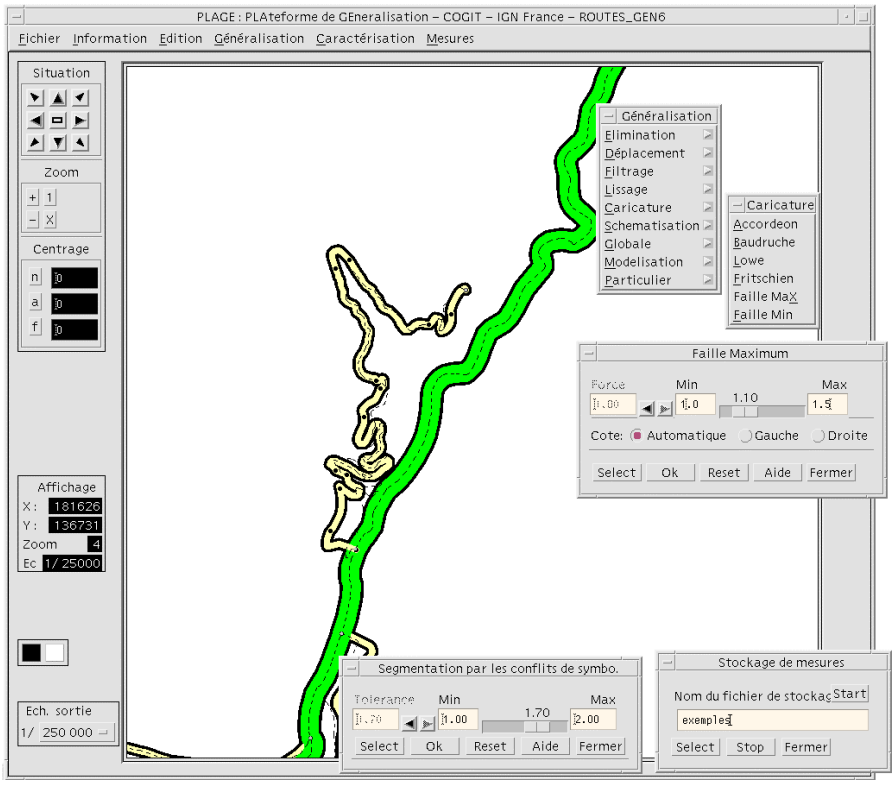
Paramètre	**Valeur par défaut**
Nombre de passes d'optimisation	2
Ordre dans lequel chaque classe est séparée des autres	ordre croissant de leur occurrence
Rapport du poids donné à un exemple positif mal classé par rapport à la celui d'un exemple négatif mal classé (uniquement pour les problèmes à deux classes)	1
"Cost of theory" : rapport du poids donné dans la longueur de description à la longueur des exemples mal classés par rapport à celui de la longueur du jeu de règles	1
Considérer les attributs comme bruités ou non (ce qui modifie l'heuristique de FOIL)	bruités
Possibilité ou non de créer des tests négatifs ($\neq$ et $\notin$)	possible
Nombre d'exemples minimum que doit couvrir une règle	1

Principaux paramètres modifiables dans la version classique de RIPPER

IV. Implémentation de l'apprentissage réalisé sur les routes

Recueil des exemples

Pour les routes, nos exemples ont été recueillis sur la plate-forme de généralisation PlaGe du laboratoire COGIT [Lecordix, Plazanet et Lagrange 97]. Cette plate-forme expérimentale comporte un ensemble d'algorithmes géométriques et des outils de visualisation et de manipulation des données cartographiques. L'interface est programmé en UIL et les algorithmes sont codés en ADA. La figure suivante montre l'environnement de PlaGe nous ayant servi au recueil des exemples sur les routes.



Environnement de recueil des exemples : PlaGe

Outre l'ajout de mesures et d'algorithmes de transformation, nous avons ajouté à l'interface de PlaGe un outil permettant de calculer et stocker les mesures choisies sur un objet donné. Mais nous avons introduit à la main dans le fichier d'exemples les descripteurs abstraits d'une route, les choix des opérations à réaliser, les algorithmes applicables, ainsi que l'algorithme appliqué sur l'objet considéré. Pour que le recueil des exemples soit plus ergonomique, il serait utile de développer des outils de traçage des opérations réalisées dans PlaGe ainsi que des fenêtres de dialogue permettant de saisir puis de stocker les descripteurs abstraits déterminés interactivement durant le recueil.

Pour les bâtiments, nos exemples ont été recueillis sur le SIG LAMPS2. L'implémentation des mesures et des algorithmes de transformation ont été réalisés par Nicolas Regnauld de l'université d'Edimbourg (en Lull, langage interne à LAMPS2).

Implémentation de l'apprentissage

Les apprentissages ont été réalisés en dehors des logiciels SIG qui ont permis de recueillir les exemples et d'appliquer les règles apprises.

Nous avons réalisé, en C, un algorithme d'enchaînement des apprentissages avec RIPPER de chaque inférence sous le modèle d'ENIGME [Thomas 96] (toutefois moins ergonomique et moins générique). Nous avons utilisé une version de RIPPER mise à disposition sur internet par W. Cohen mais qui ne semble plus être disponible au jour où nous écrivons. L'algorithme d'enchaînement des apprentissages utilise en entrée trois fichiers : un fichier contenant les exemples dans un format texte (une ligne = un exemple = un ensemble d'attributs), un fichier contenant la description de chaque étape de la structure d'inférences (i.e. les attributs représentant les classes et observables de chaque inférence), et un fichier décrivant le type des attributs (numérique ou symbolique). Cet algorithme effectue les tâches suivantes :

1. Apprentissage de chaque inférence :
   1.1. Les attributs des exemples nécessaires à l'inférence sont extraits du fichier d'exemples et mis au format d'entrée de RIPPER (fichier *.data de RIPPER)
   1.2. La définition des attributs nécessaires à l'inférence est extraite du fichier décrivant tous les attributs et mise au format d'entrée de RIPPER (fichier *.name de RIPPER)
   1.3. RIPPER est lancé sur ces fichiers, il crée un fichier texte décrivant les règles apprises (fichier *.hyp de RIPPER). Il calcule également la validation-croiséee de l'inférence.
   1.4. Les règles issues du fichier des règles sont traduites en C (pour effectuer la validation-croisée, cf. point 3) et en ADA (pour être directement introduites dans PlaGe)
2. Enchaînement des règles apprises : les fichier en C et ADA des règles sont enchaînés afin de constituer un seul fichier de code enchaînant toutes les règles apprises
3. Validation-croisée (k-parties) de l'enchaînement des règles :
   3.1. Le fichier des exemples est découpé en 2 fichiers : les exemples sont répartis dans k groupes, k-1 groupes constituent le fichier des exemples d'apprentissage, 1 groupe constitue les exemples tests.
   3.2. Les étapes 1 et 2 sont appliquées sur les exemples d'apprentissage
   3.3. Les règles apprises sont testées sur les exemples tests, en comparant la classification fournie par les règles et celle présente dans les exemples. Ces informations sont conservées pour créer les matrices de confusion estimées.
   3.4. Les étapes 3.1 à 3.3 sont réitérées k fois en faisant varier le groupe constituant les exemples tests.
   3.5. L'erreur estimée de l'enchaînement des inférences est calculée en faisant la moyenne des erreurs des k tests.

Application des règles apprises

Les règles apprises sur les routes ont été appliquées par l'intermédiaire de PlaGe. Nous y avons codé en ADA le moteur de la généralisation, qui appelle directement le fichier ADA créé à l'étape 2 de l'apprentissage. Du point de vue de l'interface, ce moteur peut être lancé sur une route ou un ensemble de routes comme tous les outils de PlaGe. Pendant le traitement d'une route, le moteur affiche à l'écran les choix réalisés : la description abstraite de l'objet, l'opération choisie, les algorithmes considérés applicables et l'algorithme appliqué. Les raisons de ces choix (i.e. les règles utilisées) ne sont pas affichées, mais cela pourrait être réalisé pour aider à l'analyse du raisonnement suivi par le système.

■

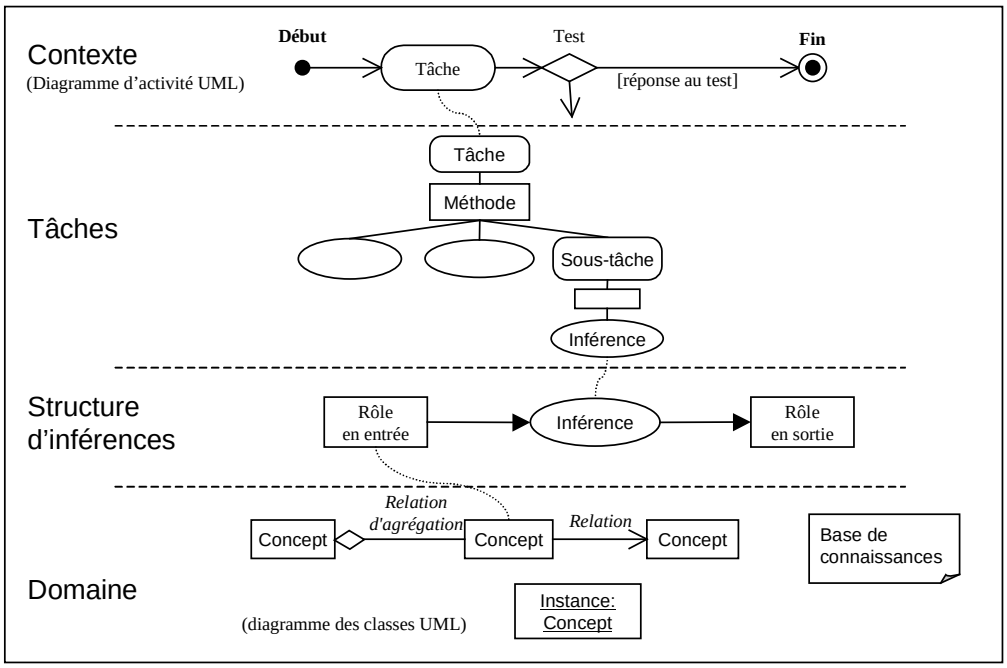
V. Résultats de l'apprentissage sur les routes

Cette annexe décrit le processus appris avec l'apprentissage par les abstractions, en suivant les formalismes de CommonKADS [Schreiber et al. 2000] et UML. Elle regroupe des figures déjà présentées dans le corps de ce mémoire au chapitre E . Des différences peuvent néanmoins exister entre les figures dans corps du mémoire et celle de cette annexe. En effet, nous ne présentons ici que les éléments utiles à l'application des règles apprises alors que le chapitre E montrait les éléments qui ont servi à l'apprentissage de des règles. Par exemple les mesures jugées inutiles pour notre problème par l'apprentissage ne sont pas reprises ici.

Nous décrivons tout d'abord le contexte d'utilisation, puis les tâches et inférences de ce processus, et enfin les éléments du domaine manipulés. Rappelons en quelques mots le formalisme CommonKADS utilisé (nous n'utilisons qu'une petite partie de ce formalisme) :

- Le diagramme des tâches présente chaque tâche du système et leur décomposition en sous-tâches.
- Le lien entre *tâche* et *sous-tâches* est fait par l'intermédiaire d'une *méthode*.
- Une sous-tâche élémentaire est une *inférence*.
- Le diagramme des inférences présente les enchaînement entre inférences et les *rôles* en entrée et sortie de ces inférences.
- Ces rôles sont les éléments du domaine manipulés.

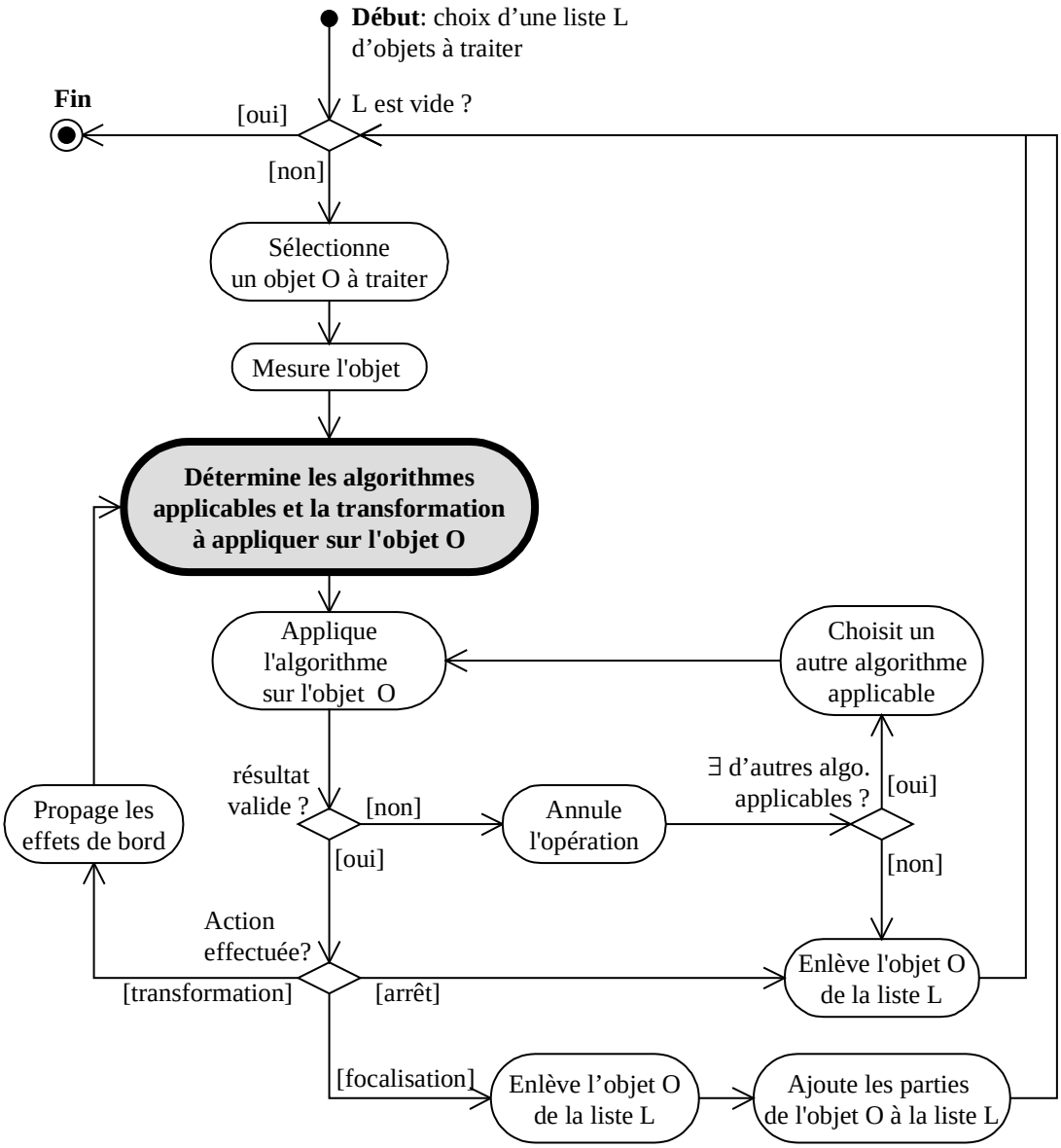
Dans notre cas ce domaine est celui de la généralisation des routes. Dans ce domaine nous décrivons, d'une part, les mesures, le langage abstrait de description, les opérations et algorithmes utilisés et, d'autre part, les bases de connaissances du système. Nous présentons tout d'abord les bases de connaissances utilisées pour produire les résultats cartographiques présentés dans ce mémoire, c'est à dire les règles de décision apprises par RIPPER [Cohen 95] avec abstraction. A titre de comparaison, nous présentons ensuite les bases de connaissances apprises par RIPPER avec une méthode de résolution de problème sans abstraction, et celles apprises par C4.5 qui apprend des arbres de décision [Quinlan 93].



Légende des diagrammes

1. CONTEXTE D'APPLICATION : MOTEUR DU PROCESSUS DE GENERALISATION.

Voir D.1.2, page 94.



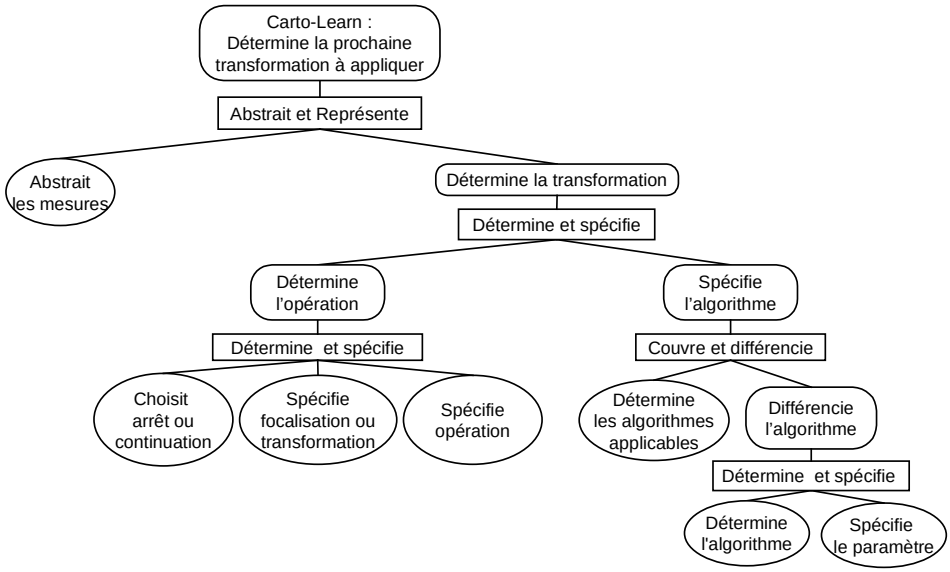
Dans la version implémentée:

- Le test "résultat valide?" vérifie que:
  - la modification n'a pas créé d'intersection interne au sein de l'arc traité
  - la modification n'a pas entraîné un écart planimétrique énorme entre les arcs avant et après traitement
  - la modification a eu un effet
- L'action "sélectionne un objet à traiter" choisit le premier de la liste.
- L'action "ajoute les parties de l'objet à la liste" les ajoute à la fin de la liste.
- L'action "choisit un autre algorithme applicable" le choisit au hasard.

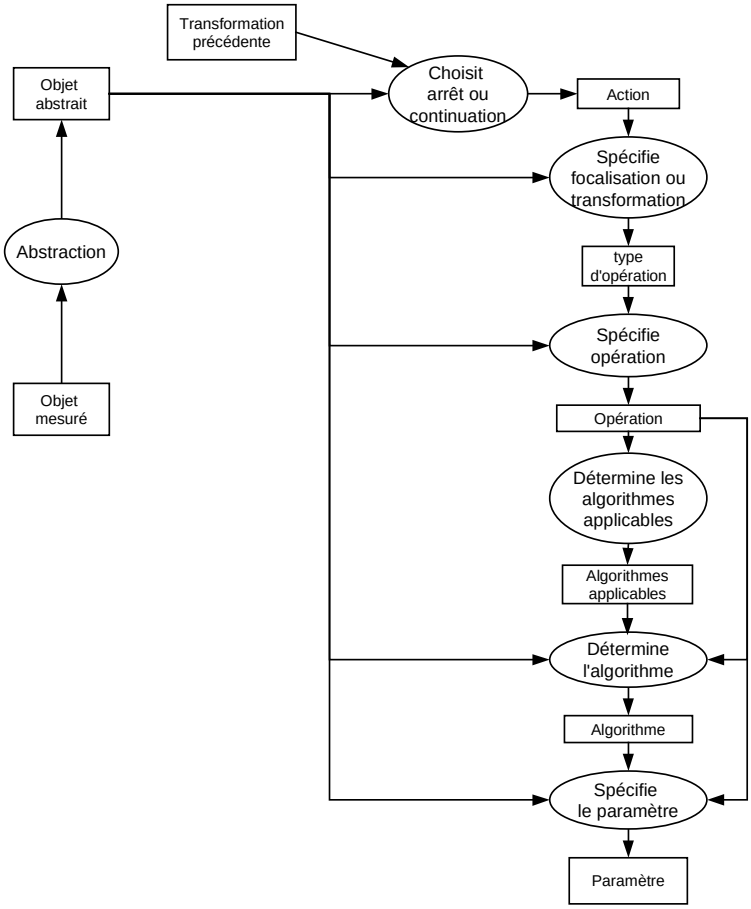
2. TACHES ET INFERENCES

Voir D.4, page 106, et E.1.5, page 129.

Tâches

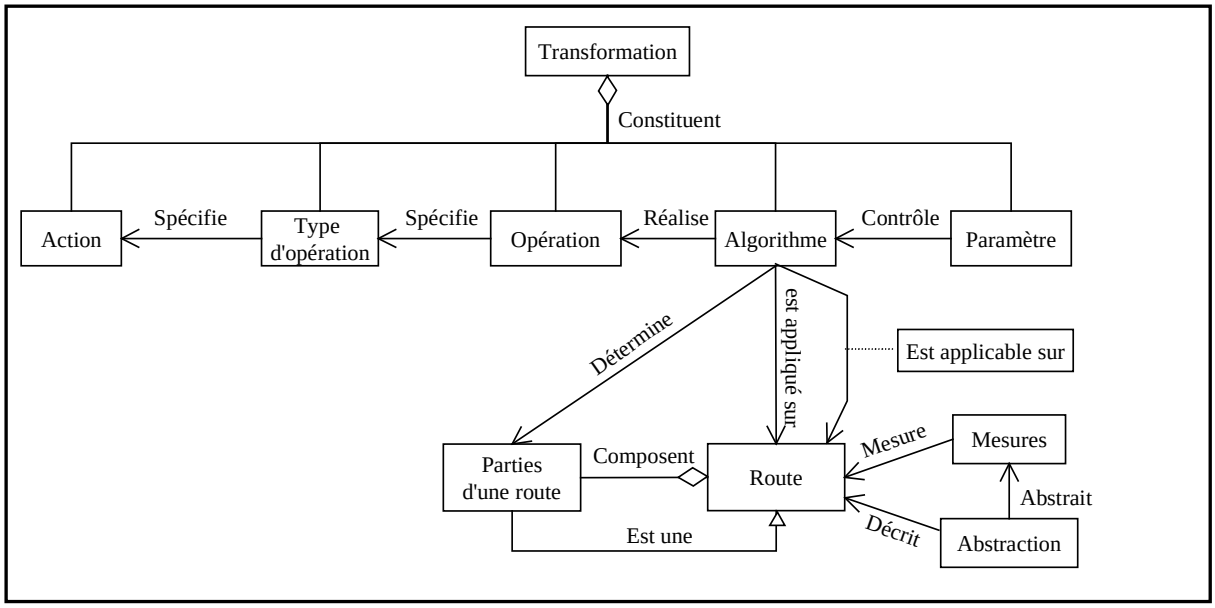


Inférences



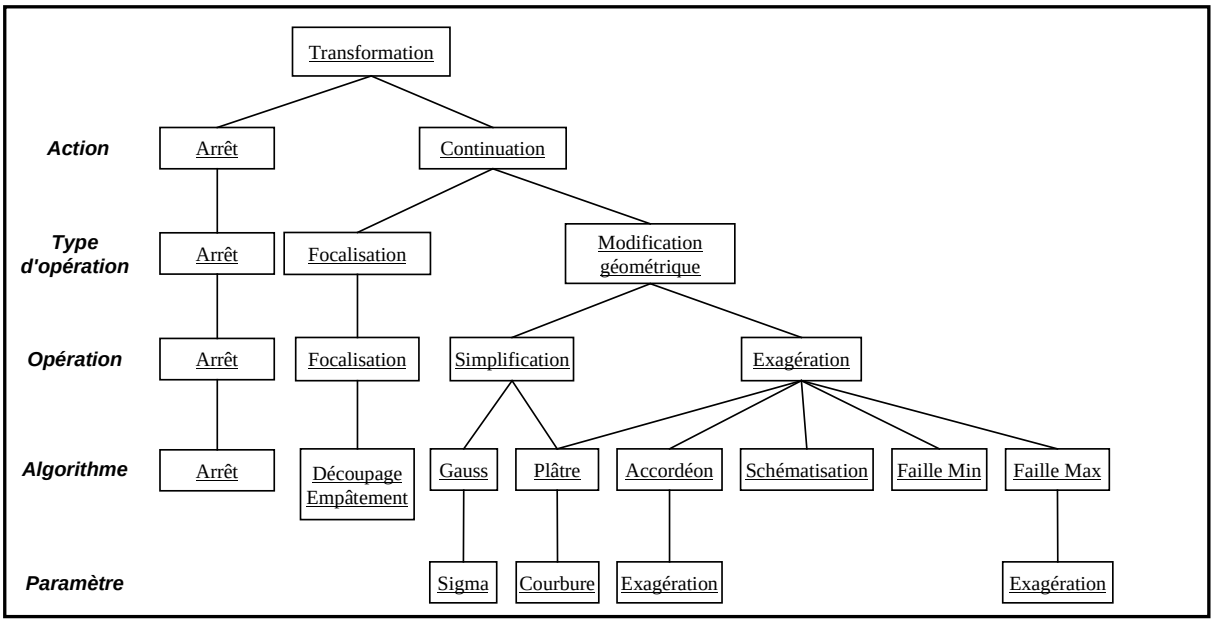
3. DOMAINE DES ROUTES, DEFINITION DES ROLES

Concepts manipulés



Treillis des relations entre les instances de la transformation

Voir E.1.4, page 128.



Mesures utilisées (Langage concret)

Voir E.1.3, page 126.

mesure	*unité**	*valeurs possibles*	*description succincte*
longueur	mm	]0,∞[	longueur de la ligne
base sur longueur		[0,1]	distance entre les extrémités de la ligne divisée par la longueur de la ligne
nombre virages		[1, ∞[	nombre de virages de la ligne (définis selon les points d'inflexion)
nombre grands virages		[1, ∞[	nombre de virages de la ligne après lissage, d'après [Plazanet 96]
taille max virages	mm	[0,∞[	hauteur du plus grand virage de la ligne, d'après [Plazanet 96]
force empâtement		[0,1]	1 – surface du symbole / (longueur de la ligne × largeur du symbole)
type empâtement		aucun monolatéral bilatéral	aucun si la ligne n'est pas empâtée, monolatéral si la ligne est empâtée d'un seul côté, bilatéral sinon.
longueur empâtement	mm	[0,∞[	longueur de la ligne en empâtement
homogénéité empâtement		[1/2,1]	max(h,1-h) avec h = longueur d'empâtement / longueur de la ligne
nombre d'arcs proches		ℕ	nombre de lignes dont le symbole intersecte celui de la ligne considérée
surface de conflit proche	mm^2	[0,∞[	somme des surfaces d'intersection entre le symbole de la ligne considérée et le symbole des lignes en conflit avec cette ligne. La notion de conflit est définie au sens de [Nickerson 88].
nb arcs proches par longueur	mm^{-1}	[0,∞[	nombre d'arcs proches divisé par la longueur de la ligne

* les unités s'entendent à l'échelle d'impression

Descripteurs abstraits utilisés

Voir E.1.2, page 124.

attribut	*valeurs possibles*
Taille	petit | moyen | grand
Sinuosité	nulle | virages peu sinueux | épingles à cheveux | hétérogène
Complexité	zéro niveau | un niveau | plusieurs niveaux
Granularité	faible | moyenne | forte
Forme générale	ligne droite | virage | courte série de virages | longue série de virages
Empâtement	nul | faible | fort | hétérogène
Environnement	libre | dense
Conflit externe	nul | peu | nombreux

4. DOMAINE DES ROUTES : BASES DE CONNAISSANCES UTILISEES PAR LES INFERENCES

Les règles suivantes sont celles utilisées dans la version finale du processus appris. Rappelons que les bases de règles de décision suivantes sont ordonnées, c'est-à-dire qu'une règle ne s'applique que si les précédentes ne s'appliquent pas.

ABSTRACTION (cf. E.2.1, page 134)

TAILLE
si longueur ≥ 13,5 **alors** *Taille = grand*
si longueur ≥ 3,4 **alors** *Taille = moyen*
sinon *Taille = petit*

COMPLEXITE
si taille max virages ≤ 0,09 **et** nombre grands virages = 1 **alors** *Complexité = zéro niveau*
si nombre grands virages ≤ 6 **alors** *Complexité = un niveau*
si base sur longueur ≥ 0,64 **alors** *Complexité = un niveau*
sinon *Complexité = plusieurs niveaux*

SINUOSITE
si nombre virages ≥ 16 **et** base sur longueur ≤ 0,61 **alors** S*inuosité = hétérogène*
si base sur longueur ≥ 0,97 **alors** *Sinuosité = nulle*
si base sur longueur ≥ 0,73 **alors** *Sinuosité = virages peu sinueux*
sinon *Sinuosité = épingles à cheveux*

FORME GENERALE
si base sur longueur ≥ 0,97 **alors** *Forme = ligne droite*
si nombre virages ≤ 2 **et** base sur longueur ≤ 0,32 **alors** *Forme = virage*
si nombre virages = 1 **alors** *Forme = virage*
si nombre grands virages ≥ 6 **alors** *Forme = longue série de virages*
sinon *Forme = courte série de virages*

EMPATEMENT
si homogénéité empâtement ≤ 0,81 **et** longueur empâtement ≥ 2,9 **alors** *Empâtement = hétérogène*
si longueur empâtement ≥ 0.9 **et** force empâtement ≥ 0,013 **alors** *Empâtement = fort*
si type empâtement ≠ aucun et force empâtement ≥ 0.008 **alors** *Empâtement = faible*
sinon *Empâtement = nul*

ENVIRONNEMENT
si surface conflit proche ≤ 0,1 **et** nb arcs proches par longueur ≤ 1,2 **alors** *Environnement = libre*
si surface conflit proche ≤ 0,5 **et** nombre arcs proches ≤ 2 **alors** *Environnement = libre*
sinon *Environnement = dense*

CHOIX de L'OPERATION (cf. E.2.2, page 138)

ACTION
si Type opération précédente = modification géométrique **et** Empâtement = nul **alors** *Action = Arrêt*
si Opération précédente = simplification **et** Empâtement = faible **alors** *Action = Arrêt*
si Algorithme précédent = plâtre **et** Empâtement = faible **alors** *Action = Arrêt*
si Complexité = zéro niveau **alors** *Action = Arrêt*
sinon *Action = Continuer*

TYPE OPERATION
si Action = arrêt **alors** *Type opération = Arrêt*
si Empâtement = hétérogène **alors** *Type opération = Focalisation*
sinon *Type opération = Modification géométrique*

OPERATION
si Action = arrêt **alors** *Opération = Arrêt*
si Type opération = focalisation **alors** *Opération = Focalisation*
si Empâtement = nul **alors** *Opération = Simplification*
sinon *Opération = Exagération*

APPLICABILITÉ DES ALGORITHMES (cf. E.2.3, page 141)

LISSAGE GAUSSIEN
si Opération = Simplification **et** Complexité ≠ plusieurs niveaux **alors** *Lissage Gaussien = applicable*
sinon *Lissage Gaussien = pas applicable*

PLATRE
si Opération = Simplification **et** Taille ≠ petit **alors** *Plâtre = applicable*
si Type opération = Modification **et** Empâtement ≠ fort **et** Empatement ≠ nul **alors** *Plâtre = applicable*
sinon *Plâtre = pas applicable*

ACCORDEON
si Empâtement = fort **et** Forme = courte série de virages **alors** *Accordéon = applicable*
sinon *Accordéon = pas applicable*

SCHEMATISATION
si Empâtement = fort **et** Complexité = plusieurs niveaux **alors** *Schématisation = applicable*
sinon *Schématisation = pas applicable*

FAILLE MIN
si Forme = virage **et** Empâtement ≠ nul **alors** *Faille Min = applicable*
sinon *Faille Min = pas applicable*

FAILLE MAX
si Forme = virage **et** Environnement = libre **et** Empâtement ≠ nul **alors** *Faille Max = applicable*
sinon *Faille Max = pas applicable*

CHOIX DE L'ALGORITHME (cf. E.2.4, page 142)

si Action = arrêt **alors** *Choix algo = Arrêt*

si Type opération = focalisation **alors** *Choix algo = Découpage conflit*

si Forme = virage **et** Empâtement = fort **alors** *Choix algo = Faille Min*
si Forme = virage **et** Environnement = dense **alors** *Choix algo = Faille Min*

si Faille Max = applicable **alors** *Choix algo = Faille Max*

si Schématisation = applicable **alors** *Choix algo = Schématisation*

si Empâtement = fort **alors** *Choix algo = Accordéon*

si Lissage Gaussien = pas applicable **alors** *Choix algo = Plâtre*

sinon *Choix algo = Lissage Gaussien*

PARAMETRAGE

PLATRE
Si Nombre virages < 4 **alors** Courbure = 150 * Largeur de symbole
Si Complexité = un niveau **alors** Courbure = 270 * Largeur de symbole
Sinon Courbure = 210 * Largeur de Symbole

LISSAGE GAUSSIEN
Si Largeur de symbole < 0.4 **alors** sigma = 150 * Largeur de symbole
Sinon sigma = 215 * Largeur de symbole

ACCORDEON
Si Complexité = plusieurs niveaux **alors** Exagération = 1.0
Si Environnement = dense **alors** Exagération = 1.0
Sinon Exagération = 1.1

5. AUTRES BASES DE CONNAISSANCES APPRISES POUR LES ROUTES

Règles apprises par RIPPER avec une méthode de réolution "directe"

Les règles de décision suivantes sont présentées à titre de comparaison avec les règles de apprises par les abstractions. Elles ont été apprises par un apprentissage direct, c'est à dire des mesures à l'algorithme à appliquer.

CHOIX DE L'ALGORITHME

si Type opération précédente = modification géométrique **et** longueur empâtement ≤ 0,6 **alors** *Choix Algo = Arrêt*
si base sur longueur ≥ 0,97 **alors** *Choix Algo = Arrêt*
si Algorithme précédent = Plâtre **alors** *Choix Algo = Arrêt*
si taille max virages ≤ 0,06 **alors** *Choix Algo = Arrêt*
si homogénéité empâtement ≤ 0,76 **et** nb arcs proches par longueur ≤ 0,3 **alors** *Choix Algo = Découpage empâtement*
si longueur base ≤ 0,29 **alors** *Choix Algo = Faille Max*
si taille moyenne virages ≥ 0,63 **alors** *Choix Algo = Faille Max*
si longueur empâtement ≥ 11,7 **alors** *Choix Algo = Schématisation*
si longueur empâtement ≥ 2,7 **et** résistance lissage ≤ 0,67 **alors** *Choix Algo = Schématisation*
si pourcentage empâtement ≥ 0,96 **et** résistance lissage ≥ 0,75 **alors** *Choix Algo = Accordéon*
si taille virages moyenne sur max ≤ 0,53 **et** nombre arcs proches par largeur ≥ 6,7 **alors** *Choix Algo = Plâtre*
si granularité ≥ 0,33 **et** longueur ≥ 2,2 **alors** *Choix Algo = Plâtre*
sinon *Choix Algo = Lissage Gaussien*

Règles apprises par RIPPER en modifiant le paramètre "cost of theory"

Les règles suivantes ont été apprises par Ripper avec la méthode de résolution complète, et en diminuant le paramètre "cost of theory". Ceci permet de donner un poids plus fort aux exemples mal classés dans le calcul de la longueur de description des règles. Ceci a pour effet de créer des règles plus détaillées qu'avec la valeur par défaut du paramètre. L'enchaînement de ces règles une erreur estimée par validation croisée de 27%. Ce qui constitue une légère amélioration par rapport aux résultats obtenus avec le paramètre par défaut (erreur estimée = 29%).

ABSTRACTION

TAILLE
si longueur ≥ 13,5 **alors** Taille = grand
si longueur ≥ 3,4 **alors** Taille = moyen
sinon Taille = petit

COMPLEXITE
si taille max virages ≤ 0,09 **et** nombre grands virages ≤ 1 **alors** Complexité = zéro niveau
si nombre grands virages ≥ 6 **et** base sur longueur ≤ 0,64 **alors** Complexité = plusieurs niveaux
si nombre grands virages ≥ 4 **et** taille max virages ≥ 0,66 **alors** Complexité = plusieurs niveaux
sinon Complexité = un niveau

SINUOSITE
si nombre virages ≥ 16 **et** base sur longueur ≤ 0,61 **alors** Sinuosité = hétérogène
si base sur longueur ≥ 0,97 **alors** Sinuosité = nulle
si base sur longueur ≥ 0,94 **et** nombre virages ≤ 1 **et** taille moyenne virages ≤ 0,06 **alors** Sinuosité = nulle
si base sur longueur ≥ 0,73 **alors** Sinuosité = virages peu sinueux
sinon Sinuosité = épingles a cheveux

FORME
si base sur longueur ≥ 0,97 **alors** Forme = ligne droite
si nombre virages ≤ 3 **et** base sur longueur ≤ 0,32 **et** nombre virages ≤ 2 **alors** Forme = virage
si nombre virages ≤ 1 **et** base sur longueur ≤ 0,58 **alors** Forme = virage
si nombre virages ≥ 5 **alors** Forme = longue série de virages
sinon Forme = courte série de virages

EMPATEMENT
si homogénéité empâtement ≤ 0,94 **et** longueur ≥ 6,4 **alors** Empâtement = hétérogène
si type empâtement = bilatéral **et** longueur empâtement ≥ 1,3 **alors** Empâtement = fort
si longueur empâtement ≥ 1 **et** longueur ≥ 2 **alors** Empâtement = fort
si force empâtement ≥ 0,011 **et** type empâtement = monolatéral **alors** Empâtement = peu
sinon Empâtement = nul

ENVIRONNEMENT
si surface de conflit proche ≤ 0,1 **et** nombre arcs proches par longueur ≤ 1,2 **alors** Environnement = libre
si surface de conflit proche ≤ 0,5 **et** nombre arcs proches ≤ 2 **alors** Environnement = libre
si surface de conflit proche ≤ 0,4 **et** nombre arcs proches par longueur ≤ 2 **alors** Environnement = libre
sinon Environnement = dense

CHOIX de L'OPERATION

ACTION
si Type opération précédente = modification géométrique **et** Empâtement = nul **alors** *Action = Arrêt*
si Opération précédente = simplification **et** Empâtement = faible **alors** *Action = Arrêt*
si Algorithme précédent = plâtre **et** Empâtement = faible **alors** *Action = Arrêt*
si Complexité = zéro niveau **alors** *Action = Arrêt*
sinon *Action = Continuer*

TYPE OPERATION
si Action = arrêt **alors** *Type opération = Arrêt*
si Empâtement = hétérogène **alors** *Type opération = Focalisation*
sinon *Type opération = Modification géométrique*

OPERATION
si Action = arrêt **alors** *Opération = Arrêt*
si Type opération = focalisation **alors** *Opération = Focalisation*
si Empâtement = nul **alors** *Opération = Simplification*
sinon *Opération = Exagération*

APPLICABILITE DES ALGORITHMES

GAUSS
si Opération = simplification **et** Complexité ≠ plusieurs niveaux **alors** Gauss = applicable
sinon Gauss = pas applicable

PLATRE
si Opération =simplification **et** Taille ≠ petit **alors** Plâtre = applicable
si Type Opération = modification **et** Empâtement ≠ fort **et** Forme ≠ virage **et** Empâtement ≠ nul **alors** Plâtre = applicable
sinon Plâtre = pas applicable

ACCORDEON
si Empâtement = fort **et** Forme = courte série de virages **alors** Accordéon = applicable
sinon Accordéon = pas applicable

SCHEMATISATION
si Empâtement = fort **et** Complexité = plusieurs niveaux **alors** Schématisation = applicable
si Empâtement = fort **et** Taille = moyen **alors** Schématisation = applicable
si Empâtement = fort **et** Algorithme précédent = Accordéon avant **alors** Schématisation = applicable
sinon Schématisation = pas applicable

FAILLE MIN
si Forme = virage **et** Empâtement ≠ nul **alors** Faille min = applicable
sinon Faille min = pas applicable

FAILLE MAX
si Forme = virage **et** Empatement ≠ nul **et** Environnement = libre **alors** Faillle max = applicable
sinon Faillle max = pas applicable

CHOIX DE L'ALGORITHME

si Action = arrêt **alors** *Choix algo = Arrêt*

si Type opération = focalisation **alors** *Choix algo = Découpage conflit*

si Forme = virage **et** Empâtement = fort **alors** *Choix algo = Faille Min*
si Faille Min = applicable **et** Environnement = dense **alors** *Choix algo = Faille Min*

si Faille Max = applicable **alors** *Choix algo = Faille Max*

si Schématisation = applicable **alors** *Choix algo = Schématisation*

si Empâtement = fort **alors** *Choix algo = Accordéon*

si Lissage Gaussien = pas applicable **alors** *Choix algo = Plâtre*

sinon *Choix algo = Lissage Gaussien*

Arbres de décision appris par C4.5 [Quinlan 93]

Les arbres de décision appris par C4.5 [Quinlan 93] suivants sont présentés à titre de comparaison avec les règles apprises par RIPPER. Les paramètres par défaut de C4.5 ont été utilisés pour construire ces arbres de décision.

ABSTRACTIONS

Taille

- longueur ≤ 3,8 → **petit**
- 3,8 < longueur ≤ 12,5 → **moyen**
- longueur > 12,5 → **grand**

Sinuosité

- base sur longueur ≤ 0,7
  - nb virages ≤ 15 → **épingles à cheveux**
  - nb virages > 15 → **hétérogène**
- 0,7 < base sur longueur ≤ 0,96 → **virages peu sinueux**
- base sur longueur > 0,96 → **nulle**

Complexité

- taille max virages ≤ 0,1
  - nb grands virages ≤ 1 → **zéro niveau**
  - nb grands virages > 1 → **un niveau**
- 0,1 < taille max virages
  - longueur ≤ 7,0 → **un niveau**
  - 7,0 < longueur ≤ 29,5
    - base sur longueur > 0,6 → **un niveau**
    - base sur longueur ≤ 0,6 → **plusieurs niveaux**
  - longueur > 29,5 → **plusieurs niveaux**

Empâtement

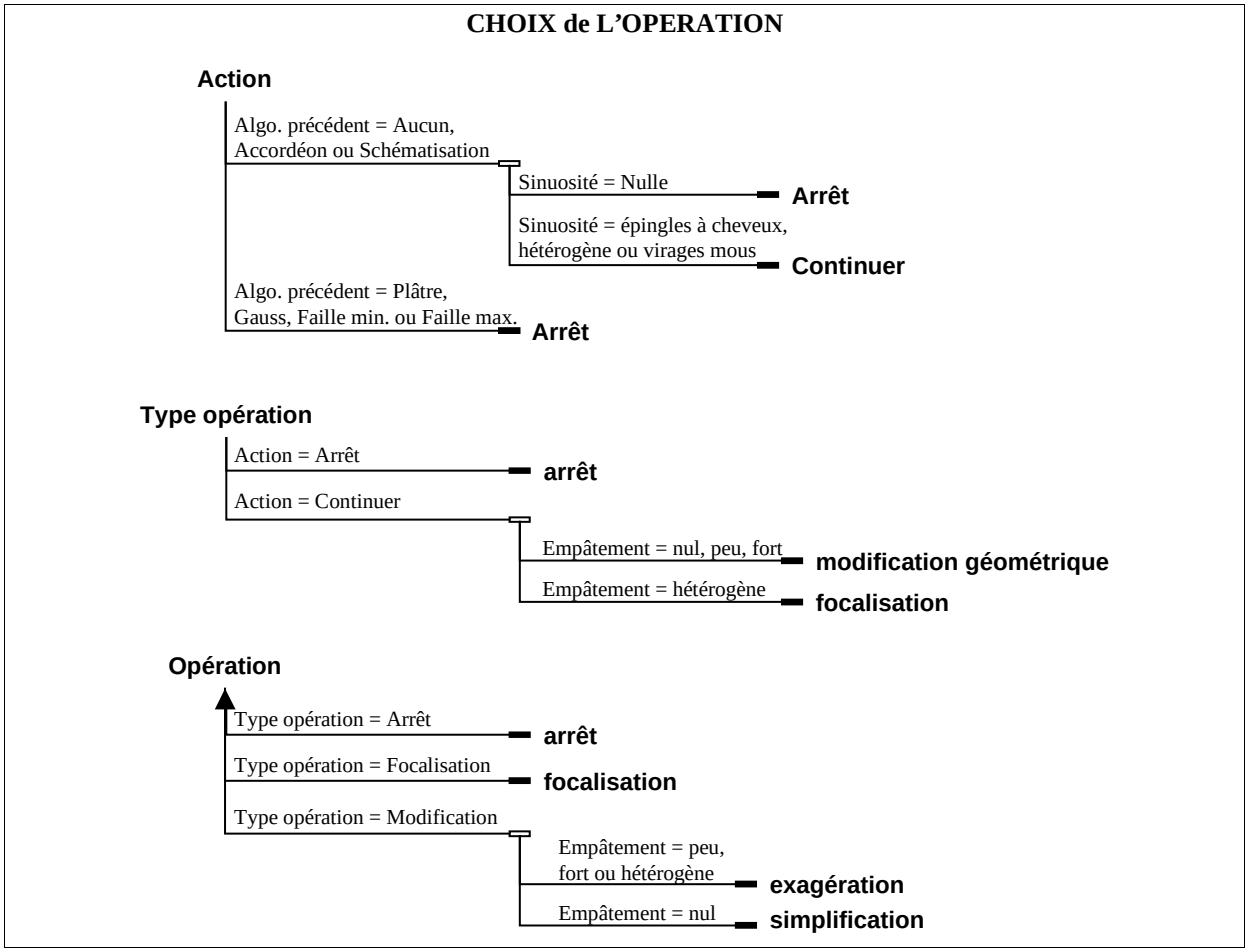
- force empâtement ≤ 0,015 → **nul**
- force empâtement > 0,015
  - longueur empâtement ≤ 1,7 → **peu**
  - longueur empâtement > 1,7
    - % empâtement ≤ 0,62 → **hétérogène**
    - % empâtement > 0,62 → **fort**

Forme

- nb virages ≤ 4
  - base sur longueur ≤ 0,96
    - nb virages ≤ 1 → **virage**
    - nb virages > 1
      - base sur longueur ≤ 0,26 → **virage**
      - base sur longueur > 0,26 → **courte série de virages**
  - base sur longueur > 0,96 → **ligne droite**
- nb virages > 4 → **longue série de virages**

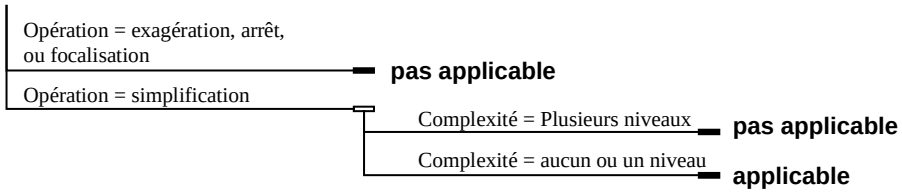
Environnement

- surface conflit proche ≤ 0,57
  - nb arcs proches ≤ 2 → **libre**
  - nb arcs proches> 2 → **dense**
- surface conflit proche > 0,57 → **dense**

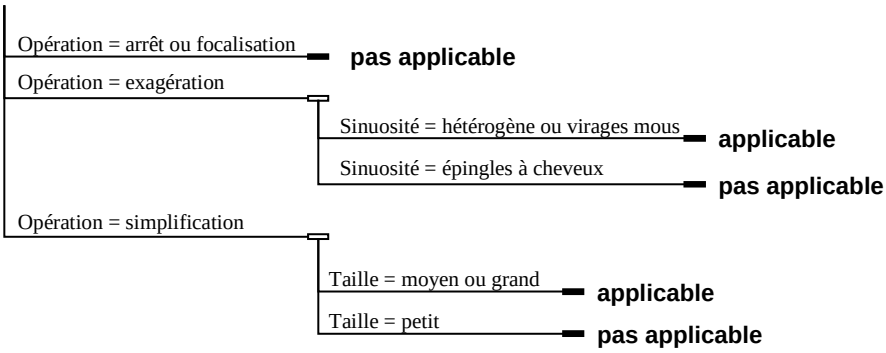

CHOIX de L'OPERATION
Action
Algo. précédent = Aucun, Accordéon ou Schématisation
Sinuosité = Nulle
Arrêt
Sinuosité = épingles à cheveux, hétérogène ou virages mous
Continuer
Algo. précédent = Plâtre, Gauss, Faille min. ou Faille max.
Arrêt
Type opération
Action = Arrêt
arrêt
Action = Continuer
Empâtement = nul, peu, fort
modification géométrique
Empâtement = hétérogène
focalisation
Opération
Type opération = Arrêt
arrêt
Type opération = Focalisation
focalisation
Type opération = Modification
Empâtement = peu, fort ou hétérogène
exagération
Empâtement = nul
simplification

APPLICABILITÉS

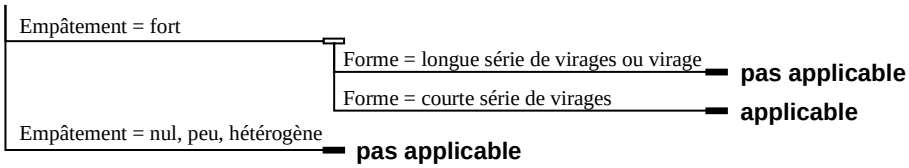
Gauss



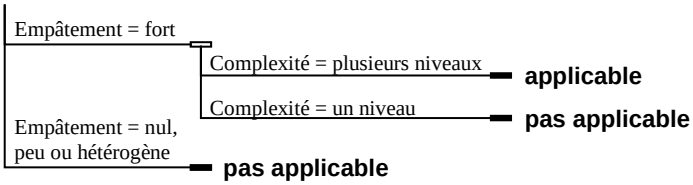
Plâtre



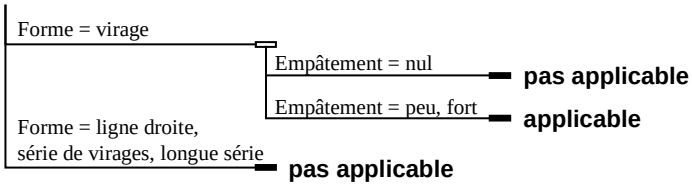
Accordéon



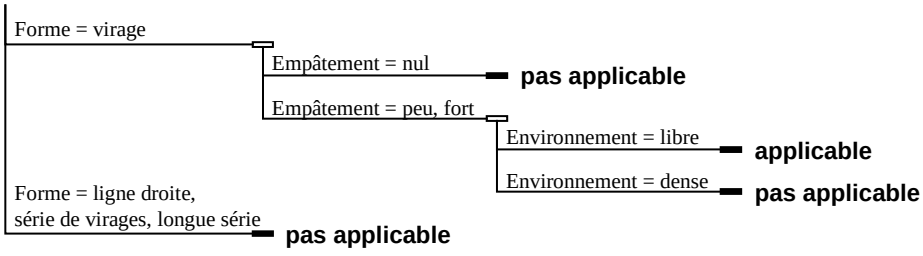
Schématisation



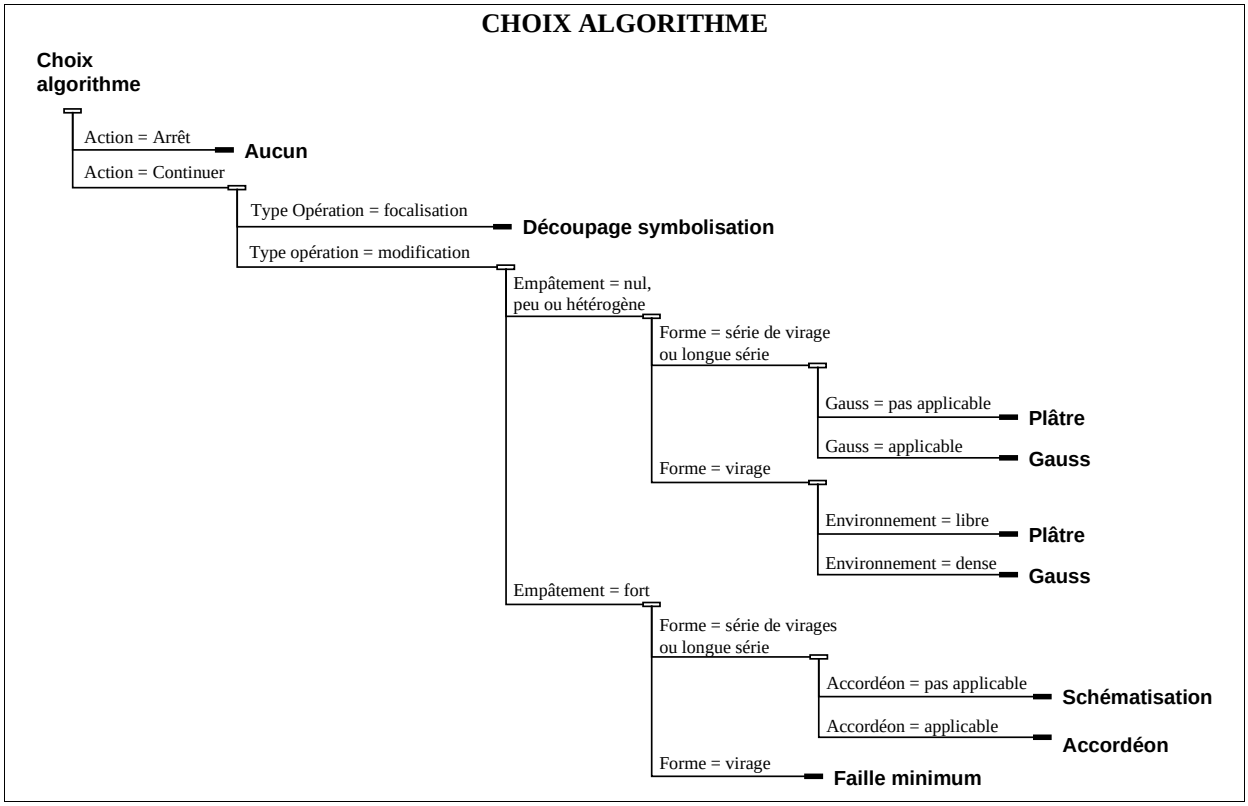
Faille Minimum



Faille Maximum



S


CHOIX ALGORITHME
Choix algorithme
Action = Arrêt
Aucun
Action = Continuer
Type Opération = focalisation
Découpage symbolisation
Type opération = modification
Empâtement = nul, peu ou hétérogène
Forme = série de virage ou longue série
Gauss = pas applicable
Plâtre
Gauss = applicable
Gauss
Forme = virage
Environnement = libre
Plâtre
Environnement = dense
Gauss
Empâtement = fort
Forme = série de virages ou longue série
Accordéon = pas applicable
Schématisation
Accordéon = applicable
Accordéon
Forme = virage
Faille minimum

VI. Résultats de l'apprentissage sur les bâtiments

Cette annexe présente les résultats de l'apprentissage réalisé sur les bâtiments. Ces tests ont été réalisés pendant la mise au point de notre approche définie au chapitre D. Pour réaliser des expérimentations aussi détaillées que celles réalisées sur les routes, il nous faudrait modifier la forme des exemples utilisés et les recueillir à nouveau.

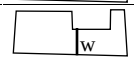
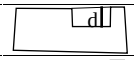
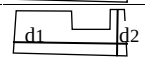
Ces tests concernent la généralisation cartographique des bâtiments de la BDTopo (BDG de l'IGN de précision métrique, d'échelle de l'ordre du 1:15.000) pour réaliser des cartes topographiques au 1:50.000.

1. MESURES ET DESCRIPTEURS ABSTRAITS

Neuf mesures et six descripteurs abstraits ont été choisis pour décrire un bâtiment (cf. tableaux ci-dessous). Notons que nous avons considéré les bâtiments isolément, contrairement aux routes dont nous décrivons l'environnement.

Attribut	*Valeurs possibles*
Forme globale	rectangle | en L | en escalier | autre
Taille	petit | moyen | grand
Nombre d'orientations principales	une | plusieurs
Granularité	faible | forte
Décrochements importants	oui | non
Existence de formes spéciales	aucune | rondes ou triangulaires

Descripteurs abstraits d'un bâtiment cartographié

Mesure	*Description sommaire*	
Surface	Surface du bâtiment divisée par la surface minimale autorisée par les spécifications de la carte.	
Longueur minimale	Longueur du plus petit segment constituant le bâtiment, divisée par la longueur minimale autorisée par les spécifications.	
Largeur minimale	Largeur minimale du bâtiment, divisée par la longueur minimale autorisée par les spécifications.	
Concavité	Mesure basée sur la comparaison de la surface du bâtiment et de la surface de son enveloppe convexe	
Compacité	Mesure basée sur la comparaison de la surface du bâtiment et de son périmètre	
Profondeur	Mesure basée sur l'écart entre le bâtiment et son enveloppe convexe	
Elongation	Mesure basée sur la comparaison de la largeur et de la longueur du bâtiment, définies en fonction de l'orientation générale du bâtiment.	
Degré d'équarrissage	Mesure estimant dans quelle mesure les angles sont droits	
Nombre de points	Nombre de couples de coordonnées décrivant le bâtiment	

Mesures utilisées pour décrire un bâtiment

2. ALGORITHMES DE TRANSFORMATION UTILISES

Quatre algorithmes ont été choisis pour transformer ces bâtiments : *dilatation*, *équarrissage* [Regnauld, Edwardes et Barrault 99], *simplification* [Ruas 88], *élargissement de la plus petite largeur* [Regnauld, Edwardes et Barrault 99]. Ces algorithmes ont des paramètres directement reliés aux spécifications de la carte. Une étude des combinaisons possibles de ces algorithmes met en valeur que, dans le contexte de nos tests de la généralisation au 1:50.000, seulement cinq combinaisons de ces algorithmes sont pertinentes [Regnauld, Edwardes et Barrault 99].

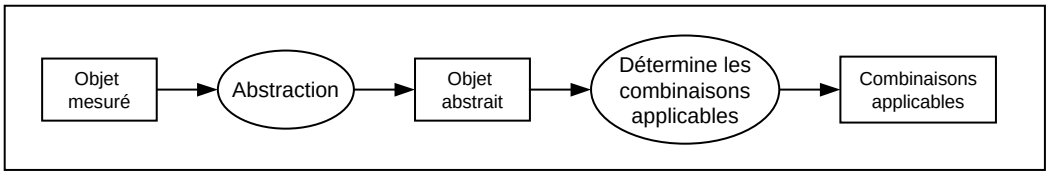
Ainsi, contrairement au cas des routes, nous ne hiérarchisons pas les algorithmes selon les opérations qu'ils peuvent réaliser. Au lieu de cela, nous considérons chaque combinaison comme un algorithme à part entière. Il ne nous est alors pas nécessaire d'apprendre quand arrêter le processus puisque la généralisation est considérée ici comme l'application d'une seule des cinq combinaisons identifiées. L'arrêt est systématique après application d'une de ces configurations. Dans la même étude de ces bâtiments, les auteurs considèrent que la seule l'utilisation de ces cinq combinaisons serait trop limitante pour des échelles plus petites que le 1:50.000. En ce cas, une approche telle que celle proposée pour les routes serait utile. Les combinaisons d'algorithmes utilisées sont présentées ci-dessous.

Dilatation
Simplification
Equarrissage
Simplification, équarrissage puis élargissement
Equarrissage, simplification puis élargissement

Combinaisons pertinentes dans le cadre de nos tests

3. METHODE DE RESOLUTION DE PROBLEME POUR LES BATIMENTS

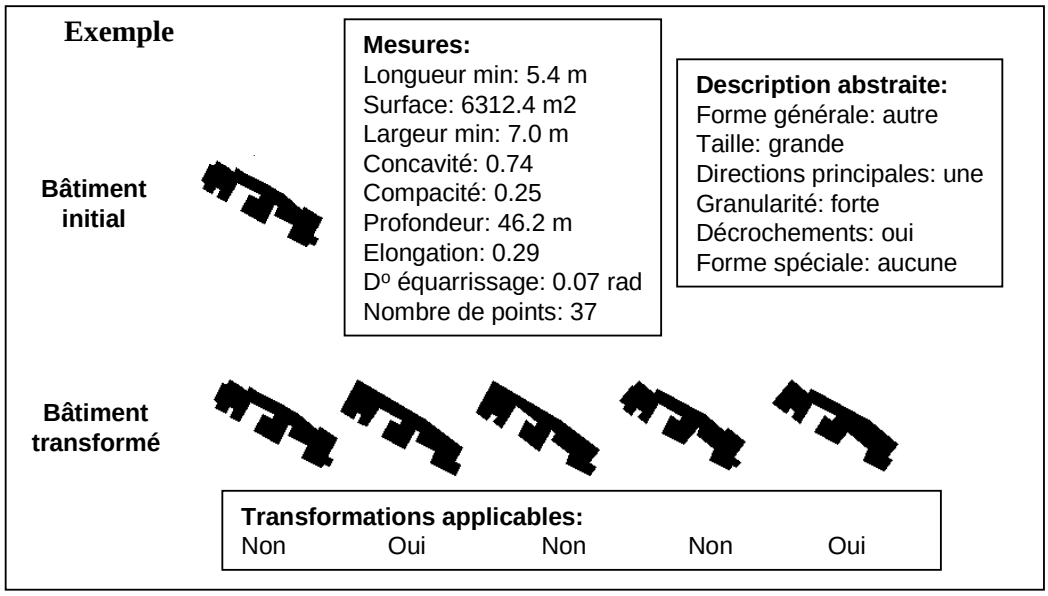
La figure suivante présente la méthode de résolution de problème définie pour les bâtiments. Comme pour les routes, celle-ci contient une phase d'abstraction, une phase de détermination des algorithmes applicables. Par contre, elle ne contient pas de phase de détermination de l'opération à réaliser. De plus, nous n'avons pas cherché à apprendre quel algorithme appliquer. Notons que si cette méthode de résolution de problème est beaucoup plus simple que celle adaptée au routes, ceci est principalement dû au fait que nous avons moins intensivement étudié les bâtiments que les routes, et non que la généralisation des bâtiments est moins complexe que celle des routes.



Méthode de résolution de problème adaptée aux bâtiments

4. EXEMPLES RECUEILLIS

Nous avons utilisé 80 exemples, représentant donc 80 transformations puisqu'une seule transformation (i.e. une des cinq combinaisons d'algorithmes) est effectuée par bâtiment. Ces bâtiments sont issus deux villes différentes : Strasbourg et Lavérune (près de Montpellier). Chaque bâtiment a été automatiquement qualifié par les neuf mesures choisies puis interactivement qualifié par les six descripteurs abstraits. Ensuite, chacune des transformation possible a été appliquée et les résultats ont été interactivement qualifiés d'*acceptable* ou non. La figure suivante présente un des 80 exemples utilisés.



5. REGLES APPRISES AVEC ABSTRACTION

Les règles suivantes présentent les résultats de l'apprentissage pour déterminer, d'une part, la valeur des attributs abstraits en fonction des mesures et, d'autre part, les applicabilités des transformations en fonction des attributs abstraits. Les pourcentages entre parenthèses représentent les erreurs estimées, par validation croisée, des hypothèses apprises. Pour les applicabilités, les premiers chiffres représentent l'erreur estimée de l'étape seule, et les deuxième chiffres représentent l'erreur estimée de l'enchaînement des inférences d'abstraction et d'applicabilité.

ABSTRACTIONS

DETERMINATION DE LA FORME GENERALE (31%)
Si Elongation ≤ 0,392 **et** Surface ≤ 5,76 **alors** *Forme = en escalier*
Si Profondeur ≥ 0,25 **et** Nombre de points ≤ 8 **alors** *Forme = en L*
Si Profondeur ≤ 0,207 **alors** *Forme = Rectangle*
Sinon *Forme = Autre*

DETERMINATION DE LA TAILLE (14%)
Si Surface ≤ 1,37 **alors** *Taille = Petit*
Si Surface ≤ 5,76 **alors** *Taille = Moyen*
Sinon *Taille = Grand*

DETERMINATION DES FORMES SPECIALES (19%)
Si Longueur minimale ≤ 0,38 **et** Nombre de points ≥ 31 **alors** *Formes spéciales = Rondes ou triangulaires*
Sinon *Formes spéciales = Aucune*

DETERMINATION DE LA GRANULARITE (9%)
Si Longueur minimale ≤ 0,86 **alors** *Granularité = Forte*
Si Surface ≤ 0,29 **alors** *Granularité = Forte*
Sinon *Granularité = Faible*

DETERMINATION DE L'EXISTENCE DE DECROCEMENTS IMPORTANTS (19%)
Si Profondeur ≥ 0,178 **et** Largeur minimale ≥ 0,75 **alors** *Décrochements importants = oui*
Sinon *Décrochement importants = non*

APPLICABILITES

DETERMINATION DE L'APPLICABILITE DE DILATATION (8%,**8**%)
Si Granularité = forte **et** Forme générale = rectangle **et** Formes spéciales = aucune **et** Taille = petit **alors** *Dilatation = applicable*
Si Taille = petit **et** Forme générale = en L **alors** *Dilatation = applicable*
Sinon *dilatation = pas applicable*

DETERMINATION DE L'APPLICABILITE DE SIMPLIFICATION (22%,**32**%)
Si Taille = Grand **alors** *Simplification = Applicable*
Si Décrochements importants = oui **et** Forme spéciale = aucune **alors** *Simplification = applicable*
Sinon *Simplification = pas applicable*

DETERMINATION DE L'APPLICABILITE DE SIMPLIFICATION-EQUARISSAGE-ELARGISSEMENT (14%,**27**%)
Si Décrochements importants = non **et** Forme générale = rectangle **alors** *Simplif-Equa-Elarg = applicable*
Si Décrochements importants = non **et** Formes spéciales = aucune **alors** *Simplif-Equa-Elarg = applicable*
Si Forme générale = rectangle **et** Taille = grand **alors** *Simplif-Equa-Elarg = applicable*
Si Forme générale = en L **et** Taille = moyen **alors** *Simplif-Equa-Elarg = applicable*
Sinon *Simplif-Equa-Elarg = pas applicable*

DETERMINATION DE L'APPLICABILITE DE EQUARRISSAGE (12%, **27**%)
Si Granularité = forte **alors** *Equarrissage = applicable*
Si Taille = moyen **et** Décrochements importants = oui **alors** *Equarrissage = applicable*
Si Taille = petit **et** Forme générale = en L **alors** *Equarrissage = applicable*
Si Décrochements importants = oui **et** Forme générale = en L **alors** *Equarrissage = applicable*
Si Forme générale = autre **et** Taille = moyen **et** Formes spéciales = Rondes ou triangulaires **alors** *Equarrissage = applicable*
Sinon *Equarissage = pas applicable*

DETERMINATION DE L'APPLICABILITE DE EQUARISSAGE- SIMPLIFICATION-ELARGISSEMENT (24%, **31**%)
Si Décrochements importants = non alors *Equa-Simplif- Elarg = applicable*
Sinon *Equa-Simplif- Elarg = pas applicable*

6. REGLES APPRISES SANS ABSTRACTION

Les règles suivantes présentent les résultats d'un apprentissage direct, c'est à dire des applicabilités des transformations en fonction des mesures, puis leurs erreurs estimées par validation croisée. Les pourcentages entre parenthèses représentent les erreurs estimées, par validation croisée, des hypothèses apprises. Celles-ci sont plus importantes que les taux d'erreur estimés pour l'enchaînement des inférences d'abstraction et d'applicabilité présentées ci-dessus.

APPLICABILITES

DETERMINATION DE L'APPLICABILITE DE DILATATION (8%)
Si Compacité ≥ 0,745 **et** Degré d'équarrissage ≤ 0,11 **alors** *Dilatation = applicable*
Si Surface ≥ 83,3 **alors** *Dilatation = applicable*
Sinon *dilatation = pas applicable*

DETERMINATION DE L'APPLICABILITE DE SIMPLIFICATION (34%)
Si Elongation ≥ 0,639 **alors** *Simplification = applicable*
Si Surface ≥ 16.0 **et** Compacité ≤ 0,37 **alors** *Simplification = applicable*
Si Surface ≥ 2,3 **et** Concavité ≥ 0,856 **alors** *Simplification = applicable*
Si Profondeur ≥ 0,339 **et** Surface ≤ 1,11 **alors** *Simplification = applicable*
Sinon *Simplification = pas applicable*

DETERMINATION DE L'APPLICABILITE DE SIMPLIFICATION-EQUARISSAGE-ELARGISSEMENT (23%)
Si Concavité ≥ 0,856 **alors** *Simplif-Equa-Elarg = applicable*
Si Profondeur ≥ 0,345 **et** Nombre points ≤ 14 **alors** *Simplif-Equa-Elarg = applicable*
Si Surface ≥ 52,5 **alors** *Simplif-Equa-Elarg = applicable*
Sinon *Simplif-Equa-Elarg = pas applicable*

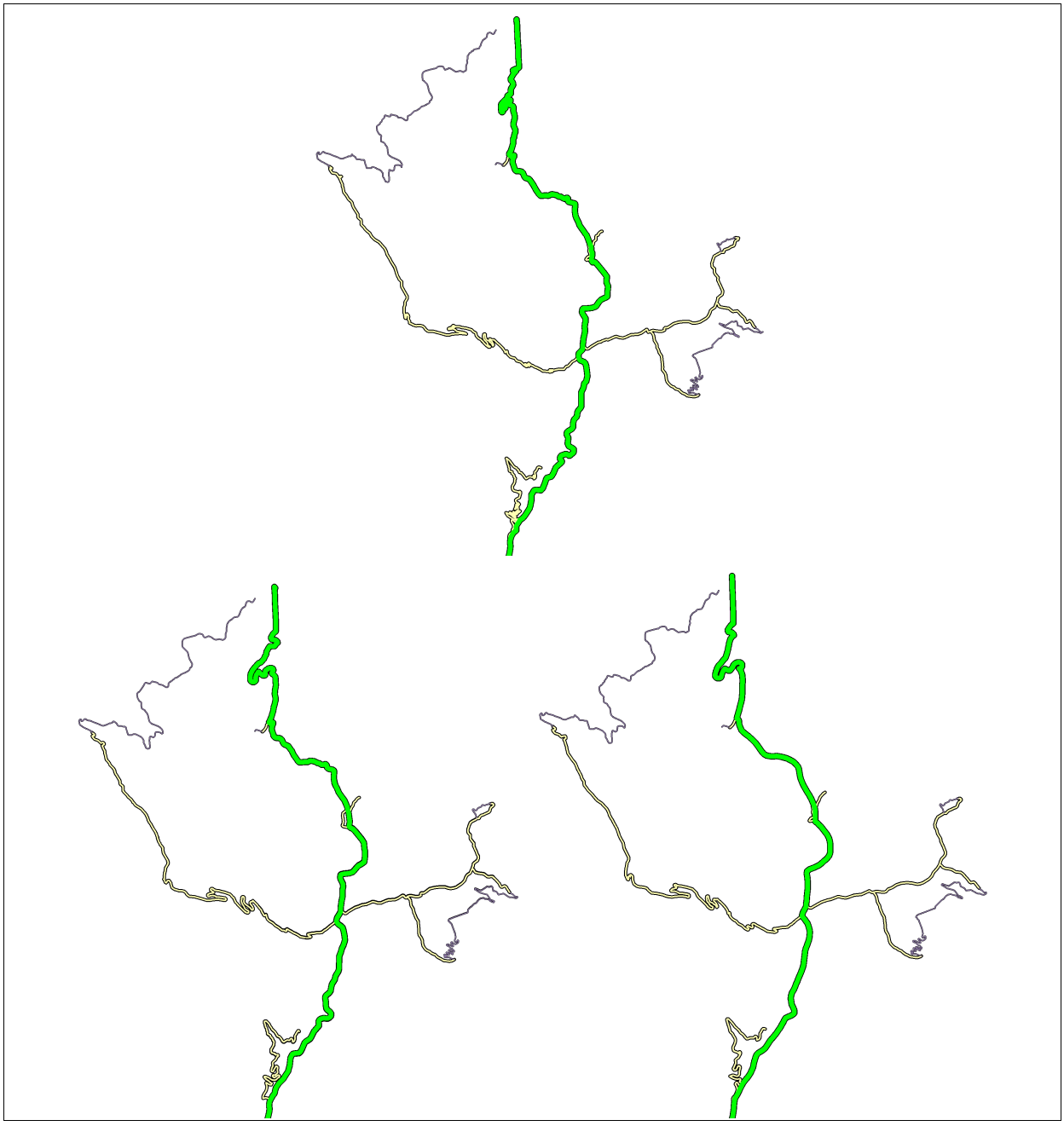
DETERMINATION DE L'APPLICABILITE DE EQUARRISSAGE (37%)
Si Profondeur ≤ 0,304 **et** Elongation ≤ 0,529 **alors** *Equarrissage = applicable*
Si Longueur minimale ≥ 0,86 **alors** *Equarrissage = applicable*
Si Surface ≥ 1,8 **et** Surface ≤ 2,4 **alors** *Equarrissage = applicable*
Si Surface ≤ 4,9 **et** Nombre points ≥ 16 **alors** *Equarrissage = applicable*
Si Surface ≤ 0,87 **et** Longueur minimale ≥ 0,55 **alors** *Equarrissage = applicable*
Si Concavité ≤ 0,674 **et** Longueur minimale ≥ 0,24 **alors** *Equarrissage = applicable*
Sinon *Equarissage = pas applicable*

DETERMINATION DE L'APPLICABILITE DE EQUARISSAGE- SIMPLIFICATION -ELARGISSEMENT (34%)
Si Degré d'équarrissage ≤ 0,23 **alors** *Equa-Simplif- Elarg = applicable*
Si Concavité ≥ 0,863 **alors** *Equa-Simplif- Elarg = applicable*
Si Largeur minimale ≥ 0,68 **et** Largeur minimale ≤ 0,78 **alors** *Equa-Simplif- Elarg = applicable*
Sinon *Equa-Simplif- Elarg = pas applicable*

VII. Résultats cartographiques sur les routes

1. RESULTATS SUR LA ZONE DES EXEMPLES D'APPRENTISSAGE

Les images suivantes montrent les résultats de l'application des règles apprises (avec abstraction) sur la zone généralisée interactivement afin de fournir les exemples d'apprentissage. Les images montrent la zone avant généralisation (en haut), les résultats de la généralisation interactive (en bas à gauche), et les résultats de la généralisation automatique par le processus appris (en bas à droite). Les principales différences résident, d'une part, dans l'absence d'opération de déplacement dans le traitement automatique et, d'autre part, dans les différences de paramétrage des lissages entre les résultats interactifs et automatiques. Ces différences de paramétrage sont volontaires. En effet, nous avons estimé que, d'un point de vue cartographique, nous n'avions pas suffisamment forcé les lissages durant le traitement interactif, et avons donc accentué ce paramétrage dans les traitements automatiques.

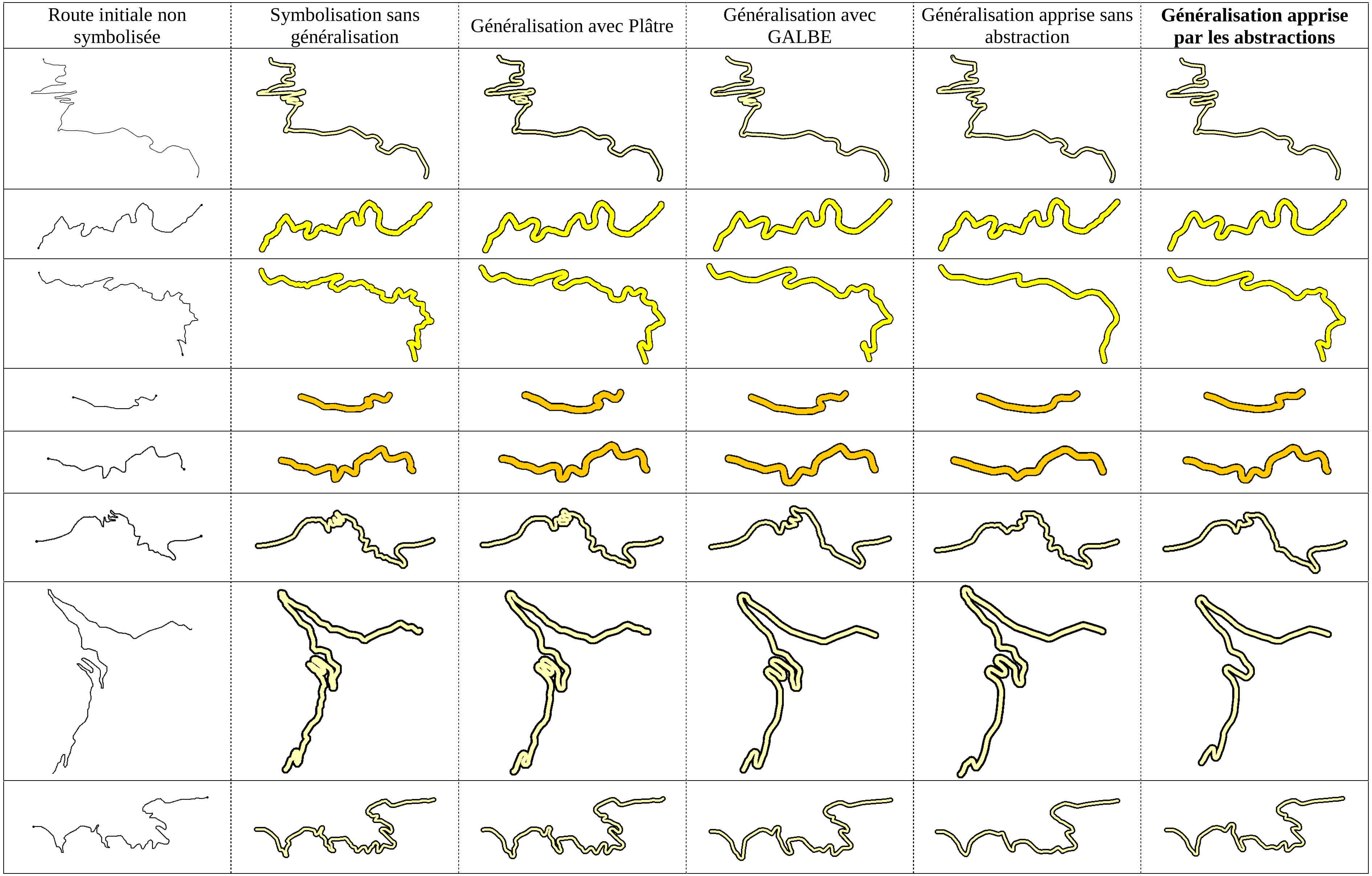


2. COMPARAISON AVEC GALBE, PLATRE, ET LE PROCESSUS APPRIS SANS ABSTRACTION

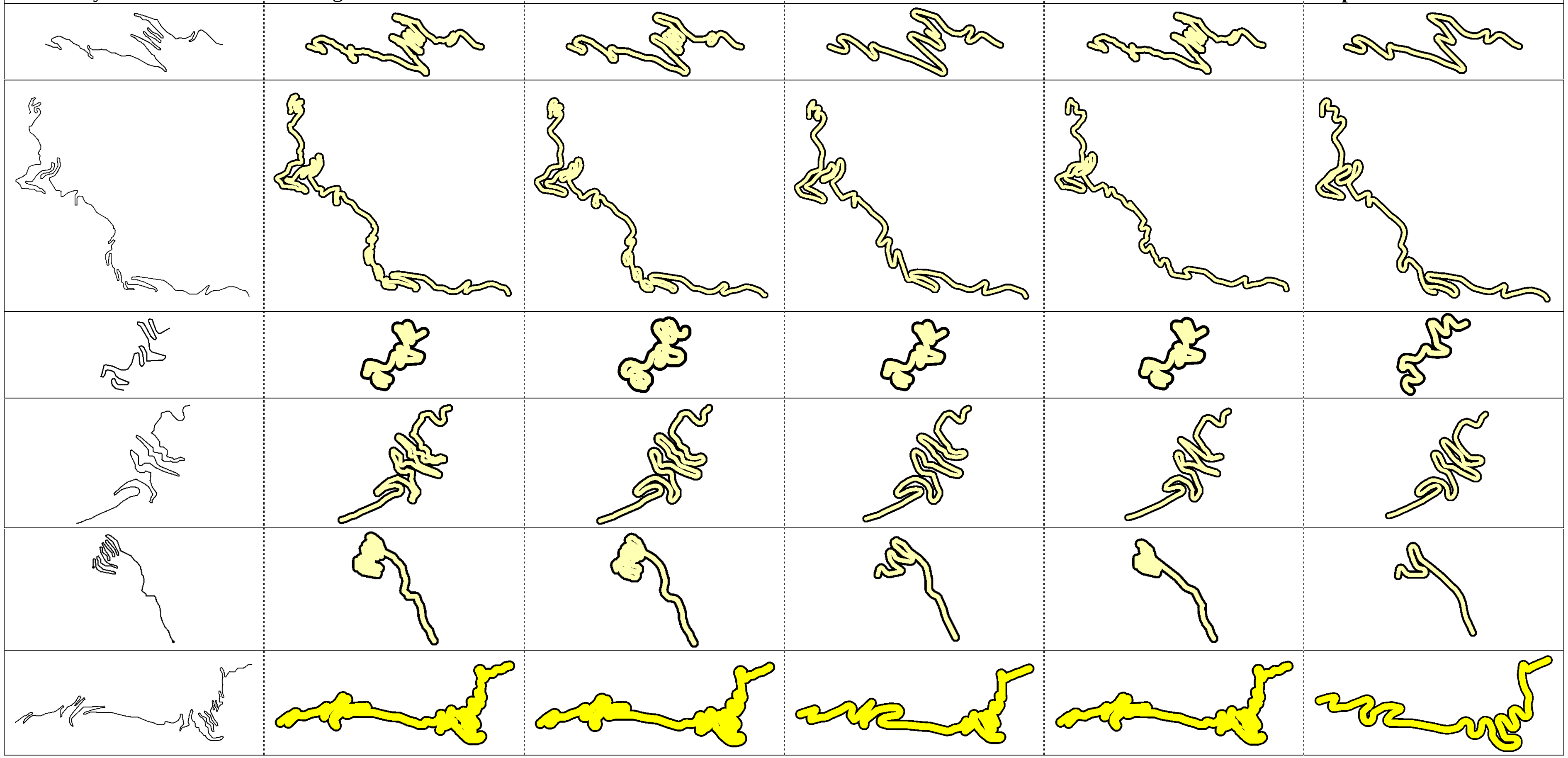
Les images des pages suivantes montrent les résultats du processus automatique appris avec abstraction. Les résultats sont comparés avec les autres processus automatiques mentionnés dans ce mémoire. Les images montrent successivement des routes :

- non symbolisées,
- symbolisées sans généralisation,
- généralisées avec GALBE (cf. chapitre B),
- généralisées avec Plâtre ([Fritsch 97], cf. annexe II-4 p.187), que nous considérons comme le meilleur algorithme de généralisation du routier comparable aux processus que nous proposons, puisqu'il est le seul, à notre connaissance, à effectuer à la fois des opérations de simplification et de caricature.
- généralisées par le système issu des règles apprises sans abstraction (cf. E.4, p.154)
- et enfin généralisées par le système issu des règles apprises avec abstraction (cf. chapitre E et annexe V p.194).

Les routes présentées sont extraites de la BDCarto sur les Alpes, les Pyrénées ou la Corse. La généralisation a été réalisée pour une cartographie au 1/250.000, c'est à dire à l'échelle utilisée pour constituer les exemples ayant servi à apprendre les règles.

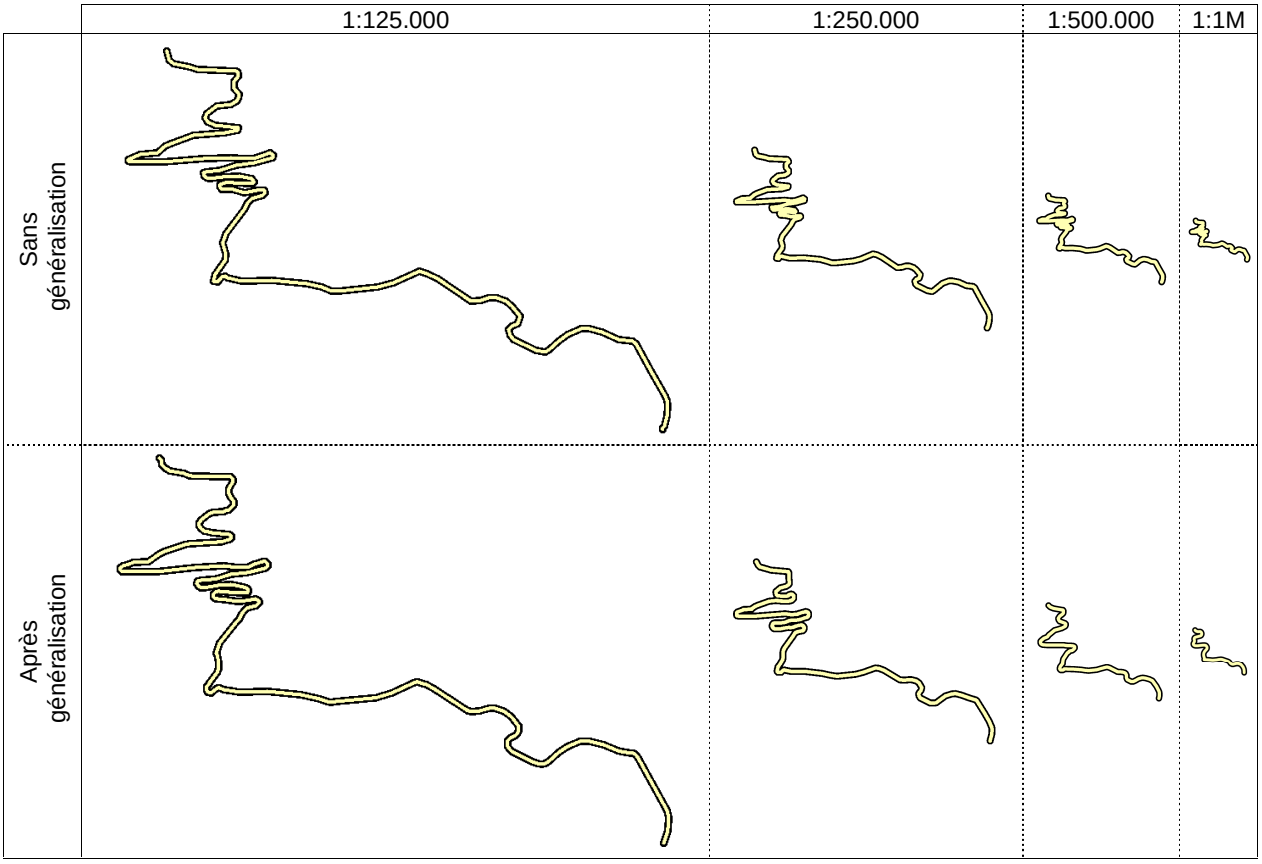

Route initiale non symbolisée
Symbolisation sans généralisation
Généralisation avec Plâtre
Généralisation avec GALBE
Généralisation apprise sans abstraction
Généralisation apprise par les abstractions

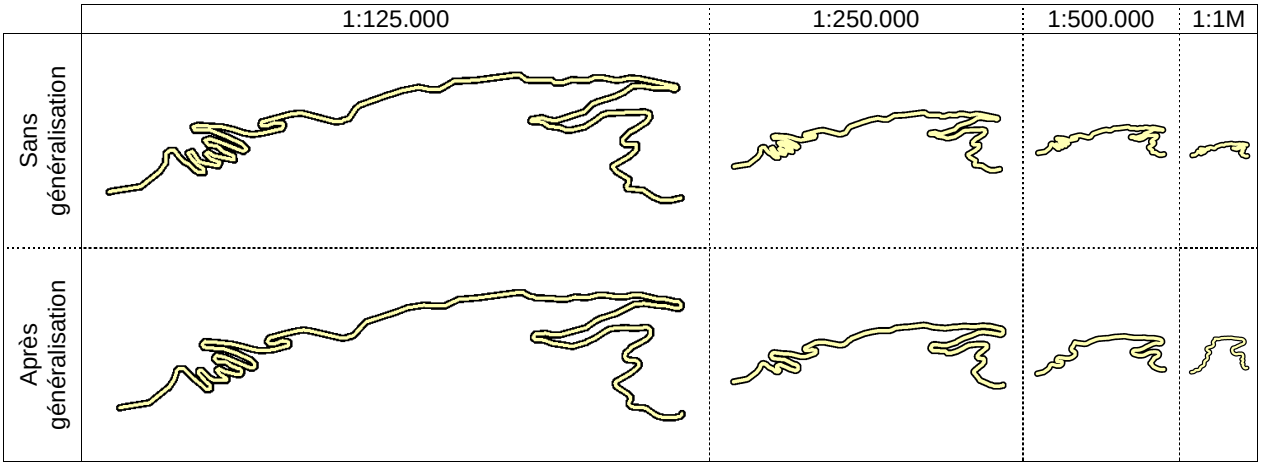
Route initiale non symbolisée	Symbolisation sans généralisation	Généralisation avec Plâtre	Généralisation avec GALBE	Généralisation apprise sans abstraction	**Généralisation apprise par les abstractions**

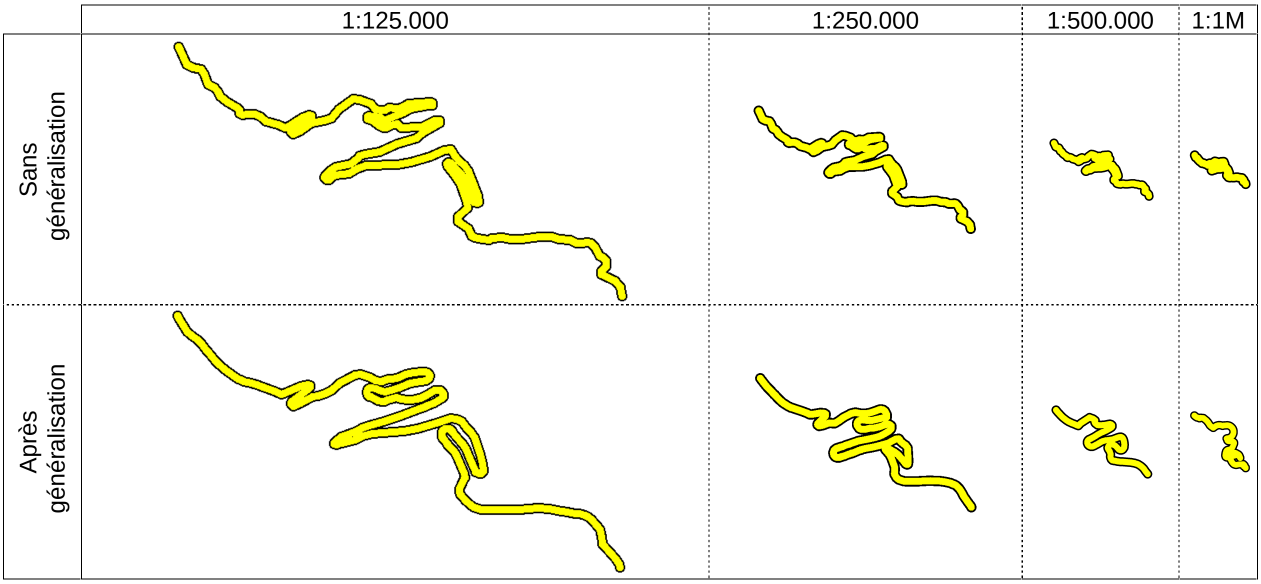


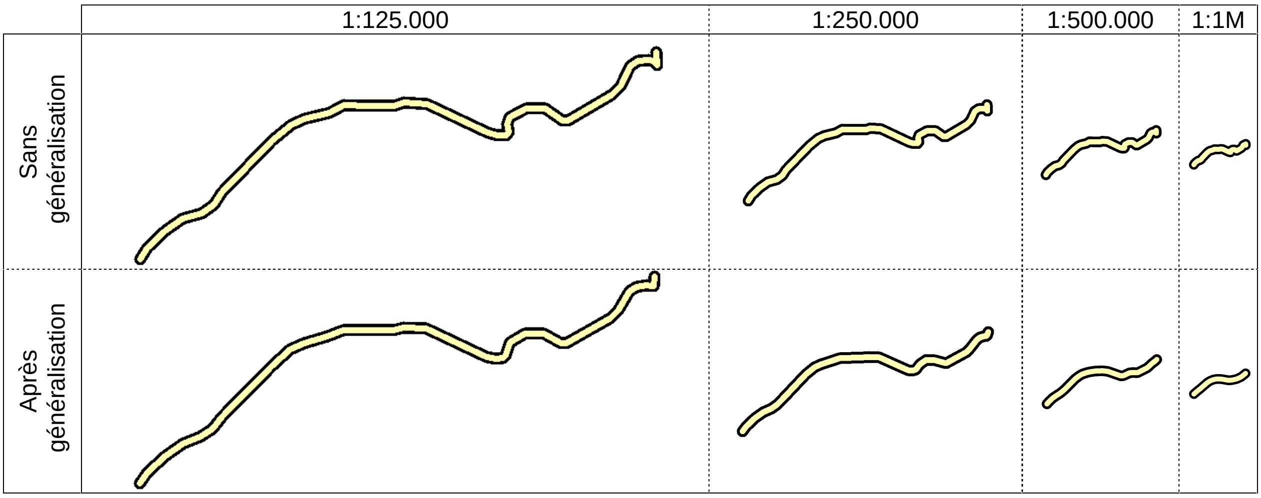
3. RESULTATS DU PROCESSUS APPRIS AVEC ABSTRACTION A DIFFERENTES ECHELLES

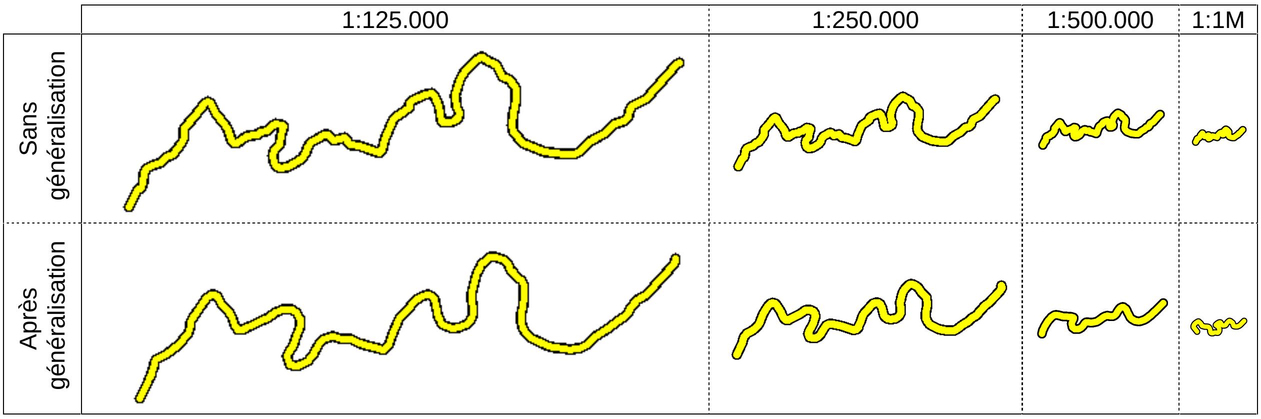
Les images suivantes montrent le résultat du processus appris sur différentes routes et à différentes échelles. Les images montrent successivement les routes avant et après traitement, pour une cartographie au 1:125.000, 1:250.000, 1:500.000 et 1:1.000.000. Les images suivantes sont issues de données de la BDCarto sur les Alpes, les Pyrénées ou la Corse, en dehors de la zone ayant fourni les exemples d'apprentissage.

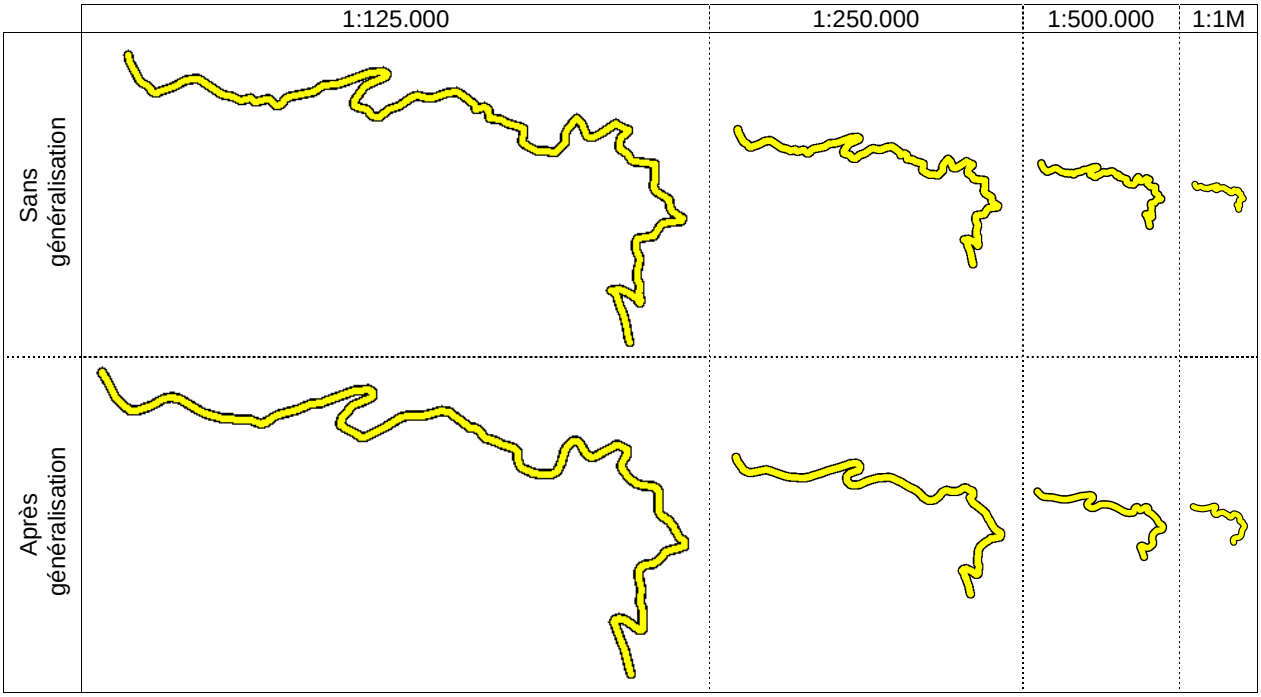


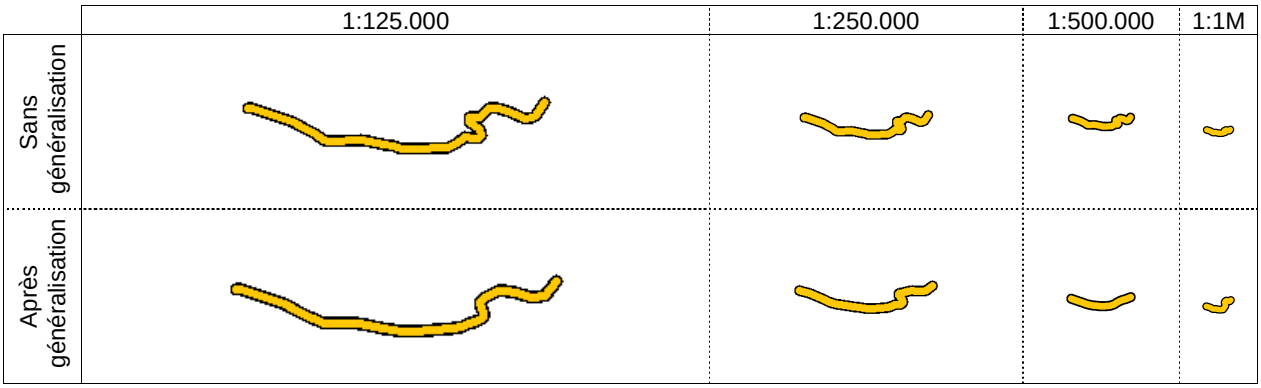


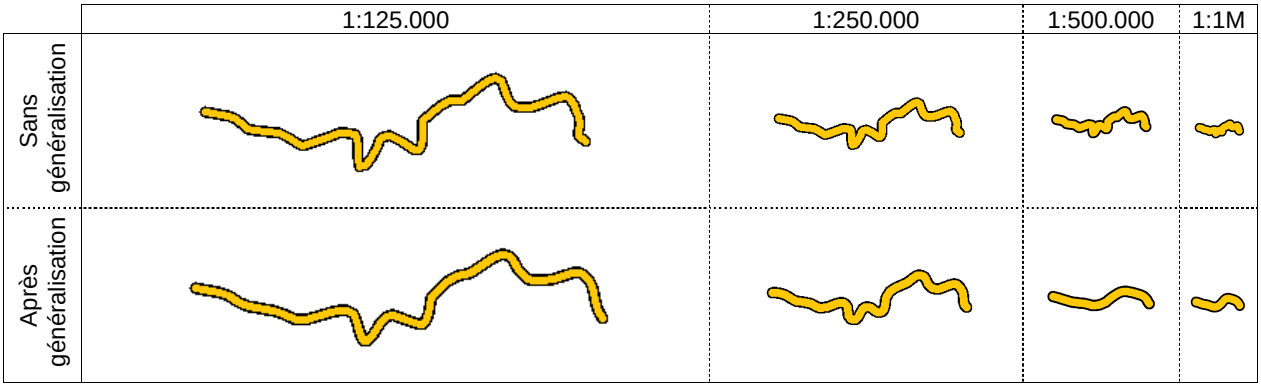

1:125.000
1:250.000
1:500.000
1:1M
Sans généralisation
Après généralisation

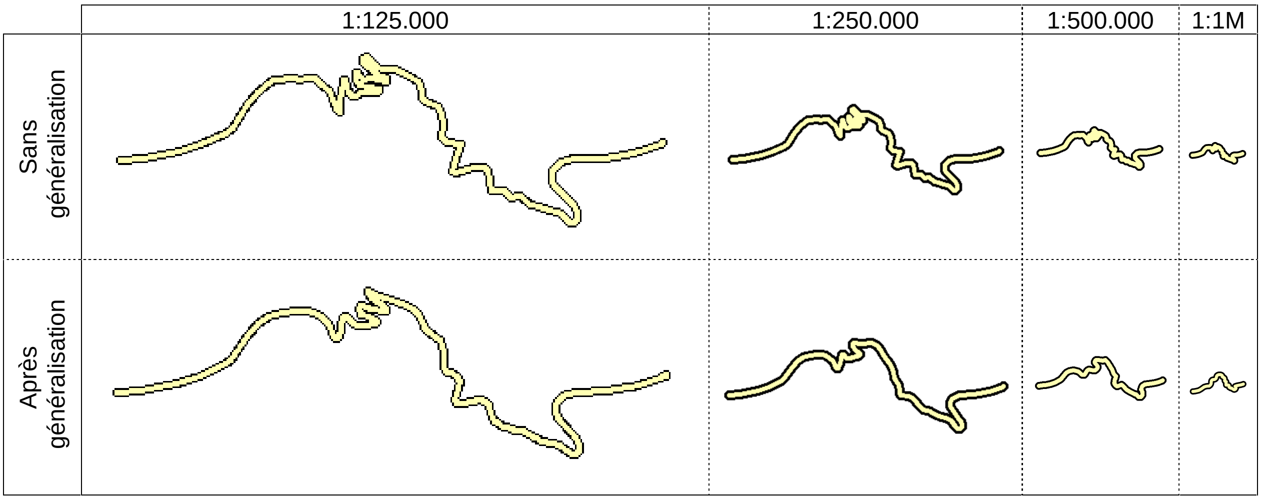

1:125.000
1:250.000
1:500.000
1:1M
Sans généralisation
Après généralisation

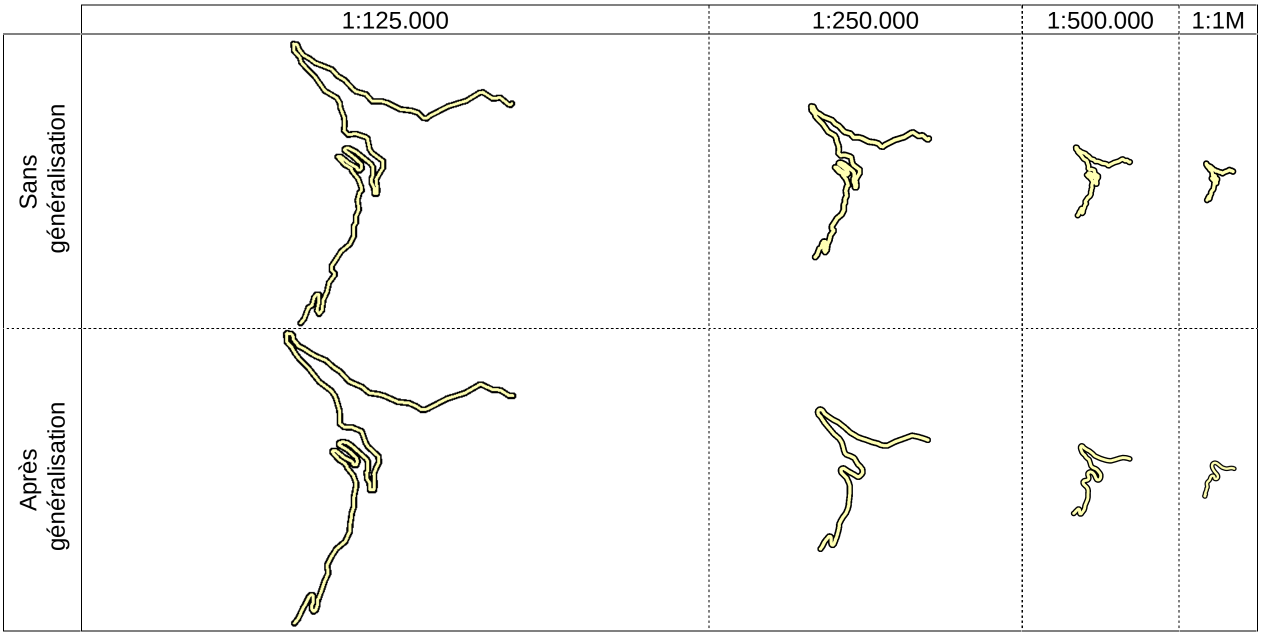

1:125.000
1:250.000
1:500.000
1:1M
Sans généralisation
Après généralisation

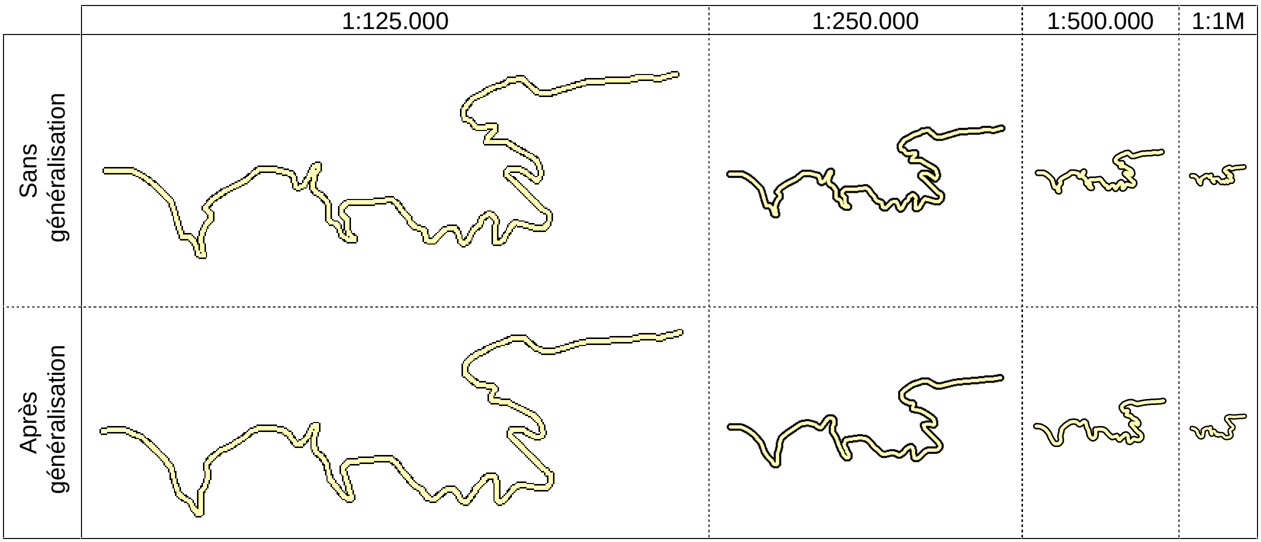

1:125.000
1:250.000
1:500.000
1:1M
Sans généralisation
Après généralisation

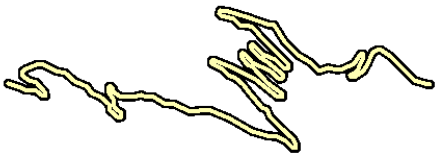
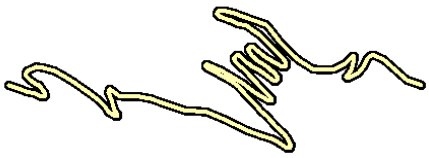

1:125.000
1:250.000
1:500.000
1:1M
Sans généralisation
Après généralisation

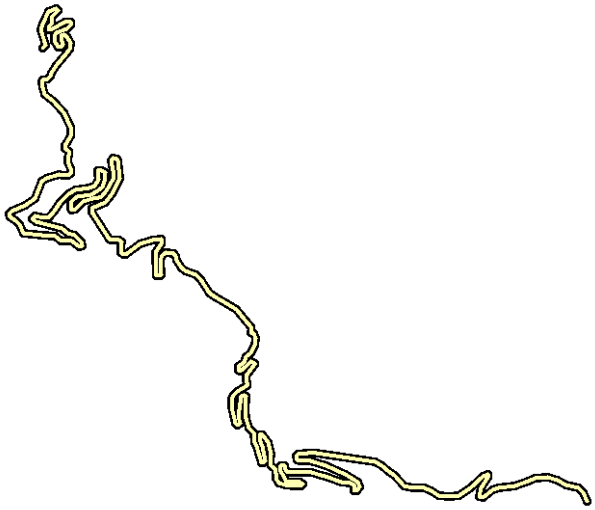
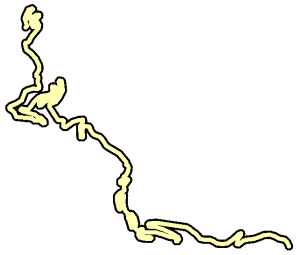
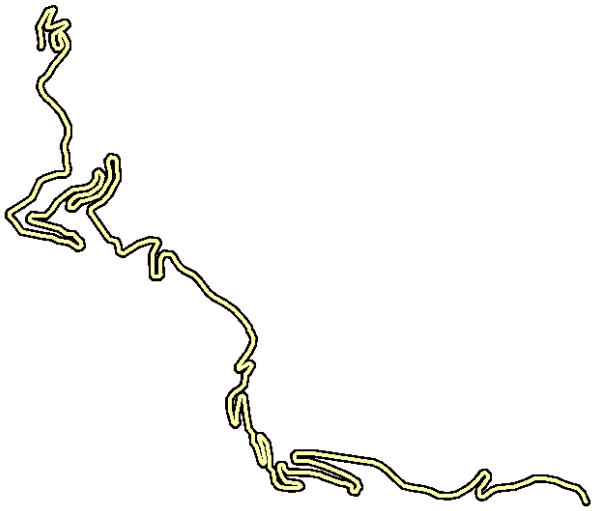
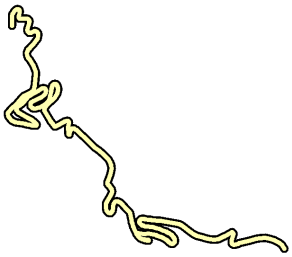
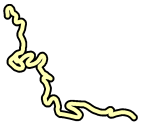

1:125.000
1:250.000
1:500.000
1:1M
Sans généralisation
Après généralisation

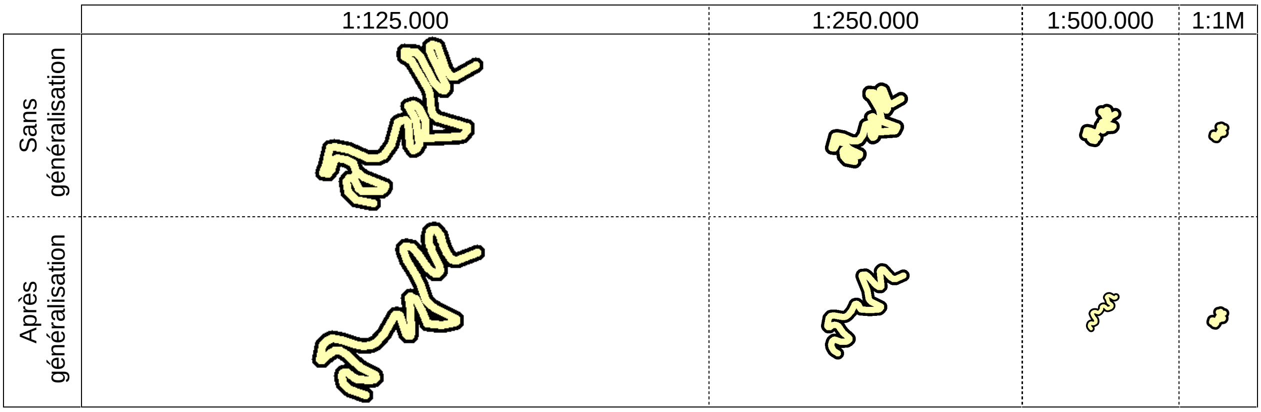

1:125.000
1:250.000
1:500.000
1:1M
Sans généralisation
Après généralisation

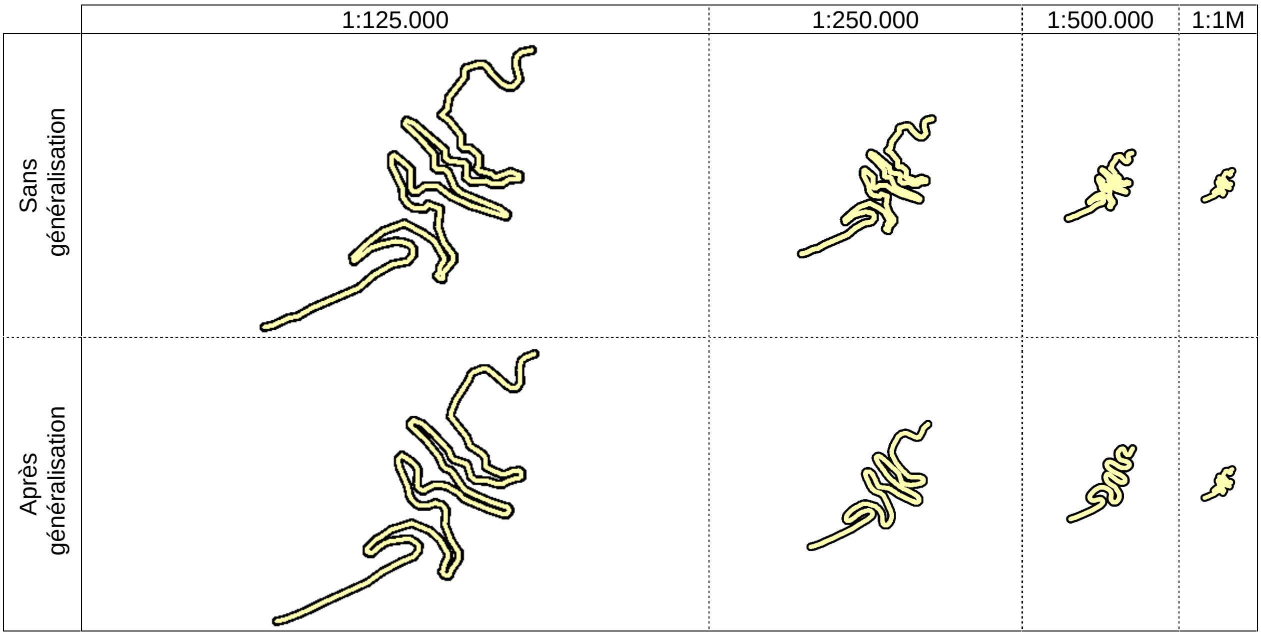

1:125.000
1:250.000
1:500.000
1:1M
Sans généralisation
Après généralisation

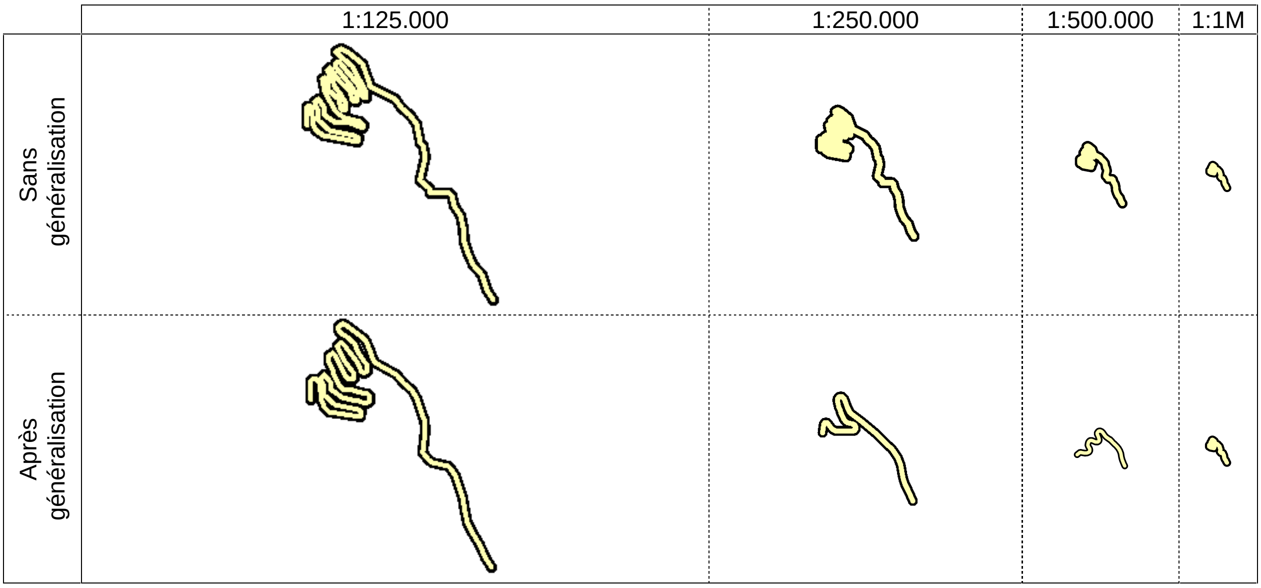

1:125.000
1:250.000
1:500.000
1:1M
Sans généralisation
Après généralisation

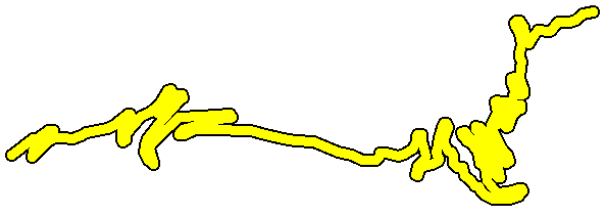
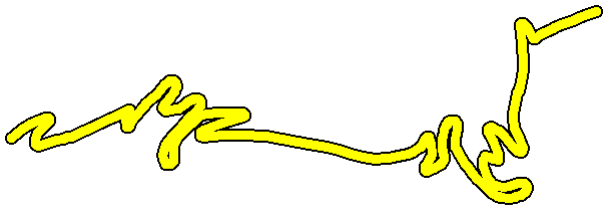
	1:125.000	1:250.000	1:500.000	1:1M
Sans généralisation				
Après généralisation				

	1:125.000	1:250.000	1:500.000	1:1M
Sans généralisation				
Après généralisation				


1:125.000
1:250.000
1:500.000
1:1M
Sans généralisation
Après généralisation


1:125.000
1:250.000
1:500.000
1:1M
Sans généralisation
Après généralisation


1:125.000
1:250.000
1:500.000
1:1M
Sans généralisation
Après généralisation

	1:125.000	1:250.000	1:500.000	1:1M
Sans généralisation				
Après généralisation				

GLOSSAIRE

◆ Ce glossaire décrit l'acception du vocabulaire faite dans ce mémoire, afin de lever les éventuelles ambiguïtés de langage.

◆ Les termes suivants, particulièrement source de confusion ou de contresens, méritent une attention particulière : BDG, échelle d'une BD, généralisation, sémantique, validation, biais, erreur.

A

Abstraction : "Action de l'esprit considérant à part un élément (qualité ou relation) d'une représentation ou d'une notion, en portant spécialement l'attention sur lui, en négligeant les autres. Résultat de cette action." [Lalande 97]

Acquisition des connaissances : Etude des outils et méthodologies pour faciliter le transfert des connaissances de l'homme vers la machine.

Apprentissage : Ensemble des changements dans un système qui lui permettent d'effectuer la même tache ou des taches similaires de manière plus efficace ou plus efficiente au cours du temps [Simon, 83]. Un système peut changer de deux manières différentes : 1/ en acquérant de nouvelles connaissances à partir de sources externes (c'est le cas de l'*apprentissage supervisé*) ou 2/ en réorganisant de lui-même ses connaissances pour les rendre plus efficientes [Shavlik et Dietterich 90, p.1].

Apprentissage de concept : *Apprentissage supervisé* où la classe des exemples ne prend que deux valeurs : "faire partie du concept à apprendre" et "ne pas faire partie du concept à apprendre" ; les exemples sont alors appelés respectivement positifs ou négatifs.

Apprentissage supervisé : Une partie des problèmes d'apprentissage où, étant donné un ensemble d'*exemples* de la forme (x,y), le but est de déterminer une fonction de classification (ou *hypothèse* h:x→y) permettant de déterminer la *classe* y en fonction des *observables* x. Ces hypothèses doivent contenir la "structure générale" des exemples afin de pouvoir prévoir la classe de nouveaux exemples inconnus..

Arbre de décision : Langage de représentation des *hypothèses*. Un test sur les *observables* permettant de séparer les *exemples* en plusieurs groupes (en général deux) est affecté à chaque nœud, et une classe est affectée à chaque feuille. Par exemple, les algorithmes d'apprentissage supervisé ID3 et C4.5 [Quinlan 86a ; Quinlan 93] créent des arbres de décision.

Attribut/Valeur (langage) : Langage de représentation des *observables*, où celles-ci sont représentées par un ensemble fixe d'attributs. Lorsque les attributs sont binaires le langage est dit d'ordre 0, lorsque les attributs peuvent prendre des valeurs symboliques ou numériques le langage est dit d'ordre 0+.

B

Base de données cartographique (BDC) : Base de données où sont représentés (et localisés) les objets graphiques d'une carte (des points, des lignes, des surfaces, du texte) avec leur symbolisation (couleur, largeur…). Une BDC est une base de données prête à afficher ou imprimer de manière lisible à une certaine échelle. Appelée aussi DCM (Digital Cartographic Model) [Brassel et Weibel 88].

Base de données géographique (BDG) : Base de données où sont représentés (et localisés) des objets géographiques (maisons, rivières…). On distingue généralement dans les BDG la géométrie des objets de leur *sémantique*. Appelée aussi DLM (Digital Landscape Model) [Brassel et Weibel 88]. A titre d'exemple de la différence entre les bases de données géographiques et cartographiques, dans une BDG les toponymes sont représentés comme des attributs d'objets (ville, rivière…) alors que dans une BDC ils sont représentés comme des objets graphiques localisés sur la carte, éventuellement sans lien explicite avec les objets qu'ils nomment. De même, les BDC ont une *échelle* bien définie, ce qui n'est pas le cas pour les BDG. Dans ce mémoire nous faisons une nette distinction entre BDG et BDC, cependant, il est à noter que d'une part, certaines bases de données sont un mélange de BDG et BDC et, d'autre part, dans la littérature le terme BDG désigne parfois sans distinction les BDG et les BDC.

Biais (d'une mesure) : erreur systématique d'une mesure, ou plus précisément écart entre l'espérance d'une série de mesures et l'espérance théorique.

Biais d'apprentissage (ou Biais d'induction) : Ensemble des contraintes d'un problème d'apprentissage pour guider la recherche d'*hypothèses*. Cette notion englobe à la fois les *biais de représentation*, les *biais de restriction*, les de *biais de préférence*. Ce terme est utilisé quand on voit l'apprentissage comme un problème de recherche dans l'espace des hypothèses [Mitchell 82]. Mitchell [1997, p.43] définit aussi ce biais comme l'ensemble des suppositions qui, associé aux *exemples* d'apprentissage, permet de justifier par *déduction* les classifications assignées par un algorithme d'apprentissage supervisé à de nouveaux exemples non classés. A titre d'exemple, le biais d'induction de l'algorithme ID3 [Quinlan 86a] peut être approché par " le concept peut être représenté par un *arbre de décision*, les petits arbres sont préférés aux grands, les arbres qui font de forts gains d'information près de la racine de l'arbre sont préférés aux autres ". NB : si les *biais de mesure* sont sources d'erreur, les *biais d'apprentissage* sont nécessaires et ne sont pas des erreurs.

Biais de préférence (ou Biais de recherche) : Préférences (ou ordre partiel) dans l'espace des *hypothèses*. Ces biais permettent de choisir entre deux *hypothèses* possibles de l'espace défini par les biais de représentation et de restriction. Le principe de *longueur de description minimale* (ou MDL) est un exemple de biais de préférence.

Biais de représentation (ou Biais de Langage) : Formalisme de représentation utilisé pour décrire les *exemples* et les *hypothèses*. Les *arbres de décision*, les *règles de décision*, les *règles de production* ou les *réseaux de neurones* sont des exemples de biais de représentation.

Biais de restriction : Contraintes sur le langage des *hypothèses*, ce biais restreint l'espace des hypothèses défini par le *biais de représentation*. Ce biais est parfois inclus dans le terme *Biais de représentation.* Par exemple, Les *arbres de décision* avec au plus k nœuds sont des exemples de biais de restriction des arbres de décision.

C

Cohérence (d'une hypothèse) : Une hypothèse H est dite cohérente avec un jeu d'exemples d'apprentissage S, si tous les exemples de S sont classés par H comme ils le sont dans S.

Cohérence (d'un jeu d'exemples) : Un jeu d'*exemples d'apprentissage* est dit cohérent s'il n'y existe pas deux exemples ayant des observables identiques et une classification différente. Un jeu d'exemples d'apprentissage est dit cohérent par rapport à un biais d'apprentissage, s'il existe au moins une hypothèse dans ce biais cohérente avec ce jeu.

D

Déduction : "Opération par laquelle on conclut rigoureusement, d'une ou de plusieurs propositions prises pour prémisses, à une proposition qui en est la conséquence nécessaire, en vertu des règles logiques" [Lalande 97].

E

Echelle d'une carte : Rapport de longueur entre les objets représentés sur la carte et les objets géographiques. NB : l'échelle d'une carte au 1/100.000 (1cm=1km) est " 1/100.000 " et non " 100.000 ", de ce fait, dériver une carte au 1/250.000 à partir de données au 1/100.000 revient à réduire l'échelle, et non à l'augmenter, contresens souvent rencontré dans le langage courant. Généralement on nomme " grandes " les échelles supérieures au 1/15.000 (comme les plans de ville) ; " moyennes " les échelles entre 1/25.000 et 1/250.000 (comme les cartes de randonnées ou d'un département en France), et " petites " les échelles inférieures au 1/500.000 (comme une carte de France ou du monde).

Echelle d'une base de données géographique : Théoriquement, la notion de représentation graphique étant absente des BDG, celles-ci n'ont pas d'échelle, mais peuvent être qualifiées par leurs résolutions (métriques et thématiques) et leur précision. Cependant en pratique, pour faciliter la compréhension du contenu d'une BDG, on la qualifie souvent par une échelle, celle des cartes qui peuvent en être dérivées relativement directement. Par conséquent, cette notion d'échelle de BDG est imprécise et doit être considérée de manière très relative.

Erreur apparente (d'une hypothèse) : Etant donné un jeu d'*exemples d'apprentissage* S, un algorithme d'apprentissage L, et une *hypothèse* H apprise par L sur S, l'erreur apparente de H est le pourcentage d'exemples de S mal classés par H, par rapport à leur classification dans S. L'erreur apparente ne permet pas d'évaluer la qualité des hypothèses apprises : elle est généralement largement inférieur à l'*erreur réelle* de H.

Erreur réelle (d'une hypothèse) : Etant donné l'ensemble D des exemples possibles (en général inconnu) et une *hypothèse* H, l'erreur réelle de H sur D est la probabilité de mal classifier un exemple de D par H. autrement dit, l'erreur réelle permet de mesurer le pouvoir de prédiction d'une hypothèse. Cette erreur est en général inconnue est approchée par l'*erreur estimée*.

Erreur estimée (d'une hypothèse) : Approximation de l'erreur réelle d'une hypothèse à partir d'un sous-ensemble de l'ensemble des exemples possibles. Cette erreur est souvent évaluée par le taux d'erreur de classification sur un jeu d'*exemples tests* différents du jeu d'*exemples d'apprentissage*, ou par *validation croisée*.

Exemple : En apprentissage un exemple est constitué d'*observables* et d'une *classe*. On désigne sous ce terme à la fois les *exemples d'apprentissage* (classés par un expert) et les objets inconnus lors de l'apprentissage (classés ou non par un expert) sur lesquels s'appliquent une *hypothèse* (e.g. on dit "classer un nouvel exemple par une hypothèse apprise").

Exemple naturel : Un exemple indépendamment de toute représentation. [Zucker 96]

Exemple d'apprentissage : *Exemple* ayant servi à construire une hypothèse. A opposer à l'*exemple test* qui sert à évaluer une hypothèse.

Exemple test : *Exemple* que l'on utilise pour évaluer une *hypothèse* H. C'est à dire que l'on classe l'exemple test par H et que l'on compare cette classification avec celle a priori de l'exemple test. Pour que cet exemple serve réellement à évaluer le pouvoir de prédiction de H il ne doit pas faire partie des *exemples d'apprentissage* qui ont servi à créer H.

Extension (d'un concept): Ensemble des entités constituant un concept. Voir *Intension*.

G

Généralisation (sens général et sens en IA) : "Opération par laquelle, reconnaissant des caractères communs entre plusieurs objets singuliers, on réunit ceux-ci sous un concept unique dont ces caractères forment la compréhension" [Lalande 97]. L'*apprentissage supervisé* généralise des *exemples* pour créer des *hypothèses*.

Généralisation (sens en cartographie) : Traditionnellement création d'une carte à partir d'une autre carte à une *échelle* plus grande. Ce terme désigne aussi le passage d'une *base de données géographique* à une *base de données cartographique* à un niveau de détail plus faible. En cas de confusion avec l'acception du terme en IA nous utilisons dans ce mémoire le terme " généralisation cartographique ".

Géométrie (en SIG) : Dans une base de données géographique, partie de l'information stockée sur les objets géographiques relative à leur position et à leur forme, à opposer à *sémantique*. Cette géométrie est en général représentée sous forme *vecteur* ou *rasteur*.

H

Hypothèse : En *apprentissage supervisé*, fonction reliant l'espace des *observables* à celui des *classes*. Ces hypothèses peuvent être de la forme de *réseaux de neurones*, de *règles*, d'*arbres de décision*, etc.

I

Induction : "Opération mentale qui consiste à remonter d'un certain nombre de propositions données, généralement singulières ou spéciales, que nous appellerons inductrices, à une proposition ou à un petit nombre de propositions plus générales, appelées induites, telles qu'elles impliquent toutes les propositions inductrices" [Lalande 97]

Intension : Décrire un concept en intension c'est donner des propriétés nécessaires et suffisantes sur des entités pour qu'elles appartiennent à ce concept.

L

Langage de la logique d'ordre 0. Langages manipulables par la logique des *propositions*. Par exemple la règle " $(A \wedge B) \Rightarrow C$ " est exprimée dans un langage d'ordre 0.

Langage de la logique d'ordre 1. Langages manipulables par la logique des *prédicats*. Par exemple la règle " $(A=B) \Rightarrow C$ " est exprimée dans un langage d'ordre 1, ce qui se voit plus immédiatement si on la réécrit sous la forme " $(\exists x \mid A=x \wedge B=x) \Rightarrow C$ ".

Logique des propositions ou **Logique d'ordre 0** : logique des formules composées des connecteurs " et, ou, donc, équivalent à, non " et de variables booléennes. Si ces variables peuvent être valuées (par ex. A=5, ou A=bleu) on parle de logique des propositions valuées ou d'ordre 0+.

Logique des prédicats ou **Logique d'ordre 1** : logique des formules composées des connecteurs " et, ou, donc, équivalent à, non ", des quantificateurs " pour tout, il existe ", et de variables.

Longueur de Description Minimale (principe de). Biais particulier de préférence sur les hypothèse : entre deux hypothèses apprises sur un jeu d'exemples, on préfère celle qui minimise la somme des longueurs de description de cette hypothèse et des exceptions à cette hypothèse. Ceci revient donc à préférer les hypothèses qui permettent de définir de la manière la plus compacte l'ensemble des exemples.

O

Observables : Partie descriptive d'un *exemple*, c'est à dire l'exemple sans sa classe.

Overfitting : On dit qu'une hypothèse couvre trop (overfit) des exemples quand elle est trop spécifique aux exemples, au détriment de sa capacité généralisatrice (capacité à bien classer un nouvel exemple quelconque). Ce

problème de l'overfitting est au cœur de l'apprentissage automatique. L'apprentissage par cœur est un extrême de l'overfitting.

R

Rasteur (représentation) : représentation de la *géométrie* des objets géographiques utilisant une partition de l'espace (en général une grille régulière): la géométrie d'un objet est alors stockée sous la forme de l'ensemble des cellules qu'il traverse. Appelé aussi mode maillé. Voir *Vecteur.*

Règles de décision : ensemble de règles ordonnées. Soit par exemple une base de règles de décision avec trois règles "A→ B", "C→ D" et "∅→E", on peut alors les lire comme "Si A alors B, sinon si C alors D, sinon E". L'algorithme RIPPER [Cohen 95] apprend des règles de décision.

Règles de production : ensemble de règles non ordonnées, sous la forme "si condition alors conséquent". Si plusieurs règles sont en conflit c'est alors au moteur d'inférence de résoudre ce conflit. L'algorithme CHARADE [Ganascia 87] apprend des règles de production.

Réseaux de neurones artificiels (ou plus simplement **Réseaux de Neurones**) : Paradigme d'apprentissage inspiré par la structure biologique des neurones dans le cerveau.

S

Sciences de l'Information Géographique (SIG) : Sciences qui traitent des données acquises sur des objets ou phénomènes localisés à la surface de la Terre, naturels ou liés à l'activité humaine. Leurs objectifs sont, d'une part, la modélisation, le traitement, et la restitution de l'information sous forme de cartes ou de bases de données, et d'autre part, la recherche de processus d'analyse de cette information pour l'aide à la décision. Voir *Système d'Information Géographique.*

Sémantique *(sens général)* : Relatif à la signification, au sens.

Sémantique *(en SIG)* : Dans une base de données géographique, partie de l'information stockée sur les objets géographiques qui n'est pas liée à leur *géométrie* ni à leur *topologie*. Par exemple la fonction d'un bâtiment (mairie, industrie…) est contenue dans sa sémantique alors que sa localisation et sa forme sont contenues dans sa géométrie. Cette restriction de définition par rapport au sens général peut faire penser à tort que la géométrie d'un objet ne contient pas de sens [Ruas 99]. Dans les premiers et plus nombreux modèles d'implémentation des *BDG*, la sémantique correspond aux attributs d'un objet géographique, et la géométrie est un lien vers des objets géométriques (points, lignes, surfaces) ; dans d'autres modèles la géométrie est aussi vue comme un attribut de l'objet.

Système d'Information Géographique (SIG) : logiciel dédié à l'intégration, la création, la transformation, l'analyse et la représentation de données géoréférencées (localisées sur la terre). En géographie ce terme prend généralement un sens plus large et inclut également les données, ainsi que les moyens humains et matériels mis en œuvre pour gérer ces données. Dans ce mémoire nous nous limitons au sens de logiciel. NB : le sigle SIG est utilisé à la fois pour *Sciences de l'Information Géographique* et Système d'Information Géographique, en cas de confusion possible nous utilisons le terme de "logiciel SIG" dans ce dernier cas.

V

Validation croisée (d'une hypothèse) : Méthode d'estimation de l'*erreur réelle* d'une hypothèse H à partir du jeu d'exemples d'apprentissage S qui a permis de créer H. Cette méthode est utilisée lorsque le nombre d'exemples disponibles n'est pas assez grand pour pouvoir le séparer en un jeu d'exemples tests et un jeu d'exemples d'apprentissage destiné à créer H. Elle consiste à séparer au hasard S en k parties, puis apprendre, avec le même *biais d'induction* que celui qui a permis de créer H, une hypothèse H_i sur (k-1) parties de S, et ensuite évaluer le taux d'erreur de classification de H_i sur la partie restante. Ce processus est réitéré k fois en faisant permuter la partie restante. Le taux d'erreur évalué par validation croisée est la moyenne des taux d'erreurs des k tests.

Vecteur (représentation) : représentation de la géométrie des objets géographiques à partir de primitives points, lignes, surface ou volumes. La géométrie d'un objet est alors stockée sous la forme de listes de coordonnées définissant le contour ou le squelette de l'objet. Voir *rasteur.*

■

REFERENCES

Abbas I. et Hottier P. 1993. Contrôle du tracé planimétrique d'une carte – Contrôle ponctuel et contrôle linéaire. *Bulletin d'information de l'IGN n.61* : *Bilan de la Recherche 92*, pp.30-36.

Affholder J.-G. 1993. Road modelling for generalization. *NCGIA research initiative 8, Specialist Meeting on Formalizing Cartographic Knowledge*, Buffalo, Etats-Unis, pp.23-36.

AGENT 1999. *Selection of Basic Algorithms*. Report DD2 of the AGENT project, ESPRIT/LTR/24939.

Aha D.W. 1992. Generalizing from case studies: A case study. *Proc. of the 9th International Machine Learning Conference*, pp.1-10, Aberdeen, Scotland, Morgan Kaufmann.

Ahn J. et Freeman H. 1983. A program for automatic name placement. *Proc. of the 6th AutoCarto conference*, Ottawa, Vol.2, pp.444-453.

Ai T., Guo R. et Liu Y. 2000. A Binary Tree Representation of Curve Hierarchical Structure Based on Gestalt Principles. *Proc. of the 9th Spatial Data Handling Symposium*, Pékin, sec.2a, pp.30-43.

Ali K.M. et Pazzani M.J. 1996. Error reduction through learning multiple descriptions. *Machine Learning*, 24(3), pp.173-202.

Andrienko G. et Andrienko N. 1999. Knowledge Engineering for Automated Map Design in DESCARTES. *Proc. of the 7th ACM International Symposium on Geographic Information Systems*, Kansas City, pp.66-72.

Armstrong M.P. 1991. Knowledge Classification and organisation. *Map Generalization : Making Rules for Knowledge Representation*, Buttenfield et McMaster (eds), Longman Scientific & Technical, Harlow, Essex, pp.86-102.

Badaud J., Witkin A., Baudin M. et Duda R. 1986. Uniqueness of the Gaussian Kernel for Scale-Space Filtering. *IEEE Transaction on Pattern Analysis and Image Processing*, Vol.8, n.1, pp.26-33.

Bader M. et Barrault M. 2000. Improving Snakes for Linear Features Displacement in Cartographic Generalisation. *Proc. of 4th International Conference on GeoComputation* (CD-ROM – 10 pages)

Bader M. et Weibel R. 1997. Detecting and Resolving Size and Proximity Conflicts in the Generalisation of Polygonal Maps. *Proc. of 18h International Cartographic Conference*, Stockholm, Vol.3, pp.1525-1532.

Bard S. 2000. *Révision d'une base de connaissances, Application à la généralisation cartographique.* Rapport de stage de DESS Méthodes Quantitatives en Gestion et Aménagement de l'Espace, université de Metz.

Bard S. et Mustière S. 2001. Revision of cartographic generalisation rule bases funded on interactive alteration analysis. *Proc. of 20th International Cartographic Conference*, Hong-Kong. A paraître.

Barrault M. 1998. *Le placement cartographique des écritures: résolution d'un problème à forte combinatoire et présentant un grand nombre de contraintes variées.* Thèse de doctorat, université de Marne-la-Vallée.

Barrault M., Bader M. et Weibel R. 2000. Preserving Topology during Conflict Removal between Symbolised Roads in Cartographic Generalisation. *Proc. of GIScience 2000*, Savannah, Georgie, pp.117-119.

Beard K. 1991. Constraints on rule formation. *Map Generalization : Making Rules for Knowledge Representation*, Buttenfield et McMaster (eds), Longman Scientific & Technical, Harlow, Essex, pp.32-58.

Bel Hadj Ali A. et Vauglin F. 1999. Geometric Matching of Polygons in GISs and assessment of Geometrical Quality of Polygons. *Proc. of the International Symposium on Spatial Data Quality*, Hong-Kong, pp.33-43.

Bergadano F. et Giordana A. 1988. A knowledge intensive approach to concept induction. *Proc. of 5th International Conference on Machine Learning*, Ann Arbor, MI, Morgan Kaufmann, pp.305-317.

Bertolotto M. et Egenhofer M.J. 1999. Progressive Vector Transmission. *Proc. of 7th ACM International Symposium on Geographic Information Systems*, Kansas City, pp.152-157.

Bertin J. 1967. *Sémiologie Graphique.* Gauthier-Villars, Paris.

Bisson G. 1994. From Inductive Learning to Knowledge Acquisition through Explanations. *Proc. of ECAI'94 workshop on the integration of Machine Learning and Knowledge Acquisition*, Amsterdam, pp.1-11.

Blum A. et Mitchell T. 1998. Combining labeled and unlabeled data with co-training. *Proc. of 11th Annual Convention on Computational Learning Theory*, pp.92-100.

Board C. 1967. Maps as models. *Models in Geography*, Chorley et Haggett (eds), Methuen. pp.671-725.

Boffet A. 2000. Creating Urban Information for Cartogrpahic Generalization. *Proc. of the 9th Spatial Data Handling Symposium*, Pékin, sec.3b, pp.4-17.

Brassel K. et Weibel R. 1988. A review and conceptual framework of automated map generalization. *International Journal of Geographical Information Systems*. Vol. 2, no. 3, pp.229-244.

Brazdil P. et Henery R. 1994. Chapter 10, Analysis of Results. *Machine Learning, neural and statistical classification*, Michie, Spiegelhalter et Taylor (eds), Ellis Horwood, pp.175-212.

Breiman L. 1996. Bagging predictors. *Machine Learning*, 24(2), pp.123-140.

Brewer C.A. 1994. Color Use Guidelines For Mapping and Visualization. *Visualization in modern cartography*, MacEachren et Taylor (eds), Elsevier Science (Pergamon), pp.123-147.

Brophy M. 1973. An Automated Methodology for Linear Generalization in Thematic Cartography. *Proc. of American Congress of Surveying and Mapping*, pp 300-314.

Brown A. et Van Elzaker C.P.J.M. 1993. The Use of Colour in the Cartographic Representation of Information Quality Generated by a GIS. *Proc. of 16^{h} International Cartographic Conference*, Cologne, Allemagne, Vol.2, pp.707-720.

Buttenfield B. 1984. *Line Structure in Graphic and Geographic Space*. Ph.D. thesis, Dept. of Geography, University of Washington.

Buttenfield B. 1991. A rule for describing line feature geometry. *Map Generalization : Making Rules for Knowledge Representation*, Buttenfield et McMaster (eds), Longman Scientific & Technical, Harlow, Essex. pp.150-171.

Buttenfield B. 1999. Sharing Vector Geospatial Data on the Internet. *Proc. of 19^{h} International Cartographic Conference*, Ottawa, Vol.3, pp.581-590.

Buttenfield B., Weber C. et Jelinski D. 1995. A Case Study for Hypermedia Cartography : Radial Growth in Trembling Aspen at Waterton Lakes National Park. *Proc. of ACSM/ASPRS annual convention, 12th Autocarto*, Charlotte, Vol. 4, pp.32-40.

Cestnik B., Kononenko I. et Bratko I. 1987. ASSISTANT-86: A knowledge-elicitation tool for sophisticated users. *Progress on Machine Learning*, Bratko et Lavrac (eds), Bled, Yougoslavie: Sigma Press.

Chan P.K. et Stolfo S.J. 1997. On the Accuracy of Meta-Learning for Scalable Data Mining. *Journal of Intelligent Information Systems*, 8(1), pp.5-28.

Cherkauer K.J. 1996. Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks. *Working notes of the AAAI workshop on Integrating Multiple Learned Models*, pp.15-21.

Chevaleyre Y. et Zucker J.-D. 2000. Noise Tolerant Rule Induction for Multi-Instance Data. *ICML workshop on Attribute Value and Relational Learning : Crossing the boundaries*, Stanford, Etats-Unis.

Chrisman N.R. 1982. A theory of cartogrpahic error and its measurement in digital data bases. *Proc. of ACSM ASPR annual convention, 5th AutoCarto*, pp.159-168.

Chrisman N.R. 1991. Constitutional and Societal Components of Cartographic Research. *Advances in Cartography*, Müller (ed), Elsever Applied Science, pp.231-242.

Clancey W. 1983. The Epistemology of a Rule Based Expert System – A framework for Explanation. *Artificial Intelligence journal*, 20(3), pp.215-251.

Clancey W. 1985. *Heuristic Classification*. Rapport de l'université de Stanford, STAN-CS-85-1066.

Clausen H. et Mark D.M. 1991. Vehicle Navigation Systems. *Advances in Cartography*, Müller (ed), Elsever Applied Science, pp.161-179.

Clemen R.T. 1989. Combining forcasts: A review and annoted bibliography. *International Journal of Forecasting*, n.5, pp.559-583.

Cohen W. 1995. Fast Effective Rule Induction. *Proc. of 12th International Conference on Machine Learning*, Lake Tahoe, Etats-Unis, Morgan Kaufmann, pp.115-123.

Cuenin R. 1972. *Cartographie Générale.* Eyrolles, Paris.

David J.-M., Krivine J.-P. et Simmons R. 1993. *Second Generation Expert Systems.* Springer Verlag: Berlin.

Davis R., Shrobe H. et Szolovits P. 1993. What is a Knowledge Representation ? *AI Magazine*, 14(1), pp.17-33.

De Berg M., Van Kreveld M. et Schirra S. 1995. A New Approach to Subdivision Simplification. *Proc. of ACSM/ASPRS annual convention, 12th Autocarto*, Charlotte, Vol. 4, pp.79-88.

Pierrot Deseilligny M., Stamon G. et Suen C.Y. 1998. Veinerization : a new shape descriptor for flexible skeletonization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5),pp.505-521.

Devijver P.A. et Dekesel M. 1982. Insert and delete algorithms for maintining dynamic Delaunay triangulations. *Pattern Recognition Letters*, déc. 82, pp.73-77.

Dietterich T.G. 1997. Machine Learning Research: Four Current Directions. *AI Magazine*, 18(4), p.97-136.

Dietterich T.G. et Bakiri G. 1995. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, n.2, pp.263-286.

Dobson M.W. 1985. The future of perceptual cartography. *Cartographica*, 22(2), pp.27-43.

Douglas D.H. et Peucker T.K. 1973. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or it's Caricature. *The Canadian Cartographer journal*, 10(2), pp.112-122.

Duchêne C. 2001. Road generalisation using agents. *Proc. of 9th Annual Conference on GIS Research in United Kingdom*, Glamorgan, à paraître.

Dutton G.H. 1981. Fractal enhancement of cartographic line detail. *The American Cartographer*, Vol.8, n.1, pp.23-40.

Eastman J.R. 1985. Cognitive Models and Cartographic Design Research. *Cartographic Journal*, 22(2), pp.95-101.

Eastman J.R. 1987. Graphic Syntax and Expert Systems for Map Decision. *Proc. of ASPRS - ACSM Annual Convention*, Baltimore, pp.87-96.

Edwardes A.J. and Regnauld N. 2000. Preserving the pattern of density in urban network simplification**.** *Proc. of the 1st International Conference on Geographic Information Science*, Savannah, Georgia, pp.104-107.

Feigenbaum E.A. 1981. Expert Systems in the 1980s. *State of the art report on machine intelligence*, Bond (ed.). Maidenhead: Pergamon-Infotech.

Forrest D. 1995. Don't break the Rules or Helping Non-Cartographers to Design Maps: an Application for Cartographic Expert Systems. *Proc. of 17th International Cartographic Conference*, Barcelone, vol.1, pp.565-569.

Forrest D. 1999. Developing Rules for Map Design: A Functional Specification for a Cartographic-Design Epert System. *Cartographica*, vol.26, n.3, pp.31-52.

Feund Y. et Schapire R.E. 1996. Experiments with a new boosting algorithm. *Proc. of the 13th International Conference on Machine Learning*, pp.148-156.

Fritsch E. 1997. *Représentations de la Géométrie et des Contraintes Cartographiques pour la Généralisation du Linéaire Routier*. Thèse de doctorat, université de Marne-la-Vallée, déc. 97.

Furnkranz J. et Widmer G. 1994. Incremental reduced error pruning. *Proc. of the 11th International Conference on Machine Learning*, pp.70-77.

Gama J. et Brazdil P. 1995. Characterization of Classification Algorithms. *Proc. of 7th Portuguese Conference on Artificial Intelligence*, Pintro-Ferreira et Mamede (eds), Springer-Verlag, pp.189-200.

Ganascia J.-G. 1987. *AGAPE et CHARADE : deux techniques d'apprentissage symbolique appliquées à la construction de bases de connaissances.* Thèse d'état, université Paris-Sud, Orsay.

Ganascia J.-G. 1991. Deriving the Learning Bias from Rule Properties. *Machine Intelligence* (12) Clarendon Press, pp.151-167..

Ganascia J.-G., Thomas J. et Laublet P. 1993. Integrating Models of Knowledge and Machine Learning. *Proc. of European Conference on Machine Learning*, Vienne, Autriche, pp.396-401

Ginsberg M. 1997. *Essentials of Artificial Intelligence*. Morgan Kaufmann : San Mateo

Giordana A. et Saitta L. 1990. Abstraction: a general framework for learning. *Working notes of the AAAI workshop on Automated Generation of Approximations and Abstraction*, Boston, MA.

Giordana A., Saitta L., Sebag L. et Botta M. 2000. Analysing relational learning in the phase transition framework. *Proc. of 17th International Conference on Machine Learning.*

Goodchild M.F. 1977. Statistical aspects of the polygon overlay problem. *Harvard papers on GIS, vol.6*, Addison-Wesley, MA, pp.1-20.

Hangouët J.-F. 1998. *Approches et Méthodes pour l'Automatisation de la Généralisation Cartographique ; Application en bord de ville.* Thèse de doctorat, université de Marne-la-Vallée.

Harrie L. 1998. *Generalisation methods for propagating updates between geographic data sets*. Thèse de Licence, université de Lund, Suède.

Harrie L. et Sarjakoski T. 2000. Generalisation of Vector Data Sets by Simultaneous Least Squares Adjustement. *International Archives of Photogrammetry and Remote Sensing*, Vol.XXXIII, Part B4, Amsterdam, pp.348-355.

Haussler D. 1988. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 36, pp.177-221.

Højholt P. 1998. Solving local and global space conflicts in map generalisation using finite element method. *Proc. of 8th International Symposium on Spatial Data Handling*, Vancouver, pp.679-689.

Imhof E. 1975. Positioning Names on Maps. *The American Cartographer*, n.2, pp.128-144.

Ishizaki S. et Lokuge I. 1995. Intelligent Interactive Dynamic Maps. *Proc. of ACSM/ASPRS annual convention, 12th Autocarto*, Charlotte, Vol. 4, pp. 1-48.

Iwasaki Y. 1990. Reasoning with Multiple Abstraction Models. *Working notes of the AAAI workshop on automated Generation of Approximations and Abstraction*, Boston, pp.122-134.

João E.M. 1991. The role of the user in GIS generalization. *Cognitive and Linguistic Aspects of Geographic Space*, Mark et Frank (eds), Kluwer Academic, pp.493-506.

João E.M. 1998. *Causes and Consequences of Map Generalisation*. Taylor and Francis: Londres.

Jones C.B. 1989. Cartographic Name Placement with Prolog. *Proc. of 9th AutoCarto*, Baltiomore, pp.231-240.

Jordan M.I. et Jacobs R.A. 1994. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2), pp.181-214.

Kalousis A. et Hilario M. 2000. A comparison of Inducer Selection via Instance-Based and Boosted Decision Tree Meta-Learning. *Proc. of the 5th International Workshop on Multistrategy Learning*, Guimaraes, pp.233-247.

Keller S. 1994. On the use of case-based reasoning in generalization. *Proc. of 6th International Symposium on Spatial Data Handling*, Edimbourg, Vol. 2, pp. 1118-1132.

Keller S. 1995. Potentials and limitations of artificial intelligence techniques applied to generalization. *GIS and Generalisation, Methodology and Practice*, Müller, Lagrange et Weibel (eds), Taylor & Francis: Londres, pp.135-147.

Kerber R. 1992. ChiMerge: Disretization of Numeric Attributes, *Proc. of the 10th National Conference on Artificial Intelligence* (AAAI), pp.123-128.

Kilpeläinen T. 2000. Knowledge Acquisition for Generalization Rules. *Cartography and Geographic Information Science*, vol.27, n.1, pp.41-50.

Kohavi, R. et M. Sahami 1996. Error-Based and Entropy-Based Discretization of Continuous Features. *Proc. of the 2nd International Conference.on Knowledge Discovery and Data Mining (KDD)*, pp.114-119.

Kolácňý A. 1969. Cartographic information – a fundamental concept and term in modern cartography. *Cartographic Journal*, n.6, pp. 47-49.

Koussoulakou A. et Kraak M.J. 1992. Spatio-Temporal Maps and Cartographic Communication. *The Cartographic Journal*, Vol. 29, pp. 101-108.

Kraak M.-J. et MacEachren A. M. 1994. Visualization of the temporal Component of Spatial Data. *Proc. of 6th International Symposium on Spatial Data Handling*, Edimbourg. Vol.1, pp.391-409.

Krygier J.B. 1994. Sound and Geographic Visualization. *Visualization in modern cartography*, MacEachren et Taylor (eds), Elsevier Science, Pergamon, pp.149-168.

Kyndt J. 1997. *Approche mécanique de la généralisation*. Rapport de stage, DESS de Mathématiques Appliquées, université Pierre et Marie Curie, Paris VI.

Lagrange F. et Landras B. 1999. *Application des réseaux de neurones à l'apprentissage des valeurs paramétriques des algorithmes de généralisation cartographique*. Rapport de stage, Ecole Navale, Lanvéoc-Poulmic.

Lagrange F., Landras B., Mustière S. 2000. Machine Learning Techniques for Determining Parameters of Cartographic Generalisation Algorithms. *International Archive of Photogrammetry and Remote Sensing (Proc. of ISPRS'2000)*, vol.33, partie B4, pp.718-725.

Lalande A. 1997. *Vocabulaire technique et critique de la philosophie.* 4ème édition (1ère éd. 1926) Quadrige / Presses Universitaires de France, Paris.

Lamy S., Ruas A., Demazeau Y., Jackson M., Mackaness W.A. et Weibel R. 1999. The Application of Agents in Automated Map Generalisation. *Proc. of 19th International Cartographic Conference*, Ottawa.,Vol 2, pp. 1225-1234.

Lecordix F., Plazanet C. et Lagrange J.-P. 1997. A Platform for Research in Generalization: Application to Caricature. *GeoInformatica*, 1997, 1(2), pp. 161-182.

Lemarié C. et Raynal L., 1996. Geographic Data Matching: First Investigations for a Generic Tool. *Proc. of GIS/LIS'96*, Denver, Colorado, pp.405-420.

Le Roux B. 1994. *Eléments d'une approche constructive de la modélisation et de la réutilisation en acquisition des connaissances.* Thèse de doctorat de l'université Paris 6.

Le Roux B., O'Hara K., Outtandy S., Shadbolt N., Laublet P. et Motta E. 1993. *The Vital Library of Knowledge Modelling*, VITAL Deliverable 2.1.5.

Lowe D.G. 1988. Organization of Smooth Image Curve at Multiple Scales. *Proc. of 2nd International Conference on Computer Vision*, pp.558-567.

MacEachren A.M. et Taylor F. 1994. *Visualization in modern cartography*. Elsevier Science (Pergamon)

MacEachren A.M. 1995. *How Maps Work. Representation, Visualization and Design.* Guilford Press: New-York.

Martynenko A. et Leontiev V. 1995. The Electronic Maps System: Scientific Basis, Methods and Technology. *Proc. of 17th International Cartographic Conference*, Barcelone, Vol.2, pp. 2881-2885.

McMaster R.B. 1983. *Mathematical measures for the evaluation of simplified lines on maps*. Ph.D thesis, université du Kansas, Etats-Unis.

McMaster R.B. 1989. The Integration of Simplification and Smoothing Algorithms in Line Generalization. *Cartographica*, 26(1), pp.101-121.

McMaster R.B. 1995. Knowledge Acquisition for Cartographic Generalization : experimental methods. *GIS and Generalization*, Müller, Weibel et Lagrange (eds), Taylor and Francis, pp.161-180.

McMaster R.B. et Mark D. 1991. The design of a graphical user interface for knowledge acquisition in cartographic generalization. *Proc. of GIS/LIS conference*, Atlanta, Vol.2, pp.311-320.

McMaster R.B. et Shea K.S. 1988. Cartographic *Generalization in Digital Environnement: A framework for Implementation in a GIS*. *Proc. of GIS/LIS conference*, San Antonio, Texas, pp.240-249.

McMaster R.B. et Shea K.S. 1992. *Generalization in Digital Cartography*. Association of American Geographers, Washington.

Midtbø T. 2000. Visualisation of the Temporal Dimension in Multi-Media Presentations of Spatial Phenomenon. *Proc. of the 9th International Symposium on Spatial Data Handling*, Pékin, sec.1a, pp.3-13.

Michalski R.S. 1980. Pattern recognition as a rule-guided inductive inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4), pp.349-361.

Mitchell T.M. 1980. *The Need for Biases in Learning Generalizations.* Rutgers Computer Science Department Rapport Technique CBM-TR-117, mai 1980. Republié dans *Readings in Machine Learning*, Shavlik et Dietterich (eds), Morgan Kaufmann, 1990, pp. 184-191.

Mitchell T.M. 1982. Generalization as Search. *Artificial Intelligence*, no.18, pp.203-226.

Mitchell T.M. 1997. *Machine Learning*. McGraw-Hill International Editions, Singapour.

Monier P. 1997. *Caractérisation du relief en vue de son traitement numérique. Application à la généralisation de l'orographie.* Thèse de doctorat, université Louis Pasteur, Strasbourg.

Monmonier M. 1991. *How to Lie with Maps.* Chicago: University of Chicago Press.

Mooney R., Shavlik J., Towell G. et Gove A. 1989. An Experimental Comparison of Symbolic and Connectionist Learning Algorithms. *Proc. of the 11th International Joint Conference on Artificial Intelligence*, Detroit, MI, pp.75-780.

Morrison J.L. 1994. The Paradigm Shift in Cartography: The Use of Electronic Technology, Digital Spatial Data, and Future Needs. *Proc. of 6th International Symposium on Spatial Data Handling*, Edimbourg. Vol.1, pp.1-15.

Muggleton S. 1995. Inverse entailment and progol. *New Generation Computing*, Vol.13, pp.245-286.

Müller J.-C. 1991. Generalization of Spatial Databases. *Geographical Information Systems: Principles and Applications*, Maguire, Goodchild et Rhind (eds), Longman Scientific & Technical, Vol.1, pp.457-475.

Müller J.-C. et Mouwes P.J. 1991. Knowledge acquisition and representation for rule based map generalization: An example from the Netherlands. *Proc. of GIS/LIS conference*, Vol.1, pp.58-67.

Mustière S. 1995. *Mesures de la qualité de la généralisation du linéaire.* Rapport de stage, DESS de Cartographie, université Paris I – Sorbonne.

Mustière S. 1998a. An Algorithm for Legibility Evaluation of Symbolised Lines. *Proc. of 4th InterCarto conference*, Barnaul, Russie, pp. 43-47.

Mustière S. 1998b. GALBE: Adaptive Generalisation. The need for an Adaptive Process for Automated Generalisation, an Example on Roads. *Proc. of 1st GIS'PlaNet conference*, Lisbonne, Portugal, CDROM, 6 p.

Mustière S. et Duchêne C. 2001. Comparison of different approaches to combine road generalisation algorithms: GALBE, AGENT and CartoLearn. *4th ICA Workshop on generalisation*, Hong-Kong, à paraître sur le site web de la commission en généralisation de l'ACI : http://www.geo.unizh.ch/ICA-bin/documents.

Mustière S. et Lecordix F. 2000. De nouveaux outils de généralisation numérique. *Bulletin du Comité Français de Cartographie*, n.163, mars 2000, pp.40-44.

Mustière S., Zucker J.-D. et Saitta L. 2000a. Abstraction et Changement de Langage pour Automatiser la Généralisation Cartographique. *Actes du 12ème congrès Francophone de Reconnaissance de Formes et Intelligence Artificielle*, Paris, Vol.1, pp.411-418.

Mustière S., Zucker J.-D. et Saitta L. 2000b. An Abstraction-Based Machine Learning Approach to Cartographic Generalisation. *Proc. of 9th International Symposium on Spatial Data Handling*, Pékin, sec.1a, pp.50-63.

Nédellec C. et Causse K. 1992. Knowledge refinement using knowledge acquisition and machine learning methods. *Proc. of the 6th European Knowledge Acquisition Workshop: Current Developments in Knowledge Acquisition*, pp.171-190.

Nickerson B.G. 1988. Automated Cartographic Generalization for Linear Features. *Cartographica*, 25(3), pp.15-66.

Nickerson B.G. 1991. Knowledge engineering for generalization. *Map Generalization : Making Rules for Knowledge Representation*, Buttenfield et McMaster (eds), Longman Scientific & Technical: Harlow, Essex, pp.40-55.

Ninio J. 1996. *L'empreinte des sens. Perception, mémoire, langage*. 3ème éd. (1ère éd. 1989), Odile Jacob: Paris.

Nyerges T. 1991. Representing Geographic Meaning. *Map Generalization : Making Rules for Knowledge Representation*, Buttenfield et McMaster (eds), Longman Scientific & Technical: Harlow Essex, pp. 59-85.

Peng W. et Tempfli K. 1996. An Object-Oriented Design for Automated Database Generalization. . *Proc. of the 7th International Symposium on Spatial Data Handling*, Delft, Vol ., sec.4b, pp.15-29.

Perkal J. 1958. Proba obiektyunej generalizacji. *Geodezia I Kartografia*, 1958, 7(2), pp.130-142. Traduit en 1966 par R. Jackowski: An Attempt at Objective Generalization, dans *Michigan Inter-University Consortium of Mathematical Geographers*, Discussion Paper no 10, University of Michigan.

Petchenik B.B. 1975. Cognition in Cartography. *Proceedigns of 2nd AutoCarto*, Washington, pp.183-193.

Plazanet C. 1996. *Enrichissement des bases de données géographiques : analyse de la géométrie des objets linéaires pour la généralisation cartographique (application au routes).* Thèse de doctorat, université de Marne-la-Vallée.

Plazanet C., Martini Bigolin N. et Ruas A. 1998. Experiments with Learning Techniques for Spatial Model Enrichment and Line Generalization. *GeoInformatica*, 2(4), dec. 98, pp.315-333.

Popelinsky L. et Brazdil P. 2000. Combining the Principal Components Method with Decision Tree Learning. *Proc. of 5th International Workshop on Multi-Strategy Learning*, Guimaraes, Portugal, 2000.

Pravda J. 1995. Some Developmental Traits Of Theoretical Cartography. *Proc. of 17th International Cartographic Conference*, Barcelone, vol.1, pp. 909-913.

Quinlan J.R. 1986a. Induction of decision Trees. *Machine Learning*, n.1, Kluwer Academic Publishers, pp.81-106.

Quinlan J.R. 1986b. The Effect of Noise on Concept Learning. *Machine Learning – An Artificial Intelligence Approach – Vol II*, Michalski, Carbonell et Mitchell (eds), Morgan Kaufman, pp. 149-166.

Quinlan J.R. 1990. Learning logical definitions from relations. *Machine Learning*, 5(3), pp.239-266.

Quinlan J.R. 1993. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann, San Mateo.

Quinlan J.R. 1995. MDL and categorical theories (continued). *Proc. of the 12th International Conference on Machine Learning*, Californie, pp.464-470.

Quinlan J.R. 1996. Bagging, Boosting and C4.5. *Proc. of the 13th National Conference on Artificial Intelligence (AAAI'96)*, Portland, pp.725-730.

Ramoni M. et Sebastiani P. 1999. *An introduction to the Robust Bayesian Classifier*. KMi Technical Report KMi-TR-79, Knowledge Media Institute, The Open University.

Rase W.D. 1991. Computer-Assisted Map Design. *Advances in Cartography*, Müller (ed), Elsever Applied Science, pp.181-200.

Regnauld N. 1998. *Généralisation du bâti: Structure spatiale de type graphe et représentation cartographique.* Thèse de doctorat, Laboratoire d'Informatique de Marseille.

Regnauld N. 2001. Constraint based mechanism to achieve automatic generalisation using agent modelling. *Proc. of 9th Annual Conference on GIS Research in United Kingdom*, Glamorgan, à paraître.

Regnauld N., Edwardes A.J. et Barrault M. 1999. Strategies in Building Generalisation : Modelling the Sequence, Constraining the Choice. *3rd ICA Workshop on generalisation*, Ottawa. Publié sur le site web de la commission en généralisation de l'ACI : http://www.geo.unizh.ch/ICA/Documents/Workshop99/papers.html

Reichenbacher T. 1995. Knowledge acquisition in map generalization using interactive systems and machine learning. *Proc. of 17th International Cartographic Conference*, Barcelone, Vol 2, pp. 2221-2230.

Ricci F. et Aha D.W. 1997. *Extending local learners with error-correcting output codes*. Technical Report, Naval Center for Applied Research in Artificial Intelligence.

Richeldi, M. et M. Rossotto 1995. Class-Driven Statistical Discretization of Continuous Attributes, *Proc. of the 12th European Conference on Machine Learning*, pp.335-338.

Rieger M. et Coulson M. 1993. Consensus for confusion: Cartographer's knowledge of generalization. *Cartographica*, 30(2&3),pp.69-80.

Rock I. 1983. *The logic of perception*. MIT Press, Cambridge, MA.

Ruas A. 1988. *Généralisation d'immeubles.* Rapport de stage, Ecole Nationale des Sciences Géographiques.

Ruas A. 1998a. Method for buiding displacement in automated map generalisation. *International Journal of Geographical Information Science*, vol.12, n.8, pp789-803.

Ruas A. 1998b. First results on the OEEPE test on generalisation. *OEEPE Newsletter*, 1998-vol.1, pp.5-10.

Ruas A. 1999. *Modèles de généralisation de données géographiques à base de contraintes et d'autonomie.* Thèse de doctorat, université de Marne-la-Vallée.

Ruas A. et Lagrange J.-P. 1994. Modélisation pour la généralisation. *Proc. of European conference on GIS*, Paris, Vol.1, pp.37-47.

Ruas A. et Lagrange J.-P. 1995. Data and knowledge modelling for generalization. *GIS and Generalization, Methodology and Practice*, Müller, Lagrange et Weibel (eds), Taylor and Francis, Londres, pp.73-90.

Ruas A. et Plazanet C. 1996. Strategies for Automated Generalization. *Proc. of the 7th International Symposium on Spatial Data Handling*, Delft, pp.319-336.

Rumelhart D.E., Hinton G.E. et Williams R.J. 1986. Learning internal representations by error propagation. *Parralel Distributed Processing*, Rumelhart et McClelland (eds), Cambridge, MIT Press.

Saafeld A. 1999. Topologically Consistent Line Simplification with the Douglas-Peucker Algorithm. *Cartography and Geographic Information Science*, vol.26, n.1, pp.7-18.

Saitta L. et Zucker J.-D. 1998. Semantic Abstraction for Concept Representation and Learning. *Proc. of Symposium on Abstraction, Reformulation and Approximation*, Pacific Grove, Californie, pp. 543-551.

Schaffer C. 1994. A conservation law for generalisation performance. *Proc. of 11th International Conference on Machine Learning.Morgan Kaufmann*, pp. 259-265.

Schreiber G., Wielinga B. et Breuker J. 1993. *KADS, a principled approach to knowledge based system development*, KBS Academic Press.

Schreiber G., Akkermans H., Anjewierden A., de Hoog R., Shadbolt N., Van de Velde W. et Wielinga B. 2000. *Knowledge Engineering and Mamagement. The CommonKADS Methodology*, MIT Press, Londres.

Sester M. 1998. Interpretation of Spatial Data Bases using Machine Learning Techniques. *Proc. of 8th International Symposium on Spatial Data Handling*, Vancouver, pp. 88-97

Sester M. 2000. Generalization Based on Least Squares Adjustement. *International Archives of Photogrammetry and Remote Sensing*, Vol.XXXIII, Part B4, pp.931-938.

Shavlik J.W. et Dietterich G.D. 1990. *Readings in Machine Learning*. Morgan Kaufmann, San Mateo.

Shavlik J.W., Mooney R.J. et Towell G. 1991. Symbolic and neural computation : An experimental approach. *Machine Learning*, n.6, pp.111-114.

Shortliffe E. 1976. *Computer Based Medical Consultations : MYCIN*. American Elsevier, 1976.

Simon H.A. 1983. Why should machines learn ? *Machine Learning : An artificial intelligence approach*, (Michalski, Carbonell et Mitchell eds), Morgan Kaufmann, pp.25-33.

Soares C. et Brazdil P. 2000. Ranking Classification Algorithms with Dataset Selection: Using Accuracy and Time Results. *Proc. of the 5th International Workshop on Multistrategy Learning*, Guimaraes, pp.249-260.

Spiess E. 1995. The need for Generalisation in GIS environnement. *GIS and Generalization, Methodology and Practice*, Müller, Lagrange et Weibel (eds), Taylor & Francis: Londres, pp.31-46.

Stephan E.M. 1995. Interactive Visualization of Environmental Data. *Proc. of 17th International Cartographic Conference*, Barcelone, vol. 2, pp.2200-2205.

Swiss Society of Cartography 1987. *Cartogrpahic Generalization. Topographic Maps.* Cartogrpahic Publication Series. Swiss Society of Cartography éditeur. 2ème édition.

Taylor F. 1994. Cartographic Visualization and Spatial Data Handling. *Proc. of 6th International Symposium on Spatial Data Handling*, Edimbourg, vol.1, pp.16-26.

Thom R. 1993. *Prédire n'est pas expliquer*. Flammarion: Paris.

Thomas J. 1996. *Vers l'intégration de l'apprentissage symbolique et l'acquisition de connaissances basée sur les modèles: le système ENIGME.* Thèse de doctorat, université Pierre et Marie Curie, Paris VI.

Thórisson K.R. 1994. Simulated Perceptual Grouping: An Application to Human-Computer Interaction. *Proc. of the 16th Annual Conference of the Cognitive Science Society*, Atlanta, pp.876-881.

Töpfer F. et Pillewizer W. 1964. The principles of Selection, A Means of Cartographic Generalization**.** *XXth International Geographical Congress* (repris dans *Cartographic Journal*, 1966, Vol.3, n.1, pp.10-16).

Valiant L. 1984. A theory of the learnable. *Communications of the ACM*, 27(11), pp. 1134-1142.

Van Dalen D. 1983. *Logic and Structure*. Springer-Verlag: Berlin.

Vauglin F. 1997. *Modèles statistiques des imprécisions géométriques des objets géographiques linéaires.* Thèse de doctorat, université de Marne-la-Vallée.

Wang Z. 1992. An Intelligent Interface for Application of Graphics Elements. *Proc. of 5th International Symposium on Spatial Data Handling*, Charleston, Vol.2, pp.391-400.

Ware J.M., Jones C.B. et Bundy G.L. 1995. A triangulated Spatial Model for Cartographic Generalisation of Areal Objects. *Proc. of 2nd International Conference on Spatial Information Theory*, pp.173-192.

Weger G. 1994. *Cours de Cartographie.* Cours de l'Ecole Nationale des Sciences Géographiques. Corrigé et réédité en 1998.

Weibel R. 1989. *Konzepte und Experimente zur Automatisierung der Reliefgeneralisierung.* Thèse de doctorat, Geographisches Institut, Universität Zurich.

Weibel R. 1991. Amplified intelligence and rule-based systems. *Map Generalization, Making rules for Knowledge Representation*, Buttenfield et McMaster (eds), Longman Scientific & Technical, Harlow, Essex. pp. 86-102.

Weibel R. 1996. A Typology of Constraints to Line Simplification. *Proc. of 7th Intrnational Symposium on Spatial Data Handling*, Delft, Pays-Bas, Vol.2, sec.9a, p.1-14.

Weibel R., Keller S. et Reichenbacher T. 1995. Overcoming the Knowledge Acquisition Bottleneck in Map Generalization : the Role of Interactive Systems and Computionnal Intelligence. *Proc. of 2nd International Conference on Spatial Information Theory (COSIT 95)*. pp. 139-156.

Weiss S.M et Kapouleas I. 1989. An empirical comparison of pattern recognition, neural nets and machine learning classification methods. *Proc. of the 11th International Joint Conference on Artificial Intelligence*, Los Altos, pp.781-787.

Werschlein T. et Weibel R. 1994. Use of Neural Networks in Line Generalisation. *Proc. of 5th European Conference and Exhibition on Geographic Information Systems (EGIS 94)*, Paris, Vol.1, pp.77-85.

Wolpert D.H. 1992. Stacked Generalization. *Neural Networks*, 5(2), pp.241-260.

Wolpert D.H. and Macready W.G. 1995. No Free-Lunch Theorems for Search. Technical Report SFI-TR-95-02-010, The Santa Fe Institute, Santa Fe, Etats-Unis.

Wood M. et Clifford H. 1995. Do Map Readers Really Notice and Use Generalization? The Perceptual Consequences of Line Simplification in a Task-Oriented Thematic Map Analysis Experiment. *Proc. of 17th International Cartographic Conference*, Barcelone, Vol. 1, pp. 118-127.

Wood M. et Goodwin A.H. 1995. Digital Terrain Models in Tourism : An investigation into the Development and Use of Computers-Generated 3D Views for Route-Finding in Mountain Areas. *Proc. of 17th International Cartographic Conference*, Barcelone, Vol.2, pp.1497-1502.

Yoshida K. et Motoda H. 1990. Towards Automatic Generation of Hierarchical Knowledge Bases. *Proc. of AAAI Workshop on Automatic Generation of Approximations and Abstractions*, Boston, pp. 98-109

Zhang L. et Tian Z. 1997. Refinement of Douglas-Peucker Algorithm to move the segments towards only one side. *Proc. of 18h International Cartographic Conference*, Stockholm, Vol. 2, pp 831-855.

Zucker J.-D. 1996. *Appariements et changements de représentation pour l'apprentissage symbolique*. Thèse de doctorat, université Pierre et Marie Curie, Paris VI.

Zucker J.-D. 1998. Abstraction for Concept Representation. *Proc. of the 4th International Workshop on Multistrategy Learning*, Lorenza Saitta (ed), Brescia, Italie,.

Zucker J.-D. et Ganascia J.-G. 1996. Representation Changes for Efficient Learning in Structural Domains. *Proc. of the 13th international conference of machine learning, ICML*, Bary, Italie, pp.543-551.

Zucker J.-D., Mustière S. et Saitta L. 2000. Learning Abstraction and Representation Knowledge: an Application to Cartographic Generalisation. *Proc. of the 5th International Workshop on Multistrategy Learning (MSL 2000)*, Guimaraes.

■

Apprentissage Supervisé pour la Généralisation Cartographique

Cette thèse a pour contexte l'automatisation de la généralisation cartographique, processus de création d'une carte à partir d'une base de données géographique trop détaillée. Pour réaliser cela, de nombreux algorithmes existent pour transformer la géométrie des objets géographiques à représenter sur la carte, mais aucun d'entre eux n'est générique. Nous adoptons alors une approche *pas à pas, adaptative et focalisée*, où le traitement d'un objet nécessite l'application de plusieurs algorithmes sur des espaces de travail adéquats. Dans ce contexte, il faut définir des règles permettant de choisir quels algorithmes appliquer sur un objet donné à partir de la description de celui-ci par un ensemble de mesures numériques.

Un processus d'enchaînement des algorithmes est mis au point empiriquement pour la généralisation des routes. L'efficacité et les limites de ce processus conduisent à envisager l'utilisation de l'apprentissage automatique supervisé pour acquérir les connaissances nécessaires à un système expert cartographique. Notre problème d'apprentissage se caractérise par la recherche de règles *efficaces et compréhensibles* à partir d'exemples peu nombreux, bruités et de description riche. Un apprentissage classique produit alors des règles de faible qualité.

Pour améliorer cela, nous guidons l'apprentissage par les connaissances du domaine en décomposant notre problème d'apprentissage en plusieurs sous-problèmes plus simples : nous apprenons tour à tour à *abstraire* puis à *choisir comment transformer* les objets géographiques manipulés. La phase d'*abstraction* consiste à reformuler la représentation des observables sous la forme d'un ensemble restreint de nouveaux attributs symboliques. La phase de *choix de transformation* consiste à déterminer quelle transformation réaliser en fonction de la description abstraite de l'objet. L'introduction de cette phase d'abstraction permet d'apprendre des règles cartographiques à la fois plus efficaces et plus compréhensibles qu'un apprentissage direct. Elle permet d'améliorer ainsi la qualité cartographique des résultats obtenus.

Supervised Machine Learning for Cartographic Generalisation

The context of this work is the automation of cartographic generalisation, which is the process of creating maps from over-detailed geographic databases. Many generalisation algorithms exist to transform the geometry of geographic objects to be represented on the map, but none of them is generic. We use a *step by step, adaptive and focalised* approach, where one geographic object must be transformed by the mean of several algorithms on adapted working spaces. In this context, rules must be determined to choose which algorithms to apply on an object, according to a description of it by the mean of a set of numeric measures.

A process to chain algorithms is empirically developed for road generalisation. The efficiency and the limits of this process incite to use supervised machine learning techniques to acquire the knowledge necessary to a cartographic expert system. Our learning problem can be characterised as the search for *efficient and understandable* rules from few, noisy and large examples. Then, a classical learning algorithm provides rules with a poor quality.

In order to improve the results quality, background knowledge is used to guide the learning process while decomposing the learning problem in several simpler sub-problems : we successively learn to *abstract* and to *choose the transformation* to apply on the geographic objects. The *abstraction* phase changes the object representation from a large set of numeric measures to a small set of new symbolic attributes. The *transformation choice* phase determines which transformation to apply according to the abstract description of the object. The introduction of this abstraction phase enables to learn more efficient and understandable rules than a direct learning. It so enables to improve the cartographic quality of the results.