Vers un Traitement Linéaire des Interrogations Spatiales

*Vincent ORIA**
University of Alberta
Edmonton, Canada T6G 2H1
e-mail : oria@cs.ualberta.ca

Jean-Marc SAGLIO
E.N.S.T** - 46, rue Barrault
75634 Paris cedex 13 - France
e-mail : saglio@inf.enst.fr

Mots Clés : Bases de données spatiales, relations spatiales, interrogations et requêtes spatiales, langage d'interrogation, décomposition d'objets

Résumé

Dans une base de données spatiales, les relations topologiques et les relations de direction permettent de positionner les objets, les uns par rapport aux autres. Les objets spatiaux sont de grande taille et on préfère souvent opérer le traitement des requêtes d'abord sur des approximations afin de réduire l'ensemble des objets avant de leur appliquer les opérateurs spatiaux. Nous avons proposé un langage intermédiaire composé de deux sous-langages complémentaires, permettant le calcul des requêtes spatiales suivant les deux étapes. L'originalité de ce langage se trouve dans le fait qu'il ramène l'étude des relations spatiales à des études linéaires. Nous montrons que la décomposition en trapézoïdes permet une implantation également linéaire du langage.

Abstract

We have designed a logic intermediate spatial query language independent of any storage model based on the spatial relationships of objects (i.e. the positions of objects relative to each other). The language is composed of two complementary sub-languages. The first one uses an object approximate and results in the selection of a set of candidate objects, which are investigated more in detail using the second one. We have showed that this language can be easily computed by a procedural execution using a small set of operators. The manipulated pieces of data are intervals both in the two sub-languages. In this paper, we show how the decompositon of objects into trapezoids allows a straight line processing for the language.

---

* Ce travail a été réalisé lorsque l'auteur était encore affilié à l'ENST - Paris

** Ecole Nationale Supérieure des Télécommunications

INTRODUCTION

Les relations spatiales permettent, dans le cadre d'un langage d'interrogations de bases de données spatiales, de formuler des requêtes sur les positions relatives des objets les uns par rapport aux autres. On distingue deux types de relations spatiales : les *relations de direction* et les *relations topologiques*. Les relations de direction sont les relations telles que "à gauche de", "à droite de", "au dessus de", "en dessous de" [Davis 86], [Chang 88] ou "à l'est de", "à l'ouest de", "au nord de", "au sud de" des applications géographiques [Peuquet 87], [Dutta 89], [Franck 91]. Les relations topologiques sont toutes les relations qui concernent la frontière, l'intérieur et l'extérieur des objets [Egenhofer 91]. La plupart des études antérieures se sont intéressées exclusivement à l'un ou à l'autre type de relations spatiales. La structure d'index 2D-string [Chang 88, Chang 89] est une approche permettant l'indexation des objets spatiaux à partir de projections symboliques des objets sur les différents axes. La même approche a été reprise par Guesgen [Guesgen 89] dans le cadre d'un langage logique, mais son approche reste limitée aux objets de formes rectangulaires.

Des approches théoriques de modèles pour bases de données spatiales commencent à voir le jour [Egenhofer 91, Kanellakis 90, Grumbach 94, Paredaens 94 et Papadimitriou 96]. En attendant une base théorique universelle, nous avons repris l'approche de la projection symbolique pour définir les opérateurs d'un langage logique de filtrage (ProLo : Projection Logic). Nous avons également proposé un langage de raffinement (PaLo : Path Logic), complémentaire à ProLo [Cheiney 95, Oria 94]. Ces travaux ont permis de ramener le traitement des relations spatiales d'un espace donné à des traitements sur des espaces unidimentionnels.

Généralement, on préfère décomposer les objets spatiaux en objets plus simples avant le calcul des relations spatiales. La décomposition a pour but de remplacer un algorithme coûteux par un ensemble d'algorithmes simples et rapides. Plusieurs techniques de décomposition, notamment en géométrie algorithmique [Preparata 88, Boissonat 96], permettent de ramener un polygone en composantes simples et disjointes. D'après la comparaison des méthodes de décomposition [Kriegel 91], la décomposition en trapézoïdes introduite par Asano Ta. et Asano Te. [Asano 83], présente des propriétés intéressantes. Les composantes produites par cet algorithme sont des trapézoïdes avec deux côtés horizontaux, c'est-à-dire parallèles à l'axe des abscisses.

Après un rappel sur le langage ProLo-PaLo dans la première partie, la seconde proposera un algorithme linéaire pour le traitement des requêtes du langage dans le cadre d'un plan.

1- LE LANGAGE ProLo-PaLo

Comme nous l'avons déjà montré dans [Cheiney 95], [Oria 94], le langage ProLo-PaLo est un langage intermédiaire composé de deux sous-langages complémentaires. Il ne nécessite aucune structure de données particulière et peut être utilisé au dessus de tout modèle spatial. Le sous-langage ProLo permet d'exprimer des requêtes filtres dont les résultats seront examinés plus en détail à l'aide de requêtes écrites en PaLo. Cette partie présente les deux sous-langages.

1.1 - LE LANGAGE DE FILTRAGE ProLo(Projection Logic)

La logique des projections permet de ramener l'étude des relations spatiales entre les objets d'un espace de dimension n, à n études sur chacun des domaines élémentaires. Pour cela, on projette préalablement les objets sur les axes et les résultats des projections sont utilisés pour définir des relations spatiales binaires de base.

L'exemple de la figure 1 montre que les projections des objets O_1 et O_2 s'intersectent sur l'axe 1 tandis que la projection de l'objet O_2 précède celle de l'objet O_1 sur l'axe 2. Ce sont ces notions d'intersection et de précédence qui sont reprises dans le langage ProLo, à travers les opérateurs "avant" et "intersecte". Ces deux opérateurs servent à définir des formules atomiques qui, combinées à l'aide de connecteurs logiques classiques, conduisent à des formules de ProLo plus complexes.

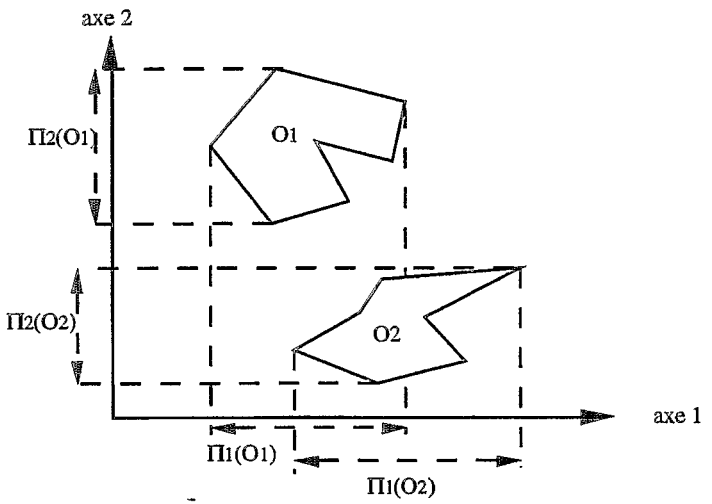


Figure 1 : *Projections d'objets dans un plan*

1.1.1 - Les opérateurs de ProLo

La projection des objets sur les axes donne une représentation grossière des objets. Il n'est pas nécessaire de définir des opérations aussi exactes que l'adjacence, l'inclusion et le chevauchement. Nous regroupons ces trois opérations en une seule qui teste l'intersection. Le langage ProLo n'a donc que deux opérateurs par axe. Ce sont les opérateurs *"avant"* et *"intersecte"* définis de la manière suivante pour l'axe i :

- $i\_avant\ (p_i) : O_1 p_i O_2 \Leftrightarrow \forall x \in P_i(O_1),\ \forall y \in P_i(O_2) : x < y$
- $i\_intersecte\ (I_i) : O_1 I_i O_2 \Leftrightarrow \exists x : x \in P_i(O_1) \wedge x \in P_i(O_2)$.

1.1.2 - Définition des formules

Le langage ProLo se définit pour un espace de dimension n par récurrence comme suit :

- *termes* : une variable sur les objets, une constante objet sont des termes.
- *formules atomiques* : si O_1 et O_2 sont des termes, alors $O_1\ q\ O_2$, où $q \in \{p_1, ..., p_n, I_1, ..., I_n\}$, est une formule atomique.
- *connecteurs logiques* : si f et g sont des formules alors
  $f \wedge g$ est une formule ;
  $\neg f$ est une formule.

Les autres connecteurs logiques se rajoutent comme étant des abréviations. Les propositions connectives $f \vee g$, $f \Rightarrow g$, $f \Leftrightarrow g$ abrègent respectivement $\neg(\neg f \wedge \neg g)$, $(g \vee \neg f)$, $(f \Rightarrow g) \wedge (g \Rightarrow f)$. La constante booléenne *vrai* est l'abréviation de $(f \vee \neg f)$ et *faux* celle de $(\neg vrai)$.

Le langage ProLo se fonde sur l'utilisation d'approximations par les boîtes d'encombrement minimums. Le résultat d'une interrogation en ProLo nécessite un examen plus fin à l'aide d'un langage tel que PaLo.

1.2 - LE LANGAGE DE RAFFINEMENT PaLo(Path Logic)

A la différence de ProLo qui opère à partir des axes, PaLo préconise le tracé de *chemins* (droites paralèlles à un des axes) et étudie les relations entre des parties d'objets intersectant chaque chemin.

Un chemin est une droite parallèle à un des axes et est matérialisé en fixant comme constantes les valeurs de n-1 composantes d'un espace de dimension n. Le domaine pour lequel il n'a pas été fixé de constante est appelé *domaine actif*. Considérons que, dans l'exemple de la figure 2, l_1 ou l_2 sont des chemins avec pour domaine actif le domaine de l'axe 1. Le chemin l_1 coupe bien les objets O_1, O_2 et O_3. Si on considère les intersections du chemin l_1 avec les objets respectifs O_1, O_2 et O_3, on obtient les segments $l_1(O_1)$, $l_1(O_2)$ et $l_1(O_3)$ qu'il est possible de comparer. Par exemple, on peut dire que $l_1(O_1)$ est contigu à $l_1(O_2)$ et que $l_1(O_1)$ est avant $l_1(O_3)$. Nous donnons la définition des opérateurs du langage PaLo et la définition des formules du langage.

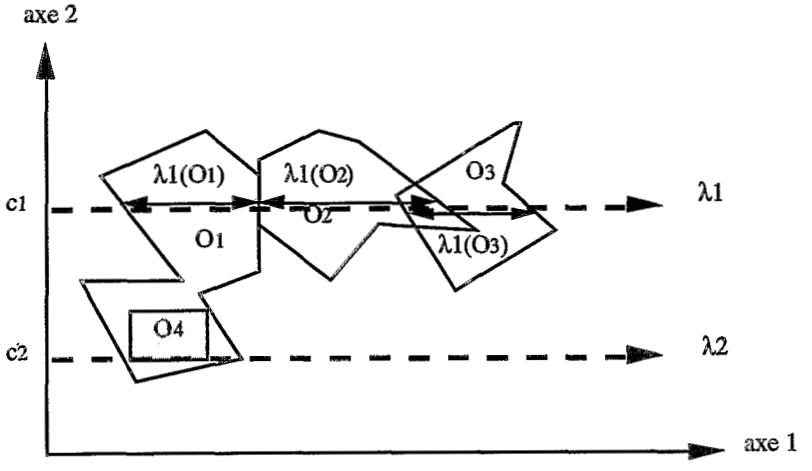


Figure 2 : *Chemins et objets*

1.2.1- Les opérateurs de PaLo

Soit $\lambda : \vec{y} = \vec{c}$, un chemin de domaine actif D_i et soient $\vec{X} = (x_1,...,x_n)$ et $\vec{Y} = (y_1,...,y_n)$ des variables de D ; on définit sur l les opérateurs contenus dans le tableau décrit par la figure 3. Ces opérateurs rappellent ceux définis par Allen [Allen 83] pour des intervales temporels.

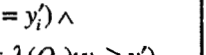
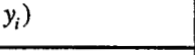
Op. sur l	Signification	Représentation
$O_1 \prec_\lambda O_2$ l_avant	$\forall \bar{X} \in \lambda(O_1), \forall \bar{Y} \in \lambda(O_2): x_i < y_i$	
$O_1 \preceq_\lambda O_2$ l_joint	$(\exists \bar{X}' \in \lambda(O_1), \exists \bar{Y}' \in \lambda(O_2): x_i' = y_i') \wedge$ $(\forall \bar{X} \in \lambda(O_1): x_i \leq x_i') \wedge (\forall \bar{Y} \in \lambda(O_2): y_i \geq y_i')$	
$O_1 \rhd_\lambda O_2$ l_achev al	$(\forall \bar{X} \in \lambda(O_1), \forall \bar{Y} \in \lambda(O_2): x_i \leq y_i) \wedge$ $(\exists \bar{X} \in \lambda(O_1), \exists \bar{Y} \in \lambda(O_2): x_i = y_i)$	
$O_1 \otimes_\lambda O_2$ l_dans	$(\exists \bar{X} \in \lambda(O_1), \exists \bar{Y} \in \lambda(O_2): x_i \leq y_i) \wedge$ $(\exists \bar{Y} \in \lambda(O_2), \forall \bar{X} \in \lambda(O_1): x_i \geq y_i)$	

Figure 3 : *Les opérateurs spatiaux de PaLo pour un chemin l*

De même, les autres opérateurs logiques s'ajoutent comme étant des abréviations. Une formule en PaLo est indéfinie, si elle contient au moins un terme pour lequel l'intersection avec le chemin est vide. Dans le cas de la figure 2, l'expression $(O_4 \prec_{\lambda 2} O_2)$ est indéfinie.

1.2.2 - Définition des formules

Le langage PaLo est un langage logique à deux types : un type objet et un type chemin.

- *termes de type objet :* une variable sur les objets et une constante objet sont des termes de type objet.
- *termes de type chemin :* une variable sur les chemins, une constante chemin sont des termes de type chemin.
- *formule atomique* : si O_1 *et* O_2 sont des termes de type objet et l, un chemin d'un domaine actif D_i, alors $O_1 \theta O_2$ où $\theta \in \{\prec_\lambda, \preceq_\lambda, \rhd_\lambda, \otimes_\lambda\}$ est une formule atomique.
- *quantificateurs* (ne sont utilisés que pour les chemins) : si l est une variable libre sur les chemins et f_λ est une formule sur le chemins l, alors :
  $(\exists \lambda) f_\lambda$ est une formule,
  $(\forall \lambda) f_\lambda$ est une formule.
- *les opérateurs logiques* : si f et g sont des formules :
  $f \wedge g$ est une formule,
  $\neg f$ est une formule.

Dans l'exemple de la figure 2, suivant le chemin l_1, l'expression des relations spatiales est donnée par $(O_1 \preceq_{\lambda 1} O_2) \wedge (O_1 \prec_{\lambda 1} O_3) \wedge (O_1 \rhd_{\lambda 1} O_3)$ et suivant le chemin l_2, on a $(O_4 \otimes_{\lambda 2} O_1)$.

Nous avons montré dans [Cheiney 95, Oria 94] que le langage ProLo-PaLo permet de calculer des requêtes de recouvrement et de disjonction et qu'il est indépendent de toute implantation. Néamoins, le fait d'avoir ramené le calcul des relations spatiales à des calculs linéaires nous a fait penser à une implantantion linéaire que nous proposons. Pour la suite, du fait des résultats préalables que nous utilisons, nous allons nous restreindre à une étude sur un plan.

2 - LA DÉCOMPOSITION DES OBJETS EN TRAPÉZOÏDES ET LA MINIMISATION DES CHEMINS DE PaLo

Les objets spatiaux sont en général de grandes tailles et pour les manipuler, il est préférable de les décomposer en objets simples plus facile à traiter [Préparata 88]. Le trapézoïde et le triangle en sont des exemples. Le trapézoïde présente la caractétistique suivante qu'il a deux côtés parallèles aux axes et donc pouvant supporter des chemins de Palo. Nous présentons la technique de décomposition en trapézoïdes [Asano 83] avant de proposer un algorithme de traitement des requêtes basé sur cette méthode et qui minimise le nombre de chemins.

2.1 LA DÉCOMPOSITION EN TRAPÉZOÏDES

L'algorithme de décomposition des objets en trapézoïdes utilise la méthode du *plane-sweep* [Preparata 88]. L'idée de base de la décomposition en trapézoïdes est d'émettre, à partir des sommets, un ou deux rayons à l'intérieur du polygone jusqu'à toucher un arc. L'algorithme se compose des deux étapes suivantes :

- Trier les sommets du polygone pour construire la liste des points événements (*event point list*), c'est-à-dire, une liste triée de tous les points que l'algorithme aura à traiter.

- Traiter la liste des points événements (sommets) en passant de l'un au suivant et en émettant à chaque fois un rayon de partition (*event point scheduling*). Avec ce processus chaque rayon et son successeur forment un trapézoïde (figure 4). Dans certains cas, on peut avoir des trapézoïdes dégénérés, c'est-à-dire, des triangles avec seulement un côté horizontal.

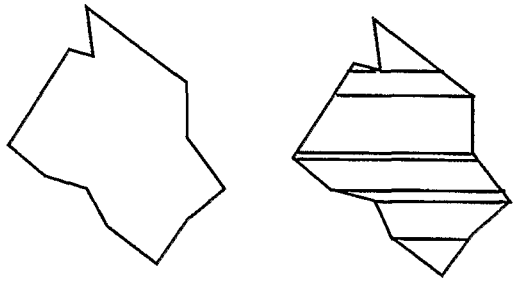


Figure 4: *Exemple de décomposition en trapézoïdes*

Les principales propriétés de la décomposition en trapézoïdes sont :

- par construction, les trapézoïdes possèdent deux côtés horizontaux ; ce qui restreint le processus de décomposition au choix des points d'arrêt *(event point list)* de la *sweep-line* (ligne verticale de balayage de l'espace);

- les trapézoïdes sont des composantes simples ;

- un polygone à n sommets se décompose en m trapézoïdes, avec $m < n$;

- la décomposition est facile à implanter avec un temps d'exécution de l'ordre de $O(nlogn)$ où n représente le nombre de sommets du polygone et le nombre de composantes produites est, dans le pire des cas, égal à trois fois le nombre optimal [Asano 83].

2.2 - LE TRAITEMENT DES INTERROGATIONS ET LA DÉCOMPOSITION EN TRAPÉZOÏDES

Une caractéristique essentielle des trapézoïdes est qu'un ou plusieurs trapézoïdes peuvent être appochés par des rectangles. Le test d'intersection de deux objets revient à tester les intersections de leurs trapézoïdes respectifs. Notre but est de ramener le test d'intersection de deux objets à des tests d'intersection des REM (Rectangles d'Englobants Minimums) des trapézoïdes qui les composent et de ne traiter en détail que les trapézoïdes pour lesquels une intersection est détectée. Autrement dit, le langage ProLo sera utilisé sur les REM des trapézoïdes et PaLo ne s'appliquera que sur les trapézoïdes pour lesquels ProLo aura détecté une intersection.

2.2.1 Algorithme naïf

Soient $R = <r_1,..., r_n>$ et $S = <s_1,..., s_m>$ les séquences respectives des trapézoïdes obtenus par décomposition des objets R et S. On note par $<REM(r_1),..., REM(r_n)>$ et $<REM(s_1),..., REM(s_m)>$ les séquences respectives des rectangles englobants des trapézoïdes R et S. Vérifier que R et S s'intersectent à l'aide de ProLo-PaLo revient à :

- calculer à l'aide de ProLo toutes les paires (r, s) avec $r \in \{r_1,..., r_n\}$ et $s \in \{s_1,..., s_m\}$ telles que $REM(r)$ intersecte $REM(s)$.

- calculer à l'aide de PaLo toutes les paires (r, s) avec $r \in \{r_1,..., r_n\}$ et $s \in \{s_1,..., s_m\}$ telles que r intersecte s.

La solution naïve consiste à comparer à l'aide de ProLo chaque trapézoïde $r \in \{r_1,..., r_n\}$ à tous les trapézoïdes $s \in \{s_1,..., s_m\}$ avec une complexité de $O(n.m)$. Cette solution, qui est meilleure à la solution de comparer l'objet entier R à l'objet entier S, n'est pas satisfaisante car elle n'exploite pas la répartition des objets dans l'espace. Nous en proposons une autre.

2.2.2 L'algorithme du tri-d'abord

La méthode du tri-d'abord consiste à ordonner les trapézoïdes des deux objets suivant l'axe des ordonnées et à ne comparer que ceux qui sont proches. Les étapes de la méthode du tri-d'abord sont :

- ordonner les trapézoïdes des objets R et S suivant l'axe des ordonnées à l'aide de la relation d'ordre partiel définie de la façon suivante :
$O_1 < O_2$ si $\exists x \in P_2(O_1), \forall y \in P_2(O_2) : x \leq y$ avec P_2 la projection des objets sur l'axe des ordonnées ;

- rechercher les paires (r, s) $r \in \{r_1,..., r_n\}$ et $s \in \{s_1,..., s_m\}$ telles que r précède s ou s précède r ;

- effectuer les deux étapes de la solution naïve à partir du résultat de l'étape précédente.

Appliquons la méthode du tri-d'abord à l'exemple de la figure 5 :

- l'ordonnancement des trapézoïdes est : $s_3 < r_4 < s_2 < s_1 < r_3 < r_2 < r_1$;

- la deuxième étape fournit comme paires de trapézoïdes candidats à l'intersection (r_4, s_3), (r_4, s_2), (r_3, s_1) ;

- l'application de ProLo sur le résultat de l'étape précédente donne des résultats positifs pour la paire (r_3, s_1) ;

- l'application de PaLo est également positif pour (r_3, s_1).

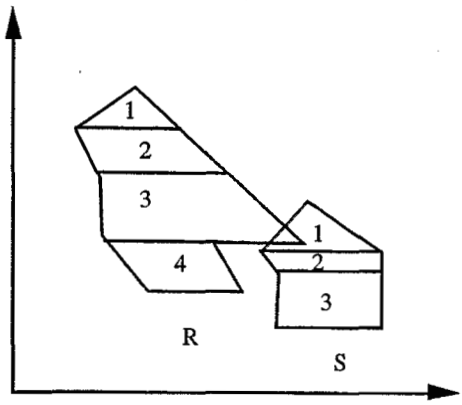


Figure 5 : *Un exemple de test d'intersection d'objets après décomposition en trapézoïdes*

Cette solution permet d'éviter l'application systématique de PaLo à tous les trapézoïdes. Faut-il pour autant parcourir tous les chemins coupant chaque paire de tapézoïdes candidats ?

Proposition : *Soient r et s deux trapézoïdes dont l'un au moins est non dégénéré. On note par* c_1^r *et* $c_2^r (c_1^r < c_2^r)$ *les coefficients des chemins qui supportent les deux côtés parallèles du trapézoïde r. De même on note par* c_1^s *et* $c_2^s (c_1^s < c_2^s)$ *les coefficients des chemins qui supportent les deux côtés parallèles du trapézoïde s. Si r et s s'intersectent alors il existe* $c = coeff(l)$ *et* $c \in \{[c_2^r, c_1^s], [c_2^s, c_1^r]\}$ *tel que l'on ait rqs où* $\theta \in \{\preceq_\lambda, \rhd_\lambda, \otimes_\lambda\}$, *c'est-à-dire que l(r) et l(s) s'intersectent.*

Preuve : $c \in \{[c_2^r, c_1^s], [c_2^s, c_1^r]\}$ représente la zone d'intersection de $P2(r)$ et de $P2(s)$. Une condition nécessaire pour qu'un point $p(x, y)$ appartienne aux trapézoïdes r et s (si on suppose r<s au sens de la relation d'ordre definie dans la section 2.2.2) est que $c_2^r \leq y \leq c_1^s$ en particulier pour $y = c_2^r$ ou $y = c_1^s$.

La proposition ramène le test d'intersection de deux trapézoïdes à l'étude de la répartition des deux objets sur deux chemins. Sur chacun des chemins, la partie d'un des objets sur le chemin est donnée par un des côtés parallèles de l'un des trapézoïdes.

CONCLUSION

Les formules des langages ProLo et PaLo font référence à des domaines unidimentionnels. En effet, que ce soit au niveau de PaLo ou de ProLo, une requête se caractérise par un ensemble de formules à vérifier sur un ensemble donné de modèles linéaires. Ce qui nous a permis d'envisager un traitement linéaire des requêtes formulées à l'aides de ces langages. L'implantation linéaire de ProLo-PaLo impose une minimisation du nombre de chemins. Bien que le modèle s'applique à un espace quelconque, nous avons abordé la minimisation des chemins dans le cadre d'un plan afin de tirer profit des différents travaux effectués en géométrie algorithmique [Préparata 88].

En nous aidant des techniques connues de décomposition des objets en éléments simples (notamment en trapézoïdes), nous avons montré que l'étude de la répartition des trapézoïdes sur seulement deux chemins conduit au test d'intersection de deux trapézoïdes. De plus, les trapézoïdes peuvent être approchés par des rectangles. Cette remarque permet d'appliquer ProLo plusieurs fois de manière à localiser les trapézoïdes susceptibles de s'intersecter et de n'utiliser Palo que sur ces trapézoïdes uniquement. Nous avons montré que tester l'intersection de deux trapézoïdes revient à étudier, à l'aide de PaLo, la répartition de ces trapézoïdes sur uniquement deux chemins. De nouveaux algorithmes de décomposition existent [Boissonat 95], les travaux futurs chercheront à en tirer profit. Nous envisageons également d'étendre les résultats présentés ici à des espaces de dimension supérieure à deux.

BIBLIOGRAPHIE

[Allen 83] ALLEN J. F., *"Maintaining Knowledge about Temporal Intervals"*, CACM, vol. 26(11), November 1983

[Asano 83] ASANO Ta., ASANO Te., *"Minimum Partition of Polygonal Regions into Trapezoids"*, in Proc. of 24th Annual Symposium on Fondations of Computer Science, 233-241, 1983

[Boissonat 95] BOISSONAT et YVENIC, *"Géométrie Algorithmique", Ediscience, 1995*

[Chang 88] CHANG N. S., YAN C. W., DIMITROFF D. C., ARNDT T., *"An Intelligent Image Database System"* IEEE Transactions On Software Enginnering, Vol 14, N°5, May 1988

[Chang 89] CHANG S. K., JUNGERT E., LI Y., *"The Design of Pictorial Database Based Upon the Theory of Symbolic Projection"* In Proc. of 1st International Symposium on Design and Implementation of Large Spatial Databases (SSD), Santa-Barbara USA, July 1989

[Cheiney 95] CHEINEY J.-P. , ORIA V., *"Spatial Databases Querying with Logic Languages"*, in Proc. of 4th Int. Conf. on Database Systems for Advanced Applications, Singapore, April, 1995

[Davis 86] D AVIS E. *"Representing and Acquiring Geographic Knowledge"*, Morgan Kaufmann Publishers Inc., Los Altos, CA, 1986

[Dutta 89] DUTTA S. *"Qualitative Spatial Reasonning : a Semi-Quantitative Approach Using Fuzzy Logic"* In proc. of Design and Implementation of Large Spatial Databases (SSD'89), Santa Barbara, July, 1990

[Egenhofer 91] EGENHOFER M.J., *"Reasoning about Binary Topological Relations"* In Proc. of 2nd symp. on the Design of Large Spatial Databases (SSD'91), Zurich, Switzerland, 1991

[Franck 91] FRANCK A. *"Qualitative Spatial Reasonning About Cardinal Directions"*, In Autocarto 10, Baltimore, MD, March 1991

[Grumbach 94] GRUMBACH S., SU J., *"Finetely Representable Databases"*, in Proc. of 13th ACM Symposium on Principles Database Systems (PODS), Miniapolis, USA, 1994

[Guesgen 89] GUESGEN H. W., *"Spatial Reasoning Based on Allen's Temporal Logic"*, Technical Report TR-89-049, International Computer Science Institute, Berkeley, August 1989

[Kanellakis 90] KANELLAKIS P. C., KUPER G. M., REVESZ P. Z. *"Constraint Query Languages"*, "in Proc. of 9thn 1th ACM Symposium on Principles Database Systems (PODS),Atlantic City, NJ , USA, 1990

[Kriegel 91] KRIEGEL H-P., BRINKHOFF T.,SCHEIDER R., *"The Combination of Spatial Access Methods and Computational Geometry in Geographic Database Systems "* in Proc. of 2nd Symp. on the Design of Large Spatial Databases (SSD'91), Zurich, Switzerland, 1991

[Oria 94] Oria V. *"Construction d'une Base de données d'Images : Etudes d'une Approche orientée Objets et d'un Langage logique d'Interrogations spatiales"*, Thèse de l'ENST-Paris, septembre 1994

[Papadimitriou 96] PAPADIMITRIOU C. H., SUCIU D., VIANU V., *"Topological Queries in Spatial Dtabases"*, in Proc. of 15th ACM Symposium on Principles Database Systems (PODS), Montréal, Canada, 1996

[Paredaens 94] PAREDAENS J., VAN DEN BUSSCHE J., VAN GUCHT D., *"Toward a Theory of Spatial Database Queries"*, in Proc. of 13th ACM Symposium on Principles Database Systems (PODS), Miniapolis, USA, 1994

[Peuquet 87] PEUQUET D. J., CI-XIANG Z., *"An Algorithm to Determine the directional Relationship between Arbitrarily-Shaped Polygons in the Plane"*, Pattern Recognition, 20(1), pp 65-74,

[Preparata 88] PREPARATA F., SHAMOS I. S., *"Computational Geometry an Introduction"*, Springer-Verlag, 1988