Randomisation, sphères et déplacements de robots

Olivier Devillers

MEMOIRE D'HABILITATION

Randomisation, sphères et déplacements de robots

présenté

par

Olivier Devillers

soutenu le 23 novembre 1993.

Jury	MM.	André Galligo	Président
		Jean Berstel	
		Jean-Daniel Boissonnat	
		André Cérézo	
		Claude Puech	
		Pierre Rosenstiehl	
Rapporteurs	MM.	Jean Berstel	
		Bernard Chazelle	
		Kurt Mehlhorn	

Table des matières

Avant-propos

Je reprends dans ce mémoire d'habilitation la plupart de mes travaux depuis mon admission au projet PRISME de l'INRIA en septembre 1989. Grâce à un groupe de travail actif et des collaborations fructueuses tant à l'intérieur qu'à l'extérieur du projet PRISME, ces années se sont révélées très enrichissantes, elles ont vu le projet PRISME participer activement à l'animation de la recherche, en France notamment par son implication dans les journées de géométrie algorithmique (1990,1992,1993) et la création de $\mathcal{GÉDÉoN}$.

Ce mémoire porte sur 14 articles présentés à des conférences ou publiés dans des journaux scientifiques. La plupart de ces articles peuvent etre trouvés à l'url : http://www.inria.fr:/prisme/personnel/devillers/

Ces articles sont souvent co-rédigés et la liste des différents co-auteurs avec leurs coordonnées est donnée plus loin.

- **• Algorithmes randomisés**
- Applications of Random Sampling to On-line Algorithms in Computational Geometry
- Randomization Yields Simple $O(n \log^\star n)$ Algorithms for Difficult $\Omega(n)$ Problems
- A Semi-Dynamic Construction of Higher Order Voronoi Diagrams
- Fully Dynamic Delaunay Triangulation in Logarithmic Expected Time per Operation
- Dynamic location in an arrangement of line segments in the plane
- Dog Bites Postman: Point Location in the Moving Voronoi Diagram and Related Problems
- Robust and efficient implementation of the Delaunay tree
- **• Algorithmes sur les sphères**
- The Space of Spheres, a Geometric Tool to Unify Duality Results on Voronoi Diagrams
- Output Sensitive Construction of Delaunay Triangulations of Constrained Sets of Points

- An Algorithm for Constructing the Convex Hull of a Set of Spheres in Dimension d
- Computing Connolly Surfaces
- **• Déplacements de robots**
- Simultaneous Containment of Several Polygons: Analysis of the Contact Configurations
- Motion Planning of Legged Robots: the Spider Robot Problem
- Computing the Union of 3-Colored Triangles

Liste des co-auteurs

Jean-Daniel Boissonnat INRIA,
BP 93, 06902 Sophia-Antipolis cedex.
Adresse électronique : `Jean-Daniel.Boissonnat@sophia.inria.fr`.

André Cérézo Université de Nice,
Parc Valrose, 06034 Nice cedex.

Leonbattista Donati INRIA,
BP 93, 06902 Sophia-Antipolis cedex.

Jacqueline Duquesne INRIA,
BP 93, 06902 Sophia-Antipolis cedex.
Adresse électronique : `Jacqueline.Duquesne@sophia.inria.fr`.

Mordecai Golin Hong Kong University of Science & Technology,
Clear Water Bay, Kowloon, Hong Kong.
Adresse électronique : `golin@cs.ust.hk`.

Stefan Meiser Institut Max Planck.
Max Planck Institut für Informatik, D-6600 Saarbrücken, Germany.
Adresse électronique : `meiser@mpi-sb.mpg.de`.

Franco P. Preparata Université de Brown.
Dep. of Computer Science, Brown University, Providence, RI 02912 (USA)
Adresse électronique : `franco@cs.brown.edu`.

René Schott CRIN,
B.P.239, 54506 Vandœuvre cedex.
Adresse électronique : `schott@loria.crin.fr`.

Monique Teillaud INRIA,

BP 93, 06902 Sophia-Antipolis cedex.

Adresse électronique : `Monique.Teillaud@sophia.inria.fr`.

Mariette Yvinec CNRS,

URA 1376, Laboratoire I3S, 250 Rue Albert Einstein, 06560 Valbonne.

Adresse électronique : `Mariette.Yvinec@sophia.inria.fr`.

Introduction

La géométrie algorithmique n'a pas toujours été mon domaine de prédilection. La plupart des spécialistes la font naître entre 1975 et 1980 avec la thèse de M. I. Shamos [Sha78] et Jean-Daniel Boissonnat retraçait récemment son histoire et ses perspectives dans son mémoire d'habilitation [Boi92]. Bien qu'ayant débuté mes travaux de recherche bien après 1978, mon premier domaine d'intérêt fut différent. Ma thèse portait sur des problèmes assez appliqués ayant trait à la synthèse d'images et plus particulièrement au tracé de rayons. Même si certains problèmes pratiques qui m'étaient posés alors avaient nécessité l'utilisation d'outils théoriques complexes tels que étude de complexité et géométrie stochastique, mes préoccupations théoriques restèrent rigoureusement motivées par l'application que je désirais en faire et je ne sus pas prendre le recul nécessaire. Les spécialistes de géométrie algorithmique que je côtoyais alors au LIENS me semblaient un peu loin des réalités concrètes de l'informatique et les complexités théoriques, les grands $O()$ et $\Omega()$ peu représentatifs de l'intérêt pratique d'un algorithme. Bref, à l'époque j'ai remis le nez sur mon clavier et continué à «écrire du code».

Lors de mon arrivée à l'INRIA, le domaine d'application changeait, la synthèse d'images cédait la place à la planification de trajectoires, mais mon état d'esprit restait essentiellement le même, c'est à dire celui d'un praticien, il semblait à l'époque que mon expérience des structures hiérarchiques de subdivision de l'espace pouvait être intéressante pour modéliser l'espace libre des robots. La suite des événements en décida autrement.

La thèse de Francis Avnaim [Avn89] qui abordait la planification de trajectoires sous l'angle «géométrie algorithmique» avec tous les aspects théoriques que cela comporte m'a intéressé et m'a prouvé que cela n'était pas incompatible avec des applications pratiques ; le logiciel PIAF développé par Francis est utilisé dans l'industrie de la découpe. Quelques temps après, Jean-Daniel Boissonnat me demanda de relire l'article sur l'analyse randomisée de l'arbre de Delaunay [BT89]. Ce fut mon premier contact avec la randomisation. Je fus séduit par cette approche qui combinait des probabilités discrètes, donc assez simple, à des algorithmes également simples et élégants. Je m'intégrai alors dans les travaux suivants, et afin de ne pas rompre brusquement avec mon passé de programmeur invétéré, je me lançai dans la programmation de l'arbre de Delaunay et de ses avatars.

J'avais découvert un domaine qui alliait le caractère formel et la rigueur des mathématiques à l'aspect concret et visuel de objets géométriques, le tout en gardant des possibilités d'applications pratiques. Je m'éloignai donc de la synthèse d'images et de ses aspects parfois légèrement bricolés pour me convertir à la géométrie algorithmique, rendue encore plus attrayante par la présence à

Sophia d'une équipe dynamique et présente au niveau international, dans les projets ESPRIT, les conférences et journaux du domaine.

Les travaux présentés dans ce mémoire d'habilitation sont tout d'abord résumés ci-dessous, puis un certain nombre de publications sont regroupées à la suite. Le résumé comme les publications sont organisés en trois parties.

La première est consacrée aux algorithmes randomisés. Nos travaux sur ce sujet, préfiguré par l'arbre de Delaunay, ont permis notamment l'obtention d'un schéma général d'algorithmes *en-ligne*, des algorithmes dynamiques pour la triangulation de Delaunay et les arrangements de segments, et d'algorithmes pour le squelette d'un polygone simple et la triangulation de Delaunay connaissant l'arbre couvrant minimal meilleur que les algorithmes déterministes.

La deuxième partie traite de problèmes concernant les sphères, avec des résultats sur la triangulation de Delaunay 3D de points contraints à appartenir à un petit nombre de plans, et sur l'enveloppe convexe de sphères. Ces résultats font intervenir deux modélisations différentes des sphères en dimension d sous la forme de points en dimension $d+1$.

Des problèmes d'origine plus robotique sont abordés dans la dernière partie : le placement de plusieurs robots polygonaux se déplaçant en translation dans un environnement polygonal et la planification de trajectoires pour des robots à pattes.

Chapitre I

Algorithmes randomisés

Les algorithmes randomisés[1] sont un de nos sujets d'intérêt majeur. Les algorithmes classiques de géométrie algorithmique sont généralement complexes et difficiles à mettre en œuvre. Une tendance récente consiste à utiliser des algorithmes plus simples, dont la complexité n'est pas optimale dans tous les cas, mais seulement en moyenne sur certains choix aléatoires fait par l'algorithme. Bien que la notion d'algorithmes randomisés ne soit pas propre à la géométrie algorithmique nous nous limiterons ici à des applications à ce domaine. Contrairement à un algorithme classique qui déduit un résultat à partir de certaines données de départ en suivant un chemin bien déterminé, l'algorithme randomisé a le choix entre plusieurs chemins pour parvenir à ses fins. L'intervention du hasard dans ce type de méthodes réside dans la manière de choisir le chemin que va finalement suivre l'algorithme. L'analyse de l'algorithme est alors faite en moyenne sur les différents chemins possibles. Aucune hypothèse n'est faite sur la répartition des données. Un algorithme non randomisé est appelé déterministe. Il est important de noter que seul le déroulement de l'algorithme est aléatoire et que le résultat est parfaitement déterminé, ainsi que les données sur lesquelles on ne fait aucune hypothèse de type probabiliste tels que que «les points suivent une loi de Poisson»...

Plusieurs techniques pour introduire du hasard dans un algorithme existent. Par exemple, à chaque fois qu'une bifurcation se présente à l'algorithme et que deux chemins sont envisageables pour continuer le calcul, il suffit de choisir à pile ou face. Pour les algorithmes division-fusion,[2] la division peut être faite de manière aléatoire [Cla87]. *Quicksort* peut être vu de cette manière si on le décrit de la façon suivante. Etant donnés n nombres à trier, on choisit au hasard l'un

[1] de nombreuses réserves ayant été émises sur l'anglicisme que voilà, je me dois à une petite justification. Tout d'abord nous n'avons pas trouvé de mot à racine latine acceptable, «aléatoirisé» n'ayant vraiment pas séduit nos oreilles. Les médecins utilisent déjà «randomisé» abondamment : un essai thérapeutique est randomisé si les malades qui vont recevoir le médicament sont tirés au sort. Enfin F. P. Preparata propose aux puristes une éthymologie acceptable : «randomiser» : du français «randonner» (courir impétueusement, c'est-à-dire au hasard), du francique «rant» (cf. l'allemand moderne *rennen*, courir) [ℊℰ𝒟ℰℴ𝒩12].

[2] encore un problème de langue, si le *divide and conquer* ne semble pas contesté en anglais, par contre an français on peut entendre : la traduction mot à mot «diviser pour conquérir», ou bien «diviser pour régner» ou encore une version plus lisse, ni guerrière ni politique, «division-fusion» qui a le mérite de rappeler que après avoir subdivisé le problème en sous-problèmes, il faudra regrouper les résultats partiels en un résultat global.

de ces nombres qu'on appelle le pivot et l'on divise notre ensemble de nombres en trois sous ensembles: le pivot, les nombres plus petits que le pivot et les nombres plus grands que le pivot. Les sous-ensembles sont triés récursivement et il ne reste plus qu'à concaténer les trois résultats pour obtenir le résultat final. Ici le hasard est intervenu dans le choix du pivot, c'est-à-dire dans la manière de diviser le problème en sous problèmes.

La catégorie d'algorithmes randomisés à laquelle nous nous sommes plus particulièrement intéressés est celle des *algorithmes randomisés incrémentaux*. Dans ce genre d'algorithmes le hasard n'intervient pas de prime abord, il s'agit en fait d'algorithmes dont le comportement dépend de l'ordre d'insertion des données. Si l'on considère que l'ordre dans lequel les données sont fournies fait partie du problème alors de tels algorithmes n'ont rien de randomisé, par contre si l'on considère que les données sont fournies en bloc, sans ordre apparent, l'algorithme se voit obligé de choisir entre une multitude de chemins correspondant aux différents ordres possibles sur les données.

Clarkson et Shor [CS89] proposèrent un schéma d'algorithmes utilisant une structure appelée *graphe de conflits*. Ces algorithmes sont incrémentaux, c'est-à-dire que les données sont examinées à tour de rôle, et pour chaque nouvelle donnée un résultat partiel est mis à jour. Le graphe de conflits contient certaines relations entre les résultats partiels et les données non encore considérées, ces relations sont appelées conflits. L'aspect randomisé est simplement assuré par le choix d'un ordre aléatoire pour l'examen des données.

Cette technique du graphe de conflits, malgré son caractère incrémental, impose la connaissance de toutes les données lors de la phase initiale, car les données non considérées doivent être présentes dans le graphe de conflits. Notre travail a permis de rendre ces algorithmes *en ligne* [BDS+92]. La structure que nous avons développée, baptisée *graphe d'influence*, contient l'histoire des différents résultats partiels. Elle permet d'insérer les données de manière semi-dynamique, en déterminant les conflits de la nouvelle donnée avec tous les résultats partiels. Ceux-ci se déduisent les uns des autres au fur et à mesure que l'on «déroule» l'histoire des insertions. Cette technique peut s'appliquer à de nombreux problèmes: calcul d'arrangements, d'enveloppes convexes, de graphes de visibilité, de diagrammes de Voronoï généralisés et, avec quelques modifications, de diagrammes de Voronoï d'ordre k [BDT93]. Tous ces algorithmes sont des outils de base en géométrie algorithmique, en particulier dans le cadre de la robotique où beaucoup de problèmes de planification de trajectoires nécessitent des calculs d'arrangements ou bien utilisent des propriétés de réduction sur des diagrammes de Voronoï.

Après ces premiers résultats nous avons étudié la possibilité de supprimer des données dans les structures randomisées afin d'obtenir des algorithmes totalement dynamiques. L'approche que nous avons utilisée est toujours basée sur le graphe d'influence et sur la connaissance de l'histoire de la structure qui nous intéresse. Dans les algorithmes précédents, seule l'insertion de nouveaux objets était permise. Si l'on autorise maintenant la suppression d'objets une première possibilité consiste à continuer à stocker l'histoire de la construction [Sch91], mais une telle approche peut conduire à ce que la taille de l'histoire soit beaucoup plus importante que le nombre d'objets effectivement présents à l'instant qui nous intéresse. La technique que nous avons préféré employer consiste à reconstruire l'histoire à chaque suppression. Lorsqu'une donnée doit être supprimée, elle est détruite dans le résultat courant, mais aussi dans tous

les résultats intermédiaires entre son insertion et l'étape actuelle; le passé de notre structure est alors modifié pour ne tenir compte que de l'histoire des insertions des objets effectivement présents. Nous avons ainsi obtenu une structure totalement dynamique permettant une mise à jour en temps logarithmique (en moyenne) de la triangulation de Delaunay dans le plan [DMT92a]. La même approche s'est également révélée efficace pour le calcul d'un arrangement de segments [DTY92].

Ces résultats ont été implantés pour le calcul des diagrammes de Voronoï d'ordre supérieurs, des diagrammes de Voronoï de segments et des arrangements de segments. La programmation de la triangulation de Delaunay a été effectuée en accordant un soin particulier aux problèmes de dégénérescence des données et de précision numérique [Dev92b].

Paradoxalement le développement du graphe d'influence dont l'objectif était la (semi-)dynamisation d'algorithmes statiques a eu également des répercussions sur ces algorithmes statiques. En combinant les deux techniques des graphes d'influence et de conflits R. Seidel parvint à un algorithme simple de triangulation d'un polygone simple en temps $O(n \log^\star n)$ [Sei91]. Plutôt que de rechercher tous les conflits du nouvel objet dans le graphe d'influence, R. Seidel propose de débuter la recherche, non à l'origine de l'histoire, mais à quelques instants clé. Afin de pouvoir initialiser cette recherche il faut calculer un graphe de conflits à chacun de ces instants clé. Pour que cette méthode soit intéressante, il est nécessaire d'exploiter une information supplémentaire sur les données pour mener rapidement le calcul de ces quelques graphes de conflits; dans le cas de R. Seidel, il calcule la carte des trapèzes d'un ensemble de segments en utilisant le graphe d'influence puis il parvient à accélérer le processus en exploitant les relations d'adjacence entre ces segments dont on sait qu'ils forment un polygone simple.

L'analyse de R. Seidel utilise le fait que, dans ce cas particulier, il n'y a qu'un seul conflit à chaque étape de l'histoire. Après avoir généralisé cette analyse au cas de plusieurs conflits, nous avons pu proposer deux nouvelles applications : le calcul du squelette d'un polygone simple et de la triangulation de Delaunay d'un ensemble de points connaissant l'arbre couvrant minimal en temps $O(n \log^\star n)$ [Dev92a]. Le squelette, ou *edge Voronoi diagram* ou encore axe médian est une structure assez répandue, utilisée tant en géométrie algorithmique que dans d'autres domaines comme la reconnaissance des formes ou la morphologie mathématique. Ce résultat améliore les précédentes bornes connues pour le calcul du squelette d'un polygone simple. En ce qui concerne l'arbre couvrant minimal, le résultat est en fait un peu plus général : étant donné un sous ensemble connexe des arêtes de la triangulation de Delaunay, celle-ci peut être reconstruite en temps $O(n \log^\star n)$. L'utilisation d'une telle connaissance partielle du résultat est a priori loin d'être suffisante pour reconstruire le résultat complet. En utilisant comme sous-ensemble connexe de Delaunay l'arbre couvrant de longueur minimale, on trouve le résultat énoncé ci-dessus qui prouve une certaine forme d'équivalence entre la triangulation de Delaunay et l'arbre couvrant de longueur minimale.

Nous allons ci-dessous donner quelques applications de ce type d'algorithmes incrémentaux pour résoudre des problèmes géométriques. Mais auparavant, les principes généraux seront illustrés sur l'exemple du tri de n nombres. Nous ne donnerons ici qu'une idée générale des choses, les lecteurs intéressés par les détails des démonstrations sont invités à consulter les articles référencés.

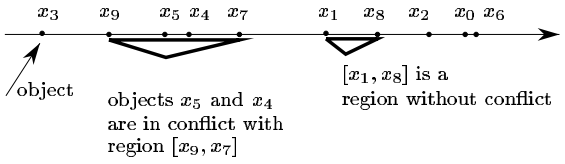

FIG. I.1 – *L'exemple du tri*

I.1 Graphe de conflits et graphe d'influence

Nous devons définir un minimum de formalisme, ou plutôt exprimer le problème que nous allons chercher à résoudre d'une manière suffisamment générale afin de recouvrir un maximum d'applications différentes.

Notre problème s'exprime en termes d'objets, régions et conflits. Les *objets* sont les données du problème: dans le cas du tri un objet est un nombre réel, un des n nombres de l'ensemble $\mathcal{S} = \{x_i\}_{i\in\{1,...,n\}}$ à trier. Une *région* peut être vue comme une certaine entité géométrique déterminée par un petit nombre (borné par une constante) d'objets. Pour le tri, une région est déterminée par deux objets x_i et x_j et est en fait l'intervalle $[x_i, x_j]$. Il faut ensuite définir une notion de *conflit* entre objets et régions; toujours dans notre exemple du tri, un nombre est en conflit avec un intervalle s'il appartient à cet intervalle (voir figure I.1).

Objets, régions et conflits étant définis, les algorithmes proposés plus loin vont permettre de calculer l'ensemble des régions définies par les objets d'un ensemble de données $\mathcal{S}$ et sans conflits avec ces objets. Le tri des éléments de $\mathcal{S}$ peut bien s'exprimer de cette façon puisqu'un intervalle $[x_i, x_j]$ est sans conflits si x_i et x_j se succèdent dans l'ensemble des points triés.

I.1.1 L'approche «stockage des conflits»

La première méthode utilisée consiste à calculer le résultat sur les k premiers objets et à stocker les conflits entre les régions déjà calculées et les objets non encore insérés dans une structure appelée graphe de conflits [CS89]. Lorsque l'on veut insérer le $(k+1)^{\text{ème}}$ objet, le graphe de conflits permet d'accéder directement aux régions en conflit avec cet objet, il faut donc supprimer ces régions du résultat courant, déduire les nouvelles régions sans conflit crées par l'insertion de l'objet et également déterminer les conflits entre les objets non encore insérés et les régions que l'on vient de créer.

Pour fixer les idées, passons tout de suite à la description de ce processus dans le cas du tri de n nombres. Supposons que les k premiers nombres $x_i, i \leq k$ aient été triés et que les conflits soient connus, c'est-à-dire que pour tout nombre $x_l, l > k$ on sache à quel intervalle $[x_i, x_j]$ le nombre x_l appartient, où $[x_i, x_j]$ est un intervalle sans conflits défini par les k premiers nombres. Dans cet exemple un objet n'est en conflit qu'avec une seule région.

Voyons maintenant comment va se dérouler l'insertion de x_{k+1}. On connaît l'intervalle $[x_i, x_j]$ le contenant, qui n'est donc plus sans conflits; il faut détruire $[x_i, x_j]$ et le remplacer par $[x_i, x_{k+1}]$ et $[x_{k+1}, x_j]$. A ce stade, les $k+1$ premiers nombres sont triés, mais il faut encore mettre à jour le graphe de conflits:

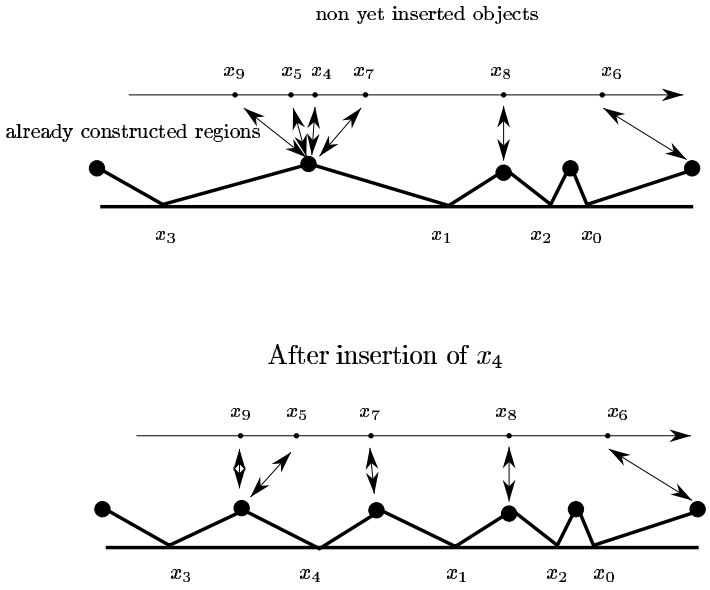

FIG. I.2 – *Le graphe de conflits*

tous les nombres qui étaient en conflit avec d'autres régions que $[x_i, x_j]$ ne sont pas affectés, et ceux qui étaient en conflit avec $[x_i, x_j]$ doivent être comparés à x_{k+1} pour décider s'ils sont maintenant en conflit avec $[x_i, x_{k+1}]$ ou $[x_{k+1}, x_j]$ (voir figure I.2). Il est à remarquer que cette dernière phase de mise à jour des conflits ressemble bigrement à celle de *Quicksort* dans laquelle on dit «pour trier récursivement les nombres dans l'intervalle $[x_i, x_j]$, je choisis parmi eux un nombre au hasard, le pivot (x_{k+1} par exemple) et je divise les nombres contenus dans $[x_i, x_j]$ entre ceux qui sont plus grands ou plus petits que le pivot.» En bref, cet algorithme n'est rien d'autre que *Quicksort*, mais vu sous un autre angle.

Cette méthode du graphe de conflits s'est révélée extrêmement efficace pour un certain nombre d'applications, mais elle présente l'inconvénient d'être statique, c'est-à-dire que tous les objets doivent être connus au début afin d'initialiser le graphe de conflits avec une unique région déterminée par zéro objets et en conflit avec tous les objets. Le paragraphe suivant va proposer une approche semi-dynamique permettant de résoudre les mêmes problèmes que le graphe de conflits.

I.1.2 L'approche «historique»

En fait le graphe de conflits peut être avantageusement remplacé par une structure qui, plutôt que de stocker les conflits des objets à insérer, détermine les régions en conflit avec tout nouvel objet. Cette approche permet d'obtenir des algorithmes semi-dynamiques, c'est-à-dire que les objets ne sont pas connus à l'avance mais seulement au moment de leur insertion. L'idée de base du graphe d'influence [BDS+92] consiste à stocker l'histoire de la construction. Lorsque l'insertion d'un nouvel objet provoque la disparition de régions en conflit avec

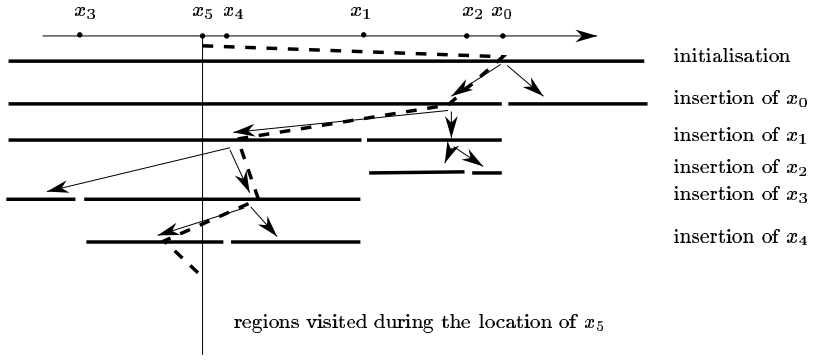

FIG. I.3 – *Le graphe d'influence*

celui-ci, elles ne sont pas supprimées mais seulement marquées inactives, les régions crées par l'insertion du nouvel objet sont reliées aux régions existantes afin de permettre la détermination des conflits. Cette idée qui consiste à utiliser l'histoire de la structure afin de localiser les conflits est apparue en géométrie algorithmique avec l'arbre de Delaunay [BT86] et a été reprise dans de nombreux travaux [Tei93, Sei91, GKS92, ...].

Dans le cas du tri, on peut détailler l'algorithme. Le graphe d'influence est un arbre binaire dont les nœuds sont des intervalles, et les deux fils d'un nœud correspondent à une partition de son intervalle en deux sous intervalles. Lorsque un nouveau nombre est inséré, il est localisé dans cet arbre binaire et la feuille qui le contient devient un nœud interne, son intervalle est coupé en deux morceaux par rapport au nouveau nombre inséré (voir figure I.3). Dans le cas du tri de nombres réels, le graphe d'influence n'est rien d'autre qu'un classique arbre binaire de recherche (sans rééquilibrage). En fait les comparaisons effectuées sont exactement les mêmes dans le cas du graphe d'influence et du graphe de conflits : si x_i et $x_j, i < j$ doivent être comparés, ils le seront lors de l'insertion de x_i dans le cas du graphe de conflits et lors de la localisation de x_j dans le cas du graphe d'influence. Cette similitude entre graphe de conflits et d'influence est d'ailleurs tout à fait générale, les tests de conflits effectués sont exactement les mêmes, ils sont seulement différés dans le cas du graphe d'influence ce qui permet d'obtenir des algorithmes semi-dynamiques.

I.1.3 Complexité

Tels qu'ils sont exposés ci-dessus, les algorithmes présentés ne sont pas vraiment randomisés. Ce sont des algorithmes incrémentaux qui mettent à jour un certain résultat (l'ensemble des régions sans conflit) dès qu'un nouvel objet est inséré. Il se trouve que si l'on fait une analyse de complexité classique, c'est-à-dire dans le cas le pire, on obtient des résultats assez décevants pour la simple raison que l'insertion du nouvel objet peut changer radicalement le résultat courant de notre algorithme.

Nous allons alors randomiser l'algorithme, c'est-à-dire introduire une dose de hasard dans son déroulement, et ceci d'une manière extrêmement simple : non

plus en insérant les points dans un ordre bien défini, imposé par l'utilisateur qui fournit les objets, mais dans un ordre aléatoire. D'un point de vue pratique, si l'on en a la possibilité, on randomisera les données, c'est-à-dire qu'on leur appliquera une permutation aléatoire ; mais l'on ne peut obtenir ainsi que des algorithmes statiques. Si l'on veut utiliser les aspects semi-dynamiques du graphe d'influence, on devra prendre les données dans l'ordre dans lequel elles se présentent. Dans bien des applications, cet ordre est suffisamment aléatoire pour obtenir de bons résultats.

L'hypothèse de base est donc que les données sont insérées dans un ordre aléatoire. Plutôt que d'effectuer ici une analyse du cas général [BDS+92, Dev92a] on va voir comment il est possible d'analyser simplement l'exemple du tri traité plus haut, à l'aide de l'analyse en arrière [Sei91].

Nous allons analyser le coût d'insertion du $n^{\text{ème}}$ nombre x_n dans le graphe d'influence (c'est-à-dire dans un arbre binaire de recherche). Rappelons que tous les intervalles sans conflit existant à un moment donné au cours du processus incrémental sont conservés dans la structure. Juste après l'insertion du $k^{\text{ème}}$ nombre, un seul intervalle $[x_i, x_j]$ (sans conflit à l'étape k) est en conflit avec x_n, alors l'hypothèse randomisée nous permet de dire que x_i est le $k^{\text{ème}}$ nombre avec probabilité $\frac{1}{k}$ et de même pour x_j, en un mot : «la région en conflit avec x_n à l'étape k a été créée à l'étape k avec probabilité $\frac{2}{k}$». On obtient alors facilement le nombre de régions en conflit avec le $n^{\text{ème}}$ nombre inséré en sommant sur l'étape de création de ces régions.

Coût d'insertion de x_n

$$
\begin{aligned}
&= \text{Nombre de régions du graphe en conflit avec } x_n \\
&= \sum_{1 \le k < n} \text{Probabilité que la région en conflit à l'étape } k \text{ ait été créée à l'étape } k \\
&= \sum_{1 \le k < n} \frac{2}{k} \\
&= O(\log n)
\end{aligned}
$$

Le coût d'insertion d'un nombre dans la structure est donc $O(\log n)$ et celui du tri de n nombres est de $O(n \log n)$.

Le même genre de techniques permet d'obtenir des résultats généraux. Le temps de calcul et l'espace mémoire utilisée par l'algorithme sont calculés en fonction de la taille moyenne des résultats intermédiaires. La moyenne est bien sur à prendre ici dans un sens randomisé, la taille moyenne du résultat intermédiaire à l'étape k est celle obtenue à partir d'un échantillon aléatoire de k objets parmi nos données initiales. Plus formellement

Théorème : *Si $f_0(k)$ est le nombre moyen de régions sans conflit déterminées par un échantillon aléatoire de taille k d'un ensemble $\mathcal{S}$ de n objets, alors les régions sans conflit déterminées par $\mathcal{S}$ peuvent être calculées en temps $O\left(\sum_{j=1}^{n} f_0(\lfloor \frac{n}{j} \rfloor)\right)$ et avec un espace mémoire $O\left(\sum_{j=1}^{n} \frac{f_0(\lfloor \frac{n}{j} \rfloor)}{j}\right)$, tant par la technique du graphe de conflits que du graphe d'influence, pourvu que les objets soient insérés dans un ordre aléatoire.*

Corollaire : *Si $f_0(k) = O(k)$ alors les régions sans conflit déterminées par $\mathcal{S}$ peuvent être calculées en temps $O(n \log n)$ et avec un espace mémoire $O(n)$*

I.2 Algorithmes complètement dynamiques

Comme on l'a vu plus haut, au prix d'une petite astuce technique (supposer que les objets sont insérés dans un ordre aléatoire) on obtient des algorithmes semi-dynamiques simples d'emploi et avec des complexités théoriques excellentes.

De nombreux travaux ont alors porté sur les algorithmes complètement dynamiques dans lesquels les objets peuvent être insérés mais aussi supprimés. Ce problème crucial a rarement une solution optimale en géométrie algorithmique. Ainsi le problème le plus emblématique de la géométrie algorithmique: l'enveloppe convexe de points dans le plan, n'a pas de solution dynamique optimale, mais seulement une solution avec une complexité $O(\log^2)$ pour chaque insertion ou suppression [OvL81].

Avec les algorithmes randomisés est venu l'espoir d'obtenir des algorithmes qui ne soient pas optimaux pour toute séquence d'insertions et de suppressions, mais le soient si on «randomise» ces insertions ou suppressions. Plus précisément on suppose que les insertions se font dans un ordre aléatoire parmi tous les ordres possibles sur les objets à insérer et que lorsqu'un objet est supprimé il s'agit de n'importe lequel des objets présents avec la même probabilité.

Plusieurs techniques ont été utilisées pour s'attaquer à ce problème. J'en distinguerai trois, conserver toute l'histoire de la structure, conserver seulement l'histoire des insertions ou ne pas utiliser l'histoire mais procéder par tirage au sort. Comme pour les graphes de conflits et d'influence nous allons illustrer ces techniques sur l'exemple d'un ensemble de nombres ordonnés, on obtient ainsi des structures de dictionnaires qui sont équilibrées «en moyenne», moins lourdes à gérer que les structures équilibrées dans le cas le pire telle que arbres bicolores ou AVL.

I.2.1 Toujours l'approche «historique»

Une idée survient naturellement. Puisque le graphe d'influence se rappelle les ordres sur toutes les séquences de nombres présents à un instant donné dans la structure, si l'on veut maintenant effectuer des suppressions, on va continuer à stocker l'histoire de la structure mais cette fois ci en tenant compte des suppressions. Si $[x, y]$ et $[y, z]$ sont deux régions sans conflit (i.e. y est le seul nombre entre x et z) et que l'on désire supprimer y, on va créer un nouveau nœud $[x, z]$ dans le graphe que l'on va déclarer fils de $[x, y]$ et $[y, z]$ (voir figure I.4).

Cette approche a été initialement développée par O. Schwarzkopf [Sch91, Sch92] et présente le grand avantage de la simplicité et de la facilité d'implantation. Il y a deux inconvénients principaux à cette méthode, le premier est que la taille du graphe construit dépend de la taille de la séquence d'insertions suppressions, et non du nombre d'objets effectivement présents, ainsi si on cherche à maintenir un ensemble de taille stable d'environ 1000 nombres triés et que l'on effectue un million d'insertions et de suppressions, la structure va avoir une taille de l'ordre de un million; ceci peut être résolu en nettoyant la structure de temps en temps, dès que l'histoire est trop grande par rapport au nombre d'objets présents, on reconstruit la structure. Le deuxième inconvénient est d'ordre plus technique, il est fréquent qu'avec ce type de structure l'arité du graphe soit plus mal contrôlée, un nœud du graphe peut avoir beaucoup de fils et il faut être capable de choisir le bon. On est alors obligé d'ajouter des structures

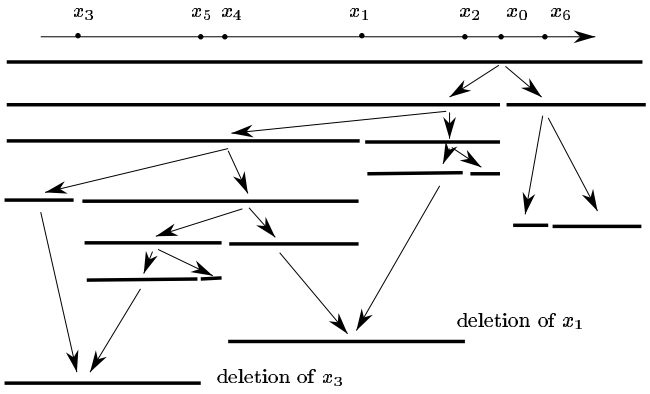

FIG. I.4 – *l'histoire des suppressions*

auxiliaires pour atteindre ce but, celles-ci compliquent un peu l'algorithme et rajoutent parfois un facteur logarithmique dans la complexité théorique.

I.2.2 Encore l'approche «historique», mais modifiable

L'approche que nous avons préférée [DMT92a, DTY92] utilise toujours l'histoire mais l'histoire des insertions seulement. Je m'explique, tant que l'on ne fait que des insertions, on procède comme précédemment, et lorsque l'on veut supprimer un objet, on essaie de réécrire une histoire conforme aux objets présents comme si l'objet supprimé n'avait jamais existé. Cette approche digne de Georges Orwell dans son roman 1984 [Orw34] nous permet de toujours manipuler le graphe d'influence défini précédemment, la structure est exactement la même, mais sa mise à jour devient beaucoup plus compliquée dans certains cas.

En ce qui concerne notre exemple fétiche cela reste assez simple. Tous les nœuds du graphe correspondant à des intervalles déterminés par l'objet supprimé sont effacés, et le fils le plus ancien de ces nœuds est promu pour remplacer le nœud détruit (voir figure I.5).

Ce genre d'approche nécessite de manipuler le graphe d'influence, non plus en l'augmentant par l'ajout de fils à ses feuilles, mais en le modifiant, y compris dans ses nœuds internes. Lorsque un nœud interne est supprimé, ses fils se voient privés de père et il faut les raccrocher à des nouveaux nœuds.

I.2.3 L'approche «pile ou face»

Dans certains cas, il existe une alternative à cette approche «historique». Ketan Mulmuley a proposé différentes approches dynamiques randomisées en géométrie algorithmique [MS91, Mul91b, Mul91c, Mul91d]. Comme précédemment nous allons illustrer l'idée de base sur l'exemple du dictionnaire. On commencera par une version statique du problème puis on verra comment la dynamiser.

Supposons que l'on dispose de n nombres triés $x_1 < x_2 < \ldots < x_n$. On va construire un échantillon de cet ensemble en tirant pour chaque nombre à pile ou face pour savoir s'il fait partie de l'échantillon. Puis on recommence pour faire un échantillon de l'échantillon et ainsi de suite jusqu'à ce qu'il n'y ait plus de nombres. Les différents niveaux de la structure sont reliés entre eux,

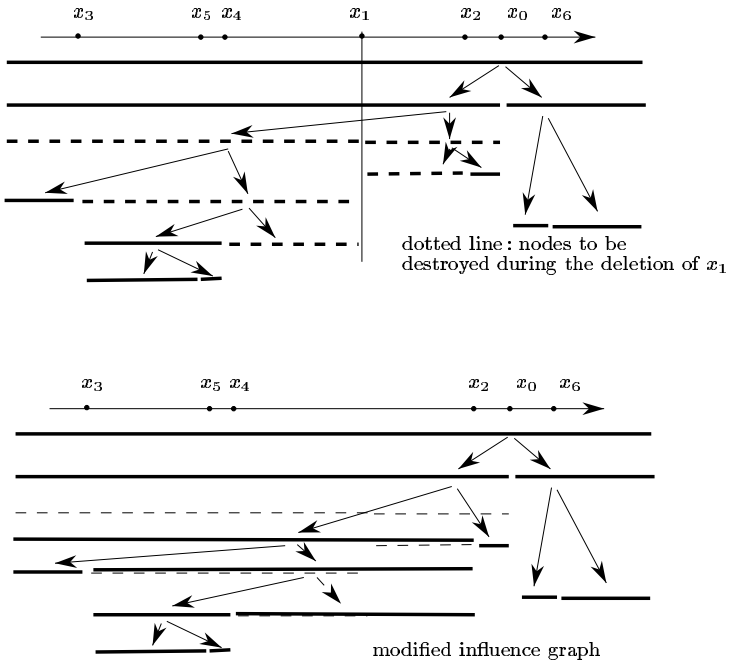

FIG. I.5 – *L'histoire des insertions*

et si maintenant on veut localiser un nouveau nombre il suffit de le localiser successivement dans les différents échantillons, la localisation à chaque étage de la structure servant de base pour la localisation au niveau inférieur (voir figure I.6).

La description ci dessus est statique, mais cette structure se maintient dynamiquement très facilement : lors de l'insertion d'un nouvel objet il suffit de le localiser au niveau le plus bas où on l'insère, puis on joue à pile ou face, si on gagne on l'insère au niveau supérieur et ainsi de suite jusqu'à ce que l'on perde. Lorsqu'on veut supprimer un objet il suffit de le supprimer à tous les niveaux de la structure où il est présent. On obtient une espèce d'arbre à arité variable dont la mise à jour est très facile et qui est équilibré en moyenne. Dans le cas du dictionnaire on peut prouver que les localisations se font en temps logarithmique et les insertions et suppressions se font en temps constant (en plus de la localisation).

I.3 Algorithmes accélérés

Faire coopérer graphe de conflits et graphe d'influence

Comme on l'a expliqué plus haut le graphe de conflits connaît à tout instant les conflits entre les objets alors que le graphe d'influence permet de déduire les conflits avec un objet à l'étape k à partir des conflits à l'étape $k-1$. L'idée a alors été introduite par Seidel [Sei91] de faire coopérer ces deux structures.

Lorsque l'on insère un objet dans le graphe d'influence on calcule les régions en conflit à l'étape 1, on en déduit ceux à l'étape 2 et ainsi de suite jusqu'à l'étape

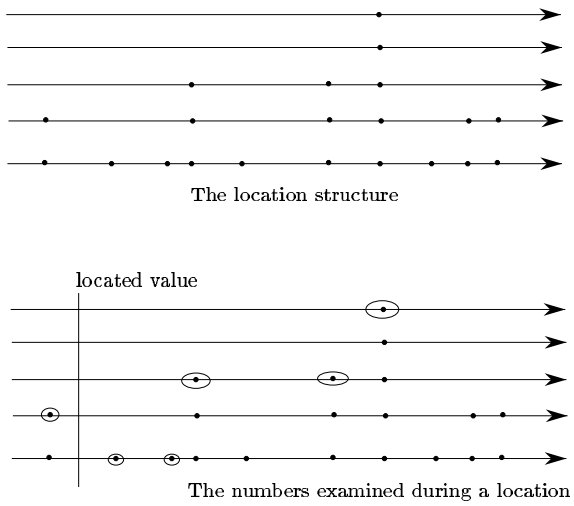

FIG. I.6 – *La structure «pile ou face»*

courante. L'idée consiste donc à ne pas démarrer cette recherche à l'étape 1 mais à une étape intermédiaire k. Il faut bien sûr trouver les conflits à l'étape k par un autre moyen. Ces conflits à l'étape k peuvent par exemple être stockés dans un graphe de conflits. Ce graphe de conflits n'est pas maintenu au cours de la construction, mais construit une fois pour toutes lors d'un arrêt à un instant clé: l'étape k. Cette combinaison de graphe de conflits et de graphe d'influence n'est envisageable que si l'on a un moyen de calculer directement le graphe de conflits à l'instant clé.

Complexité

Dans le cas où $f_0(r) = r$ on a vu que le temps de localisation dans le graphe d'influence était de l'ordre de la somme $\sum_{j=1}^{n} \frac{1}{j} = \log n$ où j représente les différentes étapes auxquels le nouvel objet est localisé, dans le cas qui nous occupe la complexité d'une localisation partielle entre les étapes k et n est donc obtenue en faisant une sommation partielle $\sum_{j=k}^{n} \frac{1}{j} = \log n/k$.

Choisir les instants clé

L'algorithme procède par phases d'insertion dans le graphe d'influence et phases de construction du graphe de conflits à des instants clé. On va supposer, comme cela sera le cas dans les applications, que les phases de construction du graphe de conflits se font en temps linéaire, et l'on va équilibrer le coût des différentes phases. Plus précisément, on va avoir une première phase d'insertion pendant un temps linéaire puis un construction de graphe de conflits, puis une nouvelle phase d'insertion pendant un temps linéaire etc...

Puisque l'insertion d'un objet dans le graphe de conflits a un coût logarithmique, en un temps linéaire on pourra traiter $\frac{n}{\log n}$ objets, le premier instant clé est donc après l'insertion des $\frac{n}{\log n}$ premiers objets. Une fois le graphe de conflits à cet instant clé construit, le coût d'insertion d'un objet est majoré, non plus seulement par $O(\log n)$, mais par $O(\log \frac{n}{n/\log n}) = O(\log \log n)$. Afin d'obtenir

un coût linéaire pour la seconde étape d'insertion, on choisit donc le second instant clé après l'insertion des $\frac{n}{\log\log n}$ premiers objets. Et ainsi de suite, le $^{\text{e}}$ instant clé et donc, la $^{\text{e}}$ construction d'un graphe de conflits est située après l'insertion des $\frac{n}{\log^{(i)} n}$ premiers objets (où $\log^{(i)}$ signifie log itéré i fois).

La totalité des objets est donc traitée lorsque $\log^{(i)} n$ est inférieur à 1, c'est-à-dire lorsque $i = \log^\star n$ (c'est la définition de $\log^\star$). L'algorithme se compose donc de $\log^\star n$ phases d'insertion et de construction de graphe de conflits, chacune de complexité linéaire, la complexité totale est donc de $O(n \log^\star n)$.

Nous verrons plus loin quelques applications de ce principe général d'accélération d'algorithmes permettant de passer d'une complexité de $O(n \log n)$ à $O(n \log^\star n)$: la triangulation et le squelette d'un polygone simple, le diagramme de Voronoï de points dont on connaît l'arbre couvrant de longueur minimale. Et on peut imaginer que cette stratégie connaîtra d'autres applications dans l'avenir.

Ce genre de stratégie par instants clé et localisations en temps $O(\log n/k)$ ne peut pas permettre de meilleur résultat de complexité pour ces problèmes que $O(n \log^\star n)$ [CMS92] alors qu'en général la seule borne inférieure connue pour ces problèmes est linéaire.

I.4 Applications

Nous allons maintenant développer quelques applications de la technique du graphe d'influence, nous nous contenterons en fait de décrire ici la manière dont chaque application particulière peut être s'intègre dans le schéma général brièvement exposé ci-dessus et introduit en détail [BDS+92, Dev92a].

Pour chacune des applications nous allons nous borner à énoncer comment il faut définir les objets, les régions et les conflits afin de pouvoir appliquer le théorème du graphe d'influence.

I.4.1 Enveloppes convexes

L'enveloppe convexe est le problème favori du géomètre algorithmique, car bien que son énoncé soit très simple sa résolution prend en compte toutes les facettes de la géométrie algorithmique. Il est donc naturel de débuter ces applications du graphe d'influence par le calcul des enveloppes convexes.

Algorithmes semi-dynamiques

Pour appliquer le schéma général du graphe d'influence au problème du calcul de l'enveloppe convexe d'un ensemble de points du plan, il suffit de définir les objets comme étant ces points et les régions comme étant les demi-plans déterminés par deux points, chaque paire d'objets déterminant donc deux régions. Enfin on dit qu'un point est en conflit avec un demi-plan si ce point appartient à ce demi-plan. Le problème du calcul de l'enveloppe convexe s'exprime alors bien comme calculer les régions sans conflits : en effet la région définie par deux objets (points) p et q est sans conflit si et seulement si pq est une arête de l'enveloppe convexe. Ici $f_0(k) = O(k)$ (la taille de l'enveloppe convexe est linéaire) le résultat suivant est donc une simple application du théorème du graphe d'influence.

Théorème : *Le graphe d'influence permet de maintenir l'enveloppe convexe de n points du plan avec un temps moyen de mise à jour de $O(\log n)$ si l'insertion des points est randomisée.*

En dimension supérieure d, les objets sont toujours des points, les régions sont toujours des demi-espaces déterminés maintenant par d points et les conflits sont définis de même. En l'absence d'hypothèses sur la répartition des points, $f_0(k)$ est borné par la taille d'une enveloppe convexe en dimension d, c'est-à-dire $O(n^{\lfloor \frac{d}{2} \rfloor})$.

Théorème : *Le graphe d'influence permet de maintenir l'enveloppe convexe de n points en dimension d avec un temps moyen de mise à jour de $O(n^{\lfloor \frac{d}{2} \rfloor - 1})$ si $d \geq 4$ et $O(\log n)$ si $d \leq 3$. L'insertion des points est supposée randomisée.*

Par dualité ces résultats s'appliquent également au problème de l'intersection de n demi-espaces en dimension d.

k-ensembles, k-niveaux Ces résultats peuvent se généraliser aux cas des k-ensembles ou par dualité des k-niveaux, dans ce cas on s'intéresse non plus aux régions sans conflits mais aux régions ayant au plus k conflits [BDT93][Mul91a].

Algorithmes dynamiques

L'algorithme présenté ci-dessus est semi-dynamique, les points peuvent être ajoutés mais non enlevés. La difficulté dans la dynamisation de l'enveloppe convexe est que, lors de la suppression d'un point, un grand nombre de points qui étaient à l'intérieur de l'enveloppe convexe peuvent apparaître sur son bord. Il est donc nécessaire de stocker les points qui n'apparaissent pas sur l'enveloppe convexe de façon à pouvoir les réutiliser le moment voulu.

La solution proposée par Clarkson, Mehlhorn et Seidel [CMS92] utilise la technique de l'histoire des insertions que nous avons développée et permet d'obtenir le résultat suivant.

Théorème : *L'enveloppe convexe de n points en dimension d peut être maintenue dynamiquement avec un temps moyen de mise à jour de $O(n^{\lfloor \frac{d}{2} \rfloor - 1})$ si $d \geq 4$ et $O(\log n)$ si $d \leq 3$. L'insertion des points est supposée randomisée, et la suppression est supposée concerner n'importe quel point présent avec la même probabilité.*

I.4.2 Arrangements

Un autre problème classique de géométrie algorithmique est le calcul de l'*arrangement* de courbes ou de surfaces, c'est à dire du découpage du plan ou de l'espace induit par un ensemble de courbes ou de surfaces.

Arrangements de segments

Une des instances les plus simples de ce problème est «étant donné un ensemble de segments dans le plan, calculer les points d'intersection». L'aspect délicat de ce problème est que la taille du résultat varie énormément selon les données, pour un ensemble de n segments, le nombre de points d'intersection t peut varier de 0 à n^2, on cherche donc à obtenir des algorithmes *à complexité dépendant de la taille de la sortie*. Un des grands classiques de la géométrie algorithmique est l'algorithme de balayage dit de Bentley Ottmann [PS85] dont

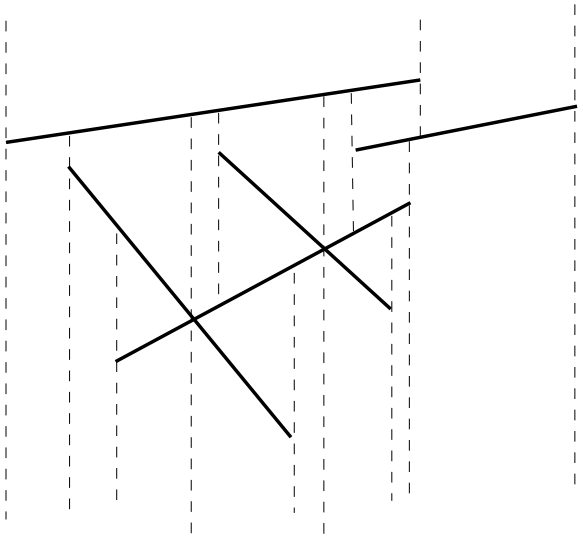

FIG. I.7 – *Décomposition trapézoïdale d'un arrangement de segments*

la complexité est $O((n+t)\log n)$, il existe également un algorithme déterministe optimal [CE92] de complexité $\Theta(n\log n + t)$. Ici encore le graphe d'influence va permettre l'obtention d'un algorithme de complexité moyenne optimale et dont la simplicité fait un excellent candidat à une implantation effective en machine.

La description de l'algorithme se réduit à la description du graphe d'influence, on cherche en fait à construire la décomposition trapezoïdale de l'arrangement : par chaque extrémité de segment et par chaque point d'intersection, on trace un trait vertical que l'on interrompt au premier segment rencontré. Pour calculer cette décomposition trapézoïdale, il suffit de dire que les objets sont les segments, les régions sont les trapèzes et qu'un segment est en conflit avec un trapèze s'il le coupe. Alors la moulinette du graphe d'influence peut digérer la chose, calculer la décomposition trapézoïdale correspond exactement à calculer les régions sans conflits. Ici, $f_0(r)$ est r plus le nombre moyen de points d'intersection entre les segments d'un échantillon aléatoire de taille r de l'ensemble des n segments. Un point d'intersection doit être compté si les deux segments qui le déterminent ont été choisis dans l'échantillon, soit avec une probabilité $\left(\frac{r}{n}\right)^2$. Donc, $f_0(r) = r + \frac{r^2}{n^2}$. On peut donc conclure.

Théorème : *Le graphe d'influence permet de maintenir la décomposition trapezoïdale d'un arrangement de n segments avec un temps moyen d'insertion optimal de* $O(\log n + \frac{k}{n})$ *si l'insertion des segments est randomisée (k désigne la taille de l'arrangement).*

La notion de décomposition trapezoïdale se généralise à des arrangements de courbes planes (de degré borné), et donc la méthode du graphe d'influence aussi. La différence tient essentiellement dans la mise en œuvre pratique qui est plus complexe et également la constante qui est cachée dans le «grand O» qui augmente très rapidement avec le degré des courbes. Cet algorithme peut également être rendu complètement dynamique [DTY92].

Arrangements de triangles

La généralisation de ce qui précède au cas tridimensionnel (pour un arrangement de triangle dans l'espace) est possible, mais nettement plus complexe. En effet la généralisation naturelle de la décomposition trapezoïdale consiste à utiliser des «murs» verticaux passant par les arêtes des triangles et par les intersections entre les triangles, mais une telle décomposition doit encore être raffinée car toutes les cellules n'ont pas une description de taille bornée. Boissonnat et Dobrindt obtiennent un résultat intéressant, légèrement sous-optimal, pour le calcul de l'enveloppe inférieure d'un arrangement de triangles (la partie visible des triangles lorsque l'on regarde depuis l'infini) [BD92]:

Théorème : *Le graphe d'influence permet de calculer l'enveloppe inférieure d'un arrangement de triangles en temps moyen* $O(n^2\alpha(n)\log n)$ *si l'insertion des segments est randomisée (*α *désigne l'inverse de la fonction d'Ackermann et est une fonction à croissance extrêmement faible).*

Calcul d'une seule composante connexe

L'algorithme ci-dessus peut également être modifié pour ne calculer qu'une seule composante connexe de l'arrangement, ceci peut se révéler très avantageux, si l'on n'est intéressé que par une composante connexe bien sûr, et si la taille de cette composante connexe est inférieure à la taille de l'arrangement total. Chazelle *et al.* on proposé un algorithme permettant de maintenir seulement la composante connexe d'un arrangement de segments contenant un point cible donné à l'avance. Il faut pour cela a chaque insertion de segment examiner si le nouveau segment découpe la composante connexe qui nous intéresse en plusieurs parties. On aboutit à un algorithme de complexité $O(n\alpha(n)\log n)$ pour le calcul d'une cellule [CEG+91].

De Berg, Dobrindt et Schwarzkopf [dBDS94] proposent une autre solution dans laquelle cet examen de la découpe éventuelle de la composante connexe recherchée est faite de façon paresseuse, non pas à chaque insertion de segment, mais une fois de temps en temps. Cette approche aboutit à la même complexité et permet une généralisation en trois dimensions.

Triangulation d'un polygone simple

La triangulation d'un polygone simple est également un des classiques de la géométrie algorithmique, et au cours des années la complexité des algorithmes pour ce problème a chuté de $O(n^2)$ à $O(n\log n)$, $O(n\log\log n)$, $O(n\log^\star n)$ et enfin Chazelle a proposé une solution optimale en temps linéaire [Cha91], mais une fois encore cet algorithme n'est optimal que d'un point de vue théorique et une mise en œuvre pratique semble assez irréaliste. La décomposition trapézoïdale permet d'obtenir facilement une triangulation de l'ensemble des segments, l'algorithme randomisé permet donc tel quel l'obtention d'un algorithme en temps moyen $O(n\log n)$.

C'est ici que rentrent en scène les algorithmes accélérés, en effet connaissant la décomposition trapézoïdale d'un échantillon des segments et le polygone simple, il est facile de construire le graphe de conflits entre les segments non encore insérés et les trapèzes déjà construits, il suffit pour cela de suivre le bord du polygone simple dans la décomposition en trapèzes. Le coût d'une telle opération est clairement linéaire en fonction de la taille du graphe de conflits construit.

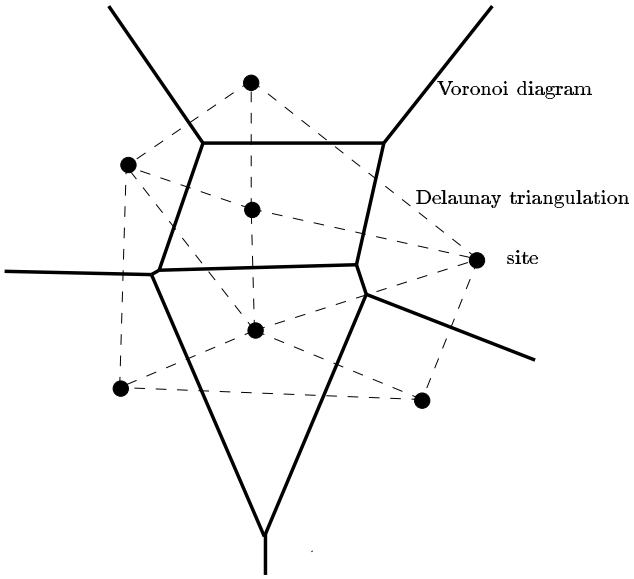

FIG. I.8 – *Le diagramme de Voronoï de 7 sites*

Nous sommes donc exactement dans le cadre décrit pour les algorithmes accélérés, et Seidel a ainsi obtenu un algorithme simple ayant une excellente complexité même si celle-ci est légèrement sous optimale [Sei91].

Théorème : *La décomposition trapezoïdale des n segments formant le bord d'un polygone simple peut être calculée en temps moyen $O(n \log^\star n)$. Une triangulation du polygone simple s'en déduit alors en temps $O(n)$.*

I.4.3 Diagrammes de Voronoï

La première application que nous avons traitée, avant même toute ébauche du schéma général du graphe d'influence, concernait le diagramme de Voronoï ou son dual, la triangulation de Delaunay.

Diagramme de Voronoï de points

Nous rappellerons tout d'abord la définition du diagramme de Voronoï dans le plan. Etant donné un ensemble de points du plan que nous appellerons les sites, le diagramme de Voronoï est la partition du plan en cellules telle que chaque cellule soit formée des points les plus proches d'un site donné que de tous les autres (voir figure I.8). Cette structure sert à résoudre le problème de la recherche du plus proche voisin : «étant donné un point du plan, quel est le site le plus proche ?» également appelé problème du bureau de poste : «Où est la poste la plus proche pour que j'y donne mon courrier ?» Le dual du diagramme de Voronoï, la triangulation de Delaunay est obtenue en joignant deux sites si leurs régions de Voronoï sont adjacentes.

Une propriété caractéristique de la triangulation de Delaunay est que le cercle circonscrit à un triangle ne contient pas de sites, ou en termes duaux, le centre du cercle circonscrit est équidistant de trois sites (les sommets du triangles) et

plus près de ceux là que de tous les autres : c'est donc un sommet de Voronoï. Fort bien, cette propriété du cercle vide nous guide donc tout droit dans la machinerie du graphe d'influence, il suffit de dire que les objets sont les sites, que les régions sont les triangles et qu'un site est en conflit avec un triangle s'il est à l'intérieur du cercle circonscrit à ce triangle et les résultats coulent (presque) tout seuls [BDS+92, Dev92a]. Nous avons pu rendre cet algorithme dynamique en utilisant la technique de l'histoire des insertions [DMT92a]. Le problème des dégénérescences tels que points alignés ou cocycliques peut être traité de façon satisfaisante [Dev92b].

Théorème : *Le graphe d'influence permet de maintenir la triangulation de Delaunay et le diagramme de Voronoï de n points du plan en temps moyen* $O(\log n)$ *par insertion et* $O(\log \log n)$ *par suppression si l'insertion des sites est randomisée et si le site supprimé est n'importe lequel des sites présents avec la même probabilité.*

Ces résultats se généralisent en dimension supérieure.

Théorème : *Le graphe d'influence permet de maintenir la triangulation de Delaunay et le diagramme de Voronoï de n points en dimension* $d \geq 3$ *avec un temps moyen d'insertion de* $O(n^{\lfloor \frac{d+1}{2} \rfloor - 1})$ *si l'insertion des sites est randomisée.*

Diagramme de Voronoï d'ordre k Les diagrammes de Voronoï d'ordre k dans lesquels les cellules ne sont plus formées des points les plus proches d'un site donné mais de k sites donnés, peuvent également être calculés avec le même genre de technique [BDT93][AS92].

Diagramme de Voronoï de segments

On s'intéresse également au cas du diagramme de Voronoï pour d'autres types de sites que des points, un des cas les plus fréquents est celui de segments ou de polygones (figure I.9).

Ce diagramme est également appelé squelette ou axe médian. Là encore le graphe d'influence peut s'appliquer. On choisit bien évidemment comme objets les segments, les régions seront associées aux arêtes du diagramme de Voronoï, qui sont définies par 4 segments, et enfin on définira la zone d'influence d'une arête de Voronoï comme l'union des cercles centrés sur les points de l'arête et touchant l'obstacle le plus proche (figure I.10); on dira qu'un segment est en conflit avec une arête s'il coupe sa zone d'influence.

On aboutit alors au résultat.

Théorème : *Le graphe d'influence permet de maintenir le diagramme de Voronoï de n segments du plan en temps moyen* $O(\log n)$ *par insertion si l'insertion des segments est randomisée.*

Le problème du chien et des postiers

Il y a deux manières de dynamiser un problème géométrique, la première que nous avons abordée jusqu'ici consiste à modifier l'ensemble des données par des insertions et des suppressions, c'est-à-dire d'une manière discrête, mais on peut également envisager des modifications continues de l'ensemble des données. Par exemple pour le diagramme de Voronoï, les sites sont toujours des points du plan, mais ces sites se déplacent, sur une trajectoire prédéterminée, en fonction du temps. Cette notion dynamique continue peut très bien être combinée avec

FIG. I.9 – *Exemple de diagramme de Voronoï de segments*

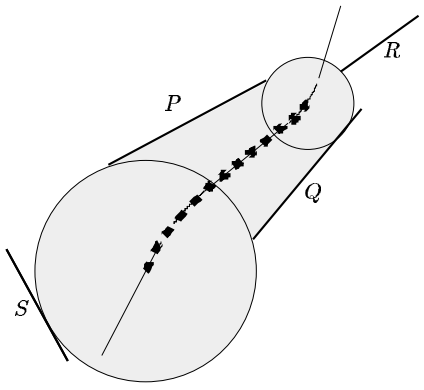

FIG. I.10 – *Definition des regions pour Voronoï de segments.*

la notion discrête précédente: on peut vouloir ajouter ou supprimer des sites et leur trajectoire et dynamiser ainsi un algorithme déjà dynamique!

La mécanique randomisée maintenant bien huilée peut également s'appliquer à ce cas, et permet même de résoudre un nouveau type de requête. Imaginons que les sites soient des postiers se déplaçant dans le plan à vitesse constante et que la requête soit la suivante: un chien, pouvant courir à vitesse v se réveille au point (x, y) au temps t et cherche à mordre un postier le plus vite possible. Attention, le postier recherché n'est pas nécessairement le plus proche du point (x, y) au temps t, il est possible que le postier le plus proche s'éloigne et qu'un autre postier qui se rapproche puisse être atteint plus rapidement.

On peut dynamiser l'algorithme en autorisant l'insertion et la suppression de postiers avec la technique de l'histoire des insertions et des suppressions [DG93].

Théorème: *Le diagramme de Voronoï d'un ensemble $\mathcal{S}$ de n sites se déplaçant chacun à vitesse constante dans le plan peut être maintenu en temps moyen $O(f_{\mathcal{S}}(n)/n + \log n)$ par insertion ou suppression, si $f_{\mathcal{S}}(r)$ désigne la taille moyenne du diagramme pour un échantillon de r sites de $\mathcal{S}$ et avec les hypothèses randomisées habituelles. Les requêtes du type «Quel est le site le plus proche au temps t d'un point donné» ainsi que les requêtes du type chien peuvent être résolues en temps $O(\log^3 n)$ à condition que le chien soit plus rapide que tous les postiers.*

Diagramme de Voronoï et arbre couvrant minimal

L'arbre couvrant minimal (euclidien) de n sites dans le plan (ou dans l'espace d'ailleurs) est l'arbre ayant comme nœuds les sites, comme arêtes des segments joignant ces sites et telles que la longueur totale de toutes les arêtes soit minimale. Il est prouvé que les arêtes de ce graphe sont un sous-ensemble de la triangulation de Delaunay et la méthode la plus efficace pour calculer l'arbre couvrant minimal consiste à calculer d'abord la triangulation de Delaunay puis d'en extraire l'arbre recherché en temps linéaire [PS85].

Le calcul, tant de la triangulation de Delaunay que de l'arbre minimal requiert au moins un temps $\Omega(n \log n)$. Un des problèmes, relativement théorique, que l'on peut se poser est de connaître la complexité de la transformation entre ces deux structures. Eh bien, la méthodes d'accélération par mélange des graphes de conflits et d'influence peut s'appliquer à la triangulation de Delaunay pour peu que l'on connaisse n'importe quel sous arbre couvrant cette triangulation, en particulier si on connaît l'arbre minimal. La connaissance de cet arbre est utilisée aux instants clé pour calculer le graphe de conflits en temps moyen linéaire. On aboutit donc au résultat suivant [Dev92a]:

Théorème: *Le diagramme de Voronoï (la triangulation de Delaunay) de n sites dont on connaît l'arbre couvrant de longueur minimale peut être calculé en $O(n \log^\star n)$.*

Squelette d'un polygone simple

La méthode d'accélération peut également s'appliquer au cas du diagramme de Voronoï de segments. On peut ici exploiter des incidences entre les segments pour arriver à calculer le graphe de conflits lors des instants clé [Dev92a]:

Théorème: *Le squelette d'un polygone simple de taille n peut être calculé en $O(n \log^\star n)$.*

Chapitre II

Algorithmes sur les sphères

A ses débuts, la géométrie algorithmique commença par faire intervenir l'objet géométrique le plus simple: le point. Puis l'on passa à d'autres objets tels que droite, segment, triangle ou plus généralement simplexe en dimension d. Lorsque l'on cherche à utiliser des objets non linéaires, tout se complique et peu de résultats sont connus à l'heure actuelle. Parmi les objets non linéaires, la sphère présente plusieurs avantages. Tout d'abord celui de la simplicité, c'est parmi les objets courbes un de ceux qui a le plus petit nombre de paramètres, ensuite celui d'une certaine utilité pratique pour modéliser des objets plus complexes, par exemple en chimie où une approximation des molécules peut être réalisées avec des sphères. Une autre raison essentielle de l'intérêt des sphères est leur correspondance étroite avec la notion de distance, tant pour la distance euclidienne que pour quelques autres notions plus exotiques tels que métrique hyperbolique ou associée à des puissances (métrique de Laguerre).

Une des origines de ces travaux sur les sphères se trouve dans l'article de Jean-Daniel Boissonnat calculant la triangulation de Delaunay de points situés dans deux plans parallèles [Boi88]. En cherchant à généraliser cet algorithme au cas de plans non parallèles, nous nous sommes aperçus que les cercles concentriques étaient naturellement transformés en faisceaux de cercles, ce qui nous a entraîné, avec l'aide d'André Cérézo, vers la géométrie hyperbolique. Après avoir généralisé l'algorithme au cas de deux ou plusieurs plans non parallèles [BCDT91] nous avons jugé utile de formaliser un peu certaines transformations géométriques faisant un usage intensif des faisceaux de cercles sous le concept d'espace des sphères [DMT92b]. Le principal résultat algorithmique obtenu est un algorithme dont la complexité dépend de la taille de la sortie t pour le calcul de la triangulation de Delaunay de n points répartis dans un petit nombre k de plans; le résultat peut être calculé en un temps $O(tk \log n)$. Cet algorithme se généralise au cas de diagrammes de puissance pour des sphères dont les centres appartiennent à k plans, ou à des diagrammes à poids multiplicatifs si les poids prennent seulement k valeurs distinctes.

Ces techniques trouvent leur application en imagerie médicale ou en microscopie où des mesures fournissent des coupes de l'objet à partir desquelles on cherche à reconstruire un modèle tridimensionnel. La triangulation de Delaunay et les notions de proximité qu'elle contient se révèlent efficaces pour cette opération [Boi85, Gei93].

D'autres applications, médicales elles aussi, nous on conduit vers une autre

utilisation des sphères. Après le niveau macroscopique de l'anatomie, nous allons passer à celui, microscopique, des molécules médicamenteuses. Afin d'étudier les interactions de diverses molécules entre elles, il est nécessaire de modéliser leur surface «active».

Nous nous sommes d'abord intéressés au calcul de l'enveloppe convexe d'un ensemble de sphères [BCD+92], celui-ci fait appel à une dualité légèrement différente entre sphères en dimension d et points en dimension $d+1$ et soulève des problèmes combinatoires non triviaux.

Puis, plus proche de la chimie, nous avons proposé un algorithme optimal de calcul de la surface de Connolly d'une molécule [BDD92].

II.1 L'espace des sphères

L'espace des sphères que nous avons introduit [DMT92b] n'est pas à proprement parler une notion originale, mais son utilisation dans ce contexte permet d'unifier et de justifier de manière élégante un certain nombre de résultats déjà existants.

Afin de simplifier la présentation, nous ne parlerons ici que de l'espace des cercles, mais la plupart des résultats se généralisent en dimensions supérieures.

L'idée de base consiste à associer au cercle C du plan, d'équation $x^2+y^2-2\varphi x-2\psi y+\chi=0$ le point C^*de coordonnées (φ,ψ,χ) dans un espace à trois dimensions : l'espace des cercles. Cette transformation présente l'intérêt de mettre en correspondance les faisceaux de cercles du plan et les droites dans l'espace des cercles. Il est également nécessaire de représenter les points et les droites du plan dans l'espace des cercles, mais ceux ci trouvent une interprétation naturelle. Les points peuvent être considérés comme des cercles de rayon nul. Dans l'espace des sphères, l'ensemble des cercles de rayon nul est le paraboloïde de révolution Π d'équation $\varphi^2+\psi^2=\chi$ qui apparaît dans de nombreuses transformations géométriques classiques. Les droites peuvent être souvent interprétées comme des cercles de rayon infini, c'est-à-dire des points à l'infini dans l'espace des sphères.

Avec ces définitions, on démontre très simplement, pratiquement sans calcul, de nombreuses propriétés en utilisant quelques résultats classiques de géométrie. Tout d'abord, on introduit la polarité par rapport à Π, en effet deux cercles du plan sont orthogonaux si et seulement si leurs représentants dans l'espace des sphères sont conjugués par rapport à Π ; on en déduit facilement que le fait qu'un point p soit à l'intérieur ou à l'extérieur d'un cercle C revient à savoir de quel côté du plan polaire de p^* par rapport à Π se trouve C^* (puisque $p^*\in\Pi$ le plan polaire n'est autre que le plan tangent).

On aboutit rapidement au lien avec le diagramme de Voronoï. Etant donné un ensemble de sites $\{p_i\}_{i\in\{1,\ldots,n\}}$, un cercle C ne contient pas p_{i_0} si et seulement si C^* est au dessus du plan polaire de $p^*_{i_0}$, donc C^* ne contient aucun des p_i s'il est au dessus de l'enveloppe supérieure des plans polaires des p^*_i. C est donc un cercle vide passant par trois sites si C^* est un sommet de cette enveloppe supérieure. Nous retrouvons ici le résultat bien connu que le diagramme de Voronoï est la projection de l'enveloppe supérieure de plans tangents au paraboloïde.

II.2 Résultats de dualité en tout genres

L'espace des cercles permet la description et la justification d'un grand nombre de transformations entre divers problèmes géométriques. La plupart de ces transformations étaient déjà connues, c'est seulement leur justification et également leur composition qui est facilitée par l'espace des cercles. Nous nous bornons ici à énumérer ces résultats donnés ailleurs avec plus de détails [DMT92b].

- Le diagramme de Voronoï de n sites est la projection de l'enveloppe supérieure de n plans (ces plans sont les plans polaires des points représentant les sites dans l'espace des cercles).
- Le diagramme de puissance de n cercles est la projection de l'enveloppe supérieure de n plans (ces plans sont les plans polaires des points représentant les cercles dans l'espace des cercles).
- Le diagramme de Voronoï à poids multiplicatifs de n sites est la projection de l'enveloppe supérieure de n paraboloïdes (ces paraboloïdes sont les images des plans polaires des sites par une transformation qui dépend du poids).
- Le diagramme de Voronoï d'ordre k de n sites est la projection de l'enveloppe supérieure de $\left(\begin{array}{c} n \\ k \end{array} \right)$ plans (ces plans sont les plans polaires des barycentres d'ordre k des sites dans l'espace des cercles).
- Le diagramme de Voronoï de n sites pour la métrique hyperbolique est obtenue en composant quelques projections à partir de l'enveloppe supérieure de n plans (ces plans sont les plans polaires des points représentant les sites dans l'espace des cercles).

Le contexte permettant d'énoncer ces résultats permet d'obtenir facilement des généralisation à des cas plus exotiques, par exemple sur les diagrammes de Voronoï de droites, les diagrammes hyperboliques d'ordre k ou les diagrammes de puissance à poids multiplicatifs.

II.3 Triangulation de points contraints

En fait l'origine de ce travail assez général sur les problèmes de dualité impliquant les cercles était relativement pratique. Jean-Daniel Boissonnat avait proposé [Boi88] un algorithme permettant de calculer la triangulation de Delaunay tridimensionnelle de points contenus dans deux plans parallèles. La motivation de cet algorithme était les problèmes de reconstruction tridimensionnelle en imagerie médicale. Un scanner par exemple fournit des données situées dans des coupes parallèles. Il semblait intéressant de chercher à exploiter cette particularité dans l'algorithme plutôt que d'utiliser un algorithme tridimensionnel général.

Notre travail a consisté à généraliser au cas de plans non parallèles et aussi au cas d'un petit nombre de plans. Toujours dans le contexte de l'imagerie médicale, ce cas peut survenir avec d'autres types de techniques de mesure telles que l'échographie.

II.3.1 Des points dans deux plans parallèles

Jean-Daniel Boissonnat avait proposé en 1988 un algorithme optimal pour calculer la triangulation de Delaunay de points appartenant à deux plans parallèles de complexité $O(n \log n + t)$ où n est le nombre de points dans les deux plans et t la taille du résultat (le nombre de tétraèdres) [Boi88]. La recherche d'un tel algorithme dans le cas général de points en 3D, optimal en fonction de la taille de la sortie est toujours à l'heure actuelle l'un des problèmes ouverts importants de la géométrie algorithmique. Nous allons maintenant décrire l'algorithme.

Soit P et Q deux plans parallèles contenant n sites. Nous rappellerons qu'en trois dimensions, les tétraèdres de la triangulation de Delaunay sont caractérisés par le fait que leur sphère circonscrite est vide (ne contient aucun site); de même les triangles de Delaunay sont caractérisés par l'existence de sphères vides passant par les trois sommets du triangle et les arêtes par l'existence de sphères vides passant par les deux sommets de l'arête.

On commence par calculer la triangulation de Delaunay des points de P, c'est-à-dire une triangulation de Delaunay dans un plan. Maintenant soit T un triangle de cette triangulation, T est un triangle de la triangulation 3D. Pour le prouver, il suffit de trouver une sphère vide passant par les sommets de T. Soit S une sphère passant par les sommets de T, $S \cap P$ est le cercle circonscrit à T dans P, il ne contient donc aucun site de P puisque T est de Delaunay dans P; S ne peut donc contenir que des sites de Q, il est maintenant facile de voir que parmi les sphères possibles il y en a une qui est tangente à Q en un point q_T et qui ne contient donc aucun site de Q, donc T est une face de la triangulation de Delaunay 3D. Pour trouver le dernier sommet du tétraèdre adjacent à T, il suffit de faire grossir la sphère tangente à Q que nous venons d'évoquer. Le premier site de Q touché par cette sphère grossissante sera le quatrième sommet du tétraèdre. L'intersection de cette sphère mobile avec Q décrit en fait un ensemble de cercles concentriques de centre q_T, c'est-à-dire que ce premier point rencontré est le site de Q le plus proche de q_T. D'autre part q_T n'est rien d'autre que la projection orthogonale du centre du cercle circonscrit à T, qui est un sommet du diagramme de Voronoï des sites de P. En résumé, on trouve pour chaque triangle de la triangulation de Delaunay de P le site de Q qui permet de le compléter en un tétraèdre en localisant dans le diagramme de Voronoï de Q la projection de tous les sommets du diagramme de Voronoï de P.

On montre ensuite par un raisonnement similaire que lorsque l'on projette le diagramme de P sur celui de Q, les intersections entre deux arêtes correspondent également à des tétraèdres de Delaunay, cette fois ci ayant deux sommets dans Q et deux sommets dans P. L'algorithme est alors le suivant : on calcule d'abord les diagrammes de Voronoï séparément dans les deux plans. Ces diagrammes de Voronoï sont obtenus comme projection verticale d'une enveloppe supérieure de plans dans l'espace des sphères. Ils sont projetés l'un sur l'autre et cette superposition nous dit comment relier dans l'espace les sites appartenant aux deux plans. Cette superposition peut être faite en temps optimal, c'est à dire en temps linéaire en fonction de la taille du résultat en utilisant la technique du balayage topologique.

Théorème : *La triangulation de Delaunay de n sites de l'espace appartenant à deux plans parallèles peut être calculée en temps $\Theta(n \log n + t)$ où t désigne la*

taille de la triangulation obtenue.

II.3.2 Des points dans deux plans non parallèles

La généralisation à deux plans non parallèles n'est pas si délicate, il suffit d'avoir les bons outils. On a dit que l'on faisait grossir la sphère passant par T et que son intersection avec Q était un ensemble de cercles concentriques, c'est-à-dire un faisceau de cercles concentriques qui devient dans l'espace des sphères une droite verticale.

Si P et Q ne sont pas parallèles, on peut appliquer le même procédé, la différence est que l'intersection des sphères avec Q n'est plus un faisceau concentrique, mais un faisceau d'axe radical la droite $P \cap Q$, faisceau qui devient dans l'espace des sphères une droite non plus verticale, mais de direction $(P \cap Q)^*$ (c'est le point à l'infini, i.e. la direction de droite représentant la droite $P \cap Q$ prise en tant que cercle de rayon infini). Tout se passe donc à peu près comme précédemment à condition de ne plus calculer les diagrammes de Voronoï 2D, c'est-à-dire en projetant l'enveloppe supérieure des plans dans la direction verticale, mais dans la direction $(P \cap Q)^*$ et en superposant les deux diagrammes obtenus ainsi pour P et Q.

Nous pouvons aussi interpréter cela légèrement différemment, pour un triangle T on cherche à localiser q_T, le point de tangence de la sphère passant par les sommets de T et tangente à Q, dans le diagramme de Voronoï de Q pour une métrique un peu spéciale dans laquelle les faisceaux de cercles concentriques sont remplacés par les faisceaux de cercles d'axe radical $P \cap Q$. Cette métrique n'est autre que la métrique hyperbolique du demi-plan de Poincaré constituée par la moitié de Q délimitée par $P \cap Q$ [BCDT91]. En fait pour calculer la triangulation de Delaunay entre deux plans non parallèles, il faut donc calculer la superposition des diagrammes de Voronoï dans les deux plans pour une métrique hyperbolique.

Théorème : *La triangulation de Delaunay de n sites de l'espace appartenant à deux plans peut être calculée en temps $\Theta(n \log n + t)$ où t désigne la taille de la triangulation obtenue.*

II.3.3 Des points dans k plans

L'algorithme ci dessus peut être légèrement modifié : plutôt que de localiser simultanément tous les sommets du diagramme de Voronoï des points de P dans le diagramme de Voronoï des points de Q en superposant les deux graphes par la technique du balayage topologique, on pourrait utiliser une structure de localisation sur le diagramme de Voronoï des points de Q et localiser chaque sommet du diagramme de Voronoï des points de P en temps logarithmique. C'est à peu de chose près ce que l'on va faire si les points n'appartiennent plus à deux plans mais à k plans différents. k est supposé être un nombre relativement petit.

La triangulation de Delaunay va être construite de proche en proche. Etant donné un tétraèdre de Delaunay *abcd*, on va chercher un de ses voisins encore inconnu, par exemple *abce*. Le problème est donc de trouver le point e, comme celui-ci appartient à l'un des k plans, on va chercher e en faisant grossir une sphère et en regardant le faisceau de cercles intersection avec chaque plan. Dans chaque plan on obtiendra un site candidat en faisant une localisation dans un

certain diagramme de Voronoï hyperbolique en temps logarithmique, puis on peut trouver le bon e parmi les k candidats possibles.

Chaque tétraèdre de la triangulation de Delaunay tridimensionnelle est donc construit en temps $O(k \log n)$ [BCDT91].

Théorème : *La triangulation de Delaunay de n sites de l'espace appartenant à k plans peut être calculée en temps $O(kt \log n)$ où t désigne la taille de la triangulation obtenue.*

Ce résultat est également généralisable au cas où les sites ne sont plus contraints à appartenir à k plans mais à k sphères. On peut aussi passer aux dimensions supérieures, si on cherche à calculer la triangulation de Delaunay en dimension d avec des points contraints à appartenir à un petit nombre de sous-espaces affines de dimension inférieure. Ce résultat s'applique également à des diagrammes de puissance si les centres des sphères sont contraints à appartenir à des sous-espaces affines.

II.4 Enveloppe convexe

Pour résoudre un autre type de problème, nous avons été amenés à introduire une autre représentation des cercles en trois dimensions.

Le problème est celui du calcul de l'enveloppe convexe d'un ensemble de n sphères (de dimension $d-1$) en dimension d. Nous exposerons l'algorithme sur le cas des cercles du plan. Le plan est plongé dans un espace de dimension trois, et chaque cercle est complété en un cône de révolution d'angle au sommet α, α étant une constante identique pour tous les cônes. L'enveloppe convexe des cercles dans le plan est l'intersection du plan avec l'enveloppe convexe des cônes. L'enveloppe convexe des cônes est constituée d'une partie supérieure correspondant à l'enveloppe convexe des sommets des cônes, de portions de cônes et de faces tangentes à deux cônes et passant par une arête de l'enveloppe convexe des sommets des cônes.

Comme tout est relié à l'enveloppe convexe des sommets des cônes, on peut en fait calculer cette enveloppe convexe de points en dimension trois puis repérer les portions de cette enveloppe convexe qui correspondent bien à des éléments de l'enveloppe convexe des sphères.

La généralisation en dimension d procède de même en calculant une enveloppe convexe de points en dimension $d+1$ et par la sélection des portions intéressantes de cette enveloppe convexe. La complexité de l'enveloppe convexe de points en dimension p est $O(n^{\lfloor \frac{p}{2} \rfloor})$. Si d est pair, $n^{\lfloor \frac{d}{2} \rfloor} = n^{\lfloor \frac{d+1}{2} \rfloor}$, l'enveloppe convexe des sphères a une complexité identique à celle des points. Par contre si d est impair, $n^{\lfloor \frac{d}{2} \rfloor} < n^{\lfloor \frac{d+1}{2} \rfloor}$, il y a un facteur n de différence. En dimension trois, on peut prouver que ce facteur est effectif, c'est-à-dire que l'on connaît des exemples dans lesquels l'enveloppe convexe de sphères est quadratique alors que l'enveloppe convexe de points est linéaire. En dimension impaire supérieure à cinq, l'optimalité de la borne n'est que conjecturée [BCD+92].

Théorème : *L'enveloppe convexe de n sphères en dimension d peut être calculée en temps $O(n \log n + n^{\lceil \frac{d}{2} \rceil})$ ce qui est optimal en dimension paire et en dimension 3.*

Cet algorithme se généralise au cas d'objets homothétiques. Si on a n copies homothétiques d'un objet convexe de complexité k en dimension d, l'enveloppe

convexe de ces objets peut être calculée en temps $O(kn \log n + kn^{\lceil \frac{d}{2} \rceil})$. Cette généralisation correspond au cas de la somme de Minkowski de deux convexes.

II.5 Surface de Connolly

La surface de Connolly est utilisée en chimie pour la modélisation des surfaces d'interaction entre molécules, notamment lors de la recherche de nouvelles substances thérapeutiques. La molécule est représentée par un certain nombre de sphères correspondant aux différents atomes qui la composent et on cherche alors l'ensemble des positions qu'une sphère de rayon ρ donnée, dite sphère test, peut occuper dans l'espace sans pénétrer la molécule. La surface de Connolly est l'enveloppe de ces sphères tests. Pour $\rho = 0$ on trouve seulement l'union des sphères de la molécule, et pour $\rho = \infty$ on retrouve l'enveloppe convexe.

Cette surface peut être calculée en utilisant divers outils de géométrie algorithmique tels que unions de sphères et enveloppes convexes [BDD92].

Chapitre III

Déplacements de robots

Notre application privilégiée est historiquement la robotique et principalement le problème de la planification de trajectoires.

Succédant à d'autres travaux, notamment de Bernard Faverjon et Pierre Tournassoud, où l'efficacité pratique avait nécessité une bonne dose d'empirisme, la thèse de Francis Avnaim [Avn89] démontrait que les préoccupations théoriques n'était pas incompatibles avec une programmation effective et efficace. Francis Avnaim effectuait une étude exhaustive du problème de la planification de trajectoires pour un polygone en translation dans le plan, puis en translation et rotation et enfin pour d'autres robots à trois degrés de liberté.

Le problème de la planification du mouvement simultané de plusieurs objets en translation était également abordé, et ce sujet a particulièrement retenu mon attention.

Un de nos objectifs [Dev93] était d'utiliser des configurations particulières du systèmes : les doubles contacts afin de réduire la combinatoire de la recherche d'une solution au problème du placement. Il est possible de prouver que ces configurations existent toujours dans certains cas particuliers : dans le cas de deux robots, de trois robots convexes ou de trois robot dans un environnement rectangulaire. On en déduit un algorithme en $O(n^2)$ (*resp.* $O(n^3)$) pour détecter un tel placement dans le cas de deux (*resp.* trois) polygones convexes en translation dans un environnement polygonal quelconque. Par contre dans le cas de trois robots quelconques dans un environnement quelconque ou de plus de trois robots, il est possible que de telles configurations n'existent pas, et on donne des exemples de tels systèmes.

Dans le domaine de la planification de trajectoires, nous nous sommes également intéressés à certains cas de robots un peu exotiques ! En particulier le robot à pattes.

Une instance simple du problème consiste à modéliser le robot par un corps ponctuel auquel sont attachées des pattes ayant un rayon d'action fixé R, nous avons appelé ce robot le robot araignée [BDDP95]. Nous supposons que dans un cas difficile, ce robot araignée peut poser ses pieds qu'en un certain nombre de points du plan. Une première solution peut être fournie par le diagramme de Voronoï d'ordre k si le robot a k pattes, malheureusement cette solution ne permet pas une prise en compte correcte des problèmes de stabilité. Une seconde solution, plus complexe, conduit à calculer l'arrangement des cercles centrés sur les points d'appui (ce que l'on peut faire par les méthodes randomisées décrites

plus haut). Si le corps du robot se trouve dans une région de cet arrangement, l'ensemble des points d'appui accessibles est alors parfaitement déterminé et la contrainte de stabilité est intégrée en calculant l'intersection de cette région avec l'enveloppe convexe des points d'appui accessibles. L'algorithme basé sur ce principe fait intervenir des résultats et structures de données complexes pour obtenir une complexité optimale: l'espace libre du robot araignée est de complexité $\Theta(n^2)$ dans le cas le pire, et il peut être calculé en temps $O(n^2 \log n)$. De plus cet algorithme, à défaut d'être sensible à la taille de la sortie, est sensible à la taille de l'arrangement des cercles ce qui permet d'abaisser sa complexité dans un bon nombre de cas pratiques.

Lorsque l'on cherche à généraliser à des robots à pattes plus complexes de nouveaux problèmes apparaissent. Si l'on suppose que les pattes du robot ne sont plus attachées à un corps ponctuel mais à un corps polygonal le problème devient nettement plus difficile. La difficulté majeure est que ce robot n'est plus invariant par rotation et, par conséquent, il est nécessaire d'ajouter une troisième dimension à l'espace des configurations, mais la généralisation du problème de la stabilité est également non triviale. Pour ce robot l'enveloppe convexe n'est plus satisfaisante et il est nécessaire de la remplacer par le problème géométrique suivant, que nous avons résolu avec une complexité optimale: étant donné un ensemble de points coloriés, trouver l'union des triangles ayant pour sommets trois de ces points de couleurs différentes [BDP91].

III.1 Déplacement de plusieurs robots

On étudie ici le déplacement, ou plutôt le placement de plusieurs robots polygonaux se déplaçant en translation dans un environnement décrit de manière polygonale dans le plan. Ce problème a notamment des applications dans l'industrie de la découpe où il est important de minimiser les chutes lorsque l'on veut obtenir plusieurs pièces dans un morceau de matière première donné. Nous ne traitons ici que le problème en translation qui correspond à des matériaux «orientés», ayant une trame, tel que le tissu.

Etant donné un système de un ou plusieurs robots se déplaçant dans un environnement encombré d'obstacles, on appelle configuration du système une position du robot, ou des robots. Une configuration est dite libre si les robots ne rencontrent ni l'environnement ni ne se rencontrent les uns les autres. Dans la suite les robots ainsi que l'environnement seront supposés polygonaux. L'espace des configurations d'un système de q robots en translation est un espace à $2q$ dimensions, et l'ensemble des configurations libres $\mathcal{L}$ est un polyèdre de cet espace. Lorsque deux robots se touchent (ou un robot touche l'environnement) on parle de configuration de contact. Ces configurations de contact peuvent être classifiées. Dire qu'un sommet d'un objet doit être en contact avec une arête d'un autre objet, c'est imposer une contrainte linéaire sur les paramètres du système; le point représentant la configuration de contact doit se trouver dans un certain hyperplan. On en déduit facilement que, sous certaines hypothèses de non dégénérescence, les facettes (faces de dimension $2q-1$) de l'espace libre correspondent à un contact de type sommet arête et que les faces de dimension k correspondent à $2q-k$ contacts entre des sommets et des arêtes. En particulier les sommets de l'espace libre correspondent à $2q$ contacts qui sont donc obtenus comme intersection de $2q$ hyperplans. Ces paires, sommet d'un objet et arête

d'un autre objet (ou de l'environnement), définissent donc les contacts, il peut arriver que la même paire d'objets soit impliquée deux fois, mais avec des arêtes et sommets différents, on parle alors de double contact.

Le calcul complet de l'espace libre est difficile et a une complexité exponentielle dans la dimension de l'espace des configurations c'est à dire $2q$. L'utilisation de doubles contacts aurait pu réduire un peu la complexité du problème. Malheureusement, en général, l'existence de tels double contacts ne peut être prouvée sauf dans certains cas très particuliers. La preuve de l'existence de doubles contacts dans ces cas là permet effectivement d'obtenir quelques gains de complexité, assez faibles il faut le reconnaître [Dev93]. Un des ingrédients essentiels des démonstrations est le fait qu'un sommet de l'espace libre qui n'est pas un double contact est nécessairement convexe.

Existence de doubles contacts

- Dans un système de deux objets polygonaux dans un environnement polygonal, il existe dans chaque composante connexe de l'espace libre un sommet correspondant à un double contact (ce résultat est très simple, le problème devient difficile à partir de trois objets).
- Dans un système de deux objets convexes dans un environnement polygonal, il existe dans chaque composante connexe de l'espace libre un sommet où chacun des deux objets est en double contact avec l'environnement.
- Dans un système de trois objets convexes dans un environnement polygonal, il existe dans chaque composante connexe de l'espace libre un sommet où l'un des objets est en double contact avec l'environnement.
- Dans un système de trois objets polygonaux à l'intérieur d'un environnement rectangulaire, il existe dans chaque composante connexe de l'espace libre un sommet correspondant à un double contact.

Non existence de double contact

- Dans un système de trois objets polygonaux dans un environnement polygonal, si une composante connexe de l'espace libre ne contient aucun double contact, alors elle est convexe et a au moins 64 sommets. Un tel système existe (figure III.1).
- Il existe un système de quatre objets convexes à l'intérieur d'un environnement rectangulaire, tel que l'espace libre ne contienne aucun double contact (et soit non vide, voir la figure III.2).

Résultats algorithmiques

Dans le cas d'objets convexes de taille m et d'un environnement de taille $n > m$, on obtient quelques gains sur la complexité des algorithmes.

- Dans le cas de deux objets, les doubles contacts peuvent être calculés en temps $O(n^2m^2 \log m)$ alors que le calcul de l'espace libre nécessite un temps de calcul $O(n^2m^2 \log^2 nm)$.

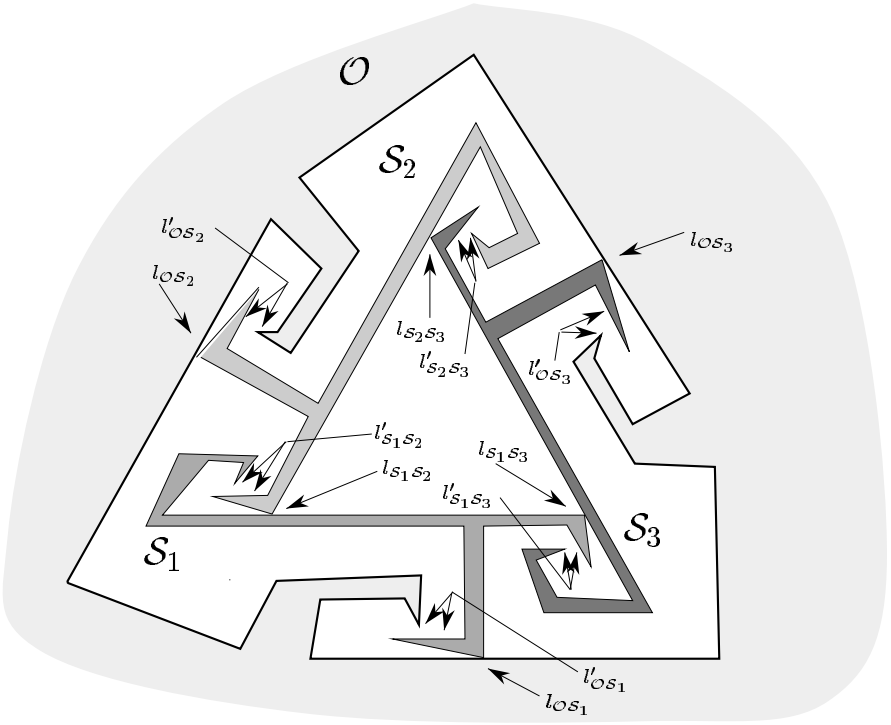

FIG. III.1 – *Trois objets sans double-contact.*

– Dans le cas de trois objets, les doubles contacts peuvent être calculés en temps $O(n^3m^3 \log m)$ alors que le calcul de l'espace libre nécessite un temps de calcul $O(n^3m^3 \log^2 nm + n^3m^5 \log m)$.

III.2 Robot à pattes

Bien que les robots à pattes soit déjà présents dans le monde des roboticiens, et étudiés pour les aspects mécaniques ou de contrôle, très peu de choses ont été faites dans le domaine de la planification de trajectoires. Nous nous plaçons ici dans un environnement difficile ou le robot à peu de possibilités pour mettre ses pattes. Le robot est alors soumis à deux contraintes, une contrainte d'accessibilité, le corps du robot pourra se trouver à un certain emplacement s'il y a des points d'appui autorisés accessibles par ses pattes ; et une contrainte de stabilité, il faut que le centre de gravité du robot soit situé à l'intérieur du polygone d'appui.

Nous ferons plusieurs types d'hypothèses simplificatrices, celles-ci pourront porter sur la modélisation du robot ou de son environnement. Nous supposerons tout d'abord que l'ensemble des points d'appui autorisés est un ensemble fini de points du plan. Ensuite, nous verrons qu'il sera possible de généraliser à des zones d'appui polygonales. Il est également nécessaire de faire des hypothèses sur le robot, le robot sera simplifié en une araignée, c'est-à-dire que nous supposerons que toutes les pattes sont de même longueur et sont attachées à un

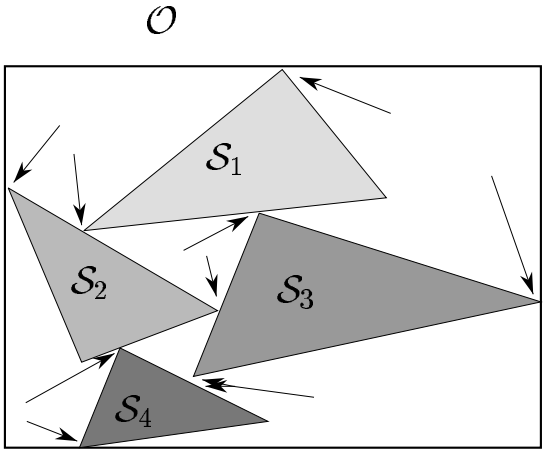

FIG. III.2 – *Quatre objets sans double-contact.*

corps ponctuel.

Comme dans le cas des robots polygonaux, on cherchera à calculer l'espace libre du robot, c'est-à-dire l'ensemble des configurations accessibles et stables. On verra ensuite comment utiliser cet espace libre pour planifier des trajectoires. Et enfin on étudiera le problème de la stabilité pour un robot non plus à corps ponctuel, mais polygonal.

III.2.1 Le robot araignée

Si on note R la longueur des pattes du robot, et G une configuration (i.e. une position du corps du robot), les points d'appui accessibles sont ceux à une distance de G plus petite que R, et la configuration est admissible si G appartient à l'enveloppe convexe des points d'appui accessibles. Une configuration peut être sur le bord de l'espace libre, soit pour des raisons de stabilité —localement le bord est alors un segment droite (une portion d'enveloppe convexe)— soit pour des raisons d'accessibilité —le bord est alors un arc de cercle de rayon R centré sur un point d'appui (celui qui est en limite d'accessibilité justement)—

Une première méthode de calcul est alors assez simple à exprimer. On calcule l'arrangement des cercles centrés sur les points d'appui et de rayon R. Dans une région Γ de cet arrangement l'ensemble des points d'appui accessibles reste le même: S_Γ. On peut donc calculer $\Gamma \cap CH(S_\Gamma)$ qui est la partie de l'espace libre à l'intérieur de Γ (CH désigne l'enveloppe convexe).

Cette méthode doit être modifiée si on veut réaliser un algorithme de bonne complexité. Tout d'abord, les bords rectilignes de l'espace libre peuvent couper l'arrangement des cercles et ces points d'intersection ne sont pas intéressants, il faut donc éviter de les calculer; ensuite calculer l'enveloppe convexe $CH(S_\Gamma)$ pour chaque région Γ de l'arrangement est excessivement coûteux. La solution consiste à utiliser un algorithme d'enveloppe convexe dynamique hors ligne, qui va permettre de calculer uniquement la différence entre $CH(S_{\Gamma_i})$ et $CH(S_{\Gamma_j})$ pour toutes les paires de régions Γ_i et Γ_j adjacentes dans l'arrangement des cercles. Cela permet d'une part de ne pas recalculer l'enveloppe convexe pour chaque cellule de l'arrangement, et d'autre part de ne connaître que la partie ju-

dicieuse de l'enveloppe convexe, celle qui va effectivement produire des sommets du bord de l'espace libre et d'éliminer ainsi les sommets inintéressants annoncés plus haut.

On aboutit alors au résultat suivant [BDDP95]:

Théorème: *L'espace libre du robot araignée autorisé à s'appuyer sur n points du plan peut être calculé en temps $O(n^2 \log n)$.*

Ce résultat est presque optimal puisque dans le cas le pire l'espace libre a une complexité $\Omega(n^2)$.

III.2.2 Planification de trajectoires pour l'araignée

Connaître l'espace libre n'est pas tout, encore faut-il pouvoir s'en servir, et dans le cas du robot à pattes il ne suffit pas de savoir qu'une configuration est stable et accessible, encore faut-il savoir où mettre les pattes du robot!

Connaissant l'espace libre, il est facile de déterminer un chemin admissible entre deux configurations de cet espace libre, mais si maintenant le robot veut suivre ce chemin il faudra décider quand il doit bouger ses pattes. Nous avons utilisé la méthode suivante [BDDP95]: tant que le robot peut déplacer son corps le long du chemin prédéterminé sans déplacer ses pattes, il le fait, et dès qu'il arrive en limite d'accessibilité ou de stabilité on détermine l'ensemble des points accessibles depuis la configuration courante, leur enveloppe convexe, et le nouveau placement des pattes réalisable qui permettra d'aller le plus loin possible sur le chemin. Il est clair qu'un tel algorithme permet de minimiser le nombre de changements de pattes le long du chemin prévu. Par contre on ne sait rien sur la manière de choisir ce chemin pour minimiser les changements de position des pattes.

III.2.3 De l'araignée à l'hémi-disque

La méthode précédente peut difficilement être généralisée au cas de points d'appui non ponctuels. Si les zones d'appui autorisées ne sont pas des points mais des segments, alors l'ensemble des points d'appui accessibles change de façon continue lors des déplacements du robot contrairement au cas précédent où il ne changeait que lorsque l'on franchissait une arête de l'arrangement des cercles.

Nous avons donc développé une méthode différente qui peut être plus facilement généralisée au cas de zone d'appui non ponctuelle. Cette méthode est basée sur la remarque suivante:

> Une configuration G du robot araignée n'est pas admissible si et seulement si, il existe un hémi-disque de centre G de rayon R ne contenant aucun point d'appui.

Nous avons en fait introduit un problème complémentaire qui n'est autre que le problème du placement d'un hémi-disque de rayon R pouvant se déplacer en translation et rotation. L'espace libre du robot araignée peut alors être obtenu en calculant l'espace libre du hémi-disque (en trois dimensions, deux de position et une d'orientation) en passant au complémentaire et en projetant le résultat dans le plan.

Théorème: *L'espace libre du robot araignée autorisé à s'appuyer sur n points du plan peut être calculé en temps $O(n^2 \alpha(n) \log n)$.*

Cette complexité est légèrement moins bonne que celle obtenue par la première méthode, mais le facteur $\alpha(n)$ est peu important[1], on a évité le recours à la technique assez lourde des enveloppes convexes hors ligne et l'algorithme se généralise aux zones d'appui polygonales.

III.2.4 Le problème de la stabilité

On étudie maintenant un robot dont le corps n'est plus ponctuel mais polygonal, c'est-à-dire que les pattes ne sont plus toutes attachées au même point du robot mais à des points différents. Nous supposons encore que l'environnement est constitué de points d'appui discrets. Etant donnée une certaine configuration du robot, les point d'appui accessibles dépendent maintenant de la patte que l'on choisit sur le robot. Nous colorions chaque patte d'une couleur différente, et nous colorions les points avec la couleur de la ou des pattes qui peuvent les atteindre. Le problème de la stabilité qui était résolu précédemment par l'enveloppe convexe se transforme ici en un problème géométrique original, «étant donné n points coloriés avec k couleurs, calculer l'union des triangles ayant des sommets de trois couleurs différentes».

Nous résolvons ce problème des triangles tricolores en utilisant un arbre de répartition des couleurs [BDP91]. On remarque tout d'abord que les deux arêtes adjacentes à un sommet convexe de cette union sont portées par les deux tangentes, issues de ce point, à l'enveloppe convexe des points ayant une couleur différente de la sienne. On construit un arbre binaire équilibré tel que dans une feuille est stockée l'enveloppe convexe des points d'une couleur donnée et dans un nœud interne est stockée l'enveloppe convexe de ses deux enfants. Un nœud à hauteur h contient donc l'enveloppe convexe des points de 2^h couleurs, la hauteur de l'arbre est donc $\log k$ et tout ensemble de $k-1$ couleurs peut être construit comme l'union des couleurs correspondant à $\log k$ nœuds. Pour calculer la tangente à droite (ou à gauche) à l'enveloppe convexe des points de $k-1$ couleurs (toutes les couleurs sauf celle du point dont sont issues les tangentes) on peut donc calculer les $\log k$ tangentes aux différents convexes stockés dans l'arbre et réalisant l'ensemble de $k-1$ couleurs, puis en sélectionner la meilleure.

A partir de cette idée, on peut obtenir un algorithme optimal [BDP91].

Théorème : *L'union des triangles tricolores formés à partir de n points coloriés peut être calculée en temps $\Theta(n \log n)$.*

[1] α est l'inverse de la fonction d'Ackermann et croît de manière extrêmement lente

Conclusion

Depuis mon arrivée au pays des géomètres algorithmiques, les algorithmes randomisés auront été mon principal sujet de recherche, et un sujet assez chaud. Nous avons pris part au récent développement de ces techniques et après la phase théorique, voici venue celle des applications. Le programme de triangulation de Delaunay dans le plan est largement distribué, tant directement que par le biais de la librairie LEDA [2]. Les implantations d'algorithmes randomisés deviennent de plus en plus fréquentes et pour des problèmes variés. Plusieurs sont actuellement en cours dans le projet PRISME.

Ce développement des applications est accompagné d'un tassement des activités théoriques. La randomisation est devenue un des ingrédients de construction des algorithmes, et nous verrons encore longtemps des algorithmes randomisés pour de nouveaux ou d'anciens problèmes. Mais les résultats sur la randomisation en tant que telle, les méthodes générales telles que les graphes de conflits ou d'influence sont maintenant assez puissantes pour ne plus connaître la même attention que ces dernières années. Une exception cependant : les algorithmes dynamiques qui ne sont pas encore vraiment formalisés dans un contexte général.

En bref, nos travaux sur la randomisation sont terminés, mais cette technique fait maintenant partie de notre arsenal privilegié pour partir à l'attaque de nouveaux problèmes, c'est maintenant pour nous un outil et non plus un objet d'étude.

L'approche utilisée avec l'espace des sphères nous a beaucoup plu, par la simplicité des démonstrations obtenues et par son formalisme efficace. Là encore, nous avons gagné un outil que nous ne manquerons pas d'utiliser dans l'avenir. L'espace des sphères verra probablement des généralisations à d'autres types d'objets que des sphères, ou a des sphères pour d'autres métriques que la métrique euclidienne.

Avec notre panoplie d'outils, les classiques de la géométrie algorithmique, ceux évoqués ci-dessus que nous avons développés et ceux que nous développerons dans l'avenir, nous essayons de diversifier notre champs d'action. Outre nos centres d'interêt historiques issus de la robotique, nous essayons de voir ce que la géométrie algorithmique peut apporter à d'autres domaines tels que vision, infographie, chimie, botanique...

[2] *Library of Efficient Data Types*

Bibliographie

[AS92] F. Aurenhammer and O. Schwarzkopf. A simple on-line randomized incremental algorithm for computing higher order Voronoi diagrams. *Internat. J. Comput. Geom. Appl.*, 2:363–381, 1992.

[Avn89] F. Avnaim. *Placement et déplacement de formes rigides ou articulées.* Thèse de doctorat en sciences, Université de Franche-Comté, Besançon, France, 1989.

[BCD+92] J. D. Boissonnat, A. Cérézo, O. Devillers, J. Duquesne, and M. Yvinec. An algorithm for constructing the convex hull of a set of spheres in dimension d. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 269–273, 1992.

[BCDT91] J.-D. Boissonnat, A. Cérézo, O. Devillers, and M. Teillaud. Output-sensitive construction of the 3-d Delaunay triangulation of constrained sets of points. In *Proc. 3rd Canad. Conf. Comput. Geom.*, pages 110–113, 1991.

[BD92] J. D. Boissonnat and K. Dobrindt. Randomized construction of the upper envelope of triangles in R^3. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 311–315, 1992.

[BDD92] J-D. Boissonnat, O. Devillers, and J. Duquesne. Computing connolly surfaces. In *IFIP Conference on Algorithms and efficient computation*, September 1992.

[BDDP95] J.-D. Boissonnat, O. Devillers, L. Donati, and F. Preparata. Motion planning of legged robots: the spider robot problem. *Internat. J. Comput. Geom. Appl.*, 5(1–2):3–20, 1995.

[BDP91] J.-D. Boissonnat, O. Devillers, and F. Preparata. Computing the union of 3-colored triangles. *Internat. J. Comput. Geom. Appl.*, 1(2):187–196, 1991.

[BDS+92] J.-D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete Comput. Geom.*, 8:51–71, 1992.

[BDT93] J.-D. Boissonnat, O. Devillers, and M. Teillaud. An semidynamic construction of higher-order Voronoi diagrams and its randomized analysis. *Algorithmica*, 9:329–356, 1993.

[Boi85] J.-D. Boissonnat. Reconstruction of solids. In *Proc. 1st Annu. ACM Sympos. Comput. Geom.*, pages 46–54, 1985.

[Boi88] J.-D. Boissonnat. Shape reconstruction from planar cross-sections. *Comput. Vision Graph. Image Process.*, 44(1):1–29, October 1988.

[Boi92] J-D. Boissonnat. Géométrie, algorithmes et robotique. Habilitation à diriger des recherches, Université de Nice, France, 1992.

[BT86] J.-D. Boissonnat and M. Teillaud. A hierarchical representation of objects: The Delaunay tree. In *Proc. 2nd Annu. ACM Sympos. Comput. Geom.*, pages 260–268, 1986.

[BT89] J.-D. Boissonnat and M. Teillaud. On the randomized construction of the Delaunay tree. Technical Report 1140, INRIA Sophia-Antipolis, Valbonne, France, 1989.

[CE92] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39:1–54, 1992.

[CEG+91] B. Chazelle, H. Edelsbrunner, L. Guibas, M. Sharir, and J. Snoeyink. Computing a face in an arrangement of line segments. In *Proc. 2nd ACM-SIAM Sympos. Discrete Algorithms*, pages 441–448, 1991.

[Cha91] B. Chazelle. An optimal convex hull algorithm for point sets in any fixed dimension. Technical Report CS-TR-336-91, Dept. Comput. Sci., Princeton Univ., Princeton, NJ, 1991.

[Cla87] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2:195–222, 1987.

[CMS92] K. L. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental constructions. In *Proc. 9th Sympos. Theoret. Aspects Comput. Sci.*, volume 577 of *Lecture Notes in Computer Science*, pages 463–474. Springer-Verlag, 1992.

[CS89] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[dBDS94] M. de Berg, K. Dobrindt, and O. Schwarzkopf. On lazy randomized incremental construction. In *Proc. 26th Annu. ACM Sympos. Theory Comput.*, pages 105–114, 1994.

[Dev92a] O. Devillers. Randomization yields simple $O(n \log^* n)$ algorithms for difficult $\Omega(n)$ problems. *Internat. J. Comput. Geom. Appl.*, 2(1):97–111, 1992.

[Dev92b] O. Devillers. Robust and efficient implementation of the Delaunay tree. Report 1619, INRIA Sophia-Antipolis, Valbonne, France, 1992.

[Dev93] O. Devillers. Simultaneous containment of several polygons: analysis of the contact configurations. *Internat. J. Comput. Geom. Appl.*, 3(4):429–442, 1993.

[DG93] O. Devillers and M. Golin. Dog bites postman: Point location in the moving Voronoi diagram and related problems. In *Proc. 1st Annu. European Sympos. Algorithms (ESA '93)*, volume 726 of *Lecture Notes in Computer Science*, pages 133–144. Springer-Verlag, 1993.

[DMT92a] O. Devillers, S. Meiser, and M. Teillaud. Fully dynamic Delaunay triangulation in logarithmic expected time per operation. *Comput. Geom. Theory Appl.*, 2(2):55–80, 1992.

[DMT92b] O. Devillers, S. Meiser, and M. Teillaud. The space of spheres, a geometric tool to unify duality results on Voronoi diagrams. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 263–268, 1992.

[DTY92] O. Devillers, M. Teillaud, and M. Yvinec. Dynamic location in an arrangement of line segments in the plane. *Algorithms Rev.*, 2(3):89–103, 1992.

[Gei93] B. Geiger. *Three-dimensional modeling of human organs and its application to diagnosis and surgical planning.* Thèse de doctorat en sciences, Ecole Nationale Supérieure des Mines de Paris, France, 1993.

[GKS92] L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7:381–413, 1992.

[MS91] K. Mulmuley and S. Sen. Dynamic point location in arrangements of hyperplanes. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 132–141, 1991.

[Mul91a] K. Mulmuley. On levels in arrangements and Voronoi diagrams. *Discrete Comput. Geom.*, 6:307–338, 1991.

[Mul91b] K. Mulmuley. Randomized multidimensional search trees: dynamic sampling. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 121–131, 1991.

[Mul91c] K. Mulmuley. Randomized multidimensional search trees: further results in dynamic sampling. In *Proc. 32nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 216–227, 1991.

[Mul91d] K. Mulmuley. Randomized multidimensional search trees: lazy balancing and dynamic shuffling. In *Proc. 32nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 180–196, 1991.

[Orw34] G. Orwell. *1984.* 1934.

[OvL81] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23:166–204, 1981.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, NY, 1985.

[Sch91] O. Schwarzkopf. Dynamic maintenance of geometric structures made easy. In *Proc. 32nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 197–206, 1991.

[Sch92] O. Schwarzkopf. *Dynamic Maintenance of Convex Polytopes and Related Structures*. Ph.D. thesis, Fachbereich Mathematik, Freie Universität Berlin, Berlin, Germany, June 1992.

[Sei91] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl.*, 1:51–64, 1991.

[Sha78] M. I. Shamos. *Computational Geometry*. Ph.D. thesis, Dept. Comput. Sci., Yale Univ., New Haven, CT, 1978.

[Tei93] M. Teillaud. *Towards dynamic randomized algorithms in computational geometry*, volume 758 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

Résumé

Ce mémoire d'habilitation présente 14 articles différents, structurés en trois parties : algorithmes randomisés, algorithmes sur les sphères et placements de robots.

Les algorithmes randomisés ont été un des sujets "chauds" de ces dernières années et nous proposons ici des travaux ayant trait à des algorithmes dynamiques ou semi-dynamiques : tout d'abord un schéma général d'algorithmes semi-dynamiques avec des applications aux diagrammes de Voronoï, aux diagrammes de Voronoï d'ordre k aux arrangements, et ensuite deux algorithmes dynamiques (permettant d'insérer et de supprimer des données) pour la triangulation de Delaunay et le calcul d'un arrangement de segments. D'autres résultats concernent des algorithmes statiques, notamment le calcul du squelette d'un polygone simple en temps $O(n \log^\star n)$.

La deuxième partie explore différentes modélisations des sphères. On peut en déduire notamment un algorithme en $O(tk \log n)$ pour la triangulation de Delaunay de n points appartenant à k plans en 3 dimensions, si t désigne la taille du résultat ; dans la cas de deux plans cet algorithme atteint une complexité optimale de $O(t + n \log n)$. Nous proposons également un algorithme de complexité $O(n^{\lceil \frac{d}{2} \rceil} + n \log n)$ pour le calcul de l'enveloppe convexe de n sphères en dimension d, et un algorithme optimal (quadratique) pour le calcul de la surface de Connolly.

La dernière partie traite de problèmes spécifiques à la planification de trajectoires, un premier chapitre concerne le cas de plusieurs robots polygonaux en translation dans le plan : certaines configurations appellées *double-contacts* peuvent jouer un rôle particulier dans certains cas. Ensuite deux résultats à propos de robots à pattes : l'analyse d'un cas simple que nous avons baptisé robot araignée, et l'étude de la stabilité d'un robot un peu plus complexe.