Modélisation géométrique par contraintes

Christophe Jermann, Dominique Michelucci, Pascal Schreck

Chapitre 1

Modélisation géométrique par contraintes

1.1. Introduction

1.1.1. *Les modeleurs déclaratifs*

Les modeleurs géométriques classiques permettent de décrire des formes géométriques très variées, mais ils ne prennent pas en compte les intentions de l'utilisateur, et n'utilisent pas son langage, ses gestes, ou son expérience du métier. Cela lui impose notamment d'acquérir un nouveau savoir-faire propre au logiciel, et ralentit la définition des maquettes numériques. Les modeleurs déclaratifs (ou la modélisation par formes caractéristiques (*features*) en CFAO) entendent combler ce manque : idéalement, l'utilisateur spécifie avec le langage et les gestes qui lui sont familiers les contraintes qui définissent l'objet géométrique ou la scène ; selon l'application, ces contraintes sont des contraintes géométriques, esthétiques (une « ligne de caractère » avec un « beau galbé »), mécaniques, médicales, économiques, etc. ; le modeleur traduit ces contraintes en une représentation interne, par exemple un système de contraintes géométriques (phase d'acquisition) ; le modeleur produit ensuite une ou plusieurs solutions qui satisfont au mieux les contraintes spécifiées (phase de génération) ; l'utilisateur prend connaissance de ces solutions et corrige sa spécification (phase d'exploration) ; un modèle satisfaisant est ainsi obtenu en quelques itérations.

Il faut distinguer deux catégories d'applications des modeleurs déclaratifs : les domaines artistiques ou ludiques (*e.g.,* animation, synthèse d'image, aménagement urbain) où le côté esthétique des scènes modélisées prime sur leur précision ; et les domaines techniques (*e.g.,* ingénierie, robotique, avionique) où les scènes doivent être

---

Chapitre rédigé par Christophe JERMANN et Dominique MICHELUCCI et Pascal SCHRECK.

définies avec une grande précision. Certaines applications comportent à la fois des aspects esthétiques et d'autres techniques (*e.g.,* conception automobile, architecture) mais on peut en général les séparer (*e.g.,* le designer s'intéresse à l'esthétique de la alors que l'ingénieur étudie son aérodynamisme et sa résistance).

La communauté de Modélisation Déclarative étudie principalement la première catégorie d'applications. Les contraintes y sont généralement imprécises et de haut niveau (*e.g.,* livre « sur » bureau, chaise « devant » table) : l'utilisateur n'a pas besoin de définir précisément la position de chaque composant de la scène mais juste d'en indiquer les grandes lignes. Afin de pallier l'imprécision de la spécification fournie, les modeleurs déclaratifs de cette communauté intègrent des modèles cognitifs permettant de contextualiser, et ainsi préciser, ces spécifications (*e.g.,* une chaise sera en général tournée vers le bureau le plus proche et posée à une distance différente selon qu'elle est occupée ou non). Ces techniques s'intègrent à la phase d'acquisition, permettant de produire en interne un modèle à base de contraintes plus précis. A l'opposé, la communauté de Modélisation par Contraintes s'intéresse plus aux domaines techniques où l'utilisateur, un ingénieur ou technicien, sait définir des contraintes précises (*e.g.,* figure cotée, forces mécaniques) en vue d'obtenir un modèle rigoureux.

On peut estimer qu'après la phase d'acquisition, ces approches convergent vers une même phase de génération qui prend en entrée une description de la scène par des contraintes, et produit les coordonnées de ses composants. Une différence est à noter cependant : en général les modèles imprécis admettent une infinité de solutions alors que les modèles précis en admettent seulement un nombre limité.

1.1.2. *Génération de modèles*

Dans ce chapitre, nous allons nous intéresser exclusivement à la phase de génération, et étudier la façon dont une spécification sous forme de contraintes est traitée pour en extraire les coordonnées des objets modélisés.

L'outil permettant la génération des modèles à partir d'une spécification à base de contraintes s'appelle un *solveur* : il résout les contraintes afin de produire les modèles solutions. Aujourd'hui, tous les solveurs analysent d'abord le système de contraintes, pour y détecter de possibles erreurs, et pour *planifier* la résolution quand le système est correct. La planification (aussi appelée décomposition) consiste à déterminer un ordre pour la résolution des composants du modèle : le système est décomposé en sous-systèmes qui doivent être résolus dans un certain ordre, puis les solutions des sous-systèmes sont assemblées pour produire une solution du modèle entier. Selon le contexte, l'utilisateur souhaite une seule solution, celle qui est intuitivement la plus proche de l'esquisse, ou bien toutes les solutions.

1.1.2.1. *Analyse de modèles*

Les méthodes d'analyse des systèmes de contraintes géométriques peuvent être regroupées en trois classes : les méthodes géométriques, les méthodes combinatoires et les méthodes numériques probabilistes.

Les méthodes géométriques (§ 1.3.1) automatisent le raisonnement géométrique ; elles produisent un plan de construction qui enchaîne les constructions géométriques simples : calculer la droite passant par deux points connus, le cercle passant par trois points connus, les points d'intersection entre 2 droites connues, entre 2 cercles connus, etc. Parfois, des techniques de raisonnement automatique sont utilisées pour ajouter des éléments participant à une construction. Ces méthodes décomposent et résolvent en même temps, d'une certaine manière, les problèmes de construction ; ainsi, dans le domaine de l'enseignement assisté par ordinateur, un plan de construction est-il la forme de solution attendue par l'utilisateur. Elles s'appliquent typiquement en 2D pour des problèmes solubles à la règle et au compas, mais ont aussi connu quelques extensions en 3D.

Les méthodes combinatoires (§ 1.3.2) s'appuient sur une abstraction des systèmes de contraintes par des graphes. Elles y détectent les sous-graphes qui représentent des sous-systèmes pouvant être résolus séparément. L'abstraction des systèmes entraîne une perte d'information qui peut fausser les résultats de ces méthodes. Par exemple, un graphe à 3 sommets liés par 3 arêtes représente aussi bien un triplet de points liés par 3 contraintes de distance qu'un triplet de droites liées par 3 contraintes d'angle ; dans le premier cas, ce graphe représente bien un système soluble alors que dans le second ce n'est pas le cas : si la somme des 3 angles ne vaut pas π, le système n'a pas de solution ; autrement il en a une infinité. Ce dernier cas est dit *singulier* en ce sens que les valeurs des 3 angles sont interdépendantes ; le cas opposé est dit *générique*. La plupart des méthodes combinatoires font l'hypothèse de la généricité du système qui interdit tout cas singulier en particulier ceux où des distances ou des angles ont des valeurs nulles.

Les méthodes numériques probabilistes (§ 1.3.3) travaillent sur le système des équations qui représentent les contraintes de la spécification (*e.g.,* la contrainte de distance égale à 5 entre les points A et B du plan peut s'exprimer par l'équation $(x_A - x_B)^2 + (y_A - y_B)^2 - 25 = 0$). Elles exploitent les propriétés algébriques de ces systèmes (redondance, calcul de base) afin d'en déduire les sous-systèmes solubles séparément. Afin d'éviter le coût du calcul symbolique de ces propriétés, elles réalisent tous les calculs numériquement sur un témoin, c'est-à-dire une solution approximative du système (*e.g.,* une esquisse fournie par l'utilisateur, ou bien un modèle respectant un sous-ensemble des contraintes).

1.1.2.2. *Résolution de modèles*

Après analyse, le système de contraintes a été corrigé et planifié, *i.e.,* sa résolution se ramène à la résolution d'un ensemble de sous-systèmes. Pour cette étape,

chaque composant du modèle doit être représenté par ses coordonnées[1], des variables à valeurs réelles, et les contraintes s'expriment par des équations et inégalités sur ces variables. Le système de contraintes devient alors un système d'équations et d'inegalités $F(U, X) = 0$, $I(U, X) \geq 0$ où X sont les variables, U sont les paramètres et F (resp. I) sont les équations (resp. inégalités). les paramètres du système sont les longueurs, angles, et autres grandeurs pas nécessairement géométriques (*e.g.,* coûts, forces) qui définissent les contraintes. La plupart des solveurs demandent que les paramètres soient fixés avant de pouvoir résoudre un système. Notons que la singularité/généricité d'un système de contraintes tient à la dépendance/indépendance de ses paramètres (cf. exemple au paragraphe précédent). Seules les contraintes métriques, qui spécifient les distances ou les angles, utilisent des paramètres ; les relations d'incidence, de parallélisme, de perpendicularité n'en utilisent pas.

Il existe de nombreuses façons de mettre un système de contraintes en équations, selon le choix des coordonnées et de l'expression de chaque contrainte. Nous ne détaillerons pas ici ces différents choix et leurs implications. Il est cependant important de savoir que ce choix peut influer de façon conséquente sur les performances de la résolution du modèle.

La résolution des sous-systèmes, et l'assemblage des sous-figures solutions, utilisent soit des formules explicites (*e.g.,* calcul du point d'intersection de 2 droites connues, calcul de la longueur du troisième côté d'un triangle connaissant les longueurs de 2 autres côtés et 1 angle), soit des méthodes d'analyse numérique (§ 1.4). En théorie, ces systèmes d'équations peuvent aussi être triangularisés par des méthodes de calcul formel comme celles des résultants, bases de Gröbner, triangularisation de Wu-Ritt. Une résolution, numérique ou formelle peut ensuite être utilisée sur chaque équation à une inconnue. De fait, ces méthodes sont utilisées dans les prouveurs géométriques du monde universitaire, mais ne sont pas employées dans les solveurs industriels, à cause de leur complexité élevée, en temps et en mémoire.

1.1.3. *Exemples*

1.1.3.1. *Une épure cotée*

Les contraintes géométriques sont très utilisées pour la cotation des pièces mécaniques ou des mécanismes, en 2D, 2D et demi, 3D. Dans les applications industrielles, une maquette numérique peut contenir plusieurs centaines ou milliers de cotes. La figure 1.1–gauche montre une esquisse cotée 2D, représentative malgré sa simplicité. Cette figure peut être transcrite en un système de contraintes tel que :

1. On parle ici de coordonnées dans un sens généralisé, le rayon d'un cercle étant lui aussi considéré comme une coordonnée par exemple.

$c_1 : inc(A, \delta_1)$ $c_5 : inc(C, \delta_2)$ $c_9 : dis(B, D, \mathbf{u})$
$c_2 : inc(A, \delta_2)$ $c_6 : inc(C, \delta_3)$ $c_{10} : dis(C, D, 6)$
$c_3 : inc(B, \delta_1)$ $c_7 : ang(\delta_1, \delta_2, 30)$ $c_{11} : dis(D, \delta_3, \mathbf{h})$
$c_4 : inc(B, \delta_3)$ $c_8 : ang(\delta_1, \delta_3, \mathbf{a})$

où inc, ang et dis désignent respectivement des contraintes d'incidence, d'angle et de distance. Les contraintes de distance et d'angle sont des contraintes métriques et prennent donc un paramètre numérique définissant une longueur ou la valeur d'un angle. Dans cette figure, les distances u et h et l'angle a sont des paramètres dont les valeurs ne sont pas précisées.

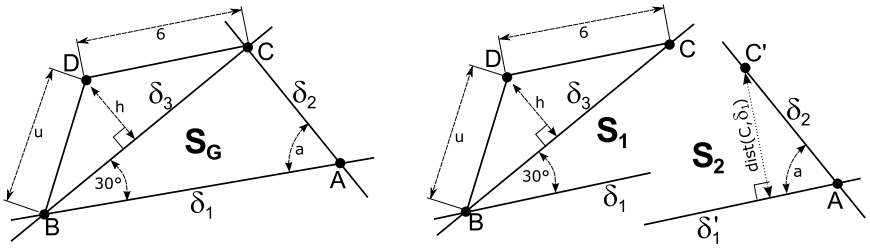

Figure 1.1. *Une esquisse cotée ; les noms des composants (droites et points) ont été ajoutés afin de clarifier la description mais n'apparaissent en principe pas sur un dessin technique*

L'analyse de ce système nous indique qu'il est bien-contraint, *i.e.,* qu'il peut être résolu et produira un nombre fini de solutions. Cette analyse pourrait aussi nous indiquer que ce système peut être résolu en plusieurs fois selon le découpage présenté à la figure 1.1–droite comme nous le verrons plus loin.

Ce système peut être résolu selon des étapes de construction simples de type règle et compas : on se donne une droite arbitraire pour δ_3 ; B est un point quelconque de δ_3 ; D est un des points d'intersection du cercle centré en B de rayon u avec une droite parallèle à δ_3 et distante de δ_3 de h ; C est un des points d'intersection de la droite δ_3 et du cercle centré en D de rayon 6 ; δ_1 est une des 2 droites passant par B, et qui fait un angle de 30°avec δ_3 ; δ_2 est l'une des deux droites passant par C et formant un angle de a°avec δ_1 ; finalement, A est le point d'intersection de δ_2 et δ_1.

Ces étapes peuvent être découvertes par analyse, et correspondent chacune à une formule de calcul simple permettant de déterminer les coordonnées d'un élément. Ces calculs ne sont en général effectués qu'après instanciation des paramètres u, h et a.Dans le cas $u = 4$, $h = 0$ et $a = 45$°on obtient : δ_3 est la droite $y = 0$; $B = (0, 0)$; $D = (\pm 4, 0)$; $C = (x_D \pm 6, 0)$; δ_1 est la droite $y = \pm\frac{1}{\sqrt{3}}x$; δ_2 est la droite $y = \pm x \mp x_C$; $A = (\pm\frac{1}{\sqrt{3}} \mp x_C, \pm\frac{1}{\sqrt{3}}x_A)$. On constate que les coordonnées de certains éléments ne sont pas uniques, *i.e.,* il existe plusieurs figures solutions. Toutefois le nombre de solutions reste fini : 16 ici. Se pose alors la question de la solution

attendue par l'utilisateur, ce qui justifie la troisième phase de modeleurs déclaratifs : la phase d'exploration qui permet à l'utilisateur de choisir une solution en parcourant l'ensemble de celles-ci ou bien en ajoutant des contraintes qui filtrent cet ensemble.

Il existe bien sûr des problèmes plus complexes pour lesquels les plans de construction ne peuvent être aussi explicites. Les méthodes géométriques et combinatoires restent toutefois capables de les planifier automatiquement, chaque étape déterminant alors plusieurs éléments simultanément au lieu d'un seul.

1.1.3.2. *Les problèmes de molécule*

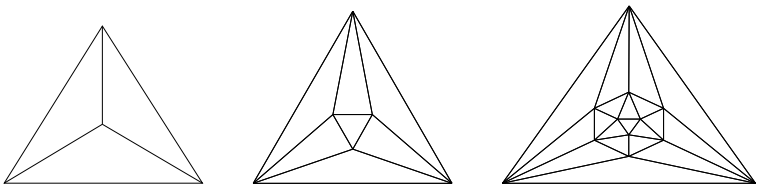

Figure 1.2. *Quelques problèmes de molécules en 3D : le tétraèdre (trivial), l'octaèdre ou plateforme de Stewart, l'icosaèdre.*

Dans le problème dit de la molécule, on considère un graphe non-orienté dont les sommets représentent des points d'un espace de dimension k (typiquement $k = 2$ ou $k = 3$) et les arêtes des contraintes de distance (Fig. 1.2). Le problème est de trouver les coordonnées $x_1, \ldots, x_k$ des sommets du graphe satisfaisant les contraintes de distance. Ce problème doit son nom à ses applications en chimie et biologie moléculaire où il s'agit de reconstituer les configurations d'une molécule à partir de quelques distances entre atomes. Il apparaît aussi en robotique, avec la plateforme de Stewart où deux triangles sont connectés par 6 vérins dont la longueur est connue. Certaines méthodes supposent que les paramètres de distances ont des valeurs génériques, ce qui interdit les plongements dégénérés.

1.1.3.3. *Les problèmes de polyèdre*

Les graphes de la figure 1.3 illustrent d'autres systèmes plus complexes de contraintes géométriques en 3D ; outre les contraintes de distance sur les arêtes, les sommets des facettes doivent être coplanaires (y compris les facettes extérieures). Les graphes de cette figure fournissent des systèmes bien contraints modulo les déplacements, c'est à dire des systèmes n'ayant qu'un nombre fini de solutions, à la position et à l'orientation près en $3D$. Ceci s'étend au graphe de tous les polyèdres eulériens (dont les nombres S, A, F des sommets, des arêtes, des faces vérifient : $S - A + F = 2$).

1.2. Définitions

Un *système de contraintes géométriques* est composé d'*objets géométriques* (les composants du modèle, *e.g.,* points, droites, sphères) et de *contraintes géométriques*

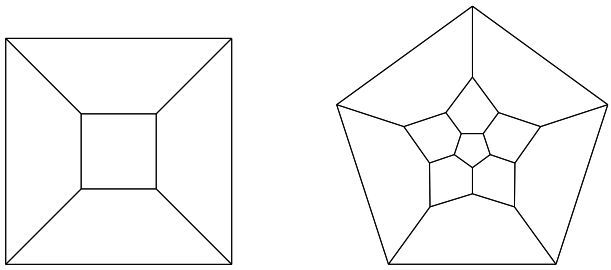

Figure 1.3. *Quelques problèmes polyèdriques en 3D : l'hexaèdre, le dodécaèdre ; ils sont bien contraints.*

(restreignant les positions possibles pour les objets, *e.g.,* distances, angles, alignements). Les contraintes géométriques sont généralement indépendantes du repère : elles ne décrivent que les dimensions et positions relatives des objets, mais pas leur position et leur orientation dans un repère. Ainsi, le modèle correspondant au système est généralement défini à isométrie près (ou *modulo les isométries*). Les isométries sont les transformations géométriques qui laissent invariants les angles et les distances. Elles comprennent les déplacements (rotations et translations) et les symétries axiales (relatives aux droites en 2D, aux plans en 3D).

Étant donné un système $\mathcal{S}$, on note $\mathcal{F}(\mathcal{S})$ l'ensemble de ses figures solutions. La *constriction* est une propriété des systèmes qui a trait à leur ensemble de figures solutions : un système $\mathcal{S}$ est *bien-contraint* si et seulement si $\mathcal{F}(\mathcal{S})$ est fini, *sur-contraint* si et seulement si $\mathcal{F}(\mathcal{S})$ est vide, et *sous-contraint* si et seulement si $\mathcal{F}(\mathcal{S})$ est infini. Lorsque l'on considère la constriction modulo les isométries, on parle de *rigidité* (ou isostaticité) : $\mathcal{S}$ est *bien-rigide* si et seulement si $\mathcal{F}(\mathcal{S})$ quotienté par le groupe des isométries se ramène à un ensemble fini de représentants ; remarquons que $\mathcal{F}(\mathcal{S})$ est alors infini et donc sous-contraint. Un système bien-rigide devient bien contraint lorsqu'on en fixe certains objets ; on dit alors qu'on fixe un repère local. D'autres groupes d'invariance peuvent être envisagés, ainsi, les similitudes sont intéressantes lorsque l'échelle n'est pas fixée ; les similitudes sont des composées d'homothéties et d'isométries et elles laissent invariants les angles et les proportions

Afin d'illustrer ces notions, envisageons plusieurs manières de spécifier un triangle dans le plan.

– tout d'abord, on peut simplement donner les coordonnées des trois sommets et obtenir trivialement un système bien contraint avec une seule solution ;

– ensuite, on peut définir le triangle en spécifiant la longueur de ses cotés, c'est-à-dire par 3 contraintes de distance, on obtient alors un système bien-rigide avec une infinité de solutions qui se déduisent toutes de 2 solutions particulières par des déplacements —à condition que l'inégalité triangulaire soit respectée ;

– enfin, le triangle pourrait être défini par 3 contraintes d'angle ; il serait alors sur-rigide si la somme des 3 angles spécifiés ne valait pas $180°$, ou sous-rigide car les longueurs de ses côtés pourraient varier tout en respectant les relations d'angles. En revanche, ce système serait bien-contraint modulo les similitudes.

La plupart des solveurs de contraintes géométriques ne savent traiter que les systèmes bien-contraints, c'est pourquoi il est important de pouvoir évaluer la constriction d'un système avant de commencer sa résolution.

1.3. Analyse et décomposition

La phase d'analyse des systèmes de contraintes géométriques a pour but d'évaluer la constriction du système, d'identifier les sous-systèmes sur-contraints pouvant poser problème, voire de réparer automatiquement les problèmes de définition lorsque cela est possible. Cette phase débouche en général sur une décomposition du système lorsque celui-ci est bien défini ; on confond donc souvent méthode de décomposition et méthode d'analyse car l'analyse est généralement effectuée au cours de la décomposition. La décomposition a pour but de ramener la résolution d'un système à celle d'un ensemble de systèmes plus simples : c'est l'application du principe *diviser pour régner*. Par décomposition, on espère améliorer les performances de la résolution. En effet, les solveurs ont en général une complexité qui n'est pas linéaire dans la taille du système.

Considérons l'exemple présenté à la figure 1.1. La partie droite de cette figure présente une décomposition de ce système en 2 parties : les sous-systèmes $\mathcal{S}_1$ et $\mathcal{S}_2$. Sans nous attarder pour le moment sur la façon d'obtenir cette décomposition, constatons que la résolution des 2 sous-systèmes produit 2 ensembles de sous-figures $\mathcal{F}(\mathcal{S}_1)$ et $\mathcal{F}(\mathcal{S}_2)$ qui peuvent être combinés en faisant coïncider les points C et C' et les droites δ_1 et δ'_1 par déplacement ; ces combinaisons produisent l'ensemble $\mathcal{F}(\mathcal{S}_G)$ des solutions de $\mathcal{S}$. Il est ici garanti que 2 sous-figures pourront toujours être assemblées en une figure solution par l'ajout dans le sous-système $\mathcal{S}_2$ d'une contrainte de distance entre C' et δ'_1 définie comme ayant la même valeur que la distance entre C et δ_1 dans le sous-système $\mathcal{S}_1$. C'est le processus de décomposition qui a automatiquement ajouté cette contrainte, parfois appelée *bord* ou *frontière*, qui vise à représenter $\mathcal{S}_1$ dans $\mathcal{S}_2$. On remarque donc que les sous-systèmes produits par décomposition, s'ils peuvent être résolus séparément, ne sont pas pour autant indépendants les uns des autres ; ici, $\mathcal{S}_2$ ne peut être résolu qu'après $\mathcal{S}_1$ puisqu'il utilise la valeur $dist(C, \delta_1)$ qui devra être lue dans les sous-figures de $\mathcal{F}(\mathcal{S}_1)$. Par ailleurs, l'ajout d'une contrainte dans $\mathcal{S}_2$ fait de lui un système de contraintes géométriques qui n'est pas à proprement parler un sous-système de $\mathcal{S}_G$. Tout ceci nous conduit à une définition générale des décompositions de systèmes de contraintes géométriques : la décomposition d'un système produit un ensemble de systèmes qui peuvent être résolus séparément *mais* en respectant un ordre (partiel) pour cette résolution, et dont les solutions peuvent être

assemblées en solutions du système initial par un opérateur d'assemblage (e.g., mise en coïncidence d'éléments par déplacement).

Cette définition, qui généralise les notions de *cluster tree* et *DR-plan* de la littérature, s'applique uniquement aux systèmes bien-contraints (ou bien-rigides), chaque composant étant un système lui-même bien-contraint. Il est crucial que chaque composant soit bien-contraint, autrement leurs ensembles de sous-figures solutions seraient infinis (ou vides) et leur assemblage ne pourrait être calculé. Lorsqu'un système est mal contraint, la décomposition n'a plus de sens : certaines méthodes retournent des composants mal-contraints, d'autres produisent un ensemble de composants bien-contraints mais qui ne couvre pas l'ensemble du système. Cependant, de nombreuses méthodes de décomposition intègrent des mécanismes de détection des cas mal contraints, permettant alors un rattrapage automatique ou un retour à l'utilisateur.

1.3.1. *Les méthodes géométriques*

Les méthodes géométriques ont été principalement définies et étudiées entre 1980 et 1995, mais des méthodes récentes les ont remis à contribution. Afin d'illustrer leur principe de fonctionnement, nous allons présenter celle proposée par Sunde en 1986. Cette méthode permet de traiter les systèmes en 2D composés de points liés par des contraintes de distance et d'angle. Elle consiste à agréger petit à petit des sous-systèmes résolus en faisant concorder certains de leurs éléments communs. Initialement, chaque segment (paire de points contraints par une distance, notés CD) et chaque paire de segments contrainte en angle (quadruplet de points liés par un angle, notés CA) est considéré comme un sous-système résolu. L'agrégation est effectuée selon des règles précises ; par exemple, deux CA partageant un segment peuvent être agrégés, trois CD partageant deux à deux un point peuvent être agrégés. Chaque agrégation correspond à la résolution d'un petit système simple permettant d'assembler les sous-figures solutions des sous-systèmes agrégés. La méthode termine lorsqu'aucune règle ne peut plus être appliquée ; si le système entier a alors été agrégé en un unique CD, la décomposition est réussie. Cette méthode a été étendue en 1992 par Verroust *et al*. qui ont proposé quelques règles d'agrégation supplémentaires. Ils ont également montré que malgré cette extension, la méthode restait incomplète, c'est-à-dire qu'il existe des systèmes de points liés par des distances et angles qui sont bien-contraints mais ne peuvent être entièrement agrégés par cette méthode.

La figure 1.4 présente la décomposition du système introduit à la figure 1.1 par une méthode géométrique ; pour simplifier la représentation de cette décomposition, nous utilisons ici une représentation du système par un graphe telle que définie dans le paragraphe suivant. Notons que ce système ne peut pas être traité par la méthode décrite ci-avant car il n'est pas composé uniquement de points liés par des distances et des angles ; il peut toutefois être traité par une méthode généralisée, due à Fudos et

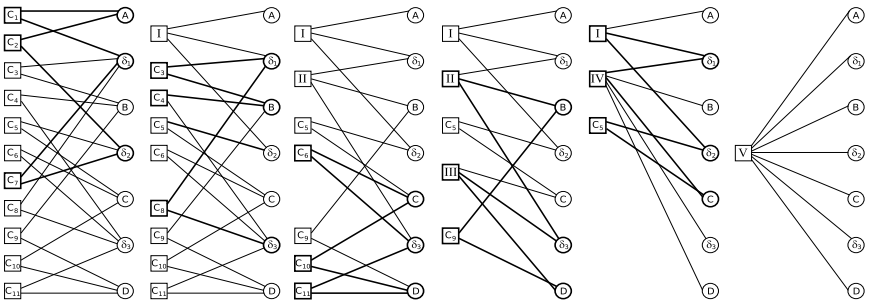

Figure 1.4. *Décomposition du système présenté à la figure 1.1 par une méthode géométrique*

al. en 1995, qui utilise les mêmes principes que celle de Sunde. Initialement, chaque contrainte induit un sous-système résolu. Afin d'agréger ces sous-systèmes, on utilise la règle qui permet d'agréger 3 sous-systèmes partageant 2 à 2 un élément. Pour commencer, les sous-systèmes 1, 2 et 7 sont ainsi agrégés en un nouveau sous-système résolu I ; puis, les sous-systèmes 3, 4 et 8 sont agrégés en un sous-système II ; etc. Le processus se termine lorsque tous les sous-systèmes ont été agrégés en un unique système résolu V : la décomposition est donc réussie.

1.3.2. *Décompositions fondées sur des graphes*

Les méthodes d'analyse et de décomposition basées sur les graphes ont été proposées et étudiées à partir des années 1990. Contrairement aux méthodes géométriques, ces méthodes ne s'appuient pas sur des règles ou des patrons, mais utilisent des propriétés de graphe afin d'identifier les sous-systèmes solubles. Dans les paragraphes suivants, nous présenterons deux approches de ce type.

1.3.2.1. *Décomposition récursive descendante*

Le système $\mathcal{S}_G$ a été découpé selon une paire d'entités géométriques (C, δ_1) dupliquée dans les sous-systèmes $\mathcal{S}_1$ et $\mathcal{S}_2$. On peut considérer que cette décomposition a été obtenue en cherchant dans $\mathcal{S}_G$ un *point d'assemblage*, c'est à dire un sous-ensemble d'objets constituant une coupe du système $\mathcal{S}_G$. Ce processus peut être réitéré dans chacun des sous-systèmes résultant d'une coupe, conduisant à la fin à un ensemble de composants indivisibles. Ce mécanisme de décomposition est dit *récursif descendant* en ce sens qu'il part du système initial et le découpe récursivement. L'élément clé de ces méthodes est l'identification des points d'assemblage : leur définition doit garantir la possibilité de réassembler les sous-figures une fois les composants résolus. Usuellement, les points d'assemblage correspondent aux paires d'articulation dans un graphe représentant le système. La figure 1.5–gauche présente le graphe correspondant à $\mathcal{S}_G$.

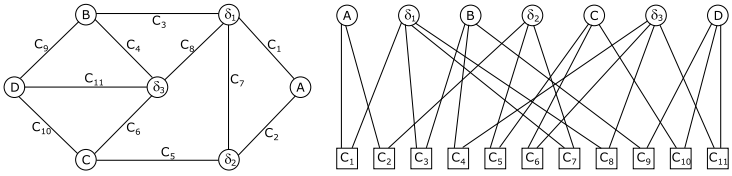

Figure 1.5. *Graphes représentant la structure du système de la figure 1.1*

Dans ce graphe, les sommets sont les entités géométriques et les arêtes représentent les contraintes. Afin d'éviter l'utilisation d'hyper-arêtes, on utilise parfois une variante bipartie de ce graphe où objets et contraintes sont des sommets, les arêtes représentant l'apparition d'un objet dans une contrainte (cf. figure 1.5–droite).

Une paire d'articulation dans un graphe est un couple de sommets dont le retrait coupe le graphe en plusieurs sous-graphes. L'identification d'un tel couple de sommets peut être réalisée en temps polynomial par un algorithme de calcul de connexité. Le couple (C, δ_1) correspond bien à une paire d'articulation dans le graphe représentant $\mathcal{S}_G$. Cette paire est dupliquée dans l'ensemble des sous-systèmes produits en découpant $\mathcal{S}_G$: une occurrence apparaît dans $\mathcal{S}_1$ et une autre dans $\mathcal{S}_2$. Dans $\mathcal{S}_1$, la position relative de C et δ_1 est bien définie. Du point de vue du graphe, cela s'explique par le fait que le sous-graphe de $\mathcal{S}_1$ est bien biconnexe[2]. En revanche, le sous-graphe restant n'est pas biconnexe, ce qui signifie que la position relative de (C', δ'_1) n'est pas bien définie ; autrement dit, ce sous-système est sous-contraint. Afin de maintenir la cohérence entre les deux sous-systèmes, une contrainte $dist(C', \delta'_1) = dist(C, \delta_1)$ est introduite dans $\mathcal{S}_2$. De façon générale, une contrainte similaire devra être introduite dans tous les sous-graphes résultant d'une coupe qui ne sont pas biconnexes. Cette contrainte, dénommée *lien virtuel*, établit un lien entre $\mathcal{S}_1$ et $\mathcal{S}_2$ et impose que $\mathcal{S}_1$ soit résolu avant $\mathcal{S}_2$.

La figure 1.6 illustre l'application complète de cette méthode sur le système de la figure 1.1. Les paires d'articulation identifiées successivement apparaissent en gras, et les liens virtuels en tirets.

La décomposition se termine lorsque tous les sous-systèmes produits sont indivisibles, c'est-à-dire qu'ils correspondent tous à des sous-graphes suffisamment connectés (triconnexes dans le cas de $\mathcal{S}_G$). L'ordre entre les sous-systèmes est induit par les liens virtuels. L'assemblage des sous-figures solutions se fait par mise en coïncidence des objets correspondants aux paires d'articulation au moyen de déplacements.

2. Un graphe est biconnexe s'il existe toujours 2 chemins sans arête commune entre toute paire de sommets.

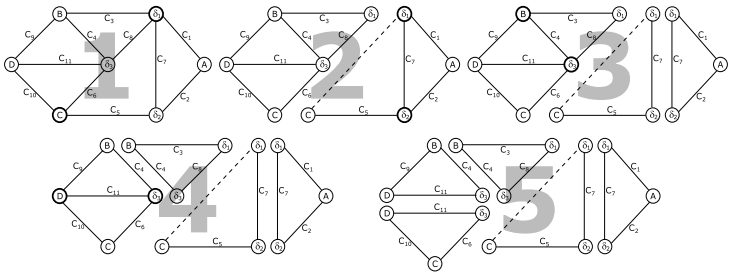

Figure 1.6. *Décomposition descendante du système de la figure 1.1*

La faiblesse de ces méthodes provient de ce que leur caractérisation des points d'assemblage est incorrecte et incomplète : elles peuvent couper le système en des sommets ne permettant pas l'assemblage (e.g., points confondus, droites parallèles) et ne peuvent parfois plus découper un système alors que cela serait encore possible.

1.3.2.2. *Décomposition récursive ascendante*

A l'inverse de la décomposition récursive descendante, on pourrait considérer que la décomposition de $\mathcal{S}_G$ a été obtenue en identifiant d'abord le composant $\mathcal{S}_1$ qui constitue un sous-système bien-contraint de $\mathcal{S}$. Une fois ce composant identifié, il a été retiré du système et remplacé par la contrainte de distance entre C' et δ'_1 qui constitue son *représentant* dans le système restant. Le processus a alors été réitéré et a identifié le sous-système $\mathcal{S}_2$ comme un nouveau composant. Ce mécanisme est dit *ascendant* car il identifie d'abord les composants avant de les assembler jusqu'à couvrir l'ensemble du système. L'étape clé des méthodes de ce type consiste à rechercher un petit composant bien-contraint. Cette recherche peut s'effectuer en utilisant des modèles de sous-systèmes bien-contraints répertoriés (méthodes à base de règles, ou de modèles), ou bien à l'aide d'une propriété structurelle (méthodes structurelles) dérivée d'une caractérisation de la rigidité due à Laman en 1970 : *un système de n points liés par m distances en 2D est rigide si et seulement si $m = 2n - 3$ et pour tout sous-système de n' points induisant m' distances, $m' \leq 2n' - 3$.*

Ce théorème ne s'applique qu'en 2D et uniquement pour les systèmes de points liés par des distances indépendantes (ou *génériques*), ce qui assure que chaque figure solution est une fonction continue des valeurs de distance et interdit, par exemple, les points alignés ou confondus. Toutes les généralisations de ce théorème à d'autres classes de problèmes en 2D ou 3D proposées à ce jour se sont avérées incorrectes. Malgré tout, une caractérisation approximative appelée *rigidité structurelle* est souvent utilisée. Elle repose sur un compte des *degrés de liberté* (DDL). Les DDL des objets représentent le nombre de mouvements indépendants qu'ils admettent ; les contraintes retirent des DDL aux objets sur lesquels elles portent. Le nombre de DDL d'un système est la somme des DDL de ses objets privée de la somme des DDL retirés par ses contraintes. Il mesure le nombre de mouvements indépendants du système.

L'intuition de la rigidité structurelle est qu'un système n'est rigide que s'il admet comme mouvement tous, et uniquement, les déplacements. En 2D et 3D, il y a respectivement 3 et 6 déplacements indépendants. Un système est donc structurellement rigide s'il admet 3 DDL en 2D (resp. 6 DDL en 3D) et que ses sous-systèmes en admettent au moins autant. Cette caractérisation est incorrecte car elle omet deux possibilités : la présence de contraintes redondantes (qui retirent conjointement moins de DDL que leur somme) et de contraintes singulières (qui retirent plus de DDL que leur somme), cf § 1.3.3. De nombreuses heuristiques ont été proposées pour traiter ces cas particuliers, sans toutefois parvenir à faire de la rigidité structurelle une caractérisation exacte de la rigidité.

Une fois un sous-système rigide identifié, les méthodes ascendantes le remplacent dans le système à décomposer par un *représentant*, qui est le plus souvent un sous-ensemble de ses objets liés par des contraintes rigidifiantes ; on parle de *frontière*.

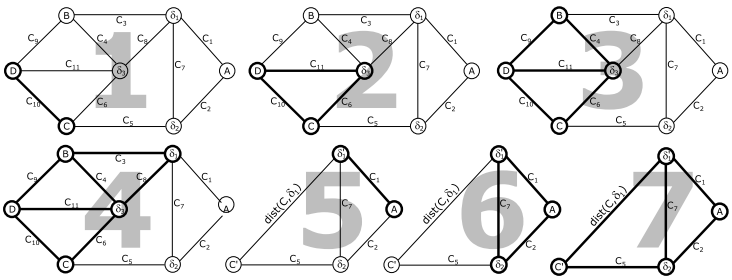

Figure 1.7. *Décomposition ascendante du système de la figure 1.1*

La figure 1.7 illustre la décomposition récursive ascendante de $\mathcal{S}_G$ en utilisant le graphe introduit à la figure 1.5. A chaque étape, le sous-graphe correspondant au composant rigide identifié apparaît en gras. Initialement, c'est le sous-graphe induit par $\{C, D\}$ qui est identifié ; effectivement, un point en 2D offre 2 DDL et une contrainte de distance en retire 1 : on obtient bien un sous-système ayant $2 + 2 - 1 = 3$ DLL en 2D. La décomposition présentée ici utilise alors une étape d'extension visant à agréger un maximum d'objets, un à un, relativement au composant rigide identifié (sinon, la frontière serait égale à ce composant !). Un objet peut être agrégé si et seulement si son ajout au composant préserve la rigidité structurelle. Chaque extension produite ainsi correspond également à un composant bien-contraint. La droite δ_3 est agrégée au composant $\{C, D\}$: elle sera positionnée en utilisant son incidence au point C et sa distance au point D. Il est alors possible d'agréger le point B, puis la droite δ_1. L'extension est terminée et le sous-graphe induit par $\{B, C, D, \delta_1, \delta_3\}$ est retiré du graphe ; pour la suite de la décomposition, il sera représenté par sa frontière, c'est à dire les objets C' et δ_1' liés par une distance. Cette dernière a pour but de maintenir dans le système restant la position de ces objets qui sera déterminée dans le premier

sous-système. La décomposition se poursuit alors de la même façon dans le système restant.

Les méthodes ascendantes identifient donc à chaque étape un composant de la décomposition. L'ordre entre les composants provient des représentants de ceux-ci : un composant ne peut être résolu qu'après tous les composants dont il contient les représentants. Dans notre exemple, $\mathcal{S}_2$ utilise la frontière représentant $\mathcal{S}_1$; $\mathcal{S}_1$ doit donc être résolu avant $\mathcal{S}_2$ afin que cette frontière soit déterminée. La combinaison des sous-figures se fera là encore en faisant correspondre par déplacement celles-ci avec leurs représentants dans d'autres sous-systèmes.

Leur inconvénient principal provient de leur caractérisation approximative de la rigidité : incomplète en cas d'utilisation de modèles de systèmes rigides, incorrectes en cas d'utilisation de propriétés structurelles. Toutefois, elles peuvent s'avérer complètes et correctes pour certaines classes de problèmes, comme les assemblages mécaniques par exemple. Par ailleurs, une caractérisation numérique probabiliste de la rigidité proposée récemment (§ 1.3.3) semble prometteuse même si elle n'a pas encore été appliquée dans ce type de méthodes.

1.3.2.3. *Autres méthodes de décomposition*

D'autres méthodes de décomposition travaillent au niveau du système d'équations représentant le système géométrique. Ces méthodes découpent le système sans dupliquer d'objets géométriques et ne pourraient donc pas produire la décomposition du système $\mathcal{S}_G$ présentée à la figure 1.1–droite.

Certaines identifient simultanément tous les composants en utilisant un algorithme de type couplage maximum sur le graphe représentant le système. Elles associent les composants de la décomposition aux composantes fortement connexes de ce graphe qui est doté d'une orientation induite par le couplage maximum. Ces méthodes ne sont pas récursives : elles produisent tous les composants simultanément. Ceci les rend souvent plus faibles que les méthodes récursives. Toutefois, le fait qu'elles travaillent au niveau des équations et des variables peut leur permettre de résoudre des demi-objets par des demi-contraintes, ce que ne pourraient pas faire des décompositions travaillant au niveau géométrique. Enfin, le fait qu'elles identifient les composants à l'aide d'une propriété de graphe les expose aux mêmes lacunes que les méthodes utilisant une caractérisation structurelle de la rigidité : elles peuvent s'avérer incorrectes en présence de redondances ou de singularités.

D'autres identifient itérativement des composants *libres* ; on dit qu'elles procèdent par *propagation des degrés de liberté*. Intuitivement, un composant est libre si les variables qu'il calcule n'interviennent dans aucune autre contrainte du système. La résolution d'un tel composant n'aura pas d'incidence sur le reste du système. Il peut être retiré pour la suite de la décomposition, entraînant la libération d'autres composants. L'identification des composants libres se fait soit au moyen de modèles de

systèmes bien-contraints, soit au moyen d'une propriété structurelle. A l'instar des méthodes récursives ascendantes, la faiblesse de ces méthodes tient principalement à leur caractérisation incomplète (modèles) ou incorrecte (propriété structurelle) de la constriction.

1.3.3. *La méthode probabiliste du témoin*

A part les méthodes algébriques, les méthodes mentionnées plus haut ne peuvent pas tester correctement l'indépendance des contraintes : tout théorème de géométrie, notamment d'incidence, peut être mis à contribution pour créer des systèmes de contraintes redondants ou contradictoires en y faisant figurer les hypothèses et la conclusion (ou sa négation) du théorème. Par exemple, un système de contraintes où l'on spécifie que les points p_1, p_2, p_3 sont alignés, ainsi que les points q_1, q_2, q_3, et où l'on impose une valeur d'angle non nulle entre les points r_{12}, r_{23}, r_{13} avec $r_{ij} = p_iq_j \cap p_jq_i$, contredit le théorème de Pappus qui stipule que ces points sont alignés. Notons que l'absence de paramètre dans les contraintes d'incidence interdit d'invoquer l'hypothèse de généricité et les techniques perturbatoires associées. Les méthodes *ad hoc* à base de règles ou combinatoires ne sont pas complètes et ne détectent que les cas courants. Des méthodes algébriques, comme la méthode de Wu, permettent de détecter toute dépendance algébrique, mais elles sont très coûteuses.

Cependant, si un témoin est disponible, il est possible de détecter ces dépendances de manière quasi certaine. Soit $F(U, X) = 0$ le système des équations découlant du système de contraintes $\mathcal{S}$; F est l'ensemble des équations qui représentent les contraintes, U est le vecteur des paramètres (distances, angles, ...) et X est le vecteur des inconnues réelles qui modélisent les coordonnées des objets géométriques de $\mathcal{S}$. Un témoin est un couple de valeurs (V, X_V) tel que $F(V, X_V) = 0$. En d'autres termes, un témoin est une solution d'une variante du système à résoudre. L'intérêt du témoin est que, sous réserve d'indépendance des valeurs de U, il a les mêmes propriétés (constriction, redondance, ...) que la cible (la solution que l'on cherche). Ainsi, l'étude du témoin peut être substituée à celle du système de contraintes. Or le témoin étant une instance numérique du système, son étude est plus simple et ne nécessite pas l'utilisation de méthodes de calcul symbolique.

Obtenir un témoin pour un système donné est généralement facile. Dans les applications où l'utilisateur fournit une esquisse sur laquelle il spécifie ses contraintes, les coordonnées et mesures dans cette esquisse constituent généralement un témoin. Dans d'autres applications (problème de la molécule, du polyèdre, ...), il peut suffire de générer aléatoirement (ou presque) des coordonnées pour les objets et d'en déduire les mesures des angles et distances adaptées.

1.3.3.1. *Etude du témoin*

Afin de déterminer la constriction d'un témoin, une étude des déformations, ou mouvements, qu'il admet est réalisée. Intuitivement, un système S est bien-contraint modulo un groupe de transformations G si et seulement si les seules déformations compatibles avec S sont les mouvements définis par G. Pour un système S bien contraint, $G = \{Id\}$ et donc S ne doit admettre aucun mouvement ; S est rigide si les seules déformations admises sont les déplacements. La présence de *flexions*, déformations ne correspondant pas aux déplacements, indique que les solutions du système peuvent être déformées continûment en d'autres solutions non-congrues par déplacement et indique une sous-rigidité ; inversement l'absence de déplacements admissibles indique une sur-rigidité. L'étude consiste donc à établir une base des mouvements admis par le témoin puis à vérifier que cette base contient bien tous les mouvements définis dans G et aucun autre. Cette étude est similaire à celle proposée en Topologie Structurale pour les systèmes de points-distances. La différence est qu'ici les calculs peuvent être réalisés numériquement grâce au témoin. Dans la suite, nous illustrerons cette étude pour la rigidité, *i.e.,* la constriction modulo les isométries directes.

Étant donné un témoin (V, X_V), il s'agit de calculer le vecteur $\dot{X}$ des mouvements infinitésimaux du témoin, tel que le témoin perturbé $X_V + \epsilon\dot{X}$ satisfasse toujours les contraintes spécifiées : $F(V, X_V + \epsilon\dot{X}) = 0$. Un développement de Taylor de cette expression nous indique que $F(V, X_V + \epsilon\dot{X}) = F(V, X_V) + \epsilon F'(V, X_V)\dot{X}^t + O(\epsilon^2)$. Ainsi $F'(V, X_V)\dot{X}^t$ doit s'annuler [3] : une base des mouvements infinitésimaux est donc donnée par le noyau de F' évaluée au témoin, qui peut-être obtenu par algèbre linéaire (*e.g.,* triangularisation de Gauss ou décomposition LU).

Il est possible de donner une base *a priori* des déplacements infinitésimaux. En 2D par exemple, elle contient t_x une translation sur l'axe des x, t_y une translation sur l'axe des y et r_{xy} une rotation autour de l'origine. Il faut alors définir l'effet de ces déplacements sur les coordonnées de tous les objets du système, par exemple :

	$\dot{x_i}$	$\dot{y_i}$	$\dot{a_l}$	$\dot{b_l}$	$\dot{c_l}$	$\dot{u_k}$	$\dot{v_k}$
t_y	1	0	0	0	$-a_l$	0	0
t_x	0	1	0	0	$-b_l$	0	0
r_{xy}	$-y_i$	x_i	$-b_l$	a_l	0	$-v_k$	u_k

où (x_i, y_i) représente un point, (a_l, b_l, c_l) représente la droite d'équation : $a_l x + b_l y + c_l = 0$, et (u_k, v_k) est un vecteur. Les autres inconnues géométriques (rayons, distances, coordonnées barycentriques, produits scalaires, aires), et non géométriques

3. Notons que F étant un système d'équations à plusieurs inconnues, F' est donc la matrice des dérivées partielles de ces équations selon toutes les inconnues, *i.e.,* la *jacobienne* du système. Son déterminant s'appelle le jacobien de F.

(coûts, forces) sont invariantes par déplacement, donc la valeur correspondante dans tous les vecteurs de la base des déplacements infinitésimaux est nulle. En 3D, on définit de même une base des déplacements infinitésimaux constituée de 3 translations (en x, y, z) et de 3 rotations (dans les plans Oxy, Oxz, Oyz).

Disposant d'une base des mouvements du témoin (noyau de la jacobienne) et d'une base des mouvements définis par le groupe de transformations G (ici, les déplacements infinitésimaux), il est possible de comparer les 2 bases et de déterminer si le système est bien-contraint modulo G.

Exemple : étude d'un système 2D

Soit un système en 2D constitué de 4 points O, A, B et C devant vérifier les relations suivantes : la distance OA est égale à u (un paramètre indéterminé), le point O est le milieu de AB, la distance OC est égale à la distance OA, AC et BC sont orthogonaux ; cette dernière contrainte résulte des précédentes, et C peut donc se trouver n'importe où sur le cercle de diamètre AB. Le système d'équations correspondant à ce système pourrait être :

$$\left\{\begin{array}{lrcl} e_1 : & (x_A - x_O)^2 + (y_A - y_O)^2 - u^2 & = & 0 \\ e_2 : & 2x_O - x_A - x_B & = & 0 \\ e_3 : & 2y_O - y_A - y_B & = & 0 \\ e_4 : & (x_C - x_O)^2 + (y_C - y_O)^2 - (x_A - x_O)^2 - (y_A - y_O)^2 & = & 0 \\ e_5 : & (x_C - x_A)(x_C - x_B) + (y_C - y_A)(y_C - y_B) & = & 0 \end{array}\right.$$

Un témoin T possible est : $O = (0, 0)$, $A = (-10, 0)$, $B = (10, 0)$, $C = (6, 8)$, $u = 10$, qui conduit à une jacobienne évaluée en ce témoin :

	x_O	y_O	x_A	y_A	x_B	y_B	x_C	y_C
$e'_1(T)/2$	10	0	-10	0	0	0	0	0
$e'_2(T)$	2	0	-1	0	-1	0	0	0
$e'_3(T)$	0	2	0	-1	0	-1	0	0
$e'_4(T)/2$	-16	-8	10	0	0	0	6	8
$e'_5(T)$	0	0	4	-8	-16	-8	12	16

Ce qui donne une base possible pour les mouvements infinitésimaux (noyau de la jacobienne) :

	$\dot{x_O}$	$\dot{y_O}$	$\dot{x_A}$	$\dot{y_A}$	$\dot{x_B}$	$\dot{y_B}$	$\dot{x_C}$	$\dot{y_C}$
v_1	1	0	1	0	1	0	1	0
v_2	0	1	0	1	0	1	0	1
v_3	0	0	0	-10	0	10	-8	6
v_4	0	0	0	0	0	0	-8	6

Une base des déplacements infinitésimaux pour ce système est :

	$\dot{x_O}$	$\dot{y_O}$	$\dot{x_A}$	$\dot{y_A}$	$\dot{x_B}$	$\dot{y_B}$	$\dot{x_C}$	$\dot{y_C}$
t_x	1	0	1	0	1	0	1	0
t_y	0	1	0	1	0	1	0	1
r_{xy}	0	0	0	-10	0	10	-8	6

On peut constater que $v_1 = t_x$, $v_2 = t_y$ et $v_3 = r_{xy}$ mais v_4 ne peut être exprimé comme une combinaison linéaire de t_x, t_y et r_{xy}. La base des mouvements contient donc bien tous les déplacements, mais contient en sus une flexion. Le témoin est donc sous-rigide. Ce résultat s'étend au système. La même méthode détermine que le sous-système OAB est pour sa part bien-rigide.

1.3.3.2. *Autres utilisations du témoin*

La méthode du témoin remplace les calculs symboliques normalement nécessaires pour l'étude d'un système par des calculs numériques effectués sur une configuration particulière (le témoin) qui a les mêmes propriétés algébriques que le système. Cette méthode peut ainsi être appliquée à d'autres études réalisables par calcul symbolique, comme la recherche de dépendances, la décomposition de systèmes, ou encore la validation de la caractérisation structurelle de la rigidité dans les méthodes à base de graphes.

À titre d'illustration, voici comment la méthode du témoin peut être utilisée pour décomposer un système modulo les déplacements. On appelle *ancre* est un sous-ensemble Y de variables permettant de fixer un repère : il en faut 3 en 2D, 6 variables en 3D. Par exemple, en 2D, si le système fixe directement ou indirectement la distance entre les points A et B, alors $\{x_A, y_A, x_B\}$ est une ancre. Toute ancre Y est contenue dans un sous-système rigide qui est maximal pour l'inclusion : pour le trouver, il suffit d'adjoindre à Y toutes les variables $x \notin Y$ telles que $Y \cup \{x\}$ reste rigide. Une méthode de décomposition récursive descendante, illustrée fig. 1.8, s'ensuit : dans un système entièrement rigide, retirer une contrainte puis calculer les sous-parties rigides maximales en utilisant des ancres ; dans chaque sous-partie rigide, réitérer le processus. Les sous-parties rigides peuvent être réassemblées par la mise en coïncidence des objets qu'elles partagent et l'utilisation de la contrainte retirée.

1.3.4. *Références bibliographiques*

Les méthodes de décomposition ont fait l'objet d'un article de fond auquel ce chapitre emprunte plusieurs exemples et explications [JER 06]. L'article regroupe une bibliographie plus importante.

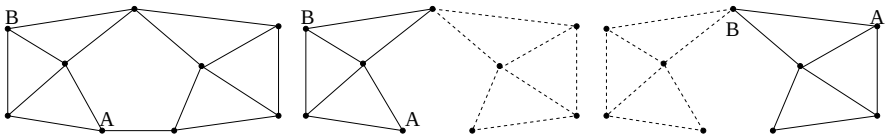

Figure 1.8. *Un système de contraintes rigide 2D. Retirer une contrainte crée un système flexible avec deux parties rigides maximales. Dans les 3 figures, la partie rigide maximale contenant A et B est en traits pleins.*

La première méthode descendante, dédiée aux systèmes de points et droites en 2D, est due à Owen [OWE 91]. Ce type de méthodes a été étudié par la suite par Fudos [FUD 97] puis Joan-Arinyo [JOA 04], qui ont défini leurs limites et leurs liens avec les approches ascendantes. Une extension en 3D, toujours basée sur l'analyse de connexité, est due à Gao [GAO 03]. Michelucci a proposé une méthode descendante où l'analyse de connexité est remplacée par un test numérique probabiliste [FOU 05, MIC 06b] qui ouvre de nouvelles perspectives.

Les premières approches récursives ascendantes sont basées sur des modèles, ont été implantées sous forme de règles dans des moteurs d'inférence logique [BUT 79, SUN 86, ALD 88, VER 92, SCH 94, DUF 98]. Les premières approches structurelles, quant à elles, reposaient initialement sur des règles structurelles très simples [HOF 95, BOU 95] avant que ne soit introduit un algorithme de flot [HOF 98] permettant de calculer la rigidité structurelle. Depuis, cette caractérisation a été utilisée dans plusieurs algorithmes de décomposition ascendante [HOF 01, JER 02, SIT 03]. D'autres groupes d'invariance sont parfois considérés [SCH 03, SCH 06b, SCH 06a].

Les méthodes à base de couplage maximum sont issues de deux communautés distinctes : celle de topologie structurale [HEN 92, LAM 98] et celle de propagation locale [LAT 96, TRO 97, BLI 98]. Quant aux méthodes à base de propagation de degrés de liberté, elles proviennent du travail original de Sutherland [SUT 63], et ont depuis été appliquées avec succès dans différents domaines de l'informatique graphique [HSU 97, TRO 98, SOS 02, TRO 06].

La méthode du témoin repose sur des principes introduits dans la communauté de Topologie Structurale [GRA 93, GRA 02]. L'approche numérique probabiliste fut reprise puis étendue à l'utilisation d'un témoin par Michelucci et al. [LAM 98, FOU 05, MIC 06b].

1.4. Résolution

Une fois le système corrigé et planifié, il faut le résoudre afin de déterminer les valeurs des coordonnées des objets. La résolution d'un système planifié consiste à résoudre chaque composant de la décomposition et à combiner les solutions partielles

obtenues en solutions globales à l'aide d'un opérateur d'assemblage. Il est à noter que chaque composant peut admettre plusieurs solutions dont chacune doit être assemblée avec toutes les solutions des autres composants : un aspect combinatoire apparaît qui peut être traité par un mécanisme de retour-arrière entre les composants.

Nous allons à présent décrire les différentes méthodes qui peuvent être utilisées pour la résolution des composants. Nous ne traiterons pas des méthodes symboliques qui fournissent seulement une forme triangularisée du système qui doit ensuite être résolue numériquement pour obtenir les valeurs des coordonnées, et dont le coût exponentiel (en espace et en temps) confine leur utilisation à certaines applications précises. Les méthodes de résolution numériques ne traitent en général que les systèmes bien-contraints ; afin de leur permettre de traiter les systèmes bien-contraints modulo un groupe de transformations, il faut fixer quelques variables du système. Par exemple, en 2D pour un système rigide, il faut fixer 3 variables qui constituent une ancre ; en 3D, 6 variables formant une ancre. Par ailleurs, elles ne peuvent opérer qu'une fois tous les paramètres des contraintes déterminés, *i.e.,* le système $F(U, X)$ devient $F(X)$ car tous les paramètres U sont remplacés par leurs valeurs.

1.4.1. *Résolution à base de modèles*

Il est possible de coder des méthodes spécifiques de résolution correspondant à des modèles de systèmes précis, comme le calcul de l'intersection de 2 droites ou de 2 cercles. Ces méthodes sont applicables dès lors que le système à résoudre correspond au modèle qu'elles implémentent. Elles sont particulièrement utilisées avec les méthodes d'analyse géométrique qui décomposent un système au moyen de modèles de sous-systèmes bien-contraints. Une méthode de résolution est alors associée à chaque modèle de sous-système.

L'avantage de ces méthodes est qu'elles offrent des performances excellentes tout en garantissant la complétude et la correction du processus, et permettent de vérifier qui plus est les singularités numériques qui pourraient intervenir lors de la résolution. Leur limite est similaire aux méthodes géométriques d'analyse : le répertoire de modèles connus limite les possibilités d'application aux seuls systèmes décomposables selon ces modèles.

1.4.2. *Méthodes numériques itératives*

L'itération de Newton-Raphson et l'homotopie (ou continuation) sont représentatives des méthodes itératives de résolution numérique.

1.4.2.1. *Méthode de Newton Raphson*

La méthode de Newton-Raphson sert à résoudre les systèmes d'équations, linéaires ou non linéaires, algébriques ou transcendantes, à partir d'une approximation d'une

solution. Cette approximation est identique à un témoin (cf. § 1.3.3) et peut donc être obtenue facilement dans le cadre de la CAO. Si X_0 est une approximation d'une racine de $F(X) = 0$, alors $X_1 = X_0 + \Delta_X$ est meilleure si $F(X_1) \approx F(X_0) + F'(X_0).\Delta_X = 0$. $F(X_0)$ et $F'(X_0)$ peuvent être évalués, et les inconnues Δ_X sont alors solutions d'un système linéaire. Si le jacobien de F est non nul, la jacobienne est inversible et $X_1 = X_0 - (F'(X_0))^{-1}F(X_0)$. À partir de X_0, la méthode de Newton-Raphson itère donc : $X_{n+1} \leftarrow X_n - (F'(X_n))^{-1}F(X_n)$ jusqu'à convergence. La méthode de la sécante est une variante qui ne remet pas à jour la jacobienne à chaque itération. Il n'est pas indispensable de calculer l'inverse de la jacobienne ; il suffit de résoudre le système linéaire permettant de déterminer Δ_X par algèbre linéaire (*e.g.,* la méthode de Gauss ou une décomposition LU).

Quand le point initial est proche, en un certain sens, d'une racine, la convergence de la méthode de Newton-Raphson est quadratique : chaque itération « double le nombre de décimales correctes ». Les bassins d'attraction de cette méthode sont des fractales (Fig. 1.11). Toutefois, la méthode peut aussi diverger (cf. fig. 1.9) ; elle fournit donc au mieux une solution par composant du système, ce qui conduit à une résolution incomplète.

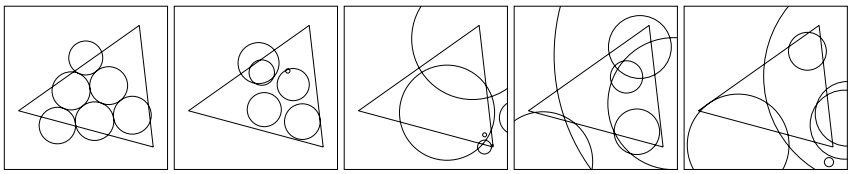

Figure 1.9. *Echec de la méthode de Newton. Voir Fig. 1.10.*

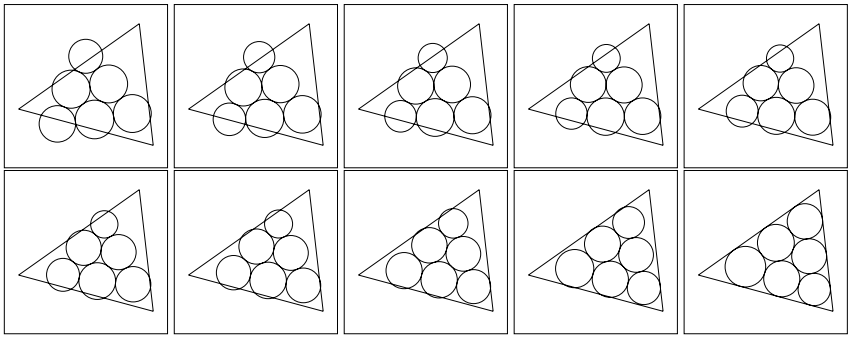

Figure 1.10. *Succès de l'homotopie. Toutes les tangences ont été spécifiées.*

1.4.2.2. *Homotopie*

La méthode par homotopie, appelée aussi par continuation, est une variante de la méthode de Newton-Raphson, au comportement plus intuitif. Ses bassins d'attraction

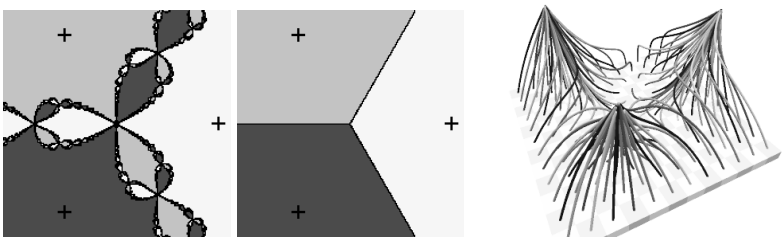

Figure 1.11. *A gauche : bassins (fractales) d'attraction de la méthode de Newton-Raphson ; au milieu, bassins (semi-algébriques) de l'homotopie ; à droite, un échantillon des courbes d'homotopie ; le système est :* $z^3 - 1 = 0, z \in \mathbb{C}$.

sont lisses ; dans le cas où le système est algébrique, ses bassins d'attraction sont des ensembles semi-algébriques. Si $F(X) = 0$ est le système à résoudre, et X_0 une approximation initiale, l'homotopie H est définie par : $H(X,t) = tF(X) + (1 - t)(F(X) - F(X_0)) = 0$. L'homotopie est donc une interpolation linéaire entre deux systèmes, d'une part le système $H(X,0) = F(X) - F(X_0) = 0$ pour $t = 0$, qui s'annule en $X = X_0$, d'autre part le système $H(X,1) = F(X) = 0$ pour $t = 1$, qui est le système à résoudre. Le système d'équations $H(X,t) = 0$ décrit une courbe ; la méthode de résolution par homotopie suit cette courbe, soit par une méthode de prédiction et correction, soit par une méthode de simplexe marcheur (*i.e.,* par approximation linéaire par morceau), depuis $t = 0$ et $X = X_0$ jusqu'à $t = 1$ où X aura la valeur d'une solution du système F. Il se peut que la courbe suivie n'atteigne pas $t = 1$, mais « redescende » (en considérant la coordonnée t comme une altitude) ; plusieurs topologies possibles pour les courbes d'homotopie sont montrées à la figure 1.12.

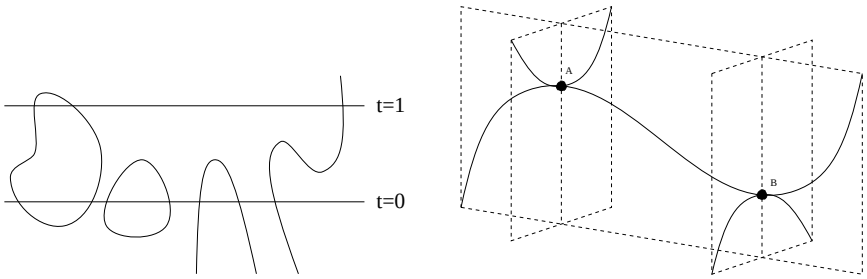

Figure 1.12. *A gauche, quelques topologies possibles pour les courbes homotopiques. A droite, une courbe d'homotopie réelle, en trait gras ; A et B sont deux coudes (turning points). En trait fin les branches complexes. Équations :* $F(X) = 3X^3 - X + \frac{1}{2}$, $G(X) = 3X^3 - X - \frac{1}{2}$, $H(t,X) = 3X^3 - X + \frac{1}{2} - t$, $A_t = \frac{13}{18}$, $A_X = -\frac{1}{3}$, $B_t = \frac{5}{18}$, $B_X = \frac{1}{3}$.

Quand le système à résoudre $F(X) = 0$ est algébrique, il est possible de trouver toutes les racines par homotopie ; l'idée est de construire un système initial I

facile à résoudre et ayant autant de solutions que le système F. Par exemple, chaque équation de degré total d dans F est représentée dans I par le produit de d équations linéaires : résoudre I revient à résoudre plusieurs systèmes linéaires. Chaque solution de I donne un point initial sur une courbe du système homotopique $H(X,t) = tF(X)+(1-t)I(X)$; chaque courbe est suivie dans l'espace complexe jusqu'à $t = 1$ (en pratique jusqu'à $t = 1 - 10^{-5}$ pour éviter les difficultés dues aux solutions singulières ou aux courbes partant à l'infini). Le suivi de chaque courbe a une complexité polynomiale, mais une quantité exponentielle de courbes doit être suivie, même quand le nombre de racines intéressantes (réelles, et non à l'infini) est très inférieur. Dans le cas algébrique, la méthode homotopique est donc complète, et correcte – modulo l'imprécision de l'arithmétique flottante : la continuation peut sauter d'une courbe à une autre.

1.4.3. *Méthodes numériques par intervalles*

Les méthodes numériques par intervalles sont de plus en plus utilisées en informatique graphique, par exemple pour trianguler des surfaces algébriques implicites. Leur principe consiste à remplacer les calculs sur des nombres réels (approchés par des flottants en machine) par des calculs sur des intervalles à bornes flottantes. Les solutions des systèmes sont alors des intervalles dont la largeur peut être rendue aussi précise que souhaitée. Ces méthodes combinent l'avantage d'offrir des calculs numériques corrects, et de permettre la recherche de toutes les solutions d'un système.

1.4.3.1. *Arithmétique d'intervalles*

Ces méthodes reposent sur l'*arithmétique d'intervalles* qui définit les opérations arithmétiques sur les intervalles à partir des opérations sur les flottants. Par exemple :

$$\begin{array}{rcll} [a,b] +_i [a',b'] & = & [& \lfloor a+a' \rfloor, \lceil b+b' \rceil \quad] \\ [a,b] \times_i [a',b'] & = & [& \lfloor \min(a\times a', a\times b', b\times a', b\times b') \rfloor, \\ & & & \lceil \max(a\times a', a\times b', b\times a', b\times b') \rceil \quad] \end{array}$$

où $\lfloor x \rfloor$ désigne le *nombre flottant* immédiatement inférieur au nombre réel x, et $\lceil x \rceil$ le nombre flottant immédiatement supérieur : les opérations flottantes doivent être arrondies vers $+\infty$ ou $-\infty$ pour être correctes. Cette arithmétique sur les intervalles ne constitue pas un corps, au sens mathématique ; par exemple un intervalle n'a ni opposé ($[0,1] - [0,1] = [-1,1]$) ni inverse. L'amplitude d'une somme (ou d'une différence) de deux intervalles est en effet la somme des amplitudes des deux intervalles arguments, et même un peu plus en tenant compte des erreurs d'arrondi. Cette arithmétique se généralise aux fonctions classiques ($\sqrt{x}$, e^x, $\log x$, $\cos x$, $\sin x$...) en décomposant les intervalles des arguments en domaines où la fonction est monotone.

L'utilisation de cette arithmétique permet de définir l'*extension intervalle* d'une fonction, qui consiste à considérer que les variables de la fonction prennent leurs valeurs dans des intervalles et à évaluer la fonction au moyen de l'arithmétique d'intervalle. L'arithmétique d'intervalles donne malheureusement des encadrements très pessimistes. Par exemple, si $x \in [0, 1]$, l'extension intervalle de la fonction réelle $x(1-x)$ s'évalue à $[0, 1]$ alors que la réponse exacte est $[0, \frac{1}{4}]$. Ceci est dû au fait qu'en substituant chaque variable dans une fonction par un intervalle, le lien entre les occurrences d'une même variable est perdu. Il convient donc d'effectuer certains calculs de façon symbolique pour que $x - x = 0$, ou d'utiliser la forme centrée (aussi dite de Taylor), qui donne des encadrements plus fins : $F(X_0 \pm W) \in F(X_0) \pm W\, F'(X_0 \pm W)$, où X_0 est le centre du vecteur d'intervalles, et W le vecteur des demi largeurs des intervalles.

1.4.3.2. *Méthode de Newton-Raphson par intervalles*

L'analyse par intervalles est ainsi utilisée pour étendre la méthode de Newton-Raphson, plus précisément dans sa variante de la sécante. Elle suit jusqu'à convergence le schéma itératif $X_{n+1} \leftarrow X_n \cap S(X_n)$, où $S(X_n) = X_n - J^{-1}F(X_n)$; les racines de F dans X_n se trouvent toutes dans X_{n+1}. Ici X_n et X_{n+1} sont des boîtes, *i.e.,* des vecteurs d'intervalles représentant les plages de valeurs possibles (domaines) pour les variables ; $F(X_n)$ est aussi un vecteur d'intervalles correspondant aux évaluations par intervalles des équations F ; J est la jacobienne en un point de X_n, typiquement le centre de X_n. L'expression $S(X_n)$ est évaluée avec la forme centrée de S, appelée opérateur de Krawczyk, ou de Krawczyk-Moore, selon les auteurs ; cette évaluation a ensuite été optimisée par Segupta et Hansel. Si $S(X_n) \subsetneq X_n$, alors S est contractante dans X_n, et X_n contient donc une seule racine, régulière. Si $X_n \cap S(X_n)$ est vide, alors X_n ne contient pas de racine. Si X_{n+1} n'est pas significativement plus petit que X_n, alors l'un des intervalles de X_n est coupé en deux (opération de *bissection*) et le processus est réitéré sur les 2 sous-vecteurs d'intervalles résultants. L'analyse par intervalles garantit correction et complétude.

De façon générale, toutes les méthodes par intervalles rendent deux listes de boîtes : l'une contenant les boîtes encadrant chacunes une solution régulière, et l'autre les boîtes où la méthode ne peut conclure. Dans cette dernière liste, les boîtes résiduelles contiennent des solutions singulières, « presque singulières », voire pas de solution du tout. La convergence de ces méthodes est ralentie autour des solutions singulières (où le jacobien s'annule) ou presque singulières.

1.4.3.3. *Réduction d'intervalles*

La communauté de programmation par contraintes réalise le compromis entre analyse par intervalles et intelligence artificielle en proposant des mécanismes de réfutation appelés *filtrages* qui permettent d'éliminer de grandes portions des intervalles de définition des variables qui ne contiennent, de façon garantie, aucune solution. Ces

méthodes considèrent les équations comme des règles de réécriture sur des intervalles. L'équation $f(x_1, ...x_n) = 0$ donne n règles :

$$X_i \leftarrow X_i \cap f_i(X_1, ..., X_{i-1}, X_{i+1}, ..., X_n), \quad i = 1, \ldots n$$

X_k est l'intervalle des valeurs possibles pour l'inconnue x_k ; la fonction f_i rend un intervalle englobant les valeurs de x_i solutions de l'équation :

$$f(X_1, \ldots, X_{i-1}, x_i, X_{i+1}, \ldots, X_n) = 0$$

où toutes les inconnues sauf x_i ont été remplacées par leurs intervalles, si bien que x_i est la seule inconnue. Ces règles peuvent être vues comme des projections des équations sur les variables. Il est parfois difficile ou impossible d'isoler une variable x_i dans une expression ; des méthodes de résolution univariée par intervalles (par exemple Newton) sont alors utilisées.

L'algorithme de filtrage utilise une file de propagation qui contient initialement toutes les projections de toutes les équations sur toutes les variables. A chaque tour, une projection est utilisée pour filtrer le domaine d'une variable ; si le domaine est effectivement réduit (en pratique, dans une proportion suffisante), toutes les projections qui utilisent le domaine de la variable réduite doivent être réintroduites dans la file de propagation. Le processus est réitéré jusqu'à ce que la file soit vide. Il s'agit donc d'un algorithme de point fixe. Selon la façon de calculer les projections, on peut atteindre différents points fixes (tous sont garantis).

Ces algorithmes de filtrage sont généralement exploités en conjonction avec l'opération de bissection : lorsque le point fixe du filtrage est atteint, le domaine d'une variable est coupé en 2 et le processus est réitéré sur chaque sous-domaine. On appelle cette catégorie de méthodes *Branch&Prune*. Les plus avancées de ces méthodes combinent analyse par intervalles et filtrage afin d'accélérer la convergence de la résolution.

1.4.4. *Références bibliographiques*

L'arithmétique d'intervalles a été introduite par Moore dans le courant des années 1960 [MOO 66]. Depuis, Neumaier [NEU 90], Kearfott [KEA 96a, KEA 96b], les partenaires du projet COCONUT [BLI 01] et bien d'autres ont écrit plusieurs livres sur l'analyse par intervalles. Stolfi et al [AND 94], et Figueiredo et al [FIG 95] ont proposé l'arithmétique d'intervalles affine pour limiter la croissance de l'amplitude des intervalles. Luc Jaulin [JAU 00], s'appuyant sur les travaux de la communauté de programmation par contraintes [DAV 87, LHO 93, BEN 94], a utilisé l'analyse par intervalles en robotique. Stolte et al [STO 01] l'ont utilisé pour discrétiser des surfaces algébriques implicites ou des arbres CSG. Michelucci [MIC 06a] l'a utilisé pour couvrir des attracteurs étranges (Julia, Hénon). Martin et al [MAR 02] ont comparé diverses arithmétiques d'intervalles pour afficher des courbes algébriques implicites.

1.5. Conclusion

Cette présentation rapide de la modélisation géométrique par contraintes ne prétend évidemment pas à l'exhaustivité. Elle laisse entrevoir la richesse d'un domaine qui, sous des dehors techniques, touche, entre autres, au calcul numérique, à l'intelligence artificielle, au calcul formel, aux interfaces homme-machine, à la logique formelle, au raisonnement géométrique et à la théorie des graphes.

Par soucis de concision, nous avons circonscrit notre tour d'horizon aux méthodes classiques de résolution de contraintes, c'est-à-dire de la synthèse d'objets ou de scènes décrits de manière non ambiguë par des contraintes géométriques. Nous n'avons pas développé certaines questions connexes, pourtant très intéressantes, comme, entre autres, celles touchant à la phase d'acquisition des contraintes , à la phase d'exploration de l'espace des solutions (objets, en particulier, de la modélisation déclarative), à la gestion de systèmes articulés, au transfert et à la standardisation des systèmes de contraintes, etc.

Par ailleurs, cette focalisation sur les méthodes classiques ne doit pas laisser croire qu'elles suffisent pour résoudre tous les problèmes de modélisation géométrique par contraintes. Diverses équipes travaillent à de nouvelles approches ou à des extensions ou perfectionnements des méthodes présentées ici. Parmi tous les sujets ouverts et actuellement à l'étude, ceux ayant trait aux contraintes en dimension trois sont particulièrement prometteurs. En voici quelques uns :

– la caractérisation combinatoire, *i.e.,* en utilisant uniquement des propriétés de graphe, de la rigidité des systèmes de distances entre points est un sujet de recherche très actif dans la communauté de la théorie des graphes ;

– en 3D, on a besoin de méthodes de résolution rapides mais aussi robustes, car des phénomènes subtils de dépendances ou d'instabilité numérique apparaissent. Des travaux récents tentent d'utiliser les bases de Bernstein plutôt que la base canonique des polynômes pour accélérer les solveurs.

– nous avons vu que l'hypothèse de généricité était omniprésente dans ce chapitre, mais elle est souvent mise en défaut. Certaines équipes cherchent donc à introduire de la sémantique pour pallier ce manque de robustesse vis-à-vis de la non-généricité.

– enfin, des équipes expriment les contraintes géométriques indépendamment de tout repère, pour les résoudre directement modulo le groupe des déplacements, ou un autre groupe. Cette méthode donne des systèmes d'équations plus petits. D'autres « algèbres géométriques » sont aussi étudiées pour formuler les contraintes.

Chapitre 2

Bibliographie

[ALD 88] ALDEFELD B., « Variations of geometries based on a geometric-reasoning method », *Computer-Aided Design*, vol. 20, n°3, p. 117-126, Butterworth-Heinemann, 1988.

[AND 94] ANDRADE M. V. A., COMBA J. L. D., STOLFI J., « Affine Arithmetic », *Interval'94*, St Petersburg, 1994.

[BEN 94] BENHAMOU F., MCALLESTER D., VAN HENTENRYCK P., « CLP(Intervals) Revisited », *International Symposium on Logic Programming, ILPS'94*, p. 124-138, 1994.

[BLI 98] BLIEK C., NEVEU B., TROMBETTONI G., « Using Graph Decomposition for Solving Continuous CSPs », *Proc. of Constraint Programming, CP'98*, vol. LNCS 1520, p. 102–116, 1998.

[BLI 01] BLIEK C., SPELLUCCI P., VINCENTE L., NEUMAIER A., GRANVILLIERS L., HUENS E., HENTENRYCK P. V., SAM-HAROUD D., FALTINGS B., *Algorithms for Solving Nonlinear Constrained and Optimisation Problems : State of the Art*, A 222 page progress report of the COCONUT project, 2001, Available at `http ://www.mat.univie.ac.at/ ~neum/glopt/coconut/cocbib.html`.

[BOU 95] BOUMA W., FUDOS I., HOFFMANN C., CAI J., PAIGE R., « Geometric constraint solver », *Computer Aided Design*, vol. 27, n°6, p. 487-501, 1995.

[BUT 79] BUTHION M., « Un programme qui résoud formellement des problèmes de construction géométriques », *RAIRO*, vol. 13, n°1, p. 73-106, 1979.

[DAV 87] DAVIS E., « Constraint Propagation with Interval Labels », *AI*, vol. 32, n°3, p. 281–331, 1987.

[DUF 98] DUFOURD J.-F., MATHIS P., SCHRECK P., « Geometric construction by assembling solved subfigures », *Artificial Intelligence Journal*, vol. 99(1), p. 73-119, 1998.

[FIG 95] DE FIGUEIREDO L., STOLFI J., « Adaptive Enumeration of Implicit Surfaces with Affine Arithmetic », *Proc. Eurographics Workshop on Implicit Surfaces*, INRIA, p. 161–170, 1995.

[FOU 05] FOUFOU S., MICHELUCCI D., JURZAK J.-P., « Numerical Decomposition of Geometric Constraints », *Proc. of Solid and Physical Modeling, SPM*, p. 143–151, 2005.

[FUD 97] FUDOS I., HOFFMANN C., « A Graph-Constructive Approach to Solving Systems of Geometric Constraints », *ACM Transactions on Graphics*, vol. 16, n°2, p. 179-216, 1997.

[GAO 03] GAO X.-S., ZHANG G.-F., Geometric Constraint Solving Based on Connectivity of Graph, Rapport n°22, Academia Sinica, Beijing, China, december 2003.

[GRA 93] GRAVER J., SERVATIUS B., SERVATIUS H., *Combinatorial Rigidity. Graduate Studies in Mathematics*, American Mathematical Society, 1993.

[GRA 02] GRAVER J., *Counting on Frameworks : Mathematics to Aid the Design of Rigid Structures*, N° 25, Mathematical Association of America, 2002.

[HEN 92] HENDRICKSON B., « Conditions for unique realizations », *SIAM journal of Computing*, vol. 21, n°1, p. 65-84, 1992.

[HOF 95] HOFFMANN C. M., VERMEER P. J., « A Spatial Constraint Problem », MERLET J.-P., RAVANI B., Eds., *Computational Kinematics'95*, p. 83-92, Kluwer Academic PUBLISHERs, 1995.

[HOF 98] HOFFMANN C., LOMONOSOV A., SITHARAM M., « Geometric Constraint Decomposition », BRUDERLIN B., ROLLER D., Eds., *Geometric Constraint Solving and Applications*, p. 170-195, Springer, 1998.

[HOF 01] HOFFMANN C., LOMONOSOV A., SITHARAM M., « Decomposition of Geometric Constraints Part I : performance measures & Part II : new algorithms », *J. of Symbolic Computation*, vol. 31, n°4, 2001.

[HSU 97] HSU C., BRÜDERLIN B., « A Degree-of-Freedom Graph Approach », *Geometric Modeling : Theory And Practice*, p. 132–155, , 1997.

[JAU 00] JAULIN L., Le calcul ensembliste par analyse par intervalles, Habilitation à diriger des recherches, Université Paris-Sud, Orsay, France, 2000, Available at : http ://www.istia.univ-angers.fr/~jaulin/hdrjaulin.zip.

[JER 02] JERMANN C., Résolution de contraintes géométriques par rigidification récursive et propagation d'intervalles, Ph.D. Thesis, UNSA, NICE, 2002.

[JER 06] JERMANN C., TROMBETTONI G., NEVEU B., MATHIS P., « Decomposition of Geometric Constraint Systems : a Survey », *International journal of computational geometryc and applications – special issue on geometric constraints*, vol. to appear, 2006.

[JOA 04] JOAN-ARINYO R., SOTO-RIERA A., VILA-MARTA S., VILAPLANA-PASTO J., « Revisiting decomposition analysis of geometric constraint graphs », *Computer Aided Design*, vol. 36, p. 123–140, 2004.

[KEA 96a] KEARFOTT R. B., KREINOVICH V., Eds., *Applications of Interval Computations*, Kluwer, Dordrecht, the Netherlands, 1996.

[KEA 96b] KEARFOTT R., *Rigorous Global Search : Continuous Problems*, Kluwer, Dordrecht, Netherlands, 1996.

[LAM 98] LAMURE H., MICHELUCCI D., « Qualitative study of geometric constraints », *Geometric Constraint Solving and Applications*, p. 234–258, Springer, 1998.

[LAT 96] LATHAM R., MIDDLEDITCH A., « Connectivity Analysis : A Tool For Processing Geometric Constraints », *Computer Aided Design*, vol. 28, n°11, p. 917-928, 1996.

[LHO 93] LHOMME O., « Consistency Techniques for Numeric CSPs », *IJCAI'93 : Proceedings of the 13th International Joint Conference on Artificial Intelligence*, p. 232–238, 1993.

[MAR 02] MARTIN R., SHOU H., VOICULESCU I., BOWYER A., WANG G., « Comparison of interval methods for plotting algebraic curves », *Comput. Aided Geom. Des.*, vol. 19, n°7, p. 553–587, Elsevier Science Publishers B. V., 2002.

[MIC 06a] MICHELUCCI D., FOUFOU S., « Interval based tracing of strange attractors », *International Journal of Computational Geometry and Applications*, vol. 16, n°1, p. 27–39, 2006.

[MIC 06b] MICHELUCCI D., FOUFOU S., « The Witness Configuration Method », *Computer Aided Design, Elsevier*, vol. 38, n°4, p. 284–299, 2006.

[MOO 66] MOORE R., *Interval Analysis*, Prentice-Hall, 1966.

[NEU 90] NEUMAIER A., *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, UK, 1990.

[OWE 91] OWEN J., « Algebraic Solution For Geometry From Dimensional Constraints », *Proc. of Solid Modeling and CAD/CAM Applications*, p. 397-407, 1991.

[SCH 94] SCHRECK P., « Implantation d'un système à base de connaissances pour les constructions géométriques », *Revue d'Intelligence Artificielle*, vol. 3, n°8, p. 223-247, 1994.

[SCH 03] SCHRAMM E., SCHRECK P., « Solving geometric constraints invariant modulo the similarity group », *Proceedings of the 2002 International Conference on Computational Science and its Applications(Montreal), Part II*, LNCS 2669 (Part III), Springer, p. 356-365, 2003.

[SCH 06a] SCHRECK P., MATHIS P., « Geometrical constraint system decomposition : a multi-group approach », *International Journal of Computational Geometry and Application*, vol. to appear, 2006.

[SCH 06b] SCHRECK P., SCHRAMM E., « Using the invariance under the similarity group to solve geometric constraint systems », *Computer-Aided Design*, vol. 38, p. 475-484, 2006.

[SIT 03] SITHARAM M., Frontier, an opensource gnu geometric constraint solver : version3 (2003) for general 2d and 3d systems, 2003, http ://www.cise.ufl.edu/ sitharam.

[SOS 02] SOSNOV A., MACÉ P., « Rapid Algebraic Resolution of 3D Geometric Constraints and Control of their Consistency », *Proceedings of the 4th International Workshop on Automated Deduction in Geometry*, 2002.

[STO 01] STOLTE N., KAUFMAN A., « Novel Techniques for Robust Voxelization and Visualization of Implicit Surfaces », *Graphical Models*, vol. 6, n°63, p. 387–412, Academic Press, 2001.

[SUN 86] SUNDE G., « Specification of Shapes by Dimensions and Other Geometric Constraints », *IFIP WG 5.2 Geometric Modeling*, 1986.

[SUT 63] SUTHERLAND I., Sketchpad : A Man-Machine Graphical Communication System, PhD thesis, Department of Electrical Engineering, MIT, 1963.

[TRO 97] TROMBETTONI G., Algorithmes de maintien de solution par propagation locales pour les systèmes de contraintes, Ph.D. Thesis, UNSA, NICE, 1997.

[TRO 98] TROMBETTONI G., « A Polynomial Time Local Propagation Algorithm for General Dataflow Constraint Problems », *Proc. of Constraint Programming*, vol. LNCS 1520, p. 432–446, 1998.

[TRO 06] TROMBETTONI G., WILCZKOWIAK M., « GPDOF : a Fast Algorithm to Decompose Under-constrained Geometric Constraints : Application to 3D Model Reconstruction », *Int. Journal of Computational Geometry and Applications (IJCGA)*, vol. 16, 2006, submitted.

[VER 92] VERROUST A., SCHONEK F., ROLLER D., « Rule Oriented Method for Parametrized Computer Aided Design », *Computer Aided Design*, vol. 24, n°6, p. 531–540, 1992.