7e Conférence Internationale de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 – Paris- France

« Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination.».

UTILISER LA SOUS-CONSTRICTION POUR RESOUDRE DES SYSTEMES DE CONTRAINTES GEOMETRIQUES

Simon E.B. THIERRY, Pascal SCHRECK, Pascal MATHIS, Arnaud FABRE

LSIIT UMR CNRS-ULP 7005
Pôle API, Boulevard Sébastien Brant
BP10413
67412 Illkirch Cedex
[thierry,schreck,mathis,fabre]@lsiit.u-strasbg.fr

RESUME : *Le problème de la résolution de contraintes géométriques dans le cadre de la CAO provient classiquement du besoin de spécifier des objets à l'aide d'un système de cotes. Les solveurs géométriques considèrent habituellement des systèmes bien-contraints, i.e. ayant un nombre fini de solutions. Nous montrons dans ce papier qu'il peut être utile de considérer des systèmes sous-contraints,* i.e. *dont les solutions forment un continuum : d'une part, pour généraliser le schéma des méthodes de décomposition habituels et, d'autre part, pour rejoindre le point de vue de l'utilisateur.*

MOTS-CLES : *CAO, résolution de contraintes géométriques, systèmes sous-contraints.*

1. INTRODUCTION

La résolution de problèmes de construction géométrique enseignée dans les lycées et les collèges a son pendant en dessin technique où, à partir d'un dessin esquissé sur lequel le concepteur impose un dimensionnement fonctionnel, on doit dessiner une pièce à l'échelle. Avec l'amélioration des connaissances en informatique et l'avènement de la conception assistée par ordinateur (CAO), les logiciels de dessin technique se sont enrichis de fonctionnalités qui résolvent automatiquement ce genre de problèmes (*Cf.* Fig. 1) :

- l'utilisateur dessine une esquisse avec le logiciel
- il impose un dimensionnement avec des outils spécifiques
- un module logiciel, un *solveur*, modifie le dessin pour que celui-ci réponde au dimensionnement.

Dans le cadre de la CAO, on peut regrouper les méthodes rencontrées dans la littérature en deux catégories :

- les approches numériques qui font appel aux possibilités de calcul des ordinateurs
- les approches symboliques qui mettent en œuvre des manipulations algébriques formelles ou des raisonnements géométriques dans le style des constructions à la règle et au compas.

Les approches numériques consistent à résoudre des systèmes d'équations réelles. Parmi les plus connues, on peut citer la méthode de Newton-Raphson, la méthode par continuation et les méthodes basées sur l'arithmétique des intervalles. Ces méthodes sont très générales, mais elles passent mal à l'échelle, c'est pourquoi dès le début de leur utilisation, elles ont été associées à des méthodes de décomposition de systèmes d'équations. En ce qui concerne les approches symboliques, seules les méthodes basées sur la géométrie sont utilisées, en effet les méthodes algébriques ont une complexité intrinsèque qui est rédhibitoire. Les méthodes géométriques opèrent en deux phases : une phase de planification, et une phase de résolution numérique. On peut y distinguer les méthodes qui sont fidèles à l'esprit des constructions géométriques pour lesquelles la première phase fournit un plan de construction qui est interprété numériquement lors de la seconde phase (Aldefeld, 1988 ; Dufourd. *et al.*, 1998), et les méthodes qui sont basées sur des propriétés géométriques générales compilées dans des algorithmes de parcours et de décomposition de graphes (Ait-Aoudia *et al.*, 1993 ; Bouma *et al.*, 1995 ; Owen, 1991 ; Hoffmann *et al.*, 1998).

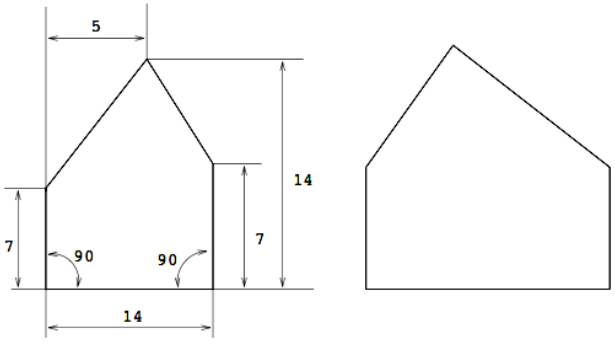

Figure 1. Une esquisse cotée (gauche) et une figure à l'échelle (droite) sans sa cotation.

Ainsi, quelle que soit la nature de l'approche, les méthodes de décomposition jouent un rôle crucial dans les sol-

veurs actuels : d'une part, elles permettent de réduire la complexité et, d'autre part, elles permettent de faire collaborer des solveurs, éventuellement de nature différente. Il est important de noter que si le théorème de Dulmage-Mendelson fournit une méthode canonique de décomposition des systèmes d'équations réelles (Lamure et Michelucci, 1998), ces méthodes doivent dans le cadre de la CAO prendre en compte le fait que les systèmes de contraintes sont *invariants par isométrie directe*. Cette caractéristique inhérente au dessin technique a été prise en compte de différentes manières : par assemblage d'objets rigides (Owen, 91 ; Sunde, 1997 ; Bouma *et al.*, 1995), en résolvant un problème de flux (Hoffmann *et al.*, 1998 ; Jermann *et al.*, 2000), par ajout/relaxation de contraintes de repères (Schreck et Mathis, 2006). Cette dernière technique a été étendue de manière à prendre en compte d'autres groupes de transformations comme celui des similitudes (Schreck et Schramm, 2006).

Les solveurs qu'ils soient numériques ou symboliques, ne donnent de bons résultats que lorsqu'en entrée ils ont des systèmes bien-contraints, c'est-à-dire spécifiant un unique objet (ou, de manière plus générale un nombre fini d'objets), ce qui correspond à des systèmes d'équations non dégénérés. Cette notion de « bonne constriction » est évidemment à relativiser pour tenir compte du groupe d'invariance du système. Dans la cas de l'invariance par isométries, différents auteurs ont imaginé des mécanismes de complétions pour rendre les systèmes bien-contraints (Joan-Arinyo *et al.*, 2003 ; Sitharam, 2005 ; Fabre et Schreck, 2007), nous pensons que considérer des groupes d'invariance généraux permet de ne pas faire de choix *a priori* ce qui permet de mieux coller avec ce que désire l'utilisateur et d'avoir des post-traitements plus performants.

Cet article illustre l'intérêt de prendre en compte des systèmes sous-contraints en vue de leur résolution par deux études liées menées dans l'équipe Informatique Géométrique et Graphique du Laboratoire des Sciences de Image, de l'Informatique et de la Télédétection. Dans un premier temps, en généralisant les travaux de Gao *et al.*, nous avons montré qu'il était utile de transformer un système bien-contraint et indécomposable en un système sous-contraint soluble après complétion *ad hoc* par une méthode symbolique géométrique (*Cf.* section 4). Cette approche permet de faire collaborer en séquence un solveur symbolique et un solveur numérique selon une architecture décrite en section 4. Dans un deuxième temps, nous considérons des objets sous-contraints du point de vue des solveurs usuels et qui pourtant ont un sens pour l'utilisateur (par exemple, un système de contraintes décrivant une paire de ciseaux). Nous exposons une méthode incrémentale permettant de représenter au fur et à mesure de la pose de contraintes le groupe d'invariance du système en train d'être défini par l'utilisateur.

Cet article est structuré comme suit. La section 2 précise la terminologie employée dans cet article. La section 3 présente les travaux du domaine qui sont relatif à la gestion des systèmes sous-contraints. La section 4 expose une méthode de résolution, appelée reparamétrisation de systèmes de contraintes, utilisant des systèmes sous-contraints. La section 5 propose une manière systématique de représenter et d'exploiter les systèmes sous-contraints. Finalement, nous concluons en section 6.

2. TERMINOLOGIE

Bien que le domaine de la résolution de contraintes géométriques soit proche de celui de la résolution d'équations, une terminologie propre s'y est développée. Nous donnons ici les quelques définitions nécessaires à la bonne compréhension de ce papier.

Un système de contraintes (C,X,A) consiste en la donnée de contraintes sous la forme d'un ensemble C de termes prédicatifs dont les variables sont classées dans l'ensemble X des inconnues ou dans l'ensemble A des paramètres. Du point de vue syntaxique qui est donné ici, les systèmes de contraintes qu'on peut décrire dépendent du *langage de contraintes* défini par l'interface utilisateur du logiciel. Ainsi en CAO, les types de contraintes rencontrées expriment, par exemple, la distance entre deux points, la distance entre un point et une droite, la distance entre deux droites parallèles, l'angle entre deux droites, etc. La figure 1 donne quelques exemples de telles contraintes exprimées graphiquement.

Pour des commodités de notation, un système de contraintes (C,X,A) est aussi noté $C(X ; A)$. Cette notation est utilisée pour chaque contrainte de C. On note par « + » l'union de système de contraintes, qui considère aussi l'union des inconnues et des paramètres, de sorte que, si C est constitué des contraintes c_i, on peut écrire $C(X;A) = \sum_i c_i(X;A)$. De la même manière, le signe « – » désigne la différence des systèmes de contraintes.

Ce coté syntaxique a évidemment son pendant sémantique qui réside dans la notion de solution à un système de contraintes dans un modèle donné du langage de contraintes. On note $F(S)$ l'ensemble des solutions du système S. Si S ne contient pas de paramètre, i.e. $A = \emptyset$, $F(S)$ peut être vu comme un ensemble de figures géométriques. Lorsque $A \neq \emptyset$, une solution est une fonction qui à toute valuation des paramètres associe une valuation des inconnues de sorte que les contraintes soient vérifiées. Un système S est dit bien-contraint si $F(S)$ est fini non vide, il est sur-contraint si $F(S)$ est vide et sous-contraint si $F(S)$ est infini. Deux systèmes S_1 et S_2 tels que $F(S_1) = F(S_2)$ sont dits équivalents.

Les systèmes de contraintes rencontrés en CAO définissent des objets solides à une isométrie directe (ou déplacement) près : on dit que de tels systèmes sont invariant par déplacement. D'une manière générale, pour un groupe $\boldsymbol{G}$ de transformations du plan ou de l'espace euclidien, on dit qu'un système de contraintes S est invariant par $\boldsymbol{G}$, ou $\boldsymbol{G}$-invariant, si pour toute transformation φ de $\boldsymbol{G}$, on a $\varphi(F(S)) = F(S)$. S est bien-contraint *modulo*

G, ou ***G***-bien-contraint, si le nombre d'orbites de $F(S)$ suivant ***G*** est fini. Un repère R pour le système S et le groupe ***G*** est un système de contrainte tel que S+R soit bien-contraint. Par exemple, pour les isométries, on obtient un repère en fixant un point et une direction parmi les inconnues d'un système de contraintes. Si le groupe ***D*** des déplacements est souvent utilisé, nous avons montré que d'autres groupes, comme celui des similitudes, étaient envisageables. Nous avons considéré les groupes de transformation composés de rotation, translation et homothéties agissant différemment sur les parties d'une figure. De tels groupes sont dits locaux. Il convient de noter que si un système est ***D***-bien-contraint, alors il est sous-contraint. Par exemple, la définition d'un triangle par les longueurs de ses trois cotés ne spécifie pas un triangle à une position particulière mais une infinité de triangles égaux.

Un solveur est un module logiciel qui prend en entrée un système de contraintes et qui, idéalement, en calcule l'ensemble des solutions. Les solveurs numériques calculent des solutions approchées à l'aide d'algorithmes itératifs dont il faut assurer la convergence. Les solveurs symboliques, ou formels, calculent une solution exacte sous la forme d'un terme syntaxique représentant soit une expression algébrique, soit un *programme de construction* exprimé dans un formalisme adéquat. Les solveurs formels permettent de résoudre des systèmes de contraintes avec des paramètres, alors que dans le cas des solveurs numériques, les paramètres doivent être instanciés par des valeurs réelles avant résolution.

Les solveurs numériques ou formels font souvent appel à des techniques de décomposition combinatoire s'appuyant peu ou prou sur la géométrie. Les méthodes les plus connues s'appuient sur la notion de graphe de contraintes où les sommets représentent les inconnues géométriques et les arêtes les contraintes entre les objets. Un exemple d'esquisse et de graphe de contraintes est donné à la figure 2. Les sommets de ce type de graphe sont valués par le degré de liberté, i.e. le nombre d'inconnues réelles, de l'objet géométrique correspondant. Sur l'exemple de la figure 2, tous les sommets ont un degré de liberté égal à 2. Les arêtes sont, elles, valuées par le degré de restriction, i.e. le nombre d'équations à inconnues réelles, correspondant à la contrainte. Sur l'exemple de la figure 2, toutes les arêtes ont un degré de restriction égal à 1.

Illustrons à l'aide de l'exemple de la figure 2, deux techniques bien connues de décomposition basées sur le graphe de contrainte. La plus simple des techniques de planification géométrique se nomme propagation des degrés de liberté et consiste à former des systèmes de deux équations à deux inconnues. La correspondance sur le graphe de contrainte consiste à dire que si deux sommets sont déterminés et reliés à un troisième sommet, alors on peut déterminer ce troisième sommet. Cela, en tenant compte de l'invariance par déplacement donne lieu à l'algorithme suivant :

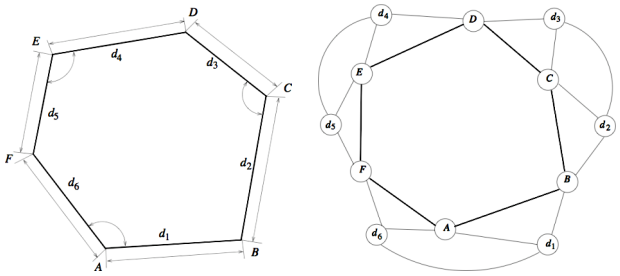

Figure 2**.** Une esquisse cotée simple (gauche) et le graphe de contraintes correspondant (droite).

```
marquer deux sommets reliés
tant qu'on peut
    - chercher un sommet non marqué
    relié à deux sommets marqués
   -   si on en trouve un
       alors on le marque,
      sinon, on arrête.
si tous les sommets sont marqués
   alors réussite,
   sinon échec
```

Cet algorithme arrive à planifier la résolution de systèmes très simples sans cependant être très performant. Ainsi, avec notre exemple, on obtient la planification suivante, en partant de A et B :
marquer A et B, puis d_1, puis d_6, puis F, échec

Parallèlement aux travaux de Sunde (Sunde, 1987), Hoffmann *et al.* ont généralisé cet algorithme à des composantes rigides nommées *clusters* (Bouma *et al.,* 1995). L'idée en est relativement simple : l'algorithme précédent est lancé et lorsqu'il s'arrête en échec, l'ensemble des objets trouvé est identifié en un cluster et l'algorithme est relancé avec les contraintes restantes pour trouver d'autres clusters. Les clusters sont ensuite assemblés suivant des schémas prédéfinis comme ceux donnés à la figure 3. Avec notre exemple, trois clusters $\{A, B, F, d_1, d_6\}$, $\{B, C, D, d_2, d_3\}$ et $\{D, E, F, d_4, d_5\}$ sont trouvés et l'assemblage en triangle est utilisé.

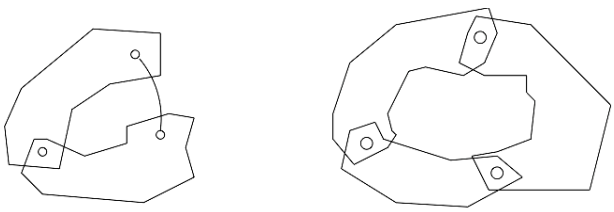

Figure 3**.** Règles d'assemblage de Hoffmann *et al.* A gauche, deux clusters ayant un objet commun et reliés par une contrainte, à droite assemblage en triangle où trois clusters ayant 2 à 2 un point en commun.

3. TRAVAUX CONNEXES

La prise en compte de systèmes géométriques sous-contraints s'est faite selon deux points de vue différents.

On retrouve d'abord ce problème dans la réalisation de mécanismes dans lesquels s'articulent plusieurs pièces rigides. (Kramer, 1991) propose une méthode basée sur l'analyse des degrés de liberté d'objets rigides. Chaque objet est associé à un repère local de l'espace qui traduit les 6 degrés de liberté de l'objet : 3 en translation et 3 en rotation. L'approche est ensuite incrémentale. Les contraintes, qui ne sont ici pas métriques mais uniquement de type incidence, coïncidence, perpendicularité des axes des repères, etc., sont données au fur et à mesure. Chacune d'elle provoque l'élimination de degrés de liberté et transforme le repère possible des objets impliqués dans la contrainte. Par la suite, cette approche a été expérimentée dans le cadre de la modélisation par caractéristiques de forme qui s'y prête particulièrement (Anantha *et al.*, 1996).

Un autre point de vue est celui du dessin technique. Poser une cotation n'est pas évident et peut être considérée comme une activité de programmation. Des auteurs de la communauté de la résolution de contraintes géométriques ont proposé des cadres pour aider le concepteur
- à ne pas oublier des cotes (complétion)
- à prendre en compte des contraintes implicites (angles droits présents sur l'esquisse, etc.)
- à détecter les redondances de contraintes.

(Zhang et Gao, 2006) exploite une idée qui vient assez naturellement pour qui utilise des solveurs procédant par décomposition pour les problèmes. Ce travail concerne les objets du plan. Un solveur, ici basé sur un couplage entité géométrique-contrainte dans le graphe de contraintes, détermine les parties bien-contraintes *modulo* les déplacements. Utilisant une méthode exposée dans (Latham et Middleditch, 1996), les systèmes bien-contraints sont triés selon un ordre imposé pour la résolution numérique. Dans cet ordre, les systèmes sont examinés deux à deux et si moins de 3 contraintes les lient, des contraintes de distances sont ajoutées de sorte à rendre élémentaires la résolution.

Envisageant le problème sous un autre angle, (Joan-Arinyo, 2003) propose une complétion du graphe de contraintes du plan (distances et angles uniquement). Face au graphe d'un système sous-contraint, la méthode s'appuie sur une version bien ou sur-contrainte d'un problème approchant. Ainsi, un problème sur-contraint, par exemple, est traité par relaxation et restriction progressive par ajout de contraintes selon un ordre de priorité.

Pour finir, la reparamétrisation d'un système bien-contraint passe également par le traitement d'un système sous-contraint. Face à l'échec d'une résolution directe, une ou plusieurs contraintes sont supprimées et le système est enrichi de nouvelles contraintes conduisant à une résolution assurément plus aisée. (Gao *et al.*, 2002) et (Gao *et al.*, 2004) initient cette approche dans l'espace en présentant une méthode agissant par propagation des degrés de liberté. Cette propagation s'appuie sur la méthode des lieux. Cependant, lorsqu'une entité (point, droite, plan) subit un nombre de contraintes plus important en regard de son degré de liberté, des contraintes excédentaires sont supprimées. Le rattrapage de ces contraintes délibérément retirées consiste principalement à ajouter des contraintes d'angle nouvelles. Les valeurs des mesures de ceux-ci sont incrémentalement cherchés dans l'espace des paramètres ajoutés.

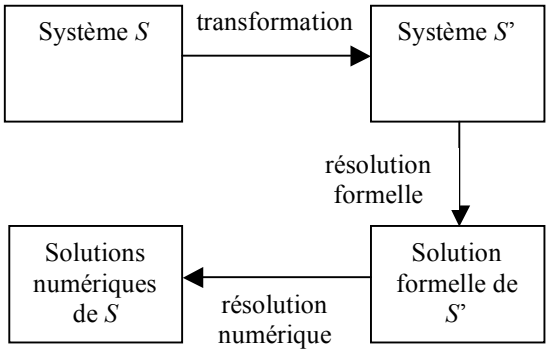

Figure 4. Organisation de la méthode par reparamétrisation.

L'approche originale que nous préconisons pour une résolution par décomposition de systèmes (Dufourd *et al.* 1997 ; Dufourd *et al.*, 1998, Schreck et Schramm, 2006, Schreck et Mathis, 2006), s'appuie sur la notion de repères locaux qui traduisent l'invariance des sous-systèmes par certains groupes de transformations. Notre démarche consiste à reprendre ces idées pour proposer un cadre unique permettant de prendre en compte les systèmes sous-contraints :
- sans ajouter des contraintes que l'utilisateur n'attend peut-être pas,
- en permettant d'explorer l'espace des solutions même si celui-ci est continu (simulation de mécanismes articulés,par exemple).

4. REPARAMÉTRISATION

Le schéma général suivi par les méthodes de décompositions combinatoires consiste à transformer un système de contraintes S en un système S' qui lui est équivalent et qui est l'union de deux ou plusieurs systèmes S_1, S_2, … S_k qui pris dans cet ordre, rendent le système facile à résoudre. Dans de nombreux cas, surtout en dimension 3, ce schéma ne permet pas de décomposer le système. En ôtant la notion d'équivalence entre les systèmes S et S', on se donne plus de marge de manœuvre pour trouver un système S' décomposable et « proche » de S, mais, naturellement, il faut un mécanisme général qui permette de retrouver les solutions de S à partir de celles de S' (*Cf.* Fig. 4). A notre connaissance, la première mise en œuvre de ce schéma apparaît dans la méthode par continuation, appelée encore méthode homotopique où le système S'

est obtenu de manière canonique et dont les solutions sont déformées continument pour retrouver celles de *S*. Gao *et al.* ont proposé une transformation syntaxique de *S* opérée soit en oubliant une contrainte et en transformant une inconnue en paramètre, soit en changeant un contrainte par une autre. Cette dernière manière de faire est évidemment plus générale puisque transformer l'inconnue x en le paramètre μ revient à ajouter l'équation $x = \mu$.

Plus formellement, le remplacement de contraintes suit le schéma suivant où, pour des raisons de simplicité, nous ne considérons que le remplacement d'une contrainte par une autre. On considère le système initial *S* sans paramètre :

$$S = (\begin{cases} c_1(x_1,\ldots,x_p) \\ \cdots \\ c_{m-1}(x_1,\ldots,x_p) \\ c_m(x_1,\ldots,x_p) \end{cases}; \{x_1,\ldots,x_p\}, \varnothing)$$

Celui-ci fournit en oubliant, par exemple, la contrainte c_m, le système S_1 qui est sous-contraint :

$$S_1 = (\begin{cases} c_1(x_1,\ldots,x_p) \\ \cdots \\ c_{m-1}(x_1,\ldots,x_p) \end{cases}; \{x_1,\ldots,x_p\}, \varnothing)$$

On complète ce système par ajout de la contrainte *d* de paramètre *k* pour donner le système *S*' :

$$S' = (\begin{cases} c_1(x_1,\ldots,x_p) \\ \cdots \\ c_{m-1}(x_1,\ldots,x_p) \\ d(x_1,\ldots,x_p;k) \end{cases}; \{x_1,\ldots,x_p\}, \{k\})$$

A la suite de Gao *et al.* mais indépendamment, nous avons étudié et implanté une méthode dans laquelle la résolution de *S*' est assurée par un solveur géométrique symbolique constitué d'un système à base de règles. Le résultat est alors donnée sous forme d'une ou plusieurs fonction définissant les inconnues en fonctions du paramètre k : $x_i = f_i(k)$. Ces fonctions, exprimées formellement, servent à remplacer les inconnues dans la contrainte c_m qui a été écartée au début de la transformation, pour donner la contrainte à résoudre :

$$c_m(f_1(k), \ldots, f_p(k))$$

où k est l'inconnue. Cette contrainte sans paramètre est résolue numériquement pour trouver les bonnes valeurs du paramètre k, qui, à leur tour, seront utilisées via les fonctions f_i pour donner les valeurs des solutions en x_i. Ce principe se généralise au cas où on remplace plusieurs contraintes. On montre que, sous certaines conditions qui sont généralement remplies dans le cadre du dessin technique, cette méthode est correcte et complète, c'est-à-dire que les valeurs qu'elle fournit sont effectivement des solutions de S, et qu'elle est capable de fournir toutes les solutions.

4.1 Résolution formelle par un système expert

La transformation de *S* en *S*' est une des étapes cruciales de la méthode. Gao *et al.* envisagent une recherche systématique de la meilleure contrainte à écarter, mais, pour cause d'explosion combinatoire, cette méthode n'est pas utilisable lorsqu'on doit enlever plusieurs contraintes. L'originalité de notre méthode est de nous appuyer sur le système à base de règles qui va résoudre le système *S*' pour savoir quelles contraintes enlever à *S* et quelles contraintes lui ajouter. Pour cela, il nous faut décrire comment fonctionnent les systèmes experts dans le cas de la résolution de contraintes géométriques.

Reprenons l'exemple donné à la figure 2. La planification trouvée à l'aide du graphe de contraintes peut être trouvée en même temps que la résolution formelle par un système expert appliquant les règles :

- si dist(A,B)=k, et A connu alors B appartient à ccr(A,k)
- si angle(d1,d2) = a et d1 connue alors dir(d2)=dir(d1)+a
- si dir(d) = a, d passe par A et A connu alors d = dpd(A,a)
- si A est sur l1, A est sur l2 et l1 et l2 sont connus alors A = inter(l1,l2)

où *dist* désigne la distance entre deux points, *ccr* le cercle définit par un centre et un rayon, *angle* la mesure de l'angle définit par deux droites, *dir* la direction d'une droite sous forme d'un angle avec l'horizontale, *dpd* la droite passant par un point donné et de direction donnée, et, enfin, *inter* l'intersection de deux lieux géométriques.

Cette application peut se faire classiquement, soit en chaînage avant –on suit alors schématiquement la propagation des degrés de liberté— soit en chaînage arrière – et on fait alors de la rétro propagation des degrés de liberté. Dans le cas de la rétro propagation des degrés de liberté, on applique le principe suivant, un objet avec deux degrés de liberté et relié à exactement deux autres objets par des contraintes peut être construit si ces objets sont eux-mêmes connus. Dans ce cas, on peut l'éliminer du système et le construire après coup.

Appliqué à notre exemple, cet algorithme ne peut même pas démarrer car aucun sommet n'est relié aux autres par exactement deux arêtes (*Cf.* Fig. 2 droite). En *oubliant* une contrainte d'angle, par exemple celle entre d_1 et d_6, on se trouve dans un cas sous-contraint où la rétro propagation fonctionne et donne la séquence inverse : d_1, d_6, *A*, *B*, d_2, *C*, *F*, d_5, *E*. Elle s'arrête en laissant le sous-graphe constitué du point *D* relié aux droites d_3 et d_4 : ce système **D**-sous-contraint est complété par une contrainte d'angle paramétrée par une valeur k entre d_3 et d_4 pour finir de résoudre le système par d_3, et fixer ainsi le repère constitué par *D* et d_4. On emploie ensuite une méthode numérique (ici, on peut employer une dichotomie ou la méthode de Newton), pour trouver la valeur de k qui va permettre de satisfaire la contrainte d'angle entre d_1 et d_6 que nous avions mise de coté. Cet exemple per-

met de montrer comment la méthode de résolution formelle guide le choix des contraintes à enlever et celles des contraintes à ajouter.

Reprenons de manière plus précise le cas d'un système expert fonctionnant en chaînage avant. On se place dans le cadre suivant : à chaque règle est associé un objet dont le degré de liberté est restreint par la règle et le nombre de degrés de liberté qui sont supprimés, ce qu'on appellera le degré de restriction de la règle. Sans perte de généralité : on peut cadencer le système expert pour qu'il s'arrête chaque fois qu'il restreint les degrés de liberté d'un objet non déjà connu. Par ailleurs, si plusieurs objets sont déterminés en même temps, on peut étendre facilement ce qui va être dit plus bas.

On considère aussi un système transactionnel où les faits déduits ou supprimés ne servent à mettre à jour la base de connaissances que lorsqu'on le dit explicitement. On a donc une opération de profil `try : BdC SExp Sys -> Bool Obj N BdC BdC`, qui à partir d'une base de connaissances (ensemble. de termes), d'un système expert et d'un système déjà résolu produit un objet non connu avec le nombre de degré de liberté retirés, les faits à ajouter et les faits à enlever. Lors de cette opération, les règles sont appliquées suivant une stratégie propre au système expert et un nombre indéfini de fois jusqu'à ce qu'un objet voie son degré de liberté être restreint.

Un pas de résolution du système expert peut ainsi être exprimé par l'opération décrite par le pseudo-code :

```
entrée: B, base de contraintes
Se, un système expert transactionnel
S1, le système de contrainte
li, liste des inconnues avec leur deg. de
   lib courant : (x,dl)
sortie: ok, booléen indiquant la réus
B, la base de contraintes modifiée
S1, le système de contrainte modifié
li, liste des inconnues avec leur deg.lib.
Algo :
ok = false
(continue,o,r,B+,B-) = try(B, Se, S1)
// essaie d'appliquer une règle supprim.
// r deg. lib de o, ajoutant des faits
// dans B+, en enlevant dans B-
tant que continue
 x = current_DOF(o,li)
 B = update(B,B+,B-)
 si x+r >= dof(o)
  alors
   continue = false, ok = true
   S1 = update_sys(S1, B)
   stocker les contraintes en surplus
  sinon
   si x+r = dof(o) alors
    continue = false, ok = true
    S1 = update_sys(S1, B)
   sinon
    li = update_list(l, (o,x+r))
    (continue,o,r,B+,B-) =
        try(B,Se,S1)
fait
retourner (ok, B, S1, l)
```

A la sortie de cet algorithme, si `ok` est égal à vrai, alors un objet vient d'être construit et l'algorithme est relancé en utilisant la base de contraintes et le système modifiés. En revanche, si `ok` est égal à faux, alors on conserve l'ancienne base de contraintes, et on peut examiner avec `l` et `B` fournis en sortie la cause de l'échec.

On appelle solveur à base de règles en un pas, un système expert utilisé pour faire des constructions et pour lequel chaque règle permet de construire un objet, c'est-à-dire où toute règle a pour conclusion $o=f(a_1,a_2, ...a_n)$ où les a_i sont connus et o fait partie des inconnues. De tels solveurs sont sans doute fastidieux à écrire, mais ils peuvent être obtenus par compilation d'un système expert classique (Wintz et Schreck, 2006) à condition que celui-ci ne boucle pas. Par ailleurs, ils peuvent être assez efficaces si on les utilise avec des tables de hachage adéquates.

Pour de tels systèmes experts, les procédures décrites ci-dessus peuvent être significativement simplifiées, et c'est un solveur de ce type que nous avons employé en faisant des essais aussi bien en chaînage avant qu'en chaînage arrière.

4.2 Transformation de systèmes.

En chaînage arrière, la stratégie consiste à choisir un des objets les moins contraints du système, disons x_0. On appelle alors voisinage de x_0 l'ensemble des contraintes mettant x_0 en jeu et on le note $N(x_0)$. En faisant la supposition habituelle que tous les autres objets sont construits, le système expert est utilisé pour construire x_0 en utilisant les contraintes de $N(x_0)$ suivant le schéma :

- si le solveur échoue à construire x_0 alors
  - essayer un autre objet peu contraint
  - si tous les objets les moins contraints provoquent un échec, ajouter des contraintes d_j en suivant une heuristique
- si le solveur réussit la construction de x_0 alors
  - soit toutes les contraintes de $N(x_0)$ ont été utilisées et le processus continue
  - soit certaines contraintes, disons c'_l, de $N(x_0)$ n'ont pas été utilisées, alors les stocker à part.

Avec les notations précédentes, le système S' à résoudre formellement est alors égal à $S - \sum_l c'_l + \sum_j d_j$.

De manière semblable, en chaînage avant, on laisse agir le système expert qui résout tout ce qu'il peut en appliquant l'algorithme présenté plus haut. En cas d'échec, l'inconnue qui est la plus déterminée après ce pas, est extraite de la liste fournie avec les contraintes qui ont été mises en jeu pour la déterminer. Une heuristique est alors utilisée pour ajouter des contraintes permettant de continuer l'algorithme donné plus haut. A la fin de ce processus, des contraintes, disons d_j ont été ajoutées par

des méthodes heuristiques et des contraintes c'_l n'ont pas été utilisées. Ici encore, le système *S*' à considérer est égal à $S - \sum_l c'_l + \sum_j d_j$.

De cette manière, que ce soit en chaînage avant ou en chaînage arrière, on est assuré de pouvoir résoudre formellement *S*' puisque la résolution se fait en même temps que la transformation du système. On appelle distance syntaxique entre *S* et *S*' le nombre de contraintes qui ont été échangées. Les heuristiques utilisées ont pour objet d'essayer de minimiser cette distance en modifiant le moins possible *S*.

La phase de résolution numérique, consiste à

- exploiter le plan de construction trouvé lors de la résolution formelle de S' pour évaluer les valeurs des inconnues en fonctions des valeurs des paramètres
- itérer un processus convergent, comme la méthode de Newton Raphson que nous avons utilisée, pour satisfaire les contraintes c'_l qui ont été mises de coté.

Le détail de cette mise en œuvre sort du cadre de cet article et nous ne la détaillerons pas plus.

5. REPRÉSENTATION DES SYSTÈMES SOUS-CONTRAINTS

Nous avons vu que les méthodes non numériques de résolution de systèmes de contraintes géométriques actuelles prennent pour hypothèse la bonne constriction du système *modulo* les isométries. Les solveurs décrits dans la littérature ajoutent donc des contraintes pour revenir à un système bien-contraint. Dans cette section, nous montrons les manques de cette approche et y proposons une alternative.

Certains systèmes non rigides, classiquement vus comme sous-contraints, représentent en fait des objets réels tels que l'utilisateur les conçoit. C'est le cas par exemple d'une paire de ciseaux, d'une lampe de bureau ou encore d'une voiture, où certains éléments sont mobiles par rapport aux autres. Par ailleurs, même dans le cas où l'objet final doit être bien-contraint *modulo* les isométries, la possibilité de représenter et de manipuler des systèmes sous-contraints peut avoir de nombreux avantages. Tout d'abord, comme on l'a vu à la section 4, la décomposition d'un système rigide peut donner lieu à des systèmes non rigides. De plus, la représentation de systèmes sous-contraints est un pré-requis pour obtenir un algorithme de construction incrémentale, permettant à un utilisateur non expert de construire peu à peu le système de contraintes tout en ayant une idée, au fur et à mesure des ajouts de contraintes, du niveau de constriction des résultats intermédiaires.

Le principe de notre approche, mise en pratique en deux dimensions, est de considérer un système sous-contraint

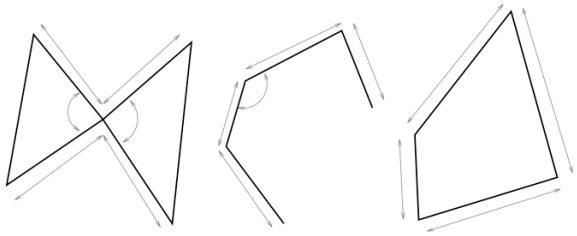

Figure 5. Exemples de systèmes sous-contraints. Notons que le système correspondant à l'ensemble des trois esquisses est lui-même un système sous-contraint.

comme un assemblage de sous-systèmes contraints *modulo* des groupes locaux (composées de translation, rotation et/ou homothétie). La connaissance des entités géométriques communes à deux sous-systèmes permet de savoir comment ces deux systèmes s'articulent. La connaissance du plus petit groupe d'invariance de chaque sous-système permet également de déterminer quels sont leurs repères. Un repère pour le système entier est alors une combinaison de repères pour chacun des sous-systèmes, à laquelle sont retirées des éléments redondants dus aux entités partagées par plusieurs sous-systèmes. Ainsi, une paire de ciseaux peut être vue comme l'assemblage de deux sous-systèmes bien-contraints *modulo* les déplacements et liés par un point commun. Chacun des ciseaux a pour repère un point et une direction. Un repère pour la paire de ciseaux consistera donc un en point et deux directions (une dans chaque ciseau), eu égard au point partagé par les deux sous-systèmes.

L'efficacité de cette méthode varie pour le moment en fonction de l'existence de chaînes fermées dans le système de contraintes géométriques. Une chaîne fermée est un système dans lequel il est impossible de trouver une décomposition du système en deux sous-systèmes non vides tels qu'il y a au maximum une entité géométrique commune aux deux sous-systèmes. Les cycles existant dans de tels systèmes rendent impossible la détermination de l'impact de la position d'un sous-système sur ses voisins, dont la position est également fonction des positions d'autres systèmes dans le cycle.

Pour les chaînes ouvertes, une première approche envisagée a été de considérer des assemblages binaires de sous-systèmes (Thierry *et al.,* 2007). La représentation d'un système sous-contraint est alors effectuée sous la forme d'un arbre où les nœuds représentent le type d'assemblage et les feuilles les sous-systèmes bien-contraints *modulo* des groupes locaux. Les types d'assemblage considérés sont au nombre de trois et dépendent des entités communes aux deux systèmes assemblés. Deux systèmes partageant un point, disposant donc d'une liberté de rotation autour de ce point, forment un assemblage avec liaison rotoïde; deux systèmes partageant une droite, disposant d'une liberté de translation le long de celle-ci, forment un assemblage avec liaison prismatique; deux systèmes sans entités géométriques communes

forment un assemblage de positionnement. Le repère d'un assemblage est facilement calculable par une récursion dans l'arbre. Pour un assemblage avec liaison rotoïde, par exemple, il sera constitué du point commun aux deux systèmes assemblés, d'une direction dans chacun d'entre eux, et des éléments de repère leur manquants encore.

Lorsque l'utilisateur ajoute une contrainte à un assemblage construit, l'obtention de la représentation de l'assemblage résultant passe par le calcul des entités communes au système engendré par la nouvelle contrainte et à l'assemblage. S'il n'y a aucune ou une seule entité commune, on sait quel type d'assemblage ajouter à l'arbre. S'il y a plusieurs entités communes, trois cas se distinguent. Dans le premier cas, la nouvelle contrainte amène la rigidification d'un assemblage. Ce cas est détecté aisément puisque cela n'arrive que lorsqu'une distance est ajoutée entre deux systèmes possédant une liaison rotoïde ou prismatique ou lorsqu'un angle est ajouté entre deux systèmes ayant une liaison rotoïde. Le deuxième cas est celui de l'ajout d'une contrainte entre deux systèmes d'un assemblage de positionnement. Il est aisé alors de construire l'assemblage avec liaison rotoïde ou prismatique résultant, selon que la contrainte ajoutée est une distance ou un angle, respectivement.

Le troisième cas, qui est celui de la création d'une chaîne fermée, a mené à une autre approche, où un système est considéré non plus comme une série d'assemblages binaires, mais comme un assemblage n-aire. Les sous-systèmes sont organisés dans un graphe de construction où la relation liant deux systèmes indique qu'un système doit être construit avant un autre. L'orientation de cette relation permet de déterminer les éléments de repères non nécessaires dans un système donné parce que déjà fournis par ce qui a été construit auparavant. Le problème des chaînes fermées est alors résolu par l'utilisation de règles de construction indiquant quels éléments de repères ne sont plus à fixer en raison de la nouvelle contrainte. Par exemple, lorsque les quatre distances d'un quadrilatère ABCD sont posées et si la dernière contrainte ajoutée est la distance entre C et D, on peut savoir que la direction de la droite (AD) n'est plus nécessaire, car le point D se trouve à l'intersection de deux cercles, l'un de centre A et l'autre de centre C. D'autres règles peuvent être ajoutées, permettant un contrôle de l'univers géométrique voulu par l'utilisateur. Il est même possible d'effectuer la reconnaissance de certains systèmes particuliers pour améliorer les performances.

Cette dernière approche, contrairement à celle de Kramer (Kramer, 1991), bénéficie d'un raisonnement géométrique poussé, conservant notamment les déductions faites sur le bord de deux systèmes, introduites dans (Schreck et Mathis, 2006). Elle présente en outre un gros avantage en matière d'interface utilisateur. En effet, comme on l'a dit, elle permet à un utilisateur non expert d'obtenir un retour sur le niveau de constriction du système qu'il conçoit au fur et à mesure de sa création. L'ajout d'une nouvelle contrainte se faisant en partant de la description du système déjà calculée, les performances sont meilleures que si l'on redémarrait – comme c'est le cas de la plupart des méthodes actuelles – l'étude du système résultant de zéro. En outre, la description des solutions par le moyen du repère du système, séparé en éléments de repères de chacun des sous-systèmes, amène une intuition très visuelle du niveau de constriction et des libertés de l'objet.

6. CONCLUSION

Dans ce papier, nous avons fait le lien entre deux méthodes utilisant la sous-constriction pour résoudre des systèmes de contraintes géométriques. À travers l'étude, d'une part d'une méthode, dite de reparamétrisation, visant à transformer un système bien-contraint en un système sous-contraint et à rattraper numériquement les contraintes enlevées et, d'autre part, d'une approche incrémentale de la conception d'un système de contraintes géométriques visant à rendre plus intuitive l'interface utilisateur et à rendre possible l'ajout efficace de nouvelles contraintes, nous avons illustré l'intérêt que représentait la considération de systèmes sous-contraints.

Ces deux séries de travaux se placent donc dans un cadre commun et nos perspectives incluent, outre des améliorations de chacune des deux méthodes séparément, un rapprochement accru des deux algorithmes, la description d'un système par le biais des repères apportant une réponse toute trouvée quant à l'identification des paramètres à faire varier pour la phase de rattrapage numérique de la reparamétrisation. Il devient même envisageable, dans certains cas, d'éviter les méthodes approximantes comme Newton-Raphson et de passer par une résolution exacte en faisant intervenir des règles de construction géométriques classiques.

BIBLIOGRAPHIE

Ait-Aoudia, S. and R. Jegou and D. Michelucci, 1993. Reduction of constraint systems. *Proceedings of Compugraphics Conference*.

Aldefeld, B, 1988. Variations of geometries based on a geometric-reasoning method. *Computer-Aided Design*, 20(3), p. 117-126.

Anantha R. and G.A. Kramer, and R.H. Crawford, 1996. Assembly modelling by geometric constraint satisfaction. *Computer-Aided Design*, 28(9), p. 707-722

Bouma, W, I. Fudos, C. Hoffmann, J. Cai and R. Paige, 1995. Geometric Constraint Solver, Computer-Aided Design, 27(6), p 487-501.

Dufourd J.-F., P. Mathis and P. Schreck, 1997. Formal resolution of geometrical constraint systems by assembling. *Proceedings of the fourth ACM symposium on Solid Modeling and Applications*, ACM Press, New-York, NY, USA, p. 271-284.

Dufourd, J. -F. and P. Mathis and P. Schreck, 1998. Geometric construction by assembling solved subfigure. *Artificial Intelligence Journal*, 99(1), p. 73-119.

.Fabre, A. and P. Schreck, 2007. A formal-numerical approach to solve 3D geometric constraint systems. *Proceddings of Geometric Modelling and Imaging Conference* (GMAI'07), p. 54-59.

Gao, H. and M. Sitharam, 2005. Combinatorial Classification of 2D Underconstrained Systems. Available on the web at the URL : http://www.cise.ufl.edu/~**sitharam**/under.pdf

Gao, X.-S. and C.M. Hoffmann and W.-Q. Yang, 2002. Solving Spatial Basic Geometric Constraint Configurations with Locus Intersection. *Proceedings of the ACM Solid Modeling Conference*, Saarbrucken, p. 95-104.

Gao, X.-S., C. M. Hoffmann, and W.-Q. Yang, 2004.Solving spatial basic geometric constraint configurations with locus intersection. *Computer-Aided Design*, 36, p. 111–122.

Hoffmann, C. and A. Lomonosov and M. Sitharam, 1998. Geometric constraint decomposition. *Geometric Constraint Solving and Applications*, B. Bruderlin and D. Roller (Eds.), p. 170-195.

Jermann, C., G. Trombettoni, G., B. Neveu, B. and M. Rueher, 2000. A Constraint Programming Approach for Solving Rigid Geometric Systems. *Proceedings of Co,straint Programming* 2000, Lectures Notes in Computer Science, p. 233-248.

Joan-Arinyo, R. and A. Soto-Riera, and S. Vila-Marta, and J. Vilaplana-Pasta, 2003. Transforming an under-constrained geometric constraint problem into a well-constrained one. *SM'03: Proceedings of the eighth ACM Symposium on Solid Modeling and Applications*, ACM Press, New-York, NY, USA, p. 33-44.

Kramer, G.A., 1991. Using degrees of freedom analysis to solve geometric constraint systems. *Proceedings of the 1th ACM Symposium of Solid Modeling and CAD/CAM Applications*, ACM Press, p. 371-378.

Lamure, H. and D. Michelucci, 1998. Qualitative Study of Geometric Constraints. In *Geometric Constraint Solving and Applications,* B. Brüderlin and D. Roller Ed., Springer, p. 234-258.

Latham, R. and A. Middleditch. 1996. Connectivity analysis: a tool for processing geometric constraints. *Computer-Aided Design*, 28(11), p.917-928.

Owen, J., 1991. Algebraic solution for geometry from dimensional constraints. *Proceedings of the first ACM Symposium on Solid Modeling and CAD/CAM Applications*, ACM Press, p. 397-407.

Schreck, P., 2001. Robustness in CAD Geometric Construction. *Proceedings of 5th International Conference IV2001*, p.111-116.

Schreck, P. and E. Schramm, 2006. Using the invariance under the similarity group to solve geometric constraint systems, Computer-Aided design, 38(5), p. 475-484.

Schreck, P. and P. Mathis, 2006 Geometrical constraint system decomposition: a multi-group approach. *International Journal of Computational Geometry and Applications*, 16(5), p. 431-442

Sunde, G., 1987. Specification of shape by dimensions and other geometric constraints. Proceedings of the Eurographics Workshop on Intelligent CAD systems,, Noordwisjkerout.

Thierry, S., P. Mathis, P. and P. Schreck, 2007. Towards an homogeneous handling of under-constrained and well-constrained systems of geometric constraints. *Proceedings of the 21st ACM Symposium on Applied Computing (SAC'07)*, Seoul, South Korea, p. 773-777

Wintz, J. and P. Schreck, 2006. Compilation de systèmes à base de connaissances pour la résolution symbolique de contraintes géométriques. *Actes des journées AFIG 2006*, Bordeaux.

Zhang, G.-F., and X.-S. Gao, 2006. Well-constrained Completion and Decomposition for Under-constrained Geometric Constraint Problems. I*nternational Journal of Computational Geometry & Applications*, 16(5), p.461-478