Une approche orientée hiérarchie de contraintes pour la résolution de contraintes géométriques

Christophe Jermann, Hiroshi Hosobe

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»

UNE APPROCHE ORIENTÉE HIÉRARCHIE DE CONTRAINTES POUR LA RÉSOLUTION DE CONTRAINTES GÉOMÉTRIQUES

Christophe JERMANN **Hiroshi HOSOBE**

LINA - Université de Nantes NII
France Japon
Christophe.Jermann@univ-nantes.fr Hosobe@nii.ac.jp

RÉSUMÉ : *Nous proposons une approche utilisant des préférences sur les contraintes afin de gérer les problèmes de contraintes géométriques sur-contraints. Cette approche s'appuie sur le paradigme des hiérarchies de contraintes dont les méthodes de résolution utilisent des algorithmes de graphes proches de ceux employés en résolution de contraintes géométriques. Cette approche nous semble très bien adaptée aux applications interactives en CAO : l'esquisse sur laquelle l'utilisateur impose usuellement ses contraintes constitue une indication des attentes de l'utilisateur et peut être intégrée comme un jeu de contraintes faibles dans le problème à résoudre ; ainsi, tout problème géométrique défini sur une esquisse devient-il sur-contraint.*

MOTS-CLÉS : *Contraintes géométriques ; Préférences ; Hiérarchies de contraintes ; Décomposition*

1. INTRODUCTION

La résolution de problèmes de contraintes géométriques est un sujet de recherche actif depuis plus de trente ans et de nombreuses méthodes ont été proposées afin de résoudre les problèmes de ce type apparaissant dans les applications de CAO, robotique, dessin, biologie moléculaire et tant d'autres (cf. (Jermann, Trombettoni, Neveu & Mathis 2006) pour un état de l'art du domaine). La plupart de ces méthodes opèrent toutefois sous l'hypothèse de bonne constriction du problème, autrement dit elles supposent que le problème à résoudre admet seulement un ensemble fini de solutions et qu'il n'a par conséquent ni trop ni trop peu de contraintes. Cette hypothèse n'est pas raisonnable en pratique, en particulier pour les outils interactifs dans lesquels l'utilisateur introduit ses contraintes de façon incrémentale sur une esquisse, le problème correspondant étant alors alternativement sur-contraint (trop de contraintes, aucune solution) et sous-contraint (trop peu de contraintes, infinité de solutions). Certains travaux récents abordent cette problématique (par exemple (Essert-Villard, Schreck & Dufourd 2000, Joan-Arinyo, Soto-Riera, Vila-Marta & Vilaplana-Pasto 2003, Hoffmann, Sitharam & Yuan 2004, Trombettoni & Wilczkowiak 2006)). Nous proposons dans cet article une alternative à ces travaux basée sur l'utilisation du paradigme des hiérarchies de contraintes. Cette approche nous semble pertinente pour plusieurs raisons :

− Dans les outils interactifs, l'utilisateur impose ses contraintes sur une esquisse qui représente très souvent une expression implicite de ses attentes ; cette esquisse décrit une position initiale pour chacune des entités du problème, positions qui peuvent être transcrites comme des contraintes *faibles* dans le problème à résoudre. Ainsi, tout problème de contraintes géométrique devient-il sur-contraint puisque les positions initiales des entités dans l'esquisse ne respectent en général pas les contraintes imposées par l'utilisateur.

− L'introduction de la notion de préférences sur les contraintes offre de nouvelles possibilités aux utilisateurs qui, au lieu d'ajouter et supprimer des contraintes à leur modèle afin d'obtenir la solution qu'ils souhaitent, peuvent simplement jouer avec les priorités données aux contraintes afin d'obtenir celle-ci.

− Les algorithmes de résolution de hiérarchies de contraintes et ceux de résolution de contraintes géométriques présentent des similarités qui rendent leur intégration assez aisée.

La suite de cet article est organisée comme suit : la section 2 rappelle les concepts nécessaires à l'exposé de la contribution, tant en résolution de contraintes géométriques qu'en résolution de hiérarchies de contraintes ; la section 3 présente la méthode que nous proposons, illustre son fonctionnement et son intérêt sur un exemple et donne ses propriétés principales ; la section 4 conclue l'article et offre des perspectives au travail qu'il présente.

2. FONDEMENTS

2.1. Contraintes géométriques

Un problème de contraintes géométriques est constitué d'un ensemble d'entités géométriques (points, plans, droites, sphères, ...) soumises à un ensemble de relations géométriques (distances, angles, incidences, colinéarités, ...). Chaque entité géométrique est représentée par un ensemble de variables dont les valeurs déterminent sa position, son orientation et ses dimensions. Chaque relation est représentée par un ensemble d'équations et d'inégalités portant sur les variables des entités qu'elle contraint. La multiplicité des variables par entité et des

équations/inégalités par contrainte induit la notion de degrés de libertés (DDL) : une entité dispose d'un nombre de DDL égal aux nombre de variables nécessaires pour définir sa position, orientation et ses dimensions de façon unique ; par exemple, un cercle en 2D dispose de 3 DDL, 2 pour la position de son centre et 1 pour son rayon ; une contrainte quant à elle retranche aux entités sur lesquelles elle porte un nombre de DDL égal au nombre de variables qu'elle permet de fixer, généralement égal au nombre d'équations la représentant ; par exemple, une contrainte de distance entre deux points en 2D retranche 1 DDL aux points qu'elle contraint, son équation de la forme $(x_1-x_2)^2+(y_1-y_2)^2=d^2$ permettant de fixer une seule variable en général.

L'approche à base de graphe proposée par Hoffmann et al. (Hoffmann, Lomonosov & Sitharam 2000), appelée HLS dans cet article, est souvent considérée comme la méthode de résolution de contraintes géométriques la plus générale du fait qu'elle s'applique à des problèmes en dimensions quelconques et n'est pas limitée à certains types d'entités ni de relations. Son étape principale utilise une abstraction du problème sous forme de réseau biparti : la source S est reliée à chaque contrainte c par un arc $S \rightarrow c$ de capacité égale au nombre de DDL de c ; chaque contrainte c est reliée à chaque entité e qu'elle contraint par un arc $c \rightarrow e$ de capacité infinie ; chaque entité e est reliée au puits T par un arc $e \rightarrow T$ de capacité égale au nombre de DDL de e. Ainsi, une distribution de flot maximum dans ce réseau représente une distribution optimale des DDL des contraintes sur les DDL des entités, autrement dit elle indique quelles contraintes permettront de fixer quelles entités. Lorsque le problème est bien-contraint, le flot maximum est parfait, c'est-à-dire que tous les arcs $S \rightarrow c$ et tous les arcs $e \rightarrow T$ sont saturés. Réciproquement, lorsque certains arcs ne peuvent pas être saturés lors du calcul d'un flot maximum, cela indique que le problème n'est pas bien contraint ; chaque arc $S \rightarrow c$ non saturé induit une partie sur-contrainte du problème, et chaque arc $e \rightarrow T$ non saturé induit une partie sous-contrainte du problème.

La méthode HLS permet de traiter les dépendances cycliques entre contraintes : elle peut identifier des sous-ensembles minimum de contraintes devant être résolues simultanément. À cette fin, le réseau est construit incrémentalement en introduisant les contraintes une à une avec tous leurs arcs incidents. À chaque nouvelle contrainte, le flot est mis à jour ; si la contrainte est saturée ainsi que toutes les entités auxquelles elle est reliée, un sous-ensemble de contraintes dépendantes a été identifié[1] ; il est alors supprimé du réseau et l'algorithme reprend ; autrement, la contrainte suivante est introduite et l'algorithme réitéré.

Dans les applications géométriques, il arrive souvent qu'un problème ne soit bien-contraint que modulo les déplacements. La méthode HLS peut travailler modulo les déplacements en utilisant une capacité fictive additionnelle valant 3 en 2D et 6 en 3D. À chaque itération, cette capacité est ajoutée temporairement à celle de l'arc $S \rightarrow c$ reliant la source à la contrainte c nouvellement introduite. Si la contrainte c est saturée ainsi que les entités auxquelles elle est reliée, un sous-problème bien-contraint modulo les déplacements est identifié ; autrement, la capacité additionnelle est retirée, le flot mis à jour, et l'algorithme procède à l'itération suivante.

Exemple 1 *Considérons un problème de contraintes géométriques 3D dans lequel un point P (3 DDL) et deux droites L_1 et L_2 (4 DDL chaque) sont soumises à cinq contraintes : c_1 fixe la position de P, c_2 fixe la position et l'orientation de L_1, c_3 impose à L_1 et L_2 d'être parallèles, c_4 impose à L_2 de passer par P et c_5 impose une distance orthogonale égale à 4 entre L_1 et L_2. Du fait des contraintes c_1 et c_2, ce problème est fixé dans le repère global et la capacité additionnelle permettant à la méthode* HLS *de travailler modulo les déplacements est inutile.*

nom	*relation*	*DDL*
c_1	$fixer(P)$	3
c_2	$fixer(L_1)$	4
c_3	$parallelisme(L_1,L_2)$	2
c_4	$incidence(P,L_2)$	2
c_5	$distance(L_1,L_2,4)$	1

Ce problème correspond au réseau biparti présenté dans la figure 1. La méthode HLS *introduit d'abord la contrainte c_1 et distribue le flot ; c_1 est saturée ainsi que le point P auquel elle est liée ; cela signifie que c_1 peut être résolue séparément afin de déterminer P. Ensuite, c_2 est introduite, distribuée, saturée ainsi que L_1 et c_2 peut donc être résolue séparément pour fixer L_1. c_3 est alors introduite et distribuée mais ne peut saturer L_2. c_4 est introduite à son tour et permet de saturer L_2 ; cela signifie que c_3 et c_4 peuvent être résolues ensemble pour fixer L_2. Toutefois, il reste encore la contrainte c_5 alors que toutes les entités ont déjà été déterminées : le problème est sur-contraint et la méthode* HLS *termine en signalant une erreur.*

2.2. Hiérarchies de contraintes

Les hiérarchies de contraintes (Borning, Duisberg, Freeman-Benson, Kramer & Woolf 1987, Borning, Freeman-Benson & Wilson 1992, Wilson 1993) permettent la prise en compte de préférences (ou priorités) sur les contraintes. Les contraintes de priorité maximale, dites contraintes *requises*, doivent nécessairement être satisfaites, alors que les contraintes de priorités moindre, dites *préférentielles* et souvent associées aux adjectifs *fortes*, *moyennes* et *faibles*, peuvent ne pas être satisfaites. Les contraintes de même priorité sont également préférées et appartiennent au même niveau de la hiérarchie.

L'ensemble des solutions d'une hiérarchie de contraintes dépend du choix d'un critère de prise en compte des priorités. Il existe des critères globaux qui maximise

[1] Ici, un processus de minimisation intervient qui supprime les entités et contraintes inutiles afin d'obtenir un sous-problème dépendant minimal.

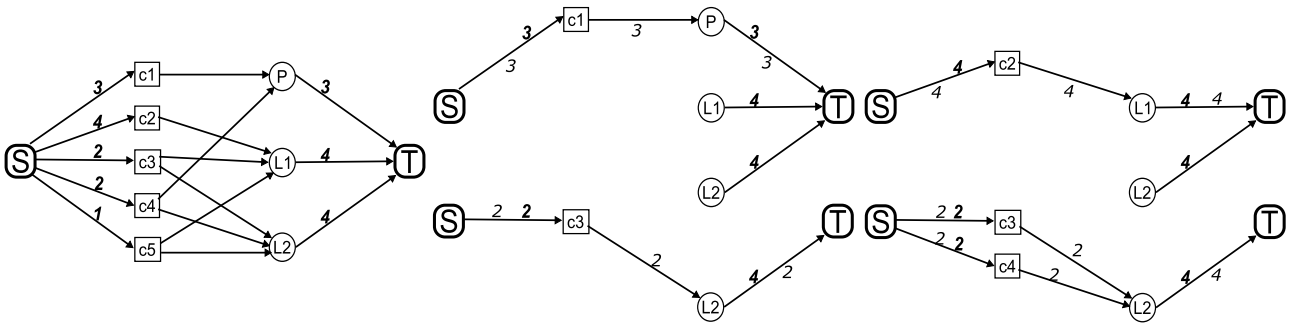

Figure 1: Application de la méthode HLS. À gauche : le réseau biparti représentant le problème géométrique. À droite : les quatre itérations de l'algorithme.

la satisfaction de l'ensemble des contraintes du problème, auxquels on oppose des critères locaux travaillant dans chaque niveau hiérarchique individuellement. Les critères globaux sont en général traités par des algorithmes d'optimisation alors que les critères locaux peuvent être traités à l'aide d'algorithmes de graphe. Dans cet article nous nous intéressons au critère local *locally-predicate-better (LPB)* qui demande qu'un maximum de contraintes soient satisfaites dans chaque niveau hiérarchiques en commençant par le plus prioritaire : une affectation a_1 est LPB-meilleure qu'une autre affectation a_2 ssi a_1 satisfait les mêmes contraintes que a_2 dans les k premiers niveaux hiérarchiques et au moins une contrainte de plus que a_2 au niveau hiérarchique $k+1$; a_1 est une LPB-solution si aucune affectation n'est LPB-meilleure qu'elle. Ainsi, les LPB-solutions d'une hiérarchie de contraintes satisfont un sous-ensemble maximal de contraintes à chaque niveau hiérarchique.

Les approches à base de graphes pour la résolution de hiérarchies de contraintes emploient une abstraction du problème sous forme de graphe biparti où les sommets sont les contraintes et les variables et une arête lie une contrainte c à une variable v si c porte sur v. les sommets-contraintes sont de plus pondérés par leur priorité. L'algorithme proposé par Gangnet et Rosenberg (Gangnet & Rosenberg 1993), appelé GR dans cet article, adopte l'approche DeltaBlue (Freeman-Benson, Maloney & Borning 1990) et est certainement la méthode à base de graphes la plus générale pour les hiérarchies de contraintes. Cet algorithme identifie un sous-ensemble de contraintes *LPB-maximal* au moyen d'un algorithme de couplage maximum incrémental qui couple d'abord les contraintes de plus haute priorité. Pour ce faire, il effectue la recherche de chemins augmentant en partant des contraintes de plus haute priorité en premier. Ainsi, les contraintes plus prioritaires sont couplées d'abord et comme un sommet couplé n'est jamais découplé par la suite dans un algorithme de couplage maximum, si un sommet ne peut être couplé, c'est nécessairement dû à une contrainte de priorité supérieure ou égale ; autrement dit, le nombre de contraintes couplées dans chaque niveau hiérarchique est maximal. Cela respecte bien la définition du critère LPB. Cet algorithme est incrémental en ce sens que l'ajout ou le retrait d'une contrainte, en contexte interactif, nécessite le recalcul du couplage seulement à partir du niveau hiérarchique de cette contrainte : les contraintes de priorité supérieure déjà couplées ne sont pas remises en cause.

Exemple 2 *Considérons une hiérarchie de contraintes discrètes où le domaine des trois variables x_1, x_2 et x_3 est $\{0,1,2,3\}$ et les quatre contraintes sont :*

priorite	*nom*	*relation*
requise	c_1	$x_1 = x_2$
$forte$	c_2	$x_2 + 1 = x_3$
$faible$	c_3	$x_1 = 0$
$faible$	c_4	$x_3 = 3$

Considérons les quatre affectations suivantes qui respectent toute la contrainte requise c_1 : $a_1 = (x_1 = 0, x_2 = 0, x_3 = 1)$, $a_2 = (1,1,2)$, $a_3 = (2,2,3)$, et $a_4 = (0,0,3)$; ces affectations satisfont respectivement les ensembles de contraintes suivantes par niveau hiérarchique : $sat(a_1) = (\{c_2\},\{c_3\})$, $sat(a_2) = (\{c_2\},\emptyset)$, $sat(a_3) = (\{c_2\},\{c_4\})$ et $sat(a_4) = (\emptyset,\{c_3,c_4\})$. Selon le critère LPB, a_2 et a_4 sont toutes deux dominées par a_1 et a_3. Toutefois, a_1 et a_3 sont incomparable pour ce critère car elles satisfont le même ensemble de contraintes fortes mais deux sous-ensembles disjoints de contraintes faibles.

Cette hiérarchie correspond au graphe biparti présenté dans la figure 2. GR *couple d'abord la contrainte requise c_1 avec la variable x_1 par exemple. Ensuite, il tente de coupler c_2 qui est la contrainte la plus prioritaire parmi les contraintes restantes ; c_2 est ainsi couplée à x_2 par exemple. Enfin, les contraintes c_3 et c_4 sont couplées ; l'algorithme peut commencer par l'une ou l'autre indifféremment puisqu'elles ont même priorité. Toutefois ces contraintes ne peuvent être couplées toutes les deux puisqu'il ne reste qu'une seule variable à associer ; le résultat obtenu dépendra donc de celle qui est couplée en premier. Si* GR *commence par c_3 (resp. c_4), il retournera l'ensemble de contraintes LPB-maximal $\{c_1,c_2,c_3\}$ (resp. $\{c_1,c_2,c_4\}$) comme montré au centre (resp. à droite) de la figure 2.*

3. HIÉRARCHIES DE CONTRAINTES GÉOMÉTRIQUES

Nous proposons d'adapter le paradigmes des hiérarchies de contraintes aux problèmes de contraintes géométriques. Cela nécessite l'introduction de la notion

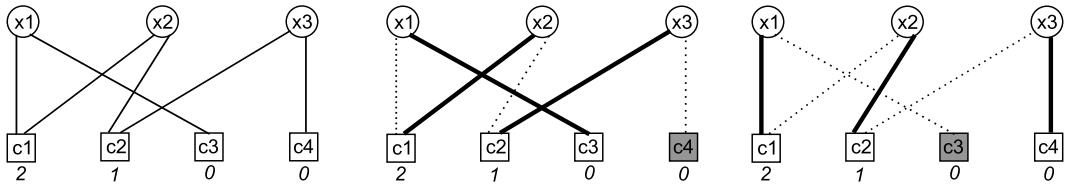

Figure 2: Application de GR. À gauche : le graphe biparti représentant la hiérarchie de contraintes. Au centre et à droite : les deux LPB-solutions retournées par GR ; les arêtes couplées sont en gras, la contrainte négligée en gris.

de priorité sur les contraintes géométriques ainsi qu'un algorithme adapté à leur traitement.

3.1. Préférences sur les contraintes géométriques

Nous pensons que l'introduction de la notion de priorité dans les outils existants est aisée et peut se faire de façon transparente pour les utilisateurs. En effet la plupart des outils demandent à l'utilisateur de définir l'ensemble des entités du problème de façon graphique via une esquisse sur laquelle ils imposent ensuite leurs contraintes. De fait, les utilisateurs positionnent les entités dans l'esquisse approximativement comme ils voudraient qu'ils soient dans la solution du problème, autrement dit l'esquisse fournit en général une indication des attentes de l'utilisateur. La configuration[2] (position, orientation et dimensions) de chaque entité dans cette esquisse peut être transcrite en un ensemble de contraintes *faibles* qui demandent de respecter préférentiellement cette configuration. Ainsi tout problème de contraintes géométriques devient-il sur-contraint puisque la configuration des entités dans l'esquisse ne respecte pas, en général, les contraintes imposées par l'utilisateur. Toutefois, comme l'esquisse est introduite comme un jeu de contraintes préférentielles, elle peut être relaxée au besoin afin de permettre la satisfaction des contraintes de l'utilisateur.

De façon toujours transparente pour l'utilisateur, il est possible d'introduire par défaut toutes les contraintes qu'il impose comme des contraintes préférentielles *fortes* plutôt qu'en tant que contraintes requises. Ainsi, si la spécification qu'il définit s'avère sur-contrainte (indépendamment de l'esquisse), elle sera relaxée automatiquement de sorte à toujours fournir une solution au lieu de n'en fournir aucune. Comme les contraintes auraient alors toutes la même priorité, la relaxation serait toutefois effectuée au hasard et ne conduirait pas nécessairement à la solution souhaitée. En explicitant l'utilisation des préférences, on pourrait alors permettre à l'utilisateur d'indiquer, parmi les contraintes qu'il impose, lesquelles il souhaite privilégier (elles seraient transférées à la priorité *requise*) et/ou lesquelles il estime secondaires (elles seraient transférées à la priorité *moyenne*). Il est bien entendu possible d'augmenter le nombre de niveaux de priorités afin de prendre en compte des préférences fines entre les contraintes. Ainsi, l'utilisateur pourrait obtenir la solution souhaitée en jouant avec les priorités d'un même ensemble de contraintes plutôt qu'en modifiant cet ensemble de contraintes.

En résumé, les préférences lèvent le problème de la redondance, permettent la prise en compte des informations implicites du problème (esquisse) et offrent un moyen alternatif pour débugger une spécification sans avoir à modifier les contraintes.

3.2. Algorithme

La méthode GR ne peut pas être appliquée telle quelle aux hiérarchies de contraintes géométriques car elle ne permet pas la gestion de contraintes à sorties multiples (contraintes retirant plusieurs DDL des entités qu'elles contraignent) ni la prise en compte des dépendances cycliques entre contraintes, deux aspects fondamentaux des problèmes de contraintes géométriques. Toutefois, son principe peut être transposé sans difficulté à l'algorithme HLS afin d'obtenir un algorithme de résolution de hiérarchies de contraintes géométriques.

La méthode que nous proposons consiste à appliquer l'algorithme HLS en introduisant les contraintes dans le réseau par ordre décroissant de priorité : les contraintes requises sont d'abord introduites, puis les *fortes*, puis les *moyennes*, et ainsi de suite. À chaque distribution de flot, si la nouvelle contrainte est saturée ainsi que les entités auxquelles elle est reliée, un sous-problème dépendant a été identifié ; il est retiré du problème est l'algorithme est réitéré. Si la contrainte est saturée mais pas les entités, l'algorithme procède à l'itération suivante. Autrement, si la contrainte ne peut être saturée cela indique la présence d'un sous-problème sur-contraint ; la contrainte dernièrement introduite est alors retirée et le flot restauré à l'état précédant son introduction. Les contraintes ainsi écartées le sont donc forcément du fait de contraintes introduites précédemment, c'est-à-dire de contraintes de priorités supérieures ou égales, ce qui respecte bien le principe du critère LPB. Les contraintes qui ont été retenues constituent donc un sous-ensemble LPB-maximal.

Exemple 3 *Les quatre itérations présentées en partie droite de la figure 1 sont aussi une illustration de notre algorithme sur le problème présenté à l'exemple 1 où on a associé aux contraintes les priorités suivantes :* c_1 *et* c_2 *sont requises,* c_3 *est forte,* c_4 *et* c_5 *sont moyennes. Ces priorités impliquent bien le même ordre d'introduction des*

[2] Outre une configuration initiale des entités, l'esquisse fournit aussi des informations de position, d'orientation et de dimensions relatives entre les entités qui pourraient aussi être introduites comme des contraintes *faibles*, ou *très faibles* du fait de leur nature redondante, dans le problème à résoudre.

contraintes que pour l'algorithme HLS *original et conduisent donc au même résultat. La dernière itération ne provoque toutefois pas la levée d'une erreur comme pour* HLS *mais conduit simplement à relaxer la contrainte* c_5. *Notre algorithme retourne donc le sous-ensemble LPB-maximal de contraintes* $\{c_1, c_2, c_3, c_4\}$.

Cet algorithme peut-être utilisé de façon incrémentale en contexte interactif, tout comme GR, l'ajout ou le retrait d'une contrainte ne nécessitant le recalcul du flot qu'à partir du niveau hiérarchique de cette contrainte.

Il est en temps polynomial puisque la distribution d'un arc issu de la source se fait en temps quadratique dans ce type de réseaux biparti et que cette distribution est réalisée une fois pour chaque contrainte du problème au plus. Ainsi l'algorithme est globalement en $O(n^3)$ pour une hiérarchie à n contraintes, indépendamment du nombre de niveaux hiérarchiques.

Notre algorithme ne permet malheureusement pas toujours d'extraire un sous-problème bien-contraint d'une hiérarchie de contraintes géométriques même s'il en existe un. Pour le comprendre, considérons à nouveau l'exemple 3. Les contraintes c_4 et c_5 ayant même priorité dans cet exemple, c_5 aurait pu être introduite dans le réseau avant c_4 conduisant l'algorithme à relaxer c_4 et à retourner le sous-ensemble LPB-maximal $\{c_1, c_2, c_3, c_5\}$. Or ce sous-ensemble de contraintes est sous-contraint car c_3 et c_5 ne permettent pas de déterminer complètement la configuration de L_2 (3 DDL retirés par les contraintes contre 4 DDL pour L_2). L'ordre dans lequel les contraintes d'un même niveau hiérarchique sont considérées peut donc influer sur le résultat obtenu. Il serait possible de calculer tous les sous-ensembles LPB-maximaux mais comme il sont potentiellement en nombre exponentiel l'algorithme ne serait alors plus polynomial. Il en résulte que le critère LPB ne permet pas de toujours extraire un sous-problème bien-contraint d'une hiérarchie de contraintes. L'utilisation d'autres critères est une piste pour lever cette limitation. L'utilisation des esquisses comme un jeu de contraintes préférentielles est aussi une solution puisque ces seules contraintes suffisent à rendre le problème bien-contraint : elles peuvent compléter n'importe quel ensemble de contraintes LPB-maximal de sorte à le rendre bien-contraint.

4. CONCLUSION

Nous avons présenté une méthode permettant la prise en compte des préférences sur les contraintes dans les problèmes de contraintes géométriques. Les préférences augmentent l'expressivité des modèles géométriques en permettant l'exploitation aisée des informations implicites d'une spécification (esquisse), permettent l'obtention d'une solution que le modèle soit sur-contraint ou sous-contraint, et offrent une alternative au débuggage des modèles pour l'obtention de la solution souhaitée.

Cette méthode réalise la fusion d'un algorithme classique de résolution de hiérarchies de contraintes selon le critère LPB et d'un algorithme classique de résolution de contraintes géométriques. La similarité de ces méthodes et leur utilisation d'une abstraction du problème sous forme de graphes rend cette fusion naturelle. L'algorithme résultant maintient les bonnes propriétés des deux algorithmes originaux.

Le potentiel de cette méthode apparaît important puisqu'elle n'est pas limitée aux problèmes de contraintes géométriques mais peut s'appliquer à toute hiérarchie de contraintes contenant des contraintes à sorties multiples et des dépendances cycliques entre contraintes, levant ainsi les principales limitations de l'algorithme GR. Notre algorithme sera ainsi implémenté dans une bibliothèque d'algorithmes de résolution de hiérarchies de contraintes afin d'être confronté à des applications issues de domaines différents.

Cet article ne fait toutefois qu'ébaucher la méthode : les détails des algorithmes originaux ont été négligés afin de clarifier le propos mais devront être réintégrés afin de définir précisément l'algorithme et de prouver ses propriétés. Une comparaison complète avec les méthodes existantes devra alors être menée. Nous espérons cette comparaison favorable puisque notre algorithme semble subsumer à la fois HLS et GR deux algorithmes de référence dans leurs domaines respectifs. Des pistes ont par ailleurs été évoquées pour lever la principale limitation de notre méthode.

References

Borning, A., Duisberg, R., Freeman-Benson, B., Kramer, A. & Woolf, M. (1987). Constraint hierarchies, *Proc. ACM OOPSLA*, pp. 48–60.

Borning, A., Freeman-Benson, B. & Wilson, M. (1992). Constraint hierarchies, *Lisp Symbolic Comput.* **5**(3): 223–270.

Essert-Villard, C., Schreck, P. & Dufourd, J. (2000). Sketch-based pruning of a solution space within a formal geo. constraint solver, *Arti. Intelligence* **124**: 139–159.

Freeman-Benson, B. N., Maloney, J. & Borning, A. (1990). An incremental constraint solver, *Communications of the ACM* **33**(1): 54–63.

Gangnet, M. & Rosenberg, B. (1993). Constraint programming and graph algorithms, *Annals of Mathematics and Artificial Intelligence* **8**(3–4): 271–284.

Hoffmann, C., Lomonosov, A. & Sitharam, M. (2000). Decomposition plans for geometric constraint systems, *Proc. J. Symbolic Computation 2000*.

Hoffmann, C., Sitharam, M. & Yuan, B. (2004). Making constraint solvers more usable: Overconstraint prob-

lem, *Computer Aided Design* **36**(4): 377–399.

Jermann, C., Trombettoni, G., Neveu, B. & Mathis, P. (2006). Decomposition of geometric constraint systems: a survey, *Internation Journal of Computational Geometry and Applications* **16**(5-6): 379–414.

Joan-Arinyo, R., Soto-Riera, A., Vila-Marta, S. & Vilaplana-Pasto, J. (2003). Transforming an under-constrained geometric constraint problem into a well-constrained one, *SM'03: Proc. of Solid modeling and applications*.

Trombettoni, G. & Wilczkowiak, M. (2006). GPDOF: a fast algorithm to decompose under-constrained geometric constraint systems: Application to 3D model reconstruction, *Int. Journal of Computational Geometry and Applications (IJCGA)* **16**.

Wilson, M. (1993). Hierarchical constraint logic programming (Ph.D. dissertation), *Technical Report 93-05-01*, Dept. Comput. Sci. Eng., Univ. Washington.