Algorithmes pour la détection de rigidités dans les CSP géométriques

Christophe Jermann
AI Lab. - EPFL, 1015 Lausanne, Switzerland
e-mail : Christophe.Jermann@epfl.ch

Bertrand Neveu **Gilles Trombettoni**
Equipe COPRIN, INRIA-I3S/CNRS-CERMICS,
2004 route des lucioles, BP 93, 06902 Sophia Antipolis, France
e-mail : Bertrand.Neveu@sophia.inria.fr
Gilles.Trombettoni@sophia.inria.fr

Résumé

La rigidité structurelle est une généralisation du théorème de Laman, une caractérisation de la rigidité des systèmes à barres en 2D. Elle est généralement considérés comme une bonne heuristique pour identifier des sous-parties rigides dans les CSP géométriques (GCSP), mais peut en réalité se tromper sur des sous-systèmes très simples car elle ne tient pas compte des propriétés géométriques vérifiées par les objets.

Dans le présent article, nous remettons en cause l'algorithme à base de flots proposé par Hoffmann *et al.* pour identifier des sous-GCSP rigides. Nous montrons que cet algorithme peut échouer à cause de la propriété de rigidité structurelle sur laquelle il est basé, mais aussi de part sa conception. Nous introduisons un nouvel algorithme de flot basé sur la rigidité structurelle étendue, une nouvelle caractérisation de la rigidité que nous avons proposée. Nous montrons que ce nouvel algorithme est correct en 2D et 3D et peut être utilisé pour répondre aux principales questions liées à la rigidité : décider de la rigidité d'un GCSP, identifier des sous-systèmes bien- ou sur-rigides, les minimiser, ...

1 Introduction

Le problèmes de satisfaction de contraintes géométriques (GCSP) apparaissent dans de nombreuses applications : CAO, robotique et biologie moléculaire en sont trois exemples. Le concept de rigidité est au cœur de nombre de ces problèmes : décider si une structure est rigide, déterminer les parties bien-, sur- et sous-rigides.

Par ailleurs, plusieurs méthodes [Kramer, 1992; Bouma *et al.*, 1995; Dufourd *et al.*, 1998; Lamure et Michelucci, 1998; Hoffmann *et al.*, 2000; Jermann *et al.*, 2000] de résolution de GCSP utilisent le concept de rigidité de façon implicite ou explicite. En particulier, les méthodes

de décomposition géométriques produisent des séquences de sous-GCSP rigides pouvant être résolus séparément puis assemblés.

Les techniques de caractérisation de rigidité peuvent être classées en deux approches : les *approches à base de règles* [Bouma *et al.*, 1995; Kramer, 1992] sont basées sur l'utilisation d'un répertoire de formes rigides connues qui ne peut couvrir tous les cas de figure, alors que les *approches structurelles* [Hoffmann *et al.*, 1997; Lamure et Michelucci, 1998] utilisent des algorithmes de flots (ou de couplage maximum) pour vérifier une propriété appelée *rigidité structurelle*, basée sur un compte des degrés de liberté dans le GCSP.

Les approches structurelles sont plus générale, bien que la rigidité structurelle ne soit qu'une approximation de la rigidité. Des heuristiques, sous forme de règles géométriques, ont parfois été proposées pour améliorer cette propriété, sans jamais permettre de caractériser complètement la rigidité. Dans [Jermann *et al.*, 2002], nous avons proposé une nouvelle caractérisation de la rigidité, appelée *rigidité structurelle étendue*, qui subsume la propriété initiale, même lorsqu'elle est améliorée par des heuristiques.

Dans le présent article, nous proposons de revenir sur l'algorithme `Dense` [Hoffmann *et al.*, 1997] pour identifier des sous-GCSP rigides. Après avoir introduit le sujet en section 2, nous montrons (cf. section 3)que cet algorithme à base de flots possède des limites dues en partie à son utilisation de la rigidité structurelle, et d'autre part au fait qu'il ne tient pas compte de la géométrie. Nous présentons 2 modifications de l'algorithme Dense qui permette d'obtenir un nouvel algorithme complet et correct vis à vis de la géométrie. Nous expliquons pour finir comment notre algorithme peut être utilisé pour répondre aux principaux problèmes liés au concept de rigidité.

2 Définitions

Dans cette section, nous donnons les définitions nécessaires à la compréhension de la contribution de cet article.

2.1 Problème de satisfaction de contraintes géométriques

Définition 1 GCSP
Un **problème de satisfaction de contraintes géométriques** *(GCSP)* $S = (O, C)$ *est composé d'un ensemble* O *d'objets géométriques et d'un ensemble* C *de contraintes géométriques.*
$S' = (O', C')$ *est un* **sous-GCSP** *de* $S = (O, C)$*, noté* $S' \subset S$*, ssi* $O' \subset O$ *et* $C' = \{c \in C | c$ *ne porte que sur des objets dans* $O'\}$*, c-a-d. que* S' *est induit par* O'*.*

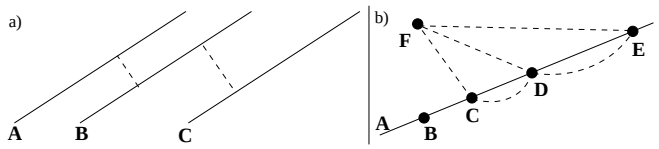

FIG. 1 – Deux exemples de GCSP

La figure 1-a présente un GCSP en 2D constitué de 3 droites liées par 2 parallélismes et deux distances droite-droite. La figure 1-b présente un GCSP en 3D constitué d'une droite et 5 points,

liés par 4 incidences point-droite et 5 distances point-point.

Hypothèses : Nous supposerons que les objets géométriques sont indéformables (pas de cercle à rayon variable par exemple) et que les contraintes ne peuvent porter que sur les positions et orientations relatives des objets (pas de fixation par rapport au repère global par exemple). Ces limitations simplifient la définition des caractérisation structurelles de la rigidité et sont nécessaires aux méthodes de résolution de GCSP basées sur la rigidité.

Sous ces hypothèses, une solution d'un GCSP est composée d'un position et d'une orientation pour chacun de ses objets qui satisfont toutes ses contraintes. Pour être résolu, un GCSP est généralement transformé en un système d'équations, chaque objet étant représenté par des variables définissant sa position et son orientation, et chaque contrainte devenant un sous-système d'équations sur les variables des objets qu'elle contraint.

2.2 Rigidité

La rigidité d'un GCSP se définit à partir des mouvements que celui-ci admet[1]. On distingue deux types de mouvements : les déformations, qui ne préservent pas les positions et orientations relatives des objets, et les déplacements (rotations et translations) qui les préservent. Intuitivement, un GCSP est **bien-rigide** s'il n'admet aucune déformation et admet tous les déplacements de l'espace géométrique considéré[2]. Un GCSP qui admet des déformations est dit **sous-rigide**, alors qu'un GCSP n'admettant pas certains déplacements, ou aucune solution, est dit **sur-rigide**. Des définitions plus formelles peuvent être trouvées dans [Jermann, 2002].

Dans l'exemple présenté en figure 1-b, le sous-GCSP CDF est rigide : un triangle est indéformable et admet tous les déplacements de l'espace ; AF est sous-rigide puisque la droite A et le point F ne sont liés par aucune contrainte ; le sous-GCSP $ACDEF$ est quant à lui sur-rigide : il est génériquement impossible de placer le point F à l'intersection des 3 sphères (contraintes de distance) dont les centres C, D et E sont alignés.

2.3 Rigidité structurelle

La **rigidité structurelle** correspond à une analyse des **degrés de liberté** (DDL) dans le GCSP. Intuitivement, un DDL représente un mouvement indépendant dans le GCSP. Plus formellement :

Définition 2 Degré de liberté (DDL)
- Objet o : DDL(o) est le nombre de variables indépendantes définissant la position et l'orientation de o.
- Contrainte c : DDL(c) est le nombre d'équations indépendantes représentant la contrainte c.
- GCSP $S = (O, C)$: DDL(S)=$\sum_O$DDL(o) - $\sum_C$DDL(c).

En 3D, un point possède 3 DDL et une droite 4 ; l'incidence point-droite retire 2 DDL, et une distance point-point 1. Ainsi, les sous-GCSP ACD, CDF et AF issus de la figure 1-b ont respectivement 5, 6 et 7 DDL.

La rigidité structurelle est une généralisation du théorème de Laman [Laman, 1970] qui caractérise la rigidité générique des systèmes à barres en 2D. Elle est basée sur l'intuition suivante :

[1] En réalité, la rigidité et les mouvements se définissent au niveau de chaque solution d'un GCSP. Cependant, comme c'est généralement le cas en CAO, on considère la rigidité au niveau du GCSP comme représentant celle de toutes ses solutions car on souhaite étudier le système sans pour autant devoir le résoudre complètement.

[2] Cette seconde condition est toujours vérifiée sous l'hypothèse que nous avons posée sur les contraintes géométriques.

si un GCSP admet moins (resp. plus) de mouvements que le nombre de déplacements (égal à $\frac{d(d+1)}{2}$ en dimension d) admis par l'espace géométrique qui le contient, alors il est sur-(resp. sous-)rigide. Plus formellement :

Définition 3 Rigidité structurelle (s_rigidité)
Un GCSP $S = (O, C)$ en dimension d est **s_rigide** *ssi DDL(S) = $\frac{d(d+1)}{2}$ et S ne contient pas de sous-GCSP sur-s_rigide.*
S est sous-s_rigide *ssi DDL(S) ¿ $\frac{d(d+1)}{2}$ et S ne contient pas de sous-GCSP sur-s_rigide.*
S est sur-s_rigide *ssi $\exists S' \subseteq S$ tel que DDL(S') ¡ $\frac{d(d+1)}{2}$.*

En pratique, la rigidité structurelle est considérée comme une bonne approximation de la rigidité [Lamure et Michelucci, 1998; Hoffmann *et al.*, 1997]. L'écart entre s_rigidité et rigidité est en réalité important (cf. [Jermann *et al.*, 2002; Jermann, 2002]). Nous illustrons ici cet écart sur 2 sous-GCSP issus de la figure 1-b : $ABCD$ est s_rigide en 3D puisque DDL($ABCD$)=6, alors que ce sous-GCSP est en réalité sous-rigide : le point B peut bouger indépendamment des points C et D sur la droite A. $ACDE$ est quant à lui sur-s_rigid car DDL($ACDE$)=5, mais il s'agit en réalité d'un sous-GCSP bien-rigide.

2.4 Rigidité structurelle étendue

La rigidité structurelle étendue (notée es_rigidité) est basée sur le concept de *degré de rigidité* (DDR). Le degré de rigidité d'un GCSP représente le nombre de déplacements admis par celui-ci. Ce nombre dépend des propriétés géométriques vérifiées par les objets du GCSP. Par exemple, le DDR d'une paire de droites en 2D est 3 si ces droites sont quelconques, alors qu'il vaut 2 si elles sont parallèles ; Le parallélisme de ces droites peut être explicite (contrainte entre les droites) mais peut aussi être induit par l'ensemble des contraintes du GCSP contenant ces droites. Dans ce dernier cas, inférer le parallélisme (et donc déterminer le DDR) est équivalent à la preuve de théorème géométrique.

Le principe de la rigidité structurelle étendue est de comparer le nombre de DDL d'un GCSP au DDR de ce même GCSP, c-a-d. le nombre de mouvements au nombre de déplacements. On peut ainsi déterminer si un GCSP admet des déformations ou non.

Définition 4 Rigidité structurelle étendue (es_rigidité)
Un GCSP $S = (O, C)$ est **es_rigide** *ssi DDL(S)=DDR(S) et S ne contient pas de sous-GCSP sur-es_rigide.*
S est sous-es_rigide *ssi DDL(S) ¿ DDR(S) et S ne contient pas de sous-GCSP sur-es_rigide.*
S est sur-es_rigide *ssi $\exists S' \subseteq S$, DDL(S')¡ DDR(S').*

La es_rigidité subsume la s_rigidté ; par exemple, es_rigidité et rigidité correspondent exactement sur tout sous-GCSP de la figure 1. Nous renvoyons le lecteur à [Jermann *et al.*, 2002] pour une comparaison plus détaillée de ces deux propriétés.

2.5 Réseau Objet-Contrainte

Un GCSP $S = (O, C)$ peut être représenté par un réseau $G = (s, V, t, E, w)$ appelé *réseau objets-contraintes* (introduit dans [Hoffmann *et al.*, 1997]). La figure 2-a dépeint le réseau objets-contraintes correspondant au GCSP de la figure 1-a.

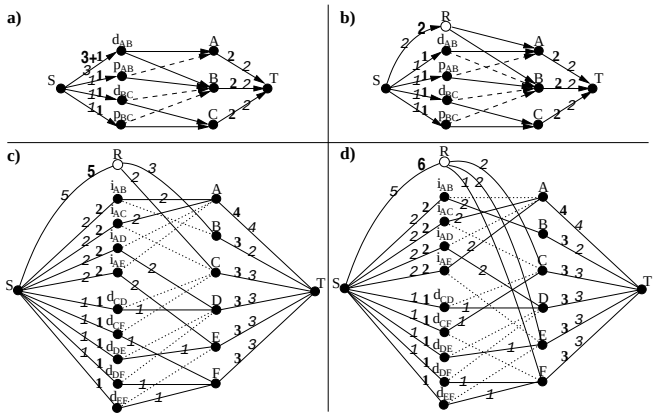

FIG. 2 – Réseaux objets-contraintes et distributions de flots par les algorithmes `Dense` et `Over-Rigid`

Définition 5 Réseau objets-contraintes (s, V, t, E, w)
- s est la source et t est le puits.
- Chaque objet $o \in O$ est représenté par un nœud-objet $v_o \in V$.
- Chaque contrainte $c \in C$ est représenté par un nœud-contrainte $v_c \in V$.
- Depuis chaque objet $o \in O$ part un arc $(v_o \rightarrow t) \in E$ de capacité $w(v_o \rightarrow t)$ =DDL(o).
- Depuis chaque contrainte $c \in C$ part un arc $(s \rightarrow v_c) \in E$ de capacité $w(s \rightarrow v_c)$ =DDL(c).
- Pour chaque objet $o \in O$ sur lequel porte une contrainte $c \in C$, il existe un arc $v_c \rightarrow v_o$ de capacité $w(v_c \rightarrow v_o) = \infty$ dans E.

Par définition, un flot dans ce réseau représente une distribution des DDL des contraintes sur les DDL des objets ; c'est ce principe qui est utilisé pour la caractérisation structurelle de la rigidité, comme nous allons le voir à la section 3.

3 Algorithmes

Notre nouvel algorithme, tout comme l'algorithme `Dense` est basé sur un calcul de flot maximum dans le réseau objets-contraintes. Cependant, il présente deux différences principales :

- il utilise la es_rigidité au lieu de la s_rigidité ;
- il effectue la distribution du flot de façon *géométriquement correcte*.

Ces deux différences sont obtenues au moyen de deux modifications majeures de l'algorithme $\widehat{\text{Dense}}$ dans la façon d'appliquer une surcharge dans le réseau objets-contraintes :

1. La surcharge appliquée dans le réseau dépend des propriétés géométriques vérifiées par les objets au lieu d'être constante.
2. La surcharge est appliquée via un nœud R dédié à cet effet, au lieu d'être appliquée au moyen des nœuds-contraintes.

Dans cette section, nous introduisons d'abord le principe de l'identification de la rigidité à l'aide d'un calcul de flots. Nous présentons alors la fonction `Distribute` qui est au cœur de

l'algorithme `Dense` et sur laquelle porte nos deux modifications. Nous expliquons enfin comment la modification de cette fonction permet de répondre de façon géométriquement correcte aux principaux problèmes liés au concept de rigidité.

3.1 Détection de rigidité à base de flot

Du point de vue géométrique, la caractérisation de la rigidité s'effectue en vérifiant qu'un GCSP n'admet que des déplacements. Sous cet angle, la détection de rigidité à base de flots peut être expliquée comme suit :

1. retirer K déplacements du GCSP en introduisant K DDL supplémentaires du côté des contraintes ;
2. vérifier si un sous-GCSP sur-contraint S' existe en calculant un flot maximum dans le réseau objets-contraintes surchargé ;
3. si tel est le cas, alors S' vérifie DDL(S')¡K.

En effet, nous avons déjà expliqué qu'un flot dans le réseau objets-contraintes représente une distribution des DDL des contraintes sur les DDL des objets, ce qui revient intuitivement à choisir quelles contraintes résolvent quels objets. Un flot maximum représente donc une distribution *optimale* des DDL dans le GCSP. Si un flot maximum ne peut saturer tous les arcs issus de la source du réseau, cela signifie que certains DDL des contraintes ne peuvent être distribués sur les objets (cf. définition 5). Il existe alors obligatoirement un sous-GCSP sur-contraint dans le GCSP.

La détection structurelle de rigidité exploite cette propriété pour identifier un sous-GCSP S' qui vérifie DDL(S')¡K pour un K donné. Pour ce faire, K DDL sont artificiellement introduits dans le réseau en provenance de la source. Selon le principe précédent, si le calcul d'un flot maximum ne permet pas de saturer tous les arcs issus de la source cela signifie que certains DDL contraintes n'ont pas pu être absorbés et qu'il existe un sous-GCSP sur-contraint. Cependant, la présence de la surcharge K implique que ce GCSP n'est peut-être pas sur-contraint, mais vérifie nécessairement DDL(S')¡K. [Hoffmann *et al.*, 1997] ont montré que S' est alors induit par l'ensemble des nœuds-objets traversés durant la dernière recherche d'un chemin augmentant dans le réseau, c-a-d. qu'il s'agit de tous les objets atteignables par un parcours du graphe résiduel partant de la source.

Selon la valeur choisie pour K, on peut alors employer ce principe pour identifier des sous-GCSP s_rigides ($K = \frac{d(d+1)}{2} + 1$), sur-s_rigides ($K = \frac{d(d+1)}{2}$), es_rigides ($K = DOR + 1$) ou sur-es_rigides ($K = DOR$).

3.2 Fonction `Distribute`

La fonction `Distribute` [Hoffmann *et al.*, 1997] est la mise en application algorithmique de ce principe. La version proposée par Hoffmann *et al.* applique la surcharge K en augmentant simplement la capacité de l'arc liant la source à une contrainte c du réseau, c-a-d. en augmentant le nombre de DDL d'une contrainte c du GCSP de K. Ceci a pour but de retirer K déplacements aux sous-GCSP induit par les objets liés à c. Cependant, retirer K déplacements à un sous-GCSP S' n'est géométriquement correct que si ce sous-GCSP possède au moins K déplacements, c-a-d. qu'il vérifie DDR(S')$\geq K$.[3]

[3] Rappelons que le DDR représente le nombre de déplacements admis par un sous-GCSP.

Considérons par exemple un segment en 3D, c-a-d. sous-GCSP constitué de 2 points liés par une contrainte de distance. Ce sous-GCSP n'admet que 5 des 6 (3 rotations + 3 translations) déplacements autorisés par l'espace 3D puisque la rotation selon l'axe défini par le segment est sans effet sur celui-ci. Ainsi, retirer 6 déplacements à un segment 3D pour vérifier s'il est rigide est géométriquement incorrect et produit assurément un résultat erroné. C'est pourtant ce que fait la fonction `Distribute` proposée par Hoffmann *et al.* lorsqu'elle applique une surcharge $K = 6$ sur une contrainte de distance liant deux points dans un GCSP en 3D.

De façon à appliquer le principe de détection structurelle de rigidité de façon géométriquement correcte, nous proposons d'introduire une contrainte virtuelle R possédant K DDL. Cette contrainte qui représente le fait de fixer K déplacements sur un ensemble d'objets, ne pourra être liée qu'aux sous-GCSP S' vérifiant DDR(S')$\geq K$. K et S' sont donc les deux paramètres d'entrée de notre version de la fonction *Distribute*.

Distribute (S : GCSP ; K : entier ; S' : GCSP) **retourne** S'' : GCSP)
Require: $K > 0$, $S' \subset S$ vérifie DDR(S')$\geq K$
Ensure: $S'' \subset S$ vérifie DDL(S'')$< K$, ou S'' est vide
  $G \leftarrow$ `Overloaded-Network`(S, K, S')
  $V \leftarrow$ `FordFulkerson`(G)
  $S'' \leftarrow$ `Object-Induced-subGCSP`(V, S)
  Return S''

La fonction `Overloaded-Network` retourne le réseau objets-contraintes correspondant au GCSP S dans lequel la contrainte R a été introduite, ainsi que les arcs $s \rightarrow R$ et $R \rightarrow o$ (pour chaque $o \in S'$) de capacités respectives K et $+\infty$. Le calcul du flot maximum dans ce réseau est effectué par la fonction `FordFulkerson` qui applique l'algorithme classique de [Ford et Fulkerson, 1962]. Cette fonction retourne l'ensemble V des nœuds-objets traversés lors de la dernière recherche d'un chemin augmentant, cet ensemble étant vide si le flot maximum sature tous les arcs issus de la source s. Enfin, la fonction `Object-Induced-subGCSP` retourne le sous-GCSP S'' induit par V. Selon la preuve de Hoffmann *et al.*, S'' vérifie DDL(S'')¡K ou S'' est vide.

Nos deux modifications de la fonction `Distribute` permettent d'une part d'appliquer la surcharge sur n'importe quel sous-GCSP, et d'autre part de ne distribuer la surcharge que vers les sous-GCSP pour lesquels cette opération est géométriquement correcte.

Exemple d'application de `Distribute` : La figure 2-a illustre l'application de la version originelle de la fonction `Distribute` sur le GCSP de la figure 1-a : `Distribute(`$S, 3, dAB$`)`. Comme la surcharge ne peut être distribuée entièrement par le calcul d'un flot maximum, la fonction conclue que le sous-GCSP AB dispose de moins de 3 DDL et le retourne. Ceci est exact puisque DDL(AB)=2 qui est inférieur à $K = 3$. Cependant, étant donné la nature géométrique du GCSP (deux droites *parallèles* en 2D), il est faux d'en conclure que ce sous-GCSP est sur-rigide.

Sur ce même sous-GCSP, notre fonction `Distribute` procèdera différemment : puisque DDR(AB)=2, la surcharge maximale applicable sur ce sous-GCSP est $K = 2$. La figure 2-b présente l'appel `Distribute(`$S, 2, AB$`)` pour la nouvelle version de la fonction. Le flot maximum parvient cette fois à saturer tous les arcs issus de la source et aucun sous-GCSP n'est donc retourné. Ceci est à la fois correct du point de vue structurel puisque DDL(AB) n'est pas strictement inférieur à $K = 2$, et du point de vue géométrique puisque ce sous-GCSP n'est pas sur-rigide.

Complexité de `Distribute` **:** La complexité de la nouvelle fonction `Distribute` est dominée par celle de la fonction `FordFulkerson` qui effectue le calcul du flot maximum. En effet, la production du réseau objets-contraintes surchargé et l'extraction du sous-GCSP induit par V peuvent être effectués en temps linéaire au plus, alors que le calcul d'un flot maximum est en $O(n^2(n+m))$, n étant le nombre de nœuds et m le nombre d'arcs dans le réseau. Cette complexité est exactement la même que celle de la fonction `Distribute` proposée par Hoffmann *et al.*.

Remarque : si cette fonction est appelée plusieurs fois, il est possible de stocker le calcul du flot précédent afin de ne le recalculer que de façon incrémentale.

3.3 Algorithmes pour la détection de rigidité

A partir de la fonction `Distribute`, on peut dériver des algorithmes répondant aux principaux problèmes liés au concept de rigidité. [Hoffmann *et al.*, 1997] ont proposé les algorithmes `Dense` et `Minimal_Dense` qui permettent d'identifier des sous-GCSP bien- ou sur-rigides et de les minimiser (en nombre d'objets).

Ces algorithmes sont entièrement reproductibles avec notre nouvelle fonction `Distribute`. Ceci nous permet de répondre aux même problème mais de façon géométriquement correcte et avec une meilleure caractérisation de la rigidité : la rigidité structurelle étendue

Dans cette section, nous présentons l'adaptation de l'algorithme `Dense` à notre fonction `Distribute`. Nous expliquerons ensuite comment utiliser notre fonction `Distribute` pour répondre aux principaux problèmes liés à la rigidité.

Algorithme `Dense` versus algorithme `Over-Rigid`

Schématiquement, l'algorithme `Dense` effectue des appels à la fonction `Distribute` pour chaque contrainte présente dans le GCSP considéré, et ce jusqu'à ce que cette fonction retourne un GCSP non-vide ou jusqu'à ce que toutes les contraintes aient été traitées. Dans cet algorithme, la surcharge est basée uniquement sur la dimension de l'espace géométrique considéré ; elle représente le nombre maximum de déplacements indépendants : 6 en 3D et 3 en 2D. L'algorithme `Dense` est sensé ne retourner que des sous-GCSP sur-rigides puisque les GCSP retournés n'admettent pas certains déplacements de l'espace géométrique considéré. Comme nous l'avons expliqué, cet algorithme est en réalité incorrect puisqu'il retire parfois plus de déplacements que n'en admet le sous-GCSP lié à la contrainte surchargée.

Pour obtenir un algorithme correct, nous proposons d'utiliser la définition de la es_rigidité et notre fonction `Distribute` : la surcharge passée en paramètre à la fonction Distribute sera le DDR du sous-GCSP sur lequel on applique la surcharge.

Ceci nous permet de définir un nouvel algorithme, appelé `Over-Rigid`, qui effectue un appel de la forme `Distribute(`S`,DDR(`S'`),`S'`))` pour chaque $S' \subset S$ jusqu'à ce qu'un sous-GCSP sur-rigide soit retourné ou que tous les S' aient été traités. En effet, un sous-GCSP S'' retourné par un appel de ce type vérifie nécessairement DDL(S'') ¡ DDR(S'), une condition suffisante pour que S'' soit sur-rigide (cf. définition 4 et lemme 1).

Bien sûr, un tel algorithme serait exponentiel en pire cas puisque le nombre de sous-GCSP dans un GCSP est égal au nombre de sous-ensembles d'objets dans l'ensemble d'objets de ce GCSP. Fort heureusement, nous montrerons (cf. section Propriétés de `Over-Rigid`) qu'il est suffisant d'appliquer la fonction `Distribute` aux sous-GCSP *DDR-minimaux* uniquement (cf. définition 1 below), ce qui produit l'algorithme suivant :

Over-Rigid (S : GCSP) retourne S'' : GCSP
Ensure: $S'' \subset S$ est sur-es_rigid ou vide
$S'' \leftarrow GCSPvide$
$M \leftarrow$ `DDR-Minimaux(`S`)` {*construit l'ensemble des sous-GCSP DDR-minimaux de* S}
while $S'' = GCSPvide$ et $M \neq \emptyset$ **do**
$S' \leftarrow$ `Pop(`M`)`
$S'' \leftarrow$ `Distribute(`S`,DDR(`S'`),`S'`)`
end while
Return S''

Definition 1 Sous-GCSP DDR-minimaux
Un sous-GCSP $S' \subset S$ *est* **DDR-minimal** *ssi il ne contient aucun sous-GCSP possédant le même DDR, c-a-d. que* $\forall S'' \subsetneq S'$, *DDR(*$S''$*)¡DDR(*$S'$*).*

Exemple d'application de `Over-Rigid`

Considérons à présent le GCSP S présenté à la figure 1-b. Soit $M = \{BC, BDF, CEF, ...\}$ l'ensemble de ses sous-GCSP DDR-minimaux résultant de la fonction `DDR-Minimaux(`S`)`. L'algorithme `Over-Rigid` procède alors comme suit :

1. A la première itération, $S' = BC$ et K =DOR(BC)=5. La figure 2-c représente l'appel `Distribute(`$S, 5, BC$`)`. Tous les arcs issus de la source s sont saturés par le calcul du flot maximum et aucun sous-GCSP sur-rigide n'est donc identifié.
2. A la seconde itération, $S' = CEF$ et K =DOR(CEF)=6. La figure 2-d présente l'appel `Distribute(`$S, 6, CEF$`)` correspondant à cette itération. Cette fois-ci, on peut constater que l'arc $s \rightarrow R$ n'est pas saturé par le flot maximum. L'ensemble des objets atteignables à partir de la source dans le graphe résiduel étant $\{A, C, D, E, F\}$, l'algorithme termine à cette itération en retournant le sous-GCSP $S'' = ACDEF$ qui est effectivement sur-es_rigide.

Sur ce même GCSP, l'algorithme `Dense`, qui a le même objectif que notre algorithme `Over-Rigid` retournerait soit un segment, soit un sous-GCSP composé de la droite A et d'un point incident, comme un sous-GCSP sur-rigide, ce qui est faux[4].

Propriétés de `Over-Rigid`

Afin de démontrer la correction et la complétude de l'algorithme `Over-Rigid`, nous utiliserons les lemmes suivants qui établissent des propriétés du concept de DDR et de la distribution de flots dans des réseaux objets-contraintes :

Lemme 1 *Soit* S *un GCSP et* $S' \subset S'' \subset S$ *deux sous-GCSP. Alors DDR(*S'*)* $\leq$ *DDR(*S''*).*

Démonstration : Chaque unité du DDR dans un sous-GCSP représente un déplacement indépendant (translation or rotation) admis par ce GCSP. L'ajout d'un nouvel objet dans ce sous-GCSP ne peut en aucun cas retirer un déplacement de ce GCSP puisque les contraintes sont indépendantes du repère global. Ainsi, DDR(S') $\leq$ DDR($S' \cup \{o\}$).□

[4]En pratique, l'algorithme `Dense` est assisté par des heuristiques qui lui permettent d'éviter des erreurs aussi triviales, mais il demeure géométriquement incorrect et peut toujours être trompé par des configurations non prises en charge.

Lemme 2 *Soit $S' \subset S'' \subset S$ deux sous-GCSP du GCSP S. Si* `Distribute(`S, K, S''`)` *retourne un sous-GCSP S_0* non vide*, alors* `Distribute(`S, K, S'`)` *retourne nécessairement un sous-GCSP S_1* non vide*.*

Démonstration : Soit $G_{S'}$ le réseau surchargé utilisé dans `Distribute(`S, K, S'`)` et $G_{S''}$ le réseau objets-contraintes surchargé utilisé dans l'appel `Distribute(`S, K, S''`)`. La seule différence entre ces deux réseaux est la présence d'arcs supplémentaires du type $R \rightarrow o$ dans $G_{S''}$ puisque $S' \subset S''$. Ainsi, la distribution de flot est nécessairement plus *difficile* dans $G_{S'}$ que dans $G_{S'}$, les capacités totales issues de la source et entrant dans le puits étant identiques dans ces deux réseaux. Donc, si un calcul de flot maximum ne peut saturer les arcs issus de la source dans $G_{S''}$, il est impossible qu'un flot maximum les sature dans $G_{S''}$.□

Correction de `Over-Rigid :`

Soit S'' un sous-GCSP *non vide* résultant de l'appel `Over-Rigid`(S). Supposons que S'' résulte de l'appel `Distribute`($S, DOR(S'), S'$) pour un sous-GCSP S donné ; S'' vérifie alors nécessairement DDL(S'') ¡ DOR(S') puisqu'il est *non vide*. De plus, par définition de la fonction `Distribute`, $S' \subset S''$. Le lemme 1 assure alors que DDR(S')≤DDR(S''), et on peut donc affirmer que DDL(S'')¡DOR(S''), c-a-d. que S'' est sur-es_rigid.□

Complétude de `Over-Rigid :`

L'algorithme `Over-Rigid` applique la surcharge sur chaque sous-GCSP dans l'ensemble M des sous-GCSP *DDR-minimaux* (généré par la fonction `DDR-Minimaux(`S`)`). Remarquons que tout sous-GCSP non DDR-minimal contient par définition un sous-GCSP DDR-minimal ayant même DDR. Le lemme 2 assure donc qu'aucun sous-GCSP sur-es_rigide ne peut être raté par l'algorithme.□

Complexité de `Over-Rigid :`

La complexité en temps de l'algorithme `Over-Rigid` dépend directement du nombre de sous-GCSP DDR-minimaux. Nous avons prouvé par énumération que la taille (en nombre d'objets) d'un sous-GCSP DDR-minimal est 2 et 2D et 3 en 3D pour des GCSP constitués de points, droites et plans sous des contraintes de distances, incidences, angles et parallélismes[5]. Ainsi, le nombre de sous-GCSP DDR-minimaux dans un GCSP de ce type constitué de n objets en dimension d est en $O(n^d)$.

Appelons C_1 la complexité de la fonction `DDR-Minimaux`, et C_2 celle de la fonction `Distribute` que nous avons étudiée à la section précédente. Alors, la complexité en pire cas de l'algorithme `Over-Rigid` est $O(C_1 + n^d * C_2)$.

C_1 est généralement la complexité de la démonstration automatique en géométrie puisque le calcul du DDR nécessite la détermination des propriétés géométriques vérifiées par les objets du GCSP. C_1 est alors exponentielle en pire cas. Cependant, cette complexité peut être polynomiale voir constante pour des classes de GCSP particulières, comme les systèmes à barres ou mécanismes génériques. qui plus est, on peut employer plusieurs heuristiques dans le calcul du DDR qui permettent de rendre la complexité moyenne plus abordable en pratique. Dans ces cas là, C_1 est généralement négligeable en comparaison de C_2, et on obtient donc une complexité globale de l'algorithme `Over-Rigid` en $O(n^{d+2} * (n + m))$. En comparaison, la complexité de l'algorithme `Dense` est en $O(m * n^2 * (n + m))$. Notre algorithme induit donc un surcoût approximativement linéaire en 2D et quadratique en 3D.

[5]La plupart des GCSP peuvent se ramener à des systèmes utilisant uniquement ces objets et contraintes primitives.

3.4 Autres algorithmes

La fonction `Distribute` peut être employée de façon similaire pour répondre aux principaux problèmes liés au concept de rigidité, à savoir :

1. Identifier des sous-GCSP bien- ou sur-rigides ; ceci est accompli en changeant la valeur de la surcharge en DDR+1 au lieu de DDR dans l'algorithme `Over-Rigid`.
2. Décider si un GCSP est rigide ; il suffit pour cela d'effectuer un compte global des DDL du GCSP (première condition de la définition 4) et d'effectuer un appel à l'algorithme `Over-Rigid` pour vérifier qu'il ne contient aucun sous-GCSP sur-rigide (seconde condition).
3. Minimiser la taille d'un sous-GCSP bien- ou sur-rigide ; pour ce faire, on utilise le principe de minimisation introduit dans l'algorithme `Minimal_Dense`) [Hoffmann *et al.*, 1997] : chaque objet est considéré à tour de rôle ; si son retrait permet toujours de trouver un sous-GCSP bien ou sur-rigide (par un appel à l'algorithme du point 1), alors l'objet est supprimé car il n'est pas nécessaire, sinon il est conservé car il est nécessaire.

Pour tout ces problèmes, l'utilisation de notre fonction `Distribute` en conjonction avec la caractérisation par es_rigidité permet l'obtention d'algorithmes géométriquement correctes qui retournent donc des résultats plus conformes à la rigidité.

3.5 Conclusion

Nous avons introduit une nouvelle fonction `Distribute` qui permet d'effectuer la détection structurelle de la rigidité de façon géométriquement correcte et d'employer la caractérisation par rigidité structurelle étendue qui est plus conforme à la rigidité que la rigidité structurelle classique.

Le concept de sous-GCSP DDR-minimal et ces propriétés est apparu comme crucial dans la définition d'une nouvelle famille d'algorithmes polynomiaux pour répondre aux principaux problèmes relatifs au concept de rigidité.

Ces algorithmes permettent de traiter des GCSP en 2D et 3D de façon géométriquement correcte. Ils peuvent prendre en compte les contraintes d'incidences et de parallélismes, contraintes omniprésentes dans les applications en robotique, CAO, biologie moléculaire et vision artificielle. Les spécificités géométriques induites par ces contraintes n'étaient pas prises en compte de façon générale par les algorithmes `Dense` et `Minimal_Dense`.

Nos algorithmes ouvrent donc une possibilité pour une utilisation plus fiable des techniques liées au concept de rigidité dans un contexte applicatif industriel, en particulier pour les méthodes de rigidification récursives qui décomposent un GCSP en sous-GCSP rigides *minimaux* à résoudre séparément [Hoffmann *et al.*, 2000; Jermann, 2002].

La suite logique de ce travail sera donc d'évaluer expérimentalement sur des applications réelles le rapport entre gain de qualité (correction géométrique et meilleure adéquation à la rigidité) et surcôut engendré par les nouveaux algorithmes.

Parallèlement, la poursuite de l'étude théorique des propriétés du DDR et de l'algorithme de flots permettront probablement d'exhiber des conditions spécifiques permettant d'améliorer en retour les performances pratiques.

Références

[Bouma *et al.*, 1995] W. Bouma, I. Fudos, C.M. Hoffmann, J. Cai, et R. Paige. Geometric constraint solver. *Computer Aided Design*, 27(6) :487–501, 1995.

[Dufourd *et al.*, 1998] J.-F. Dufourd, P. Mathis, et P. Schreck. Geometric construction by assembling solved subfigures. *Artificial Intelligence*, 99(1) :73–119, 1998.

[Ford et Fulkerson, 1962] L.R. Ford et D.R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[Hoffmann *et al.*, 1997] C.M. Hoffmann, A. Lomonosov, et M. Sitharam. Finding solvable subsets of constraint graphs. Dans *Principles and Practice of Constraint Programming CP'97*, pages 463–477, 1997.

[Hoffmann *et al.*, 2000] C.M. Hoffmann, A. Lomonosov, et M. Sitharam. Decomposition plans for geometric constraint systems. Dans *Proc. J. Symbolic Computation 2000*, 2000.

[Jermann *et al.*, 2000] C. Jermann, G. Trombettoni, B. Neveu, et M. Rueher. A constraint programming approach for solving rigid geometric systems. Dans *Principles and Practice of Constraint Programming, CP 2000*, volume 1894 of *LNCS*, pages 233–248, 2000.

[Jermann *et al.*, 2002] C. Jermann, B. Neveu, et G. Trombettoni. On the structural rigidity for gcsps. Dans *Proceedings of the* 4^th *International Workshop on Automated Deduction in Geometry*, 2002.

[Jermann, 2002] C. Jermann. *Résolution de contraintes géométriques par rigidification récursive et propagation d'intervalles*. Thèse de doctorat, Université de Nice - Sophia Antipolis, 2002.

[Kramer, 1992] G. Kramer. *Solving Geometric Constraint Systems*. MIT Press, 1992.

[Laman, 1970] G. Laman. On graphs and rigidity of plane skeletal structures. *J. Eng. Math.*, 4 :331–340, 1970.

[Lamure et Michelucci, 1998] H. Lamure et D. Michelucci. Qualitative study of geometric constraints. Dans B. Bruderlin et D. Roller, editors, *Geometric Constraint Solving and Applications*, pages 234–258. Springer, 1998.