Journées du Groupe de Travail en Modélisation Géométrique 2012, Strasbourg

Une approche par décomposition et reparamétrisation de systèmes de contraintes géométriques.

Rémi Imbach [1], Pascal Mathis [1] et Pascal Schreck [1]

[1] Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection, UMR 7005

Résumé

La décomposition des systèmes de contraintes est un avatar de la stratégie diviser pour régner *qui est indispensable dans le cadre de la résolution de systèmes de contraintes géométriques en CAO. Diverses méthodes tirant partie de l'invariance par déplacement ont été décrites dans la littérature, mais malheureusement, elles n'arrivent pas à décomposer des systèmes relativement standards, surtout en 3D. Une idée assez récente consiste à conjuguer résolution formelle et résolution numérique d'une part en modifiant le système de contraintes original pour le rendre soluble par une méthode constructive, et, d'autre part, en appliquant une méthode numérique pour rattraper les contraintes non prises en compte dans le système modifié. Dans la lignée de ces travaux, nous présentons deux nouvelles avancées. La première concerne la manière de modifier le système en tenant compte d'une méthode de décomposition et dont l'objectif est de minimiser le nombre de contraintes modifiées par composante irréductible. La seconde consiste à concevoir une méthode numérique qui tire parti du contexte géométrique pour rattraper des contraintes de manière robuste et pour produire toutes les solutions possibles.*

Mots-clés : Systèmes de contraintes géométriques, analyse des degrés de liberté, systèmes de contraintes paramétrés, résolution numérique.

1. Introduction

La résolution de contraintes géométriques est une problématique dérivée des CSP (Constraint Satisfaction Problems) qui peut s'appliquer à divers domaines comme l'enseignement assisté par ordinateur, la robotique, la chimie moléculaire ou encore la CAO. C'est ce dernier domaine qui nous intéresse dans cet article.

L'objet du dessin technique est de concevoir des objets ou des scènes géométriques dans un sens assez large qui comprend, entre autres, les mécanismes, les pièces mécaniques et les ensembles architecturaux. Dans tous les cas, il s'agit de dessiner une esquisse représentant la forme de l'objet qui sert de support à la cotation : en termes plus modernes, l'esquisse définit la topologie de l'objet dessiné tandis que les cotes qui lui sont appliquées définissent sa géométrie. Les outils spécialisés dans le traitement des contraintes géométriques extraient de cette esquisse cotée plusieurs informations dont les contraintes d'incidence, provenant de la topologie, et des contraintes métriques provenant de la cotation (voir exemple Figure 1). Le tout forme un système de contraintes géométriques, ou GCS (Geometric Constraint System) en abrégé. En ce sens, la résolution de contraintes géométriques a sa place en modélisation géométrique entre la modélisation déclarative, une esquisse cotée est en effet une spécification déclarative d'un objet, et des approches plus impératives comme la modélisation à base topologique ou la CSG. La manipulation de tels objets définis par des contraintes passe souvent par un *solveur* qui traduit la spécification déclarative en une représentation informatique plus facilement exploitable. Cette représentation peut prendre la forme

- soit d'un ensemble de valeurs numériques traduisant la géométrie d'un objet particulier, on parle alors de solveur numérique [LGL81, LM95, Mic03]. De tels solveurs utilisent une méthode itérative comme celle de Newton-Raphson ou la méthode par continuation. Elles sont générales, mais il est difficile d'obtenir toutes les solutions.
- soit d'une procédure pour construire de tels objets, on parle alors de solveur constructif [Brü93, DMS97, JAS97]. Les solveurs constructifs sont beaucoup moins

généraux, mais en revanche, ils permettent dans la plupart des cas de parcourir l'espace des solutions, ce qui est intéressant lorsqu'on veut obtenir "la solution proche de l'esquisse".

En CAO, des esquisses cotées relativement simples peuvent comporter des centaines de contraintes et, même si il y a eu beaucoup de progrès dans le domaine des solveurs, il est crucial de décomposer un système de contraintes en sous-systèmes plus petits et plus faciles à résoudre. Diverses approches ont été étudiées, parmi les plus connues on trouve [LM96, DMS97, HLS97, TSM*11, Zha11]. Malgré des degrés de sophistication divers, on rencontre des systèmes de contraintes qu'on ne peut pas décomposer avec ces méthodes. C'est typiquement le cas des systèmes définissant des polyèdres en 3D, mais aussi de systèmes de contraintes "ordinaires" en 2D.

Des travaux assez récents ont proposé de contourner le problème en modifiant le système de contraintes à résoudre en enlevant des contraintes et en les remplaçant par d'autres pour maintenir la rigidité : le système transformé est résolu de manière constructive, et on rattrape les contraintes supprimées du système de départ en utilisant une méthode numérique. On parle parfois de système pseudo-décomposable et de pseudo-décomposition. L'approche décrite dans [GHY02] utilise une technique employant la *force brute* via des essais systématiques de remplacement de contraintes et échantillonnage d'une courbe. Elle n'est utilisable que pour des petits systèmes 3D où seule une contrainte est remplacée. Plus récemment, [FS08] décrit une approche plus générale pour résoudre une plus grande famille de cas en 2D ou en 3D.

Nous proposons ici des travaux en cours de développement qui portent à la fois sur la manière de transformer le système initial S en un système S' soluble par une méthode constructive, et sur une approche numérique pour résoudre le système résultant de la solution formelle de S' et des contraintes retirées de S. Pour le premier point, notre approche se distingue des approches précédentes qui tentaient de minimiser le nombre de contraintes à changer sans tenir compte d'aucune méthode de décomposition [†]. L'idée consiste ici à modifier le système de contraintes en même temps qu'on essaye de le décomposer : nous étudions un moyen de rendre minimal le nombre de contraintes à changer *par composante rigide du système modifié*. Pour le moment, la méthode de décomposition que nous utilisons est très simple, il s'agit d'une propagation des degrés de liberté pour former des clusters à la manière d'Hoffmann *et al.* [HLS97]. En ce qui concerne les méthodes numériques, nous étudions la manière d'adapter des méthodes générales de résolution d'équations et de suivi de courbes en utilisant les particularités du cadre géométrique. L'objectif est d'utiliser des critères géométriques lus sur la figure pour rendre plus fiables les méthodes de suivi de chemins et notamment prévoir les changements de branche. Nous étudions d'autre part des méthodes numériques pour obtenir toutes les solutions du système.

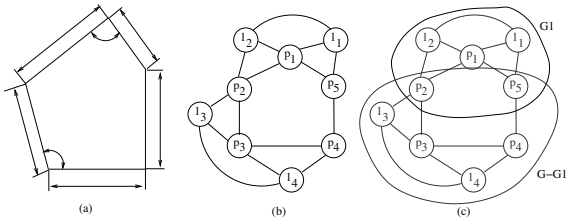

Figure 1: *Un GCS donné sous forme d'esquisse cotée et le graphe correspondant.*

La suite de cet article est organisée de la manière suivante. La section 2 introduit les notions de bases concernant les systèmes de contraintes et leur décomposition. La section 3 présente la méthode de reparamétrisation associée à la méthode des lieux. La section 4 présente en détail notre algorithme de décomposition. La section 5 explique comment nous réalisons la résolution numérique. La section 6 conclut.

2. Définitions préliminaires

2.1. GCS et graphes de contraintes

Un GCS est un ensemble de contraintes portant sur des primitives géométriques (points, droites, cercles, etc). Chaque primitive possède un degré de liberté (*DOF* pour *Degree Of Freedom*) correspondant au nombre minimum de paramètres permettant de la définir. À chaque contrainte est associé un degré de restriction (*DOR* pour *Degree Of Restriction*) qui représente le nombre de degrés de liberté qu'elle ôte aux primitives. Par exemple, dans l'espace, un point et un plan ont chacun 3 DOF alors qu'une droite en possède 4. Une contrainte de distance supprime 1 degré de liberté, une contrainte de type "être le milieu de de deux points" supprime les 3 degrés de liberté d'un point. En dimension d, avec P l'ensemble des primitives et C l'ensemble des contraintes, les GCS bien-contraints [‡] vérifient la relation suivante :

$$\sum_{p\in P} DOF(p) - \sum_{c\in C} DOR(c) = \binom{d+1}{2}$$

En CAO, les primitives sont souvent des points et droites en 2D, des points et plans en 3D ainsi que des cercles ou sphères de rayon fixé. Par ailleurs les types de primitives utilisés restreignent un seul degré de liberté ce qui explique que

†. Évidemment, un telle méthode pouvait être mise en œuvre, après modification du système, par le solveur constructif employé

‡. on dit aussi iso-rigides : ils contiennent le nombre minimum de contraintes pour définir une figure rigide

la relation précédente se trouve généralement sous la forme : $2n-3=c$ en 2D et $3n-6=c$ en 3D pour n primitives et c contraintes. Les constantes 3 et 6 correspondent aux degrés de liberté en translation et rotation des objets.

Un GCS peut être représenté par un graphe de contraintes $G=\langle V,E\rangle$ dans lequel les sommets correspondent aux primitives et les arêtes aux contraintes. Afin de simplifier le discours mais sans perte de généralité, nous considèrerons des contraintes de degré 2 et des contraintes de degré 1. Cela nous permet notamment de conserver le formalisme des graphes et d'éviter celui des hypergraphes, rendu nécessaire pour des contraintes impliquant plus de 2 primitives. L'extension aux hypergraphes est toutefois immédiate.

Si v est un sommet de G, nous noterons $DOF(v)$ le degré de liberté de la primitive associée alors que $DEG(v)$ désignera le degré du sommet dans le graphe.

Remarquons que dans la terminologie des graphes, le problème illustré par la figure 4 est habituellement noté $K_{3,3}$. Il s'agit en effet du graphe biparti complet à deux ensembles de 3 sommets.

La figure 1(a) représente un GCS en 2D avec 5 points, 5 distances et 2 contraintes d'angle. À droite, se trouve le graphe correspondant. Les sommets étiquetés p_i correspondent aux points et ceux étiquetés l_i aux droites. Les arêtes liant deux points schématisent des distances imposées, les arêtes point-droite des contraintes d'incidence et les arêtes entre droites des contraintes d'angle.

2.2. Décomposition et sous-graphes

Nous commençons par rappeler les notions élémentaires de la théorie des graphes. Sauf mention explicite du contraire, nous considérons des graphes non orientés. Soit $G=\langle V,E\rangle$ un graphe, on note $E=E[G]$ l'ensemble de ses arêtes et $V=V[G]$ l'ensemble des ses sommets. Si $V_1\subset V$, on note $G[V_1]$ le graphe induit ne contenant que les arêtes de G ayant leurs deux extrémités dans V_1. On définit de manière similaire le sous-graphe induit par un ensemble d'arêtes $E_1\subset E$ dont chaque sommet est incident à au moins une arête de E_1 et noté $G[E_1]$.

Soit un graphe G et un sous-graphe G_1 avec $E_1=E[G_1]$, on définit la différence $G-G_1$ comme étant égale à $G[V_2]$ avec $V_2=V[G[E-E_1]]$. Pour deux sous-graphes G_1 et G_2 de G, on définit l'intersection $G_{1\cap 2}=G[V_1\cap V_2]$. On sait ([MT10]) que si G_1 représente un GCS bien-contraint, il en est de même pour $G_2=G-G_1$ si et seulement si $G_{1\cap 2}$ est bien contraint et non réduit à un point en 2D et à deux points en 3D. La figure 1(c) représente un graphe G, un sous-graphe G_1 et $G_2=G-G_1$. $G_{1\cap 2}$ contient deux sommets $p2$ et $p5$ non reliés. Avec ces notations, on appelle sommet intérieur d'un sous-graphe G_1 un sommet de $G_1-G_{1\cap 2}$.

Nous supposons maintenant que G correspond à un GCS bien-contraint, que G_1 est un sous-graphe de G et que $G_2=G-G_1$. Alors, si G_1 est bien contraint et que $G_{1\cap 2}$ ne l'est pas, on peut le compléter par des informations tirées de G_1. Par exemple, dans la figure 1(b), dès que le GCS correspondant à G_1 est résolu, on peut calculer la distance $d=p2$-$p5$ et compléter $G_{1\cap 2}$ par une contrainte de distance liant ces deux points dans $G_{1\cap 2}$. Ce système complété est appelé le bord de G_1 dans G et noté $B_{G_1/G}$ [MT10].

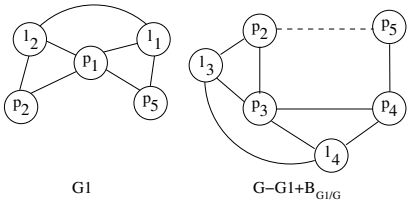

Figure 2: *Décomposition*

En termes de graphes, la decomposition d'un GCS correspondant au graphe G est une suite finie $G_1,G_2,\ldots,G_n$ où :
- G_1 correspond à un GCS bien-contraint,
- $G_2,\ldots,G_n$ est une décomposition de $G-G_1+B_{G_1/G}$

La figure 2 montre une décomposition du graphe de l'exemple précédent. Il faut noter qu'une telle décomposition n'est en général pas unique. On dit qu'elle est maximale lorsque la longueur de la suite de sous-graphes est maximale.

3. Méthode des lieux et reparamétrisation

La méthode par intersection des lieux géométriques, en abrégé méthode des lieux ou LIM (*Locus Intersection Method*), traduit chaque contrainte $c(p_i,p_j)$, où p_i (resp. p_j) est une primitive inconnue (resp. connue) en le lieu géométrique de p_i induit par $c(p_i,p_j)$. Les primitives inconnues sont construites en faisant l'intersection de ces lieux. Si un GCS peut être résolu sans changement par une telle suite d'intersections de lieux géométriques, une propagation des degrés de liberté dans le graphe correspondant fournit cette suite de constructions. LIM a un coût quadratique, mais une faible puissance de résolution. Nous détaillons cette méthode à la suite et nous montrons comment la reparamétrisation permet d'étendre largement la classe des GCS solubles.

3.1. LIM et plans de construction

La propagation des degrés de liberté peut être réalisée selon deux stratégies. La première, appelée rétro-propagation procède par déconstruction du graphe : un des sommets v satisfaisant $DEG(v)=DOF(v)$ est retiré du graphe avec les arêtes ayant ce sommet comme extrémité. Si après avoir répété cette étape autant de fois qu'il est possible, le graphe est réduit à un *repère*, ensemble de primitives déterminant un repère affine de l'espace, la résolution est un succès. Par exemple, dans le plan, si le graphe est réduit à un couple de sommets adjacents, la propagation s'achève.

La seconde stratégie, ou propagation avant, consiste à choisir un ensemble de primitives formant un repère, et à marquer les sommets correspondants comme construits. Puis, si un sommet v est adjacent à $DOF(v)$ sommets déjà construits, il est marqué lui aussi. Cette opération est appliquée jusqu'à ce qu'aucun nouveau sommet ne puisse être marqué. Lorsque tous les sommets sont marqués, le GCS est résolu.

Considérons à présent la propagation implantée par l'algorithme 1 où $DEGM(v)$ représente le nombre de sommets marqués adjacents à un sommet v.

Algorithme 1 Propagation avant des degrés de liberté

Entrées: $G = \langle V, E \rangle$: graphe de contraintes non-vide

1: Choisir un repère et marquer ses sommets
2: Choisir un sommet non-marqué v tel que $DEGM(v) = DOF(v)$, marquer v
3: Retourner à 2 tant qu'il est possible de choisir un sommet

Son application à un GCS fournit un *plan de construction*, c'est à dire une suite d'instructions permettant de construire les primitives du problème comme intersections de lieux géométriques. Pour le problème illustré par la figure 3 avec comme repère $(p_0 p_1)$, il produit le plan de construction suivant :

p_0 = fix()
  //origine du repère
l_0 = hline(p_0)
  // droite horizontale passant par p_0
p_1 = intersection_cl(p_0, k_0, l_0)
  // intersection du cercle (centre : p_0, rayon : k_0) avec l_0
p_2 = intersection_cc(p_0, k_6, p_1, k_1)
  // intersection des cercles (p_0, k_6) et (p_1, k_1)
p_5 = intersection_cc(p_0, k_5, p_2, k_8)
...

Ces deux stratégies permettent de résoudre les mêmes problèmes et traduisent une constructibilité symbolique par LIM. Les solutions numériques sont produites dans un second temps en calculant ces intersections. Remarquons qu'un plan de construction peut être vu comme une fonction des paramètres du GCS, les k_i dans l'exemple précédent.

L'évaluation numérique du plan de construction conduit à un arbre dont les sommets de profondeur n correspondent à l'évaluation de la n-ème instruction du plan de construction. À chaque fois qu'une telle instruction donne k solutions, k fils sont ajoutés au sommet correspondant. Finalement, l'arbre a autant de feuilles qu'il y a de solutions au GCS, chaque solution étant décrite par une branche de l'arbre.

Remarquons que LIM sous-tend un arbre avec potentiellement plus de branches que n'en produit l'évaluation. En effet, l'évaluation du plan de construction sur certaines branches peut échouer, par exemple quand un point dans le plan est définit comme l'intersection de deux cercles qui ne se coupent pas.

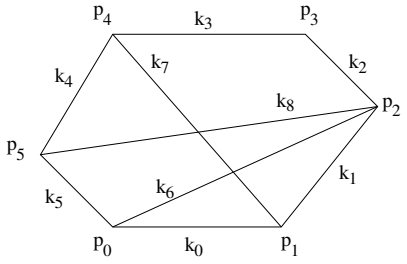

Figure 3: *Un GCS soluble par LIM. Les k_i sont les paramètres des distances*

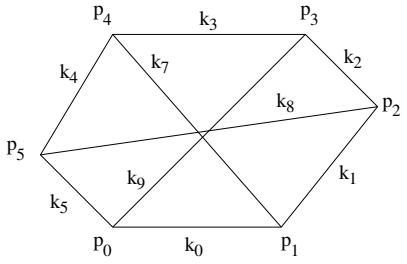

Figure 4: *Un problème non soluble par LIM*

Soit maintenant le GCS nommé $K_{3,3}$ et illustré par la figure 4 consistant à construire six points en connaissant neuf contraintes de distances entre ces points. Quel que soit le repère choisi, l'algorithme 1 échoue car chaque sommet a un degré dans le graphe strictement supérieur à son degré de liberté : les points p_i ont deux degrés de liberté mais sont contraints par trois distances.

L'approche par reparamétrisation consiste à remplacer un tel GCS par un autre, similaire, mais soluble par LIM.

3.2. Reparamétrisation

Reprenons l'exemple $K_{3,3}$ et supposons que le GCS de la figure 3 admette une solution s^* qui satisfasse la contrainte $distance(p_0, p_3) = k_9$. Alors s^* est aussi une solution du GCS $K_{3,3}$, car toutes ses contraintes sont satisfaites par s^*. Nous avons ainsi transformé le GCS G en un GCS G' en les mêmes inconnues en remplaçant certaines contraintes ($distance(p_0, p_3) = k_9$ dans l'exemple ci-dessus) par d'autres ($distance(p_0, p_2) = k_6$), avec G' soluble par LIM. Un plan de construction peut alors être formé pour G'.

Les paramètres des contraintes ajoutées (angles ou distances) sont appelées *paramètres guides* (k_6 dans l'exemple

précédent). La suite de la méthode consiste à rechercher des valeurs pour les paramètres guides de sorte que les solutions de G' soient aussi des solutions de G : c'est l'étape de résolution numérique.

[GHY02] décrit une procédure par rétro-propagation calculant à partir d'un GCS bien contraint G un nouveau GCS G' soluble par LIM. Cette méthode est réécrite en propagation avant dans l'optique d'une décomposition expliquée plus bas. La procédure s'arrête dès que le GCS induit par le sous-graphe des sommets marqués est bien contraint et contient au moins un sommet intérieur.

Notre méthode de reparamétrisation agit, elle, par propagation avant. Elle est formalisée par l'algorithme 2. À chaque itération, plusieurs sommets peuvent êtres choisis ; l'algorithme n'est donc pas déterministe et peut amener à plusieurs reparamétrisations différentes. Pour le GCS $K_{3,3}$, une de ces reparamétrisations possibles est le GCS de la figure 3.

Algorithme 2 Reparamétrisation

Entrées: $G = \langle V, E \rangle$: graphe de contraintes non-vide

1: Choisir un repère et marquer ses sommets
2: Choisir un sommet non encore marqué v tel que $DEGM(v) > 0$ et $DEGM(v) - DOF(v)$ est minimum
3: **Si** DEGM(v) < DOF(v)
   ajouter DOF(v)-DEGM(v) contraintes
   Si DEGM(v) > DOF(v)
   enlever DEGM(v)-DOF(v) contraintes
4: Marquer v
5: Soit Vm l'ensemble des sommets marqués
   Arrêter si $G[Vm]$ est bien contraint et contient un sommet intérieur
   sinon retourner à 2

3.3. Résolution numérique

On considère maintenant le plan de construction Cp du GCS reparamétré G' comme une fonction des paramètres guides $k^+ = (k_0^+, ..., k_{d-1}^+)$, et on le note $Cp(k^+)$ par omission des autres paramètres ([IMS11]).

Pour une figure produite par $Cp(k^+)$, on teste la satisfaction des contraintes supprimées en définissant la fonction $F : \mathbb{R}^d \to \mathbb{R}^d$ par $F = (F_0, ..., F_{d-1})$ où $F_i(k^+) = f_i^-(Cp(k^+)) - k_i^-$. $f_i^-(Cp(k^+))$ correspond à l'évaluation de la contrainte supprimée c_i^- de paramètre k_i^-.

En d'autres termes, $F(k^+)$ évalue, sur une figure produite par le plan de construction $Cp(k^+)$ évalué pour les paramètres $k^+ = (k_0^+, ..., k_{d-1}^+)$ les d contraintes supprimées et s'annule si elles sont satisfaites.

Les valeurs numériques des paramètres guides $(h_0, ..., h_{d-1})$ qui sont solutions de $F(h) = 0$ peuvent être obtenues en échantillonnant une boîte de $\mathbb{R}^d$ ([GHY02])

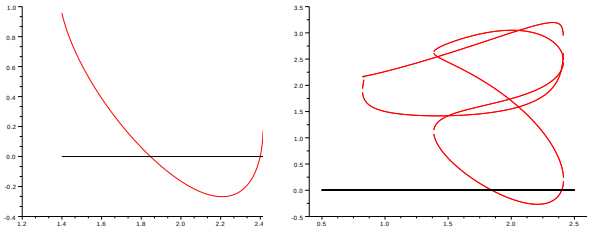

Figure 5: *À gauche : échantillonnage d'une branche du plan de construction. À droite : échantillonnage de toutes les branches du même plan de construction*

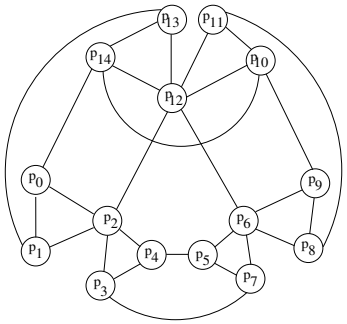

Figure 6: 15 *primitives et* 27 *contraintes. Deux contraintes sont remplacées pour résoudre le problème par LIM :*(p_1p_{13}) *et* (p_8p_{11})

et en calculant $F(x_0, ..., x_{d-1})$ sur chacune des branches du plan de construction de G' pour chaque échantillon $(x_0, ..., x_{d-1})$. La figure 5 illustre un tel échantillonnage dans le cas de la reparamétrisation de $K_{3,3}$.

Considérons à présent le GCS du plan dont le graphe de contrainte est donné par la figure 6, impliquant 15 primitives et 27 contraintes. Sa reparamétrisation amène à remplacer deux contraintes par deux autres dans le meilleur des cas et le problème est résolu en échantillonnant une fonction $F : \mathbb{R}^2 \to \mathbb{R}^2$, ce qui devient coûteux. En fait, ce coût explose quand le nombre de contraintes remplacées augmente, et il est encore aggravé par la croissance exponentielle du nombre de branches du plan de construction en fonction du nombre d'instructions.

L'idée proposée ici pour gérer cette explosion du coût consiste à choisir les contraintes à remplacer de telle sorte que le GCS G' soit décomposable en sous-systèmes bien contraints selon une méthode de décomposition donnée. Par exemple, le problème de la figure 6 peut-être transformé en le graphe de la figure 7 : trois contraintes ont été remplacées, mais le GCS induit par ce graphe est décomposable en trois sous-systèmes, solubles indépendamment, contenant chacun

un seul paramètre guide, et dont les plans de construction contiennent beaucoup moins de branches que le problème global.

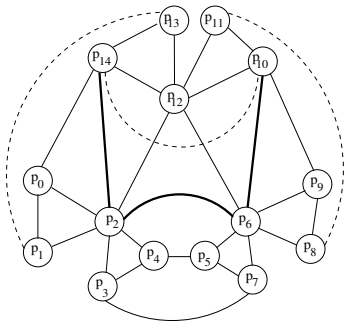

Figure 7: *Une reparamétrisation permettant la décomposition : les arêtes en pointillés sont les contraintes supprimées, celles en gras les contraintes ajoutées au GCS initial.*

Dans [FS08], la méthode de Newton-Raphson est appliquée à la fonction $F : \mathbb{R}^d \to \mathbb{R}^d$. Le principal problème vient alors du fait que F est seulement définie sur un sous-ensemble de $\mathbb{R}^d$, et la méthode de Newton peut en sortir ce qui la fait échouer. Nous utilisons ici une méthode par continuation pour trouver les zéros de F. La section 5 y est consacrée, et montre également comment le problème du domaine de définition est résolu.

3.4. Complétude et correction

Soit v une solution de G et $(f_0^+,...,f_{d-1}^+)$ la fonction évaluant les contraintes $(c_0^+,...,c_{d-1}^+)$ ajoutées dans G', il existe des paramètres $h = (h_0,...,h_{d-1})$ tels que $h_i = f_i^+(v)$. Comme v est solution de G, on a $f_i^-(v) = k_i^-$ pour $0 \leq i \leq d-1$, donc il existe une branche du plan de construction de G', noté Cp, sur laquelle $F(h) = 0$.

D'autre part, si h satisfait $F(h) = 0$ alors il existe une branche sur laquelle $Cp(h)$ est une solution v de G' vérifiant $f_i^-(Cp(k^+)) = k_i^-$ pour $0 \leq i \leq d-1$, donc v est solution de G.

4. Algorithme de décomposition avec reparamétrisation

4.1. Décomposition

Notre objectif est ici de lier propagation des degrés de liberté, reparamétrisation et décomposition de GCS.

L'algorithme suivant décompose le problème représenté par G en sous-problèmes éventuellement reparamétrés. Il met en jeu alternativement les deux algorithmes 1 et 2 vus plus haut.

L'utilisation d'une propagation avant des degrés de liberté impose la stratégie *bottom-up* suivie par cet algorithme.

Algorithme 3 Décomposition et reparamétrisation

Entrées: $G = \langle V, E \rangle$

Tant que tous les sommets ne sont pas marqués **faire**

- Appliquer l'algorithme 1 sur G autant que possible, pour chaque sous-système G_1 contenant des sommets internes. Dans G, remplacer, G_1 par son bord.
- Appliquer l'algorithme 2 sur G jusqu'à trouver un sous-système G_1 contenant des sommets internes. Remplacer G_1 par son bord.

fin Tant que

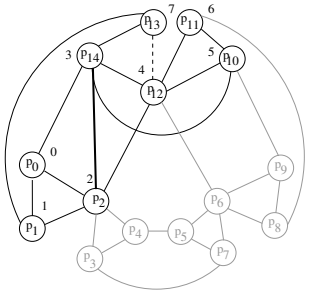

Figure 8: *Première étape de l'algorithme 2*

Sur l'exemple de la figure 1, une propagation arrière aurait conduit d'emblée à la suppression d'une contrainte, le graphe initial ne possédant que des sommets de degré au moins 3. L'algorithme 3, en revanche, utilise LIM en propagation avant autant que possible avant de recourir à une reparamétrisation. Après avoir fixé le repère $p_1 l_1$, par exemple, la construction se poursuit avec la droite l_2, les points p_2, p_5 et s'arrête faute de primitives directement constructibles. Les sommets internes p_1, l_1, l_2 sont supprimés et la distance $p_2 p_5$ ajoutée. Le graphe restant peut ensuite être parcouru sans reparamétrisation. Notons que d'autres décompositions sont possibles avec, par exemple, d'abord le triangle $p_2 p_3 p_4$ et finalement le triangle $p_2 p_4 p_5$.

Considérons maintenant l'exemple de la figure 6. L'algorithme 1 seul ne parvient pas à produire de sous-graphes avec des sommets internes. L'algorithme 2 est alors utilisé. La figure 8 illustre ce comportement après avoir fixé le repère p_0, p_1. Les sommets entourés en gras sont construits dans une première passe. Le nombre à côté de chaque sommet indique l'ordre de construction. L'arête $p_{14}\ p_2$ indique une distance ajoutée. L'ajout du point p_{13} rend le sous-graphe bien contraint. Ce point est lié au reste du graphe par 3 contraintes de distance. Deux d'entre elles serviront à la construction effective de ce point. La troisième sera rattrapée, lors de la phase numérique, en faisant varier la distance ajoutée $p_{14} p_2$.

Les sommets internes sont ensuite supprimés et le bord complété (Cf. figure 9). Ici, le bord contient les points p_2, p_{10}, p_{11}, p_{12} ainsi que 5 contraintes de distance. Parmi celles-ci, 4 figurent déjà dans le système initial, seule la

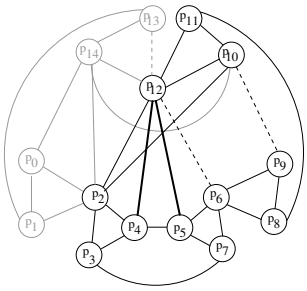

Figure 9: *Seconde étape de l'algorithme 2*

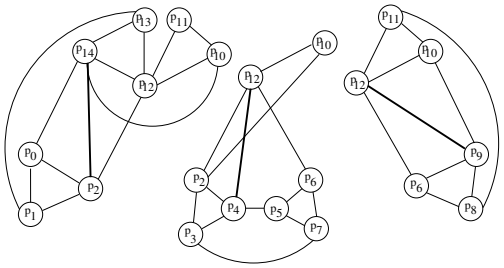

Figure 10: *Résultat d'une décomposition avec reparamétrisation*

distance p_2p_{10} est ajoutée. En choisissant p_2 et p_{12} pour nouveau repère, l'ensemble des sommets restants peut être construit en ajoutant deux contraintes qui seront alors à rattraper.

Il est toutefois possible de n'introduire que deux paramètres guides pour rendre ce problème soluble par LIM. A ce jour, la méthode naïve consistant à considérer toutes les possibilités est la seule méthode connue pour minimiser le nombre de paramètres guides. Le coût en temps de cette méthode est exponentielle avec le nombre de sommets mais la recherche de ce nombre minimum n'est pas forcément une quête judicieuse. En effet, l'introduction de 2 paramètres guides conduisent, lors de la phase numérique, à rechercher les zéros d'une fonction à deux variables. Il est préférable dans cette phase de chercher à réduire le nombre de variables des fonctions. Ainsi, on préféra 3 paramètres guides mais chacun dans un sous-graphe, ce qui amène à chercher les zéros de 3 fonctions à une seule variable. Ainsi pour une décomposition $G_1, \dots G_n$ où G_i peut contenir des contraintes ajoutées. Soit m le nombre maximum de contraintes ajoutées dans les sous-graphes G_i, l'objectif est de minimiser m. C'est précisément ce que s'attache à réaliser l'heuristique suivante.

4.2. Répartition des paramètres guides

Pour répartir au mieux des paramètres guides dans les sous-systèmes, il faudrait connaître tous les sous-graphes impliquant $2n-3$ contraintes. Cette recherche est malheureusement d'un coût exponentiel comme l'a montré [HLS97]. C'est pourquoi nous avons recours ici à une heuristique.

Supposons que G soit un système bien contraint avec une décomposition maximum en deux sous-graphes G_1 et G_2, chacun contenant des sommets internes et chacun nécessitant une reparamétrisation avec une seule contrainte. Le raisonnement suivant se généralise à davantage de sous-graphes. L'objectif est ici de trouver la répartition des deux paramètres dans les deux sous-systèmes.

Nous savons que G_1 est bien-contraint. Par ailleurs, G_2 est bien contraint si $G_{1\cap 2}$ est bien contraint ; il est sous-contraint sinon. Bien entendu, l'algorithme 3 démarre sans connaître cette décomposition. Idéalement, si un repère est fixé dans G_1, on peut espérer isoler G_1. Mais supposons qu'un repère soit fixé dans G_2 et soit Gf le premier sous-graphe déterminé par 3, nous avons alors deux cas :

- soit $G_{1\cap 2}$ est bien contraint, dans ce cas $Gf = G_2$,
- soit au contraire $G_{1\cap 2}$ n'est pas bien-contraint, ce qui est le cas le plus courant en CAO, et alors $Gf = G$

En effet, dans ce dernier cas, comme G_2 est sous-contraint, toute propagation démarrant à l'intérieur G_2 ne peut s'arrêter qu'après avoir marqué tous les sommets de G_1 pour produire le sous-graphe bien contraint Gf. Ainsi, si des contraintes on été ajoutées dans G_2, alors la dernière contrainte retirée appartenait à G_1. Cette remarque conduit à l'heuristique suivante : si deux ou plus paramètres guides sont ajoutés dans un sous-graphe Gf, l'algorithme 3 est relancé sur Gf mais en partant d'un nouveau repère impliquant la dernière contrainte retirée. Alors, le marquage des sommets a de meilleures chances d'isoler G_1 comme composante rigide modifiée.

Revenons à l'exemple de la figure 9. La méthode heuristique amène à reconsidérer le second sous-graphe car celui-ci contient deux paramètres guides. Une construction s'appuyant sur le repère p_8p_9, la distance p_8p_9 ayant été supprimée dans une première passe, conduit à une meilleure décomposition. Les arêtes en gras sur la figure 10 correspondent aux 3 paramètres guides, chacun dans un sous-graphe.

4.3. Terminaison et complexité

À chaque itération, l'algorithme extrait un sous-graphe et le remplace par son bord. Soient n et e le nombre de sommets et d'arètes du graphe de contraintes. L'algorithme 1 met en œuvre $O(n^2+e)$ opérations s'il résout le problème. Dans chaque boucle, les sommets restant sont parcourus, un sommet v est ôté et ses voisins sont mis à jour. La boucle est itérée n fois et la mise à jour des voisins nécessite de

visiter toutes les arêtes adjacentes. L'algorithme 1 échoue quand tous les repères possibles ont été essayés. Dans le plan, le nombre de repères est en $O(e)$ et pour chacun d'eux tous les sommets sont parcourus. Dans ce cas, le nombre d'opérations effectuées est en $O(ne)$. Or dans un graphe de contraintes e est proportionnel à n, on a $e = O(n)$ et l'ensemble du processus donne tous les sous-graphes possibles avec un coût de $O(n^2)$. Dans l'espace, un repère est formé par une combinaison de trois contraintes donc le nombre de repères est au plus en $O(e^3)$, donc l'algorithme a un coût en $O(n^3)$. Remarquons qu'en 3D, un repère peut contenir une contrainte ajoutée quand aucun repère ne peut être extrait du graphe de contrainte original (par exemple dans le cas d'un dodécaèdre).

Le bord est calculé en ajoutant des contraintes entre les primitives du bord. En 2D, un bord structurellement bien contraint peut être produit par des constructions de Henneberg (voir [GS09]). Dans l'espace, [TSM*11] présente un algorithme glouton pour déterminer ce bord.

L'algorithme 2 diffère d'un algorithme de propagation avant des degrés de liberté uniquement par le fait que si aucun sommet ne peut être construit, des nouvelles contraintes sont ajoutées, et les coûts de ces deux algorithmes sont les mêmes. En appliquant l'heuristique de 4.2, l'algorithme 2 est appelé $k < n$ fois où k est le nombre de contraintes supprimées (2 ou 3 en pratique). Donc, dans le pire des cas, l'algorithme général a un coût en $O(n^3)$. Dans la plupart des exemples de CAO, son coût est en $O(n^2)$.

5. Résolution numérique

La phase symbolique décrite précédemment fournit pour un GCS G plusieurs GCS solubles indépendamment par LIM pouvant contenir chacun des paramètres guides. La phase numérique a pour objectif de trouver pour chacun de ces sous-systèmes les valeurs des paramètres guides pour lesquelles les solutions de ces sous-systèmes, une fois assemblées, sont une solution de G.

Pour simplifier le discours et les notations, on considère ici que G n'est pas décomposable, et que l'on a obtenu par reparamétrisation un GCS G' et son plan de construction Cp en ajoutant à (respectivement supprimant de) G d contraintes $c_0^+,...,c_{d-1}^+$ (resp. $c_0^-,...,c_{d-1}^-$) de paramètres $k_0^+,...,k_{d-1}^+$ (resp. $k_0^-,...,k_{d-1}^-$). Les paramètres guides dont dépend Cp sont donc $k^+ = (k_0^+,...,k_{d-1}^+)$. On construit alors la fonction $F : \mathbb{R}^d \to \mathbb{R}^d$ comme indiqué plus haut.

Nous utilisons une méthode par continuation pour trouver les zéros du système $\mathcal{F}_1$ de d équations à d inconnues définit par $F(h) = 0$, dont le principe est le suivant. Connaissant une ou plusieurs des solutions d'un système $\mathcal{F}_0 : F_0(h) = 0$ en les mêmes inconnues que $\mathcal{F}_1$, appelé *système de départ*, $\mathcal{F}_0$ est déformé continûment en $\mathcal{F}_1$, appelé *système cible*, en construisant un nouveau système dépendant d'un paramètre t, $\mathcal{H} : H(t,h) = 0$ où la fonction H satisfait $H(0,h) = F_0(h)$ et $H(1,h) = F_1(h)$. Si H est $\mathcal{C}^m$, le théorème des fonctions implicites affirme qu'il existe localement, autour de chaque solution de $\mathcal{H}$ une fonction $\phi : \mathbb{R} \to \mathbb{R}^d$ telle que $H(t,\phi(t)) = 0$ où ϕ est $\mathcal{C}^{m-1}$.

Pour une solution de $\mathcal{F}_0$, l'image de la fonction ϕ fournie par le théorème des fonctions implicites est appelée un *chemin d'homotopie*. Si H est analytique, il est prouvé dans [GZ79] que ces chemins sont des variétés de dimension 1, plongées dans $\mathbb{R}^{d+1}$, difféomorphes à des intervalles de $\mathbb{R}$ ou à des cercles.

Le chemin passant par la solution donnée par l'esquisse est alors suivi jusqu'à trouver des solutions du système cible (voir la figure 11). On trouvera une description détaillée des méthodes par continuation dans [AG93].

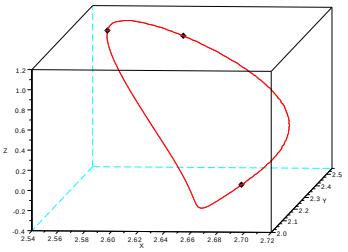

Figure 11: *Un chemin d'homotopie pour un problème nécessitant le rattrapage de 2 contraintes. Le point en $Z = 0$ est la solution initiale, les deux points en $Z = 1$ sont deux solutions trouvées lors du suivi du chemin d'homotopie.*

5.1. Méthode par continuation des paramètres

Dans le cas de la résolution de problèmes (re)paramétrés, la continuation peut être réalisée sur les paramètres du système. [MS89] justifie cette méthode dans le cas de systèmes polynômiaux ; montrons maintenant comment elle peut être adaptée à notre problème.

Une idée empruntée à [LM95] consiste à utiliser l'esquisse fournie par l'utilisateur pour construire le système de départ. Sur cette esquisse, que l'on note sk, toutes les contraintes de G sont satisfaites, mais avec d'autres paramètres. Notons k_{sk} le vecteur des paramètres des contraintes de G satisfaites par l'esquisse en remarquant que certaines de ses composantes correspondent aux paramètres des contraintes supprimées. Comme le plan de construction Cp du problème reparamétré G' dépend aussi des paramètres de toutes les contraintes de G (à l'exception de k^-), il en va de même pour la fonction F. On considère alors la fonction $F_k(h)$ qui dépend de k, le vecteur des paramètres de toutes les contraintes. On peut alors lire sur sk les paramètres des contraintes ajoutées h_{sk} et on a $F_{k_{sk}}(h_{sk}) = 0$.

Le système de départ est alors définit par $\mathcal{F}_0 : F_{k_{sk}}(h) = 0$, et le système cible par $\mathcal{F}_1 : F_{k_*}(h) = 0$, où k_* est le vecteur des paramètres des contraintes à résoudre. La fonction d'homotopie $H : \mathbb{R}^{d+1} \to \mathbb{R}^d$ est alors obtenue en déformant les paramètres k_{sk} en k_* par $H(t,h) = F_{k(t)}(h)$ où $k(t) = (1-t)k_{sk} + tk_*$ pour $t \in \mathbb{R}$.

Ici, aucune expression analytique de H n'a été calculée, puisque la fonction F est évaluée en chaque vecteur en interprétant le plan de construction *Cp* (voir 3.1). Cependant, comme chacune des instructions de *Cp* correspond à une intersection de lieux géométriques,on peut écrire ce plan de construction comme un système d'équations triangulaire par blocs, où chaque bloc correspond à une intersection associée à une primitive géométrique. Comme chaque équation de ce système est analytique et que les fonctions f_i^- évaluant les contraintes supprimées le sont aussi, H est analytique, et notamment $\mathcal{C}^\infty$ sur son domaine de définition.

Les chemins d'homotopie de H sont donc des variétés de dimension 1, et peuvent être suivies depuis les solutions connues de $\mathcal{F}_0$ jusqu'aux solutions de $\mathcal{F}_1$. Ici, une seule solution du système de départ, h_{sk}, est connue, et 5.3 montre comment suivre son chemin d'homotopie.

5.2. Les branches du plan de construction

Rappelons que $F_k(h)$ et donc H, sont des multi-fonctions (voir 3.3), et leur domaine de définition dépend de la branche considérée. Nous cherchons une caractérisation géométrique du bord de ce domaine, sur une certaine branche.

Tout d'abord, le domaine de définition de $F_k(h)$ est une intersection finie d'ensembles de définition. En effet, considérons la i-ème étape d'un plan de construction dépendant de n paramètres. Pour un ensemble de primitives construites avec les paramètres $k_0,...,k_{m_{i-1}}$, des lieux, et l' intersection correspondante, sont calculés grâce aux paramètres $k_{m_{i-1}},...,k_{m_i}$. Toutefois, cette intersection n'existe que pour certaines valeurs des paramètres, disons $(k_{m_{i-1}},...,k_{m_i}) \in \mathcal{D}_i \subset \mathbb{R}^{m_i - m_{i-1}}$. On dira que la i-ème instruction du plan de construction est définie si les lieux construits se coupent et qu'elle ne l'est pas s'ils sont disjoints ou confondus.De la même façon, les instructions précédentes sont définies pour $(k_0,...,k_{m_{i-1}}) \in \mathcal{D}_{i-1} \subset \mathbb{R}^{m_{i-1}}$. La figure produite à la i-ème étape existe donc pour $(k_0,...,k_{m_{i-1}},...,k_{m_i}) \in (\mathcal{D}_{i-1} \times \mathbb{R}^{m_i - m_{i-1}}) \cap (\mathcal{D}_i \times \mathbb{R}^{m_{i-1}})$.

Prenons un point $x = (k,h)$ du bord du domaine de définition $\mathcal{D}$ de $F_k(h)$. Comme $\mathcal{D}$ peut s'écrire $\mathcal{D} = \cap_{0 \leq i \leq l}(\mathcal{D}_i \times \mathbb{R}^{n-(m_i - m_{i-1})})$ où l est le nombre d'instructions du plan de construction, il existe un i tel que x appartienne au bord de $\mathcal{D}_i$. Il reste alors à caractériser les bords des domaines de définition $\mathcal{D}_i$, c'est-à-dire les points $x = (k,h)$ tels que toute boule centrée en x contienne à la fois des points où la i-ème instruction est définie et des points où elle ne l'est pas.

Considérons à présent le cas d'un univers géométrique 2D contenant des points, des droites, des distances et des angles (ce qui suit s'étend en 3D). On distingue deux types de points x du bord $\mathcal{D}$ d'une instruction du plan de construction. Ceux vérifiant $x \in \mathcal{D}$ et ceux tels que $x \notin \mathcal{D}$. On vérifie facilement que le premier type de points du bord correspond à des cas de tangence cercle-cercle ou cercle-droite, et le second à des situations où deux droites sont confondues ou parallèles, ou deux cercles sont concentriques et de même rayon.

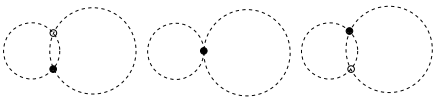

Figure 12: *La i-ème instruction du plan de construction qui consiste en l'intersection de deux cercles impose un choix entre deux solutions. Les deux figures alors construites sont les mêmes si ces cercles sont tangents.*

Si $x = (k,h)$ est sur le bord de $\mathcal{D}$ et $x \in \mathcal{D}$ alors il existe une autre branche du plan de construction sur laquelle la fonction $F_k(h)$ prend la même valeur que sur la branche actuelle. En effet, d'après ce qui a été dit plus haut, dans la figure produite par le plan de construction, deux cercles, ou un cercle et une droite, sont tangents. Si ces deux lieux sont produits par la i-ème instruction, les deux branches induites par cette instruction mènent à la construction de la même figure et $F_k(h)$ prend la même valeur sur ces deux branches. Si un chemin d'homotopie passe par un tel point x et sort du domaine de définition, on peut donc trouver une branche sur laquelle un autre chemin passe par le point x telle que ces deux chemins soient raccordables par continuité.

Les figures 12 et 13 illustrent le cas d'un suivi de chemin passant par un point du bord du domaine de définition où deux cercles deviennent parallèles. Avant de croiser ce bord (à gauche sur les figures), le chemin était suivi sur la branche induite par l'intersection marquée d'un point noir. Sur le bord (au centre sur la figure 12), les deux intersections sont confondues. En choisissant la branche induite par l'autre intersection, on peut continuer le suivi du chemin (à droite sur les figures), le mouvement des primitives des solutions ainsi que le chemin sont continus.

5.3. Suivi de courbe

Il y a principalement deux méthodes pour suivre une courbe définie par d équations indépendantes en $d+1$ variables. Ici, la courbe est définie par le système $H(x) = 0$, avec $x = (t,h)$. Comme on connaît un point x_0 vérifiant $H(x_0) = 0$, x_0 appartient à la courbe suivie.

La première méthode consiste à calculer le vecteur tangent t à la courbe en x_0 puis à calculer une solution prédite $x_0' = x_0 + \delta t$ où δ est un scalaire appelé pas de prédiction.

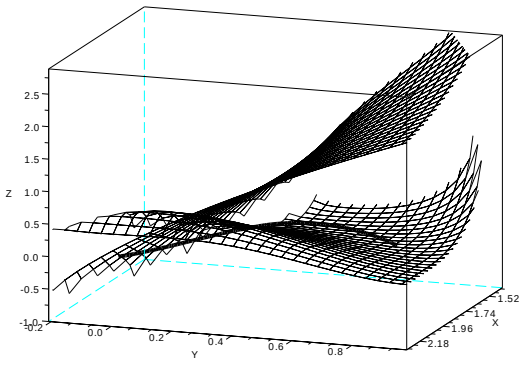

Figure 13: *Le chemin suivi, en trait plein, atteint le bord du domaine de définition et passe sur une nouvelle branche de la fonction.*

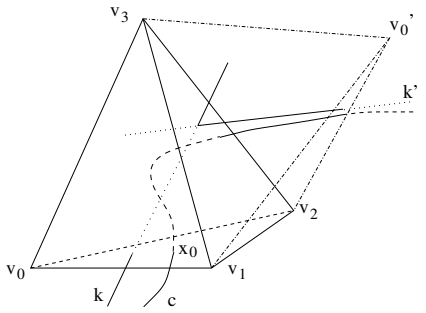

Figure 14: *Deux itérations de la méthode PLI pour suivre la courbe c. Dans $\langle v_0, v_1, v_2, v_3\rangle$ (resp. $\langle v'_0, v_1, v_2, v_3\rangle$), H^A a pour noyau k (resp. k').*

x'_0 est alors corrigé en x_1 satisfaisant $H(x_1) = 0$ en ajoutant au système une contrainte assurant l'avancement sur la courbe (par exemple x_1 appartient à l'hyperplan passant par x'_0 de vecteur normal t). La correction est effectuée par une méthode numérique itérative comme Newton-Raphson. Ces deux étapes sont répétées pour obtenir d'autres points de la courbe. L'enjeu de cette méthode est de choisir un pas de prédiction adapté à chaque courbe suivie, ce qui est abordé en utilisant l'arithmétique par intervalles dans [FM07] et [KX94].

La seconde méthode, utilisée ici, calcule un simplexe de dimension $d+1$ tel que x_0 appartienne à une de ses faces. Sur ce simplexe, H est approximée par une application affine permettant de trouver l'unique autre face du simplexe par laquelle passe la courbe. L'opération est répétée en définissant un nouveau simplexe partageant cette face avec le premier. Cette méthode, appelée interpolation linéaire par morceaux (PLI) est détaillée dans [AG97]. Nous présentons brièvement la version proposée par [DWLT90].

5.3.1. Interpolation linéaire par morceaux

Un simplexe de $\mathbb{R}^{d+1}$ est l'enveloppe convexe de $d+2$ points $v_0, ..., v_{d+1}$ de $\mathbb{R}^{d+1}$ tels que pour tout $0 \leq i \leq d+1$, v_i n'appartienne pas à l'hyperplan $\mathcal{H}_i$ passant par les points $v_0, ..., v_{i-1}, v_{i+1}, ..., v_{d+1}$. La i-ème face d'un simplexe est son intersection avec $\mathcal{H}_i$. $\langle v_0, ..., v_{d+1}\rangle$ forme un repère affine de $\mathbb{R}^{d+1}$, (i.e. la famille composée des vecteurs $v_0v_1, v_0v_2, ..., v_0v_{d+1}$ forme une base de $\mathbb{R}^{d+1}$) et l'évaluation de H en chacun des points de ce repère, si H est définie en ces points, détermine une unique application affine H^A. On notera $\langle v_0, ..., v_{d+1}\rangle$ le simplexe déterminé par les points $(v_0, ..., v_{d+1})$.

Supposons à présent que x_0 vérifiant $H(x) = 0$ appartienne à la face i de $\langle v_0, ..., v_{d+1}\rangle$. Sur $\langle v_0, ..., v_{d+1}\rangle$, la courbe suivie est assimilée au noyau de H^A qui est au moins de dimension 1 d'après le théorème du rang. Si le noyau de H^A est de dimension 1 c'est une droite, et on considère le segment $[a,b]$ définit comme l'intersection de $Ker(H^A)$ avec $\langle v_0, ..., v_{d+1}\rangle$. Si $a \in \mathcal{H}_i$ alors $\exists j \neq i \mid b \in \mathcal{H}_j$ et j est la face du simplexe par laquelle sort la courbe. On peut alors calculer un nouveau simplexe ayant sa j-ème face égale à cette dernière face, puis recommencer avec ce nouveau simplexe.

La figure 14 présente deux itérations de la méthode pour suivre une courbe de $\mathbb{R}^3$. La courbe entre dans $\langle v_0, v_1, v_2, v_3\rangle$ par la face 3 et le noyau k de l'application affine approximant H est calculé. Ce noyau intersecte la face 0 du simplexe. Dans le nouveau simplexe $\langle v'_0, v_1, v_2, v_3\rangle$ partageant sa face 0 avec s_0, un nouveau noyau k' est calculé et ainsi de suite.

Le cas où la matrice de H^A n'est pas de rang maximal, donc où son noyau est de dimension supérieure à 1, est traité dans [AG97]. En pratique, on génère les simplexes au coup par coup par réflexion : si $Ker(H^A)$ sort de $\langle v_0, ...v_{d+1}\rangle$ par la j-ème face, on prend comme nouveau simplexe $\langle v_0, ..., v_{j-1}, v'_j, v_{j+1}, ..., v_{d+1}\rangle$, où v'_j est le symétrique de v_j par rapport au centre de gravité de la face j. Notons que les calculs sont effectués en coordonnées barycentriques dans le repère formé par les sommets du simplexes courant.

5.3.2. Gestion du changement de branche

La courbe suivie peut sortir du domaine de définition et l'analyse de la figure produite par le plan de construction permet de détecter à l'avance une telle situation, avec l'instruction critique et, si elle existe, la nouvelle branche sur laquelle continue le chemin. Nous présentons maintenant une heuristique pour gérer ce changement de branche, dans le cas où l'instruction mise en défaut consiste en une intersection faisant intervenir un ou plusieurs cercles (resp. sphères) pour un GCS dans le plan (resp. l'espace).

Cette instruction amène donc à choisir entre deux intersections (voir figure 12 pour le cas de l'intersection de deux

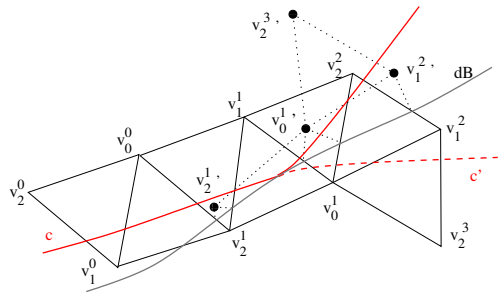

Figure 15: *Le suivi par PLI d'une courbe c passant d'une branche du plan de construction à une autre.*

cercles), que l'on nommera s_1 et s_2. On notera H_1 (resp. H_2) la fonction H sur la branche induite par le choix de s_1 (resp. s_2), B_1 (resp. B_2) le domaine de définition de H_1 (resp. H_2) et on suppose qu'avant de rencontrer dB_1 (le bord de B_1, la courbe était suivie sur la branche induite par s_1. Soit $x_* \in dB_1$ vérifiant $H_1(x_*) = 0$.

Le caractère symétrique de s_1 et s_2 nous amène à considérer que dans un voisinage de x_*, $B_1 = B_2$ et $dB_1 = dB_2$, où dB_2 est le bord de B_2, et nous noterons $B = B_i$ et $dB = dB_i$ pour $i = 1$ et 2. Dans ce même voisinage de x_*, on approxime dB par un hyperplan $\mathcal{P}$ et on considère que $x \in dB \Rightarrow H_1(x) = H_2(x)$. On construit alors dans ce voisinage une nouvelle fonction $H' : \mathbb{R}^{d+1} \to \mathbb{R}^d$ définie par :

$$H'(x) = \begin{cases} H_1(x) & \text{si } x \in B, \\ H_2(\Delta_{\mathcal{P}}(x)) & \text{si } x \notin B, \end{cases}$$

où $\Delta_{\mathcal{P}}(x)$ est le symétrique de x par rapport à $\mathcal{P}$.

Lors du suivi de la courbe c par PLI, on détecte que c s'approche du bord du domaine de définition quand H n'est pas définie en un ou plusieurs des sommets $(v_0, ..., v_{d+1})$ du simplexe courant. Après avoir détecté l'instruction de plan de construction en cause, les deux intersections s_1 et s_2 possibles, les deux fonctions H_1 et H_2 décrites précédemment et l'hyperplan $\mathcal{P}$ on applique la méthode PLI à la fonction H' construite à partir de H_1 et H_2. On suit alors une courbe c' qui est la symétrique de c par rapport à $\mathcal{P}$. Lorsque tous les sommets sont en dehors de B, on calcule les symétriques de ces sommets par rapport au bord pour construire un nouveau simplexe en lequel H_2 est complètement définie, pour continuer à suivre la courbe définie par $H_2 = 0$.

À chaque étape, l'hyperplan $\mathcal{P}$ approximant dB est calculé en considérant les arêtes ab du simplexe telles que $a \in B$ et $b \notin B$. Il y en a au moins d si au moins un des sommets est en dehors de B. On recherche alors par dichotomie pour d de ces arêtes ab un point $a' \in B$ tel que la distance de a' à dB sur l'arête ab soit plus petite qu'un $\lambda \in \mathbb{R}$ donné. Ces d points définissent l'hyperplan $\mathcal{P}$ approximant dB.

La figure 15 illustre cette procédure dans le cas d'une courbe c de $\mathbb{R}^2$. c sort de $\langle v_0^0, v_1^0, v_2^0 \rangle$ par la face 2, et le nouveau sommet v_2^1 calculé est hors du domaine de définition B de H. L'hyperplan $\mathcal{P}$ approximant dB est alors calculé grâce aux arêtes $v_0^0 v_2^1$ et $v_1^0 v_2^1$, et H est approximé par l'application affine valant $H_1(v_1^0)$ en v_1^0, $H_1(v_0^0)$ en v_0^0 et $H_2(v_2^{1,})$ en v_2^1, où $v_2^{1,}$ est le symétrique de v_2^1 par rapport à $\mathcal{P}$. Le noyau de cette application sort de $\langle v_0^0, v_1^0, v_2^1 \rangle$ par la face 1. À l'itération suivante, c sort du simplexe $\langle v_0^0, v_1^1, v_2^1 \rangle$ par la face 0, et le nouveau sommet v_0^1 n'est pas dans B. On continue ainsi jusqu'à ce qu'un simplexe ait tous ses sommets en dehors de B. Alors le nouveau simplexe est $\langle v_0^{1,}, v_1^{2,}, v_2^{3,} \rangle$ et la fonction considérée est H_2.

5.4. Pistes de travail

La méthode PLI et son adaptation aux changements de branche décrite ci-dessus est pour le moment limitée aux intersections entre un cercle (ou une sphère pour les GCS dans l'espace) et d'autres lieux géométriques. Son extension à des lieux géométriques quelconques est un de nos axes de travail. D'autre part, cette méthode suppose que le domaine de définition de la fonction définissant la courbe soit convexe. Hors il est facile de construire un exemple où ça n'est pas le cas. Il faudrait alors, pour un point de la courbe donnée, pouvoir déterminer un voisinage de ce point dans lequel le domaine de définition est convexe. Enfin, le problème de la taille des simplexes lors du suivi de la courbe reste posé. En effet, si le simplexe est trop grand, certaines variations de la courbe ne sont pas détectées, notamment quand la courbe entre et sort par une même face. Il apparaît pertinent de pouvoir adapter la taille du simplexe à la courbe suivie.

6. Conclusion

Dans le cadre de la résolution de systèmes de contraintes géométriques, les méthodes constructives jouissent de propriétés intéressantes, mais elles ne sont cependant pas assez puissantes pour résoudre de nombreux problèmes de type CAO. Ce phénomène est particulièrement marqué dans le cas de la 3D. Du point de vue inverse, les méthodes numériques efficaces ne fournissent qu'une solution du problème, qui peut ne pas satisfaire l'utilisateur. Les méthodes à base de reparamétrisation permettent de contourner ce problème en remplaçant certaines contraintes du système donné pour le rendre soluble constructivement. Il faut alors faire face à la difficulté du rattrapage numérique de ces contraintes.

Dans cet article, nous avons décrit un algorithme choisissant ces contraintes de façon à rendre le système décomposable pour obtenir des sous-problèmes avec peu de contraintes remplacées, grâce à une stratégie de répartition des contraintes ajoutées entre les sous-problèmes. Les paramètres des contraintes ajoutées menant aux solutions du problème à résoudre sont obtenues en déformant continûment l'esquisse fournie par l'utilisateur en une solution du problème grâce à un algorithme de suivi de chemin, adapté au caractère géométrique du contexte.

Nous envisageons de poursuivre ces travaux en étudiant d'autres heuristiques menant à une minimisation des contraintes remplacées dans chacune des composantes rigides, et en adaptant la reparamétrisation à d'autres stratégies de décomposition comme par exemple, la W-décomposition de [TSM*11]. Du point de vue de la résolution numérique, seules les solutions se trouvant sur le même chemin d'homotopie que la solution fournie par l'esquisse sont trouvées. Nous cherchons un moyen de trouver les autres solutions.

Références

[AG93] ALLGOWER E., GEORG K. : Continuation and path following. *Acta Numerica. Vol. 2*, Num. -1 (1993), 1–64.

[AG97] ALLGOWER E., GEORG K. : Numerical path following. *Handbook of Numerical Analysis. Vol. 5*, Num. 3 (1997), 207.

[Brü93] BRÜDERLIN B. : Using geometric rewrite rules for solving geometric problems symbolically. *Theoritical Computer Science* (1993), 291–303.

[DMS97] DUFOURD J.-F., MATHIS P., SCHRECK P. : Formal resolution of geometrical constraint systems by assembling. *Proceedings of the 4th ACM Solid Modeling conf.* (1997), 271–284.

[DWLT90] DOBKIN D., WILKS A., LEVY S., THURSTON W. : Contour tracing by piecewise linear approximations. *ACM Transactions on Graphics (TOG). Vol. 9*, Num. 4 (1990), 389–423.

[FM07] FAUDOT D., MICHELUCCI D. : A new robust algorithm to trace curves. *Reliable computing. Vol. 13*, Num. 4 (2007), 309–324.

[FS08] FABRE A., SCHRECK P. : Combining symbolic and numerical solvers to simplify indecomposable systems solving. In *ACM Symposium on Applied Computing SAC 2008* (Mar 2008), Roger L. Wainwright H. H., (Ed.), ACM, ACM Press, pp. 1838–1842. ISBN 978-1-59593-753-7.

[GHY02] GAO X.-S., HOFFMANN C. M., YANG W.-Q. : Solving spatial basic geometric constraint configurations with locus intersection. In *Proceedings of the seventh ACM symposium on Solid modeling and applications* (New York, NY, USA, 2002), SMA '02, ACM, pp. 95–104.

[GS09] GAO H., SITHARAM M. : Characterizing 1-dof henneberg-i graphs with efficient configuration spaces. In *Proceedings of the 2009 ACM symposium on Applied Computing* (New York, NY, USA, 2009), SAC '09, ACM, pp. 1122–1126.

[GZ79] GARCIA C., ZANGWILL W. : Finding all solutions to polynomial systems and other systems of equations. *Mathematical Programming. Vol. 16*, Num. 1 (1979), 159–176.

[HLS97] HOFFMANN C. M., LOMONOSOV A., SITHARAM M. : Finding solvable subsets of constraint graphs. In *CP* (1997), pp. 463–477.

[IMS11] IMBACH R., MATHIS P., SCHRECK P. : Tracking method for reparametrized geometrical constraint systems. In *13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing* (2011).

[JAS97] JOAN-ARINYO R., SOTO A. : A correct rule-based geometric constraint solver. *Computer and Graphics. Vol. 5*, Num. 21 (1997), 599–609.

[KX94] KEARFOTT R., XING Z. : An interval step control for continuation methods. *SIAM Journal on Numerical Analysis. Vol. 31*, Num. 3 (1994), 892–914.

[LGL81] LIN V. C., GOSSARD D. C., LIGHT R. A. : Variational geometry in computer-aided design. *SIGGRAPH Comput. Graph.. Vol. 15* (August 1981), 171–177.

[LM95] LAMURE H., MICHELUCCI D. : Solving constraints by homotopy. In *Proceedings of the ACM-Sigraph Solid Modeling Conference* (1995), ACM Press, pp. 134–145.

[LM96] LATHAM R. S., MIDDLEDITCH A. E. : Connectivity analysis : a tool for processing geometric constraints. *Computer-Aided Design. Vol. 28*, Num. 11 (1996), 917–928.

[Mic03] MICHELUCCI D. : Using cayley menger determinants. In *Proceedings of the Workshop on Geometric Constraint Solving)* (2003). Beijing, China (available at the URL http ://www.mmrc.iss.ac.cn/ ascm/ascm03/).

[MS89] MORGAN A., SOMMESE A. : Coefficient-parameter polynomial continuation. *Applied Mathematics and Computation. Vol. 29*, Num. 2 (1989), 123–160.

[MT10] MATHIS P., THIERRY S. E. : A formalization of geometric constraint systems and their decomposition. *Formal Aspects of Computing. Vol. 22*, Num. 2 (2010), 129–151.

[TSM*11] THIERRY S. E., SCHRECK P., MICHELUCCI D., FÜNFZIG C., GÉNEVAUX J.-D. : Extensions of the witness method to characterize under-, over- and well-constrained geometric constraint systems. *Computer-Aided Design. Vol. 43*, Num. 10 (2011), 1234 – 1249.

[Zha11] ZHANG G.-F. : Well-constrained completion for under-constrained geometric constraint problem based on connectivity analysis of graph. In *SAC* (2011), pp. 1094–1099.