22es Journées de l'Association Francophone d'Informatique Graphique, Arles, 2009

Un algorithme de complétion de systèmes de contraintes géométriques sous-contraints

Simon E.B. Thierry[1,2], Pascal Mathis[1,2] et Pascal Schreck[1,2]

[1]Université de Strasbourg
[2]LSIIT, UMR CNRS-UdS 7005

Abstract

Un système de contraintes géométriques (GCS) est dit sous-contraint s'il admet une infinité de solutions. La sous-constriction est classiquement traitée par l'ajout de contraintes au système pour se ramener au cas bien contraint. Pourtant, considérer un système sous-contraint peut être une volonté de l'utilisateur qui souhaite spécifier un objet non rigide et peut présenter un intérêt en terme d'interface utilisateur.
Cet article présente une méthode de calcul des entités géométriques à fixer pour se ramener au cas bien contraint, sans ajouter de contraintes. L'algorithme est basé sur un calcul de flux, en extension des travaux de Latham et Middleditch [LM96b]. Il permet de construire incrémentalement un GCS et, à chaque étape de la construction, de savoir quels éléments géométriques doivent être fixés pour que le système soit bien-contraint.

A geometric constraint system (GCS) is said under-constrained if it has an infinite number of solutions. The under-constrainedness is classically considered as a problem to be corrected by adding constraints to the system. Yet, solving an under-constrained system is interesting one the one hand in terms of user interface and on the other hand because the user may want to design a deformable objet.
This article presents a method to compute the geometric entities to pin down in order to get a well-constrained system, without adding constraints. The algorithme is based on a flow computation, extending the work of Latham and Middleditch [LM96b]. It allows to incrementally build a GCS and, at each construction step, know which elements need to be pinned down for the system to be well-constrained.

1. Introduction

La résolution de contraintes géométriques est une fonctionnalité clé des logiciels de Conception Assistée par Ordinateur (CAO). Le problème consiste à résoudre des contraintes de distances, d'angles, d'incidence, de tangences, *etc.* appliquées à des entités géométriques telles que des points, droites, cercles ou encore plans, sphères si la résolution porte sur des problèmes en 3D. Un système de contraintes géométriques (GCS pour *Geometrical Constraints System*) est un ensemble de telles contraintes. Il est généralement fourni par l'utilisateur qui place interactivement sous forme de cotation des contraintes sur une esquisse.

Dans un GCS, chaque entité e possède un degré de liberté $dl(e)$ et chaque contrainte c un degré de restriction $dr(c)$ correspondant au nombre de degrés de liberté qu'elle supprime. La différence $\sum dl(e) - \sum dr(c) = n$ fournit une indication sur le niveau de constriction. Un GCS est dit bien contraint s'il possède un nombre fini non nul de solutions, sous-contraint s'il en admet une infinité et sur-contraint si aucune solution n'existe. En CAO, les solveurs traitent habituellement des problèmes bien contraints mais *modulo* les déplacements. Ainsi, un triangle du plan contraint par deux distances et un angle est globalement sous-contraint (six degrés de liberté pour les trois points, sur lesquels s'appliquent trois contraintes, chacune restreignant un degré de liberté) mais admet un nombre fini de solutions à un déplacement près. Ainsi lorsqu'un système est bien contraint (*modulo* les déplacements), on a $n = 3$ en 2D et $n = 6$ en 3D. La réciproque est évidemment fausse, la figure 1 montre par exemple, pour $n = 3$, un problème en 2D contenant une partie sur-contrainte (le triangle $p_3p_4p_5$) et une autre sous-contrainte admettant une articulation.Les solveurs prennent en compte l'invariance par déplacement en fixant un repère

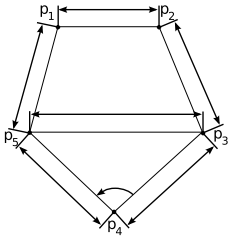

Figure 1: *GCS sur-contraint avec cinq points contraints par six distances et un angle. Si on retire une contrainte du triangle $p_3p_4p_5$ le GCS devient sous-contraint.*

lors de la résolution (en 2D, un point et la direction d'une droite par exemple).

Pour résoudre les GCS, plusieurs approches existent. L'approche algébrique consiste à traduire le GCS en un système d'équations, résolu soit symboliquement soit numériquement. Les méthodes symboliques [GC98], très couteuses, sont réservées à la résolution de petits systèmes, les méthodes numériques sont quant à elles soit couteuses (homotopie par exemple [LM96a]) soit ne fournissent qu'une solution (Newton-Raphson). D'autre part, l'approche géométrique [JASR97, Kra92, Sch94] s'appuie sur des constructions géométriques de type construction à la règle et au compas. La solution prend alors la forme d'un plan de construction où étape par étape sont indiqués les procédés de construction (intersection entre deux cercles pour déterminer la position d'un point par exemple). L'évaluation du plan de construction, en prenant en compte les multiples solutions produites par les intersections de lieux géométriques, fournit numériquement plusieurs solutions.

Il peut être cependant difficile pour l'utilisateur de saisir le nombre suffisant de contraintes pour obtenir un système bien contraint lorsque celui-ci atteint et dépasse la dizaine d'entités. Son intention est alors de laisser les parties sous-contraintes comme elles le sont sur l'esquisse. Par ailleurs, l'utilisateur peut vouloir spécifier un objet articulé (une paire de ciseaux, une lampe de bureau).

Dans cet article, nous nous intéressons à la résolution de problèmes sous-contraints. Nous supposons disposer d'un solveur géométrique capable, à partir d'un problème bien contraint, de fournir un plan de construction. La résolution que nous présentons se déroule ainsi : un algorithme de flux permet dans un premier temps de déterminer un ensemble d'éléments à fixer pour obtenir un système où le nombre de degrés de liberté est égal au nombre de degrés de restriction. Á partir de cette paramétrisation, le solveur géométrique produit un plan de construction pour ce système alors bien contraint. Pour finir, des valeurs numériques sont données aux éléments fixés et le plan de construction peut être numériquement interprété pour produire les solutions.

Cet article s'organise comme suit. La section 2 présente les travaux liés aux systèmes sous-contraints et aux travaux que nous étendons ; la section 3 présente l'algorithme de calcul des degrés de liberté à fixer ; la section 4 explique comment interpréter le flot obtenu par l'algorithme ; la section 5 donne des éléments de stratégie pour choisir une bon flot ; enfin, la section 6 conclut et présente les perspectives de ces travaux.

2. Travaux liés

Les problèmes de cinématique inverse font habituellement intervenir des systèmes sous-contraints formés de barres rigides pour lesquelles les degrés de liberté sont connus et les valeurs de paramètres (notamment les angles) déterminés par des méthodes numériques. Nous nous plaçons ici à la fois dans un cadre plus général dans lequel doivent pouvoir s'insérer de nouveaux types d'entités mais aussi dans un cadre plus étroit pour la recherche des valeurs numériques car nous supposons qu'un plan de construction pourra être fourni.

Pour le traitement de GCS sous-contraint, le principal travail réalisé dans ce domaine est présenté dans [JASRVMVP04]. Le GCS est traduit sous forme de graphe dans lequel les entités sont des sommets et les contraintes des arcs. L'idée consiste à incrémentalement enrichir le graphe de nouvelles contraintes afin de rendre le problème initial bien contraint. Parmi les différentes complétions possibles, il s'agit de trouver celle qui permettra à un solveur géométrique connu de résoudre le système complété. D'autres travaux, consistant également à compléter le GCS en ajoutant de nouvelles contraintes géométriques, existent dans la littérature [ZG06].

Notre approche est différente dans la mesure où il ne s'agit pas de compléter un GCS mais de fixer directement certains éléments. Pour cela nous étendons un algorithme de triangularisation par bloc de GCS [LM96b] s'appuyant sur la recherche d'un flux maximum dans un graphe. Cette méthode a été étendue en intégrant des connaissances géométriques [JNT03].

Les articles décrivant des méthodes géométriques permettant de résoudre des GCS sous-contraints sont peu nombreux et admettent la faiblesse de leur pouvoir de résolution, qu'il s'agisse d'analyse des degrés de liberté d'assemblages d'objets rigides [Kra92] ou de décomposition en sous-systèmes rigides [TMS07].

3. Algorithme de calcul de flux

Dans cette section, nous présentons l'algorithme permettant de calculer l'ensembles des repères d'un GCS. Il se base sur l'algorithme de Latham et Middleditch [LM96b] qui fait l'hypothèse que le système à résoudre est rigide et qu'il est donc nécessaire de fixer 3 degrés de liberté au système afin de figer les solutions dans le plan. Pour cela, une

contrainte virtuelle est ajoutée et connectée à toutes les entités géométriques. Afin de retirer l'hypothèse de la rigidité du GCS, notre algorithme ne considère pas de contrainte virtuelle à saturer impérativement. En revanche, chaque entité est liée à un sommet qui dispose d'autant de degrés de constriction que l'entité a de degrés de liberté. Ces sommets, qui représentent les éléments de repère, ne seront pas nécessairement saturés. Nous déterminons ainsi quels degrés de liberté doivent être fixés.

L'algorithme présenté ici va calculer un repère du GCS. Á partir de cette paramétrisation, un solveur, considéré connu, trouvera un plan de construction. S'il échoue, une modification du flot permettra lui fournir une autre paramétrisation. Le plan de construction sera ensuite interprété numériquement pour fournir une solution.

3.1. Principe

Le graphe de flux (dont les valuations sont toutes positives ou nulles) correspondant à un système de contraintes géométriques est donc construit ainsi :

- le graphe a un sommet source S et un sommet final F ;
- à chaque entité e correspond un sommet s_e. Chaque sommet s_e est lié à S par un arc orienté de S vers s_e, valué par le nombre de degrés de liberté de e ;
- à chaque contrainte c correspond un sommet s_c. Chaque sommet s_c est lié à F par un arc orienté de s_c vers F, valué par le nombre de degrés de restriction de c ;
- si la contrainte c est liée à l'entité e, alors le graphe de flux inclut un arc de s_e vers s_c, dont la valuation maximale est le nombre dedegrés de libertés que c peut ôter à e ;
- pour chaque entité e, un sommet r_e est ajouté, ainsi qu'un arc de s_e vers r_e et un arc de r_e vers F. L'arc de r_e vers F a pour valuation le nombre de degrés de liberté de e, qui est donc la valuation maximale de l'arc de s_e vers r_e.

Dans les représentations graphiques présentées dans cet article, le sommet S et le sommet F n'apparaissent pas explicitement. Les valuations des arcs de ces sommets étant fixes, ils nuisent à la compréhension d'un graphe représentant un GCS complexe plus qu'ils n'y aident.

L'objectif de l'algorithme décrit ici va être de calculer un flot valide, c'est-à-dire une valuation des arcs du graphe de flux, telle que :

- chaque sommet s_e correspondant à une entité est saturé : la somme des valuations des arcs sortant de s_e est égale à la valuation de l'arc allant de S vers s_e ;
- chaque sommet s_c correspondant à une contrainte est saturé : la somme des valuations des arcs entrants de s_c est égale à la valuation de l'arc allant de s_c vers F ;
- les sommets de repère ne sont pas nécessairement saturés : la valuation de l'arc entrant d'un sommet r_x est inférieure ou égale à la valuation de l'arc sortant de r_x.

Les valuations des arcs allant d'un sommet s_e vers un sommet r_e (que nous appelerons *degré de repère* de s_e dans la suite) indiquent les degrés de liberté du système et, donc, les entités à fixer – éventuellement partiellement – pour que le GCS ne soit pas sous-contraint. Dans le cas d'un système décrivant un objet rigide, la somme des valuations d'entrée des sommets r_x est ainsi 3 en 2D et 6 en 3D. L'interprétation géométrique de ces valuations est discutée en section 4.

L'ajout des sommets r_x permet d'éliminer l'hypothèse de non sous-constriction du GCS. Elle a toutefois le lourd désavantage d'empêcher la détection de systèmes sur-contraints. Ainsi, à la figure 1, qui représente un pentagone sur-contraint de par la présence de quatre contraintes dans le triangle $p_3p_4p_5$, notre algorithme associera une valuation du flux indiquant qu'il faut fixer le point p_4 et la direction p_5p_1 pour que le système soit bien contraint. Pourtant, ce système est structurellement sur-contraint. Pour cette raison, notre algorithme part de l'hypothèse – forte – que le système n'est pas sur-contraint. Nous détectons les cas de sur-constriction par une analyse de la matrice jacobienne du GCS [MF09].

3.2. Algorithme

Afin de permettre à l'utilisateur de corriger le système de contraintes géométriques s'il a trop de degrés de libertés, l'algorithme présenté ici est incrémental. Il est initialisé en considérant une entité géométrique e quelconque et le graphe de flux correspondant, contenant S, F, s_e et r_e. En valuant l'arc de s_e à r_e par le nombre de degrés de liberté de e, ce graphe de flux initial est valide.

L'ajout d'une nouvelle entité s'effectue, similairement, en ajoutant le sommet lui correspondant, le sommet correspondant à son repère, et en saturant ces deux sommets. L'ajout d'une nouvelle contrainte c se fait lorsque toutes les entités liées à cette contrainte sont présentes dans le graphe, puis en baissant les valuations d'entrée de certains sommets de repère afin de pouvoir saturer s_c. Pour ceci, nous recherchons un chemin sans cycle de s_c vers un sommet r_x tel que :

- r_x est au moins partiellement saturé : la valuation de son arc d'entrée est non nulle ;
- pour chaque sommet s_e du chemin, la valuation de l'arc lié à s_e et situé du côté de s_c est augmentable (inférieure au degré de liberté de e et au degré de restriction de la contrainte liée à cet arc) ;
- de même, pour chaque somme s_e du chemin, la valuation de l'arc lié s_e et situé du côté de r_x est diminuable (non nulle).

Une fois un tel chemin trouvé, il faut «retourner» ce chemin : pour chaque contrainte c' du chemin (c excepté), la valuation de l'arc du côté de c est diminuée de 1 et celle de l'arc du côté de r_x est augmentée d'autant. Il faut alors diminuer de 1 la valuation de l'arc d'entrée de r_x et, enfin, augmenter de 1 celle de l'arc partant de s_c.

On répète cette opération (recherche de chemin, retournement de chemin) autant de fois que le degré de restriction de

c, afin que s_c soit saturé. L'algorithme 1 détaille l'opération d'ajout d'une contrainte, en faisant appel à l'algorithme 2 pour chercher un chemin. L'algorithme 2 recherche les chemins valides avec comme sommet initial une entité géométrique donnée. L'algorithme 1 utilise donc cet algorithme sur chacune des entités liées à la contrainte à ajouter.

En outre, ces deux algorithmes utilisent une liste de degrés de repère à respecter ainsi qu'une stratégie de choix de repère. Ces deux éléments seront discutés plus avant dans la suite de l'article, il suffit pour le moment de savoir que la liste permet à l'algorithme de ne pas baisser le degré de repère d'une entité en-dessous d'une valeur indiquée par l'utilisateur.

Algorithm 1 `AjoutContrainte`

Données: G: graphe de flux actuel ; $c(e_1,...e_n)$: contrainte à ajouter ; $L = \{e_i, d_i\}$: liste des obligations de repère ; S: stratégie de choix de repère
Liste $E := (s_{e_1}, ..., s_{e_n})$
Liste $P :=$ liste vide des chemins trouvés
pour chaque entité $e \in E$ **faire**
  Arbre $T :=$ nouvel arbre avec e comme unique noeud
  $P := P +$ `RechercheChemins(`G`, `e`, `L`, `T`)`
fin faire
$p :=$ `MeilleurChemin(`P`, `S`)`
`RetournerChemin(`G`, `p`)`
Ajouter un sommet s_c lié à l'ensemble des sommets de E
Valuer les arcs de s_c par 0
Valuer l'arc de s_c au sommet initial de p par 1
renvoyer G

Algorithm 2 `RechercheChemins`

Données: G: graphe de flux ; e: entité dont on veut augmenter le degré de repère ; $L = \{r_i, d_i\}$: liste des obligations de repère ; T: arbre des chemins parcourus
pour chaque contrainte c liée à e **faire**
  si c ne relie pas e à son parent **alors**
    si c est retournable **alors**
      pour chaque entité x liée à c sauf e **faire**
        ajouter x dans T avec e pour parent
        si on peut retirer un degré de repère à x **alors**
          marquer x comme final dans T
        fin si
        `RechercheChemins(`G`, `x`, `L`, `T`)`
      fin faire
    fin si
  fin si
fin faire
renvoyer G

L'algorithme 1 est une version de l'algorithme d'ajout de contrainte, qui cherche exhaustivement tous les chemins possibles pour ensuite sélectionner le meilleur en fonction d'une stratégie de choix. Il est bien évidemment possible d'adapter ces algorithmes afin de se contenter du premier chemin trouvé au terme d'un parcours (en profondeur d'abord ou en largeur d'abord) ou de disposer d'une heuristique permettant de décider si l'on continue à chercher ou non, en fonction de la qualité du repère calculé. De même, cet algorithme a été écrit de manière simplifiée pour le cas où la contrainte avait un degré de restriction de 1, mais peut être aisément modifié pour d'autres contraintes.

Il est également possible d'adapter l'algorithme `AjoutContrainte` afin de modifier le flot sans ajouter de contrainte et ainsi chercher d'autres repères. En effet, en remplaçant la contrainte c par une entité e dans les paramètres d'`AjoutContrainte` (et en n'effectuant donc qu'un tour de boucle), il est possible de trouver un chemin avec pour sommet initial s_e et pour sommet final un sommet de repère r_x partiellement saturé. En retournant ce chemin, il est possible d'augmenter de 1 la valuation de l'arc de s_e vers r_e. Ainsi, l'utilisateur peut, à partir d'un repère qui lui est proposé, demander à essayer un autre repère. Dans le cas où il voudrait fixer plusieurs degrés de repère, nous ajoutons, après chaque augmentation de la valuation de l'arc vers un sommet de repère r_x, le couple (r_x, d_x) à la liste des obligations de repère, avec d_x le nombre de degrés de libertés que l'utilisateur veut fixer pour l'entité x. Dans l'algorithme 2, la liste des obligations de repère est prise en compte lors de la vérification de la possibilité de «retirer un degré de repère à x» : il faut pour satisfaire cette condition que le nombre de degrés de repères de x, c'est-à-dire la valuation de l'arc de s_x vers r_x, soit non nulle et supérieure à d_x quand (r_x, d_x) apparaît dans la liste des obligations de repère.

3.3. Exemple

Considérons l'exemple de la figure 2, représentant un GCS constitué d'une chaîne fermée non rigide de quatre barres et d'une cinquième barre en rotation libre autour d'une des points de la chaîne. Le graphe de flux initial est représenté à la figure 3A), avec uniquement le point p_1 et le sommet de repère r_{p_1}. La figure 3B) montre le graphe de flux obtenu après ajout de p_2 et de la contrainte de distance entre p_1 et p_2. De par la saturation de la contrainte de distance, un degré de repère a été retiré à p_2. À la figure 3C), p_3 et p_4 ont été ajoutés, ainsi que les contraintes concernant les quatre entités considérés. À la figure 3D) le point p_5 a été ajouté, ainsi que deux des contraintes de distance le concernant : celles liant p_2 et p_3 respectivement. La figure 3E) montre l'ajout de la dernière contrainte de distance (non encore saturée) et un chemin possible à retourner (en pointillé) dans lequel le degré de restriction de la nouvelle contrainte remet en cause l'un des degrés de repère de p_3. Enfin, la figure 3E) montre le flot résultant après cette exécution de l'algorithme. La figure 5 montre une interprétation géométrique de ce flot.

Comme évoqué dans la section 3.2, l'algorithme 1 peut être adapté afin de modifier le flot. La figure 4 montre une

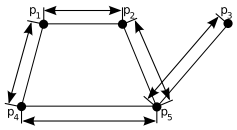

Figure 2: *Système déformable fait d'une chaîne fermée de 4 barres rigides et d'une barre en libre rotation autour d'un point de la chaîne*

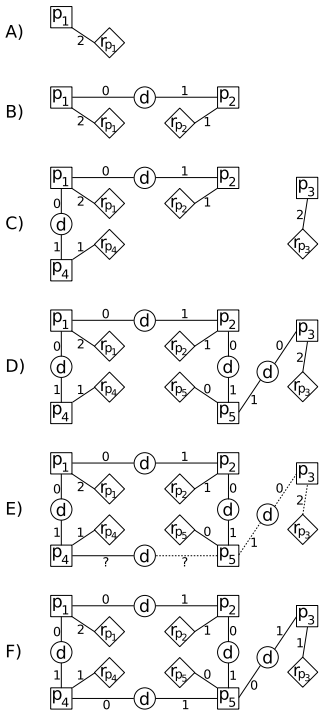

Figure 3: *Quelques étapes de l'algorithme 1 sur l'exemple de la figure 2 : A) initialisation avec un point et son repère ; B) ajout d'un point et d'une contrainte ; C) ajout de deux points et des contraintes les concernant ; D) système entier sauf une contrainte ; E) un chemin possible à retourner en pointillé ; F) un flot possible*

telle modification où, en partant du flot final de la figure 3, l'utilisateur a demandé à augmenter le degré de repère du point p_3. Le repère qui correspond à ce flot est illustré à la figure 6.

4. Interprétations géométriques d'un flot

L'algorithme présenté dans la section 3 permet de calculer un flot à partir d'un GCS. Cette section présente les interprétations que l'on peut faire d'un flot donné.

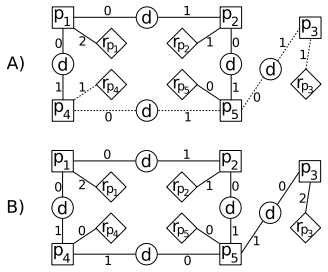

Figure 4: *Changement de flot à partir du résultat de la figure 3: A) un chemin possible à retourner pour ajouter un degré de repère à p_3 ; B) flot résultant*

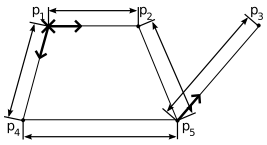

Figure 5: *Repère correspondant au flot final de la figure 3*

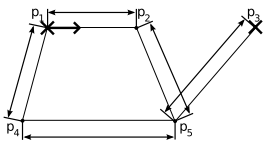

Figure 6: *Repère correspondant au flot final de la figure 4*

4.1. Repère du système de contraintes géométrique

Le flot indique, par les valuations des arcs vers les sommets de repère, combien de degrés de liberté doivent être fixés sur les différentes entités du GCS. Elle n'indique pas, toutefois, quelle est l'interprétation géométrique de la fixation de ces degrés de liberté.

Lorsqu'un sommet de repère est saturé, cela signifie que l'entité associée est entièrement fixée. L'interprétation est alors immédiate : les coordonnées de l'entité doivent être données. Lorsque le sommet de repère n'est pas saturé, en revanche, il y a plusieurs interprétations possibles.

Dans le plan par exemple, s'il faut fixer un degré de liberté d'un point, l'une ou l'autre de ses coordonnées peut être fixée. Si le sommet correspondant à ce point dans le graphe de flux est saturé grâce à une contrainte de distance, il est aussi possible de fixer, comme repère, la direction entre ce point et l'autre point contraint concerné par la contrainte de distance. D'autres interprétations sont encore possibles dans le cas où le point est lié à une contrainte d'angle.

De même, fixer un degré de liberté d'une droite peut revenir à fixer sa direction (par exemple si elle est liée à une contrainte d'incidence avec un point) ou à positionner son intersection avec une autre droite (cas d'une contrainte d'angle). Les exemples sont plus nombreux encore en 3D.

Certaines interprétations seront préférables à d'autres. La section 5.2 montre par exemple que les constructions géométriques peuvent être plus ou moins sensibles à l'échec selon la paramétrisation et son interprétation. En outre, un flot peut être valide au sens de la définition donnée

à la section 3.1 mais ses interprétations géométriques ne pas être valides géométriquement, car elles rendent le système sur-contraint. Par exemple, en considérant le GCS représenté à gauche sur la figure 8, un flot possible calculé par l'algorithme 1 est donné. Une interprétation géométrique de ce flot pourrait être de fixer le point p_1, et les directions des droites d_1 et d_2. Pourtant, l'angle entre d_1 et d_2 étant contraint, fixer ces deux directions sur-contraint le GCS. Une autre interprétation, valide quant à elle, est de fixer le point p_1, la direction de la droite d_1 et la position, sur d_1, de l'intersection de d_1 et de d_2.

Ce type de problème ne peut *a priori* pas se détecter par une analyse simple du graphe de flux et nécessite de réintroduire des connaissances géométriques, par exemple par un système à base de règles. Même cette approche peut être mise en défaut si la base de règles n'est pas suffisante. Il est toutefois possible de détecter la sur-constriction par interrogation du témoin [MF09].

4.2. Déduction d'un plan de construction

De même que dans les travaux de Latham et Middleditch [LM96b], il est possible d'opérer une triangularisation par blocs du GCS et ainsi de déduire un plan de construction, en calculant, à partir d'un flot f, un graphe $G(f)$ orienté dont les arcs ne sont pas valués. Ce graphe est calculé ainsi :

- les sommets de $G(f)$ sont les sommets du graphe de flux f, à l'exception des sommets de repère dont l'arc a une valuation nulle ;
- s'il y un arc avec une valuation non nulle entre deux sommets de f, il y a un arc orienté doublement entre ces sommets dans $G(f)$;
- s'il y a un arc avec une valuation nulle entre deux sommets de f, il y a un arc orienté vers le sommet représentant une contrainte dans $G(f)$.

L'orientation des arcs dans le graphe $G(f)$ traduit une relation de dépendance pour la construction : s'il y a un arc du sommet d'entité s_e vers le sommet de contrainte s_c, cela signifie que l'entité e devra être construite avant de pouvoir utiliser la contrainte c dans une construction géométrique permettant de construire la ou les entités contraintes par c et dont le sommet est lié à s_c par un arc avec une valuation non nulle.

À partir du graphe $G(f)$ d'un flot f, il est possible de construire un *graphe réduit* en calculant les composantes fortement connexes maximales. Ceci permet de déduire un plan de construction partiel du GCS. En effet, un arc allant d'une composante à une autre indique un ordre de construction entre ces composantes. On obtient ainsi un graphe de dépendance indiquant quelles entités doivent être construites d'abord (celles dont les sommets de repère sont saturés) puis l'ordre dans lequel les sous-systèmes doivent être construits.

La figure 7 montre le calcul du graphe réduit à partir du flot final de la figure 3. Il indique le plan de construction

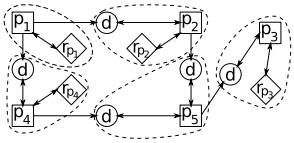

Figure 7: *Calcul du graphe réduit du flot de la figure 3F). Les composantes fortement connexes maximales sont indiquées en pointillés.*

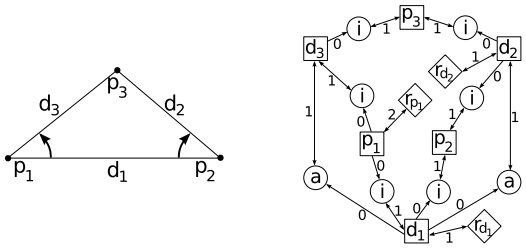

Figure 8: *Triangle invariant par similitude, avec six contraintes d'incidence point-droite et deux contraintes d'angle. À gauche : esquisse ; à droite : flot valide.*

suivante : fixer le point p_1 ; fixer la direction p_1p_2 et construire p_2 comme l'intersection d'un cercle de centre p_1 et de la droite passant par p_1 et de direction précédemment fixée ; similairement, fixer la direction p_1p_4 et construire p_4 ; construire p_5 comme l'intersection de deux cercles, de centres p_2 et p_4 respectivement ; fixer la direction p_5p_3 et construire p_3.

Au sein d'une composante, la présence de plusieurs sommets d'entités, c'est-à-dire une interdépendance de ces entités dans la construction, signifie que, dans ce flot, il n'est pas possible de déduire un ordre de construction : le sous-système correspondant à cette composante doit être construit d'un coup. La présence d'une composante fortement connexe correspondant à un sous-système que les solveurs utilisés ne savent pas résoudre mène donc à un échec du calcul d'un plan de construction.

5. Selection du bon repère

L'algorithme 1 (*cf.* section 3, p.4) permet de calculer un flot indiquant les degrés de libertés à fixer dans le GCS correspondant pour qu'il soit structurellement bien contraint. La section 4 montre qu'à un flot donné on peut associer plusieurs repères différents et un plan de construction. Cette section discute donc la qualité d'un flot en fonction de ce que l'on peut en déduire et explicite la notion de stratégie abordée dans l'explication de l'algorithme 1.

5.1. Plan de construction

Deux flots différents peuvent aboutir à deux graphes réduits différents et donc à deux plans de construction différents.

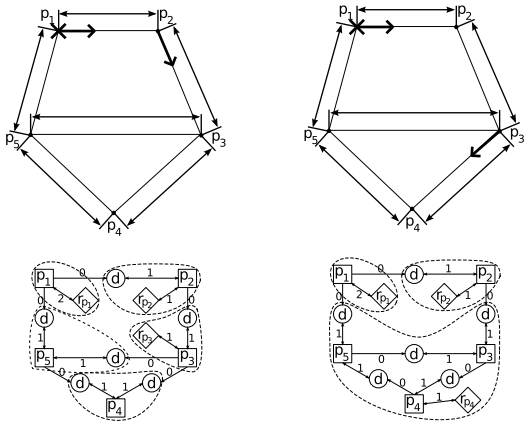

Figure 9: *Pentagone contraint par six distances. À gauche : flot permettant de déduire un plan de construction ; à droite : flot problématique (une fois p_1 et p_2 construits, tout le reste du système doit être résolu d'un coup)*

Dans certains cas, des composantes fortement connexes de grande taille apparaissent, pour lesquelles le plan de construction n'indique donc pas d'ordre de construction. Il est alors nécessaire de recourir à un solveur externe qui sera capable de trouver les différentes solutions du sous-système correspondant.

La figure 9 est un exemple de ce problème : pour le GCS, constitué de cinq points contraints par six distances, deux flots valides sont donnés. À gauche, le graphe réduit indique qu'il faut fixer p_1 et la direction p_1p_2, en déduire p_2, puis fixer la direction p_2p_3 et en déduire p_3. Ensuite, p_5 se construit comme l'intersection de cercles de centre p_1 et p_3 respectivement, puis on construit similairement p_4 à partir de p_3 et p_5. À droite, le graphe réduit indique qu'il faut fixer p_1 et la direction p_1p_2 puis construire p_2. Ensuite, il est nécessaire de connaître une construction géométrique atomique qui, à partir de p_1 et p_2 et des distances p_1-p_5 et p_2-p_3, permet de construire p_3, p_4 et p_5.

5.2. Validité des paramètres

Le repère calculé à la figure 5 est constitué d'un point et de trois directions. Pour le même GCS, le repère calculé à la figure 6, est constitué de deux points et d'une direction. Si les deux repères sont géométriquement valides, les risques de ne pas réussir à calculer une solution numérique, à partir d'une valuation des paramètres correspondant aux éléments de repère à fixer, sont plus élevés avec le repère de la figure 6 : si les deux points fixés par l'utilisateur sont trop éloignés, aucune solution n'existe.

De même, sur le GCS de la figure 10, un repère valide peut être constitué du point p_1, de la direction p_1p_2, puis de la direction p_2p_3. Cependant, certaines valeurs des paramètres

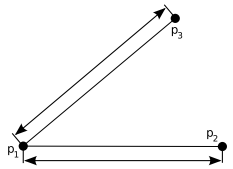

Figure 10: *Système articulé fait de deux barres rigides*

ne permettront pas de réussir une construction, par exemple si la distance p_1-p_3 est inférieure à la distance p_1-p_2 et que les deux directions choisies sont orthogonales. En revanche, si le repère choisi consiste en la fixation du point p_1 et des directions p_1p_2 et p_1p_3, alors toutes les valeurs de paramètre possibles aboutiront à une construction valide.

À partir d'études d'exemples, nous avons élaboré trois heuristiques permettant la découverte de repères offrant une bonne paramétrisation :

- défavoriser les repères avec plusieurs sommets de repère saturés et favoriser au contraire des repères avec peu de «racines» et des interprétations où les éléments de repère sont utilisés dans des constructions géométriques ;
- favoriser les repères où les différents éléments de repère interviennent «tôt» dans la construction ;
- favoriser les repères induisant des constructions géométriques où les cas d'échec sont dûs à des cas dégénérés (par exemple, construire une droite à partir de deux points) par rapport à ceux induisant des constructions où les valeurs des paramètres peuvent faire échouer la construction (par exemple l'intersection d'un cercle et d'une droite qui n'a pas été construite à partir d'un point situé dans le cercle).

La logique des deux premières heuristiques est de permettre aux constructions géométriques d'adapter la solution aux paramètres fournis par l'utilisateur. La première a en outre l'avantage d'éviter les repères où plusieurs points différents sont fixés, ce qui implique virtuellement des contraintes de distances supplémentaires entre ces points, et d'y préférer des repères où l'on fixe des directions, c'est-à-dire des ajouts virtuels de contraintes d'angles. Les contraintes d'angles étant invariantes par similitude, elles réduisent moins les libertés du système que des contraintes de distance, invariantes uniquement par déplacement.

Toutefois, il s'agit là que d'heuristiques et il est possible de trouver des exemples les mettant en défaut. Fort heureusement, dans la très large majorité des cas, les valeurs de paramètre de l'esquisse sont valides, même avec des repères déconseillés par nos heuristiques. Dans les cas où les valeurs de l'esquisse ne conviennent pas non plus, des méthodes numériques permettent de calculer de nouvelles valeurs, à partir de celles de l'esquisse [FS08].

5.3. Stratégies de calcul d'un flot

Avec les eléments de «qualité d'un repère» évoqués plus haut, on peut élaborer des stratégies de calcul du flot. Dans l'algorithme 1, la stratégie intervient pour choisir le chemins permettant un retournement et, donc, la paramétrisation du GCS qui en découle. Elle permet d'éliminer les flots interprétés en repères sur-contraignants et de comparer les chemins en fonction des plans de constructions qu'ils impliquent ou en utilisant les heuristiques citées en section 5.2.

Une telle utilisation de la stratégie n'est toutefois pas idéale, en cela qu'elle mène vers des maxima locaux, et que de bons flots ne sont peut-être atteignables que par des retournements de chemins faisant passer par des flots mal évalués. Un calcul exhaustif de tous les flots possibles serait trop coûteux.

Avec cette stratégie, notre méthode générale consiste donc à chercher une paramétrisation du GCS dont les heuristiques, notamment la triangularisation par blocs et sa traduction en plan de construction, nous permettent d'espérer que le solveur sera capable d'en déduire un plan de construction. S'il échoue, le flot est modifié pour proposer d'autres paramétrisations. Le plan de construction est ensuite interprété numériquement en se basant sur les valeurs de l'esquisse. Selon la paramétrisation et son interprétation, il est possible que certaines phases du plan de construction échouent. L'utilisation d'une méthode numérique telle que Newton-Raphson permet alors de modifier les valeurs des paramètres pour aboutir à une solution.

6. Conclusion

Nous avons exposé un algorithme consistant à traduire un GCS sous forme de graphe et à calculer un flux saturant les sommets correspondant aux entités et aux contraintes. L'ajout de sommets correspondant aux éléments de repère potentiels, non obligatoirement saturés, permet de considérer des systèmes sous-contraints et de déterminer quels degrés de liberté doivent être fixés pour que le système ait un nombre fini non nul de solutions.

Cet algorithme, qui fonctionne aussi bien pour des GCS 2D que 3D, est incrémental et permet donc à l'utilisateur de construire son GCS pas à pas. À chaque étape, l'interprétation du flot sous la forme d'un repère donne à l'utilisateur une intuition des libertés du système et donc de sa déformabilité. Ainsi, cet article ouvre de nouvelles voies pour une amélioration de l'interface utilisateur dans des logiciels de CAO s'adressant à des concepteurs non-experts.

La méthode présentée dans ce papier ne permet pas de détecter les cas de sur-constriction, ce qui est un handicap majeur pour une méthode incrémentale. Ces cas peuvent toutefois être détectés par d'autres moyens, par exemple la méthode du témoin [MF09]. Les résultats de cet algorithme posent en outre des questions sur des méthodes pour obtenir des valeurs valides des paramètres correspondant au repère, adaptées par de méthodes existantes [FS08].

Les perspectives de ces travaux incluent une étude plus poussée des composantes fortement connexes du graphe réduit. Nous espérons, en intégrant des connaissances géométriques à l'algorithme, réussir à obtenir une décomposition du système en sous-systèmes dont le groupe de bonne constriction est connu et ainsi répondre à deux problèmes actuels : la détection de la sur-constriction et la décomposition des sous-systèmes engendrés par des composantes fortement connexes de trop grande taille dans le graphe réduit.

References

[FS08] FABRE A., SCHRECK P.: Combining symbolic and numerical solvers to simplify indecomposable systems solving. In *SAC '08* (2008), ACM, pp. 1838–1842.

[GC98] GAO X.-S., CHOU S.-C.: Solving geometric constraint systems I. A global propagation approach. *Computer-Aided Design 30*, 1 (1998), 47–54.

[JASR97] JOAN-ARINYO R., SOTO-RIERA A.: A correct rule-based geometric constraint solver. *Computer and Graphics 5*, 21 (1997), 599–609.

[JASRVMVP04] JOAN-ARINYO R., SOTO-RIERA A., VILA-MARTA S., VILAPLANA-PASTO J.: Revisiting decomposition analysis of geometric constraint graphs. *Computer-Aided Design 36*, 2 (2004), 123–140.

[JNT03] JERMANN C., NEVEU B., TROMBETTONI G.: Algorithms for identifying rigid subsystems in geometric constraint systems. In *IJCAI'03* (2003), pp. 233–238.

[Kra92] KRAMER G. A.: A geometric constraint engine. *Artificial Intelligence 58*, 1-3 (1992), 327–360.

[LM96a] LAMURE H., MICHELUCCI D.: Solving geometric constraints by homotopy. *IEEE Transactions on Visualization and Computer Graphics 2*, 1 (1996), 28–34.

[LM96b] LATHAM R. S., MIDDLEDITCH A. E.: Connectivity analysis: a tool for processing geometric constraints. *Computer-Aided Design 28*, 11 (1996), 917–928.

[MF09] MICHELUCCI D., FOUFOU S.: Interrogating witnesses for geometric constraint solving. In *SPM '09* (2009), SIAM/ACM, pp. 343–348.

[Sch94] SCHRECK P.: A knowledge-based for solving geometric constructions problems. In *7th Int. Conf. on Systems research, Informatics and Cybernetics* (1994), pp. 19–24.

[TMS07] THIERRY S., MATHIS P., SCHRECK P.: Towards an homogeneous handling of under-constrained and well-constrained systems of geometric constraints. In *SAC '07* (2007), ACM, pp. 773–777.

[ZG06] ZHANG G.-F., GAO X.-S.: Well-constrained completion and decomposition for under-constrained geometric constraint problems. *IJCGA 16*, 5,6 (2006), 18–35.