N° d'ordre: 635

UNIVERSITÉ DE STRASBOURG

THÈSE

présentée pour obtenir le grade de

Docteur en informatique de l'Université de Strasbourg

par

Simon E.B. THIERRY

Décomposition et paramétrisation de systèmes de contraintes géométriques sous-contraints

Soutenue publiquement le **16 septembre 2010**
devant la commission d'examen composée de :

M. Dominique MICHELUCCI ..... *Président*
Professeur à l'Université de Bourgogne

M. Robert JOAN-ARINYO ..... *Rapporteur*
Professeur à l'Universitat Politècnica de Catalunya

M. Bertrand NEVEU ..... *Rapporteur*
Ingénieur en chef des ponts, eaux et forêts à l'École des Ponts ParisTech

M. Pascal MATHIS ..... *Examinateur*
Maître de conférences à l'Université de Strasbourg

M. Pascal SCHRECK ..... *Directeur de thèse*
Professeur à l'Université de Strasbourg

Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection

UMR CNRS-UdS 7005

Résumé

La résolution de systèmes de contraintes géométriques (GCS) a pour objectif de produire des figures qui respectent une description technique fournie par l'utilisateur sous la forme d'une esquisse cotée. Le GCS donné par l'utilisateur peut être bien contraint (il décrit un nombre fini non nul de figures), sous-contraint (une infinité de figures) ou sur-contraint (aucune solution).

Classiquement, les systèmes sous-contraints sont considérés comme des cas d'erreur que l'utilisateur doit corriger en ajoutant des contraintes. Nos travaux proposent une autre approche, qui est celle de chercher à résoudre de manière homogène tous les systèmes de contraintes géométriques qui ne sont pas sur-contraints.

Pour cela, nous proposons des algorithmes de paramétrisation, qui indiquent quels éléments du système doivent être fixés pour qu'il y ait un nombre fini de solutions, et des algorithmes de décomposition, qui permettent d'identifier les sous-systèmes bien contraints.

Ces outils ouvrent la voie à des logiciels de modélisation par contraintes accessibles à des utilisateurs non-experts : ils permettent des retours visuels intuitifs sur le niveau de constriction du système. Comme nos algorithmes sont incrémentaux, ils permettent une approche par essai/erreur où l'utilisateur corrige l'esquisse au fur et à mesure de la résolution.

Abstract

Solving geometric constraint systems (GCS) aims at yielding figures which respect geometric requirements given by the user under the form of a technical sketch. The GCS given by the user can be well-constrained (it describes a non-zero finite number of figures), under-constrained (an infinity of figures) or over-constrained (no solutions at all).

Classically, under-constrained systems are considered as errors to be corrected by the user. Our work proposes another approach, *i.e.* try to homogeneously solve all non-over-constrained systems.

For that, we propose parameterization algorithms : they indicate which elements of the system need to be anchored for there to be a finite number of solutions ; and decomposition algorithms, which allow to identify well-constrained subsystems.

These tools open the way to constraint-based modelers which are accessible to non-expert users : they give intuitive visual feedback about the constrainedness level of the system. Since our algorithms are all incremental, they allow a trial and error approach : the user corrects the sketch as the resolution goes.

R

Je remercie Bertrand Neveu, Dominique Michelucci et Robert Joan-Arinyo d'avoir accepté de faire partie de mon jury de soutenance. Je remercie tout particulièrement Bertrand Neveu et Robert Joan-Arinyo d'avoir accepté d'en être les rapporteurs. Tous les trois m'ont fait part de leurs précieux conseils pour l'amélioration du présent manuscrit et je leur en suis reconnaissant.

Je remercie les personnalités politiques de l'UdS qui ont été tantôt des collaborateurs, tantôt des adversaires de débat mais qui tous, à un moment ou à un autre, m'ont fait confiance et m'ont permis de faire avancer les choses dans le bon sens (du moins je l'espère) : Alain Beretz, Alice Couégnas, Béatrice Meier-Muller, Catherine Florentz, Éric Westhof, Hugues Dreyssé, Jean Déroche, Joëlle Hubé, Marie-Ange Joerg, Michaël Gutnic, Patrick Llerena, Richard Kleinschmager, Stéphane Heitz, Yves Strickler et tous ceux que j'oublie plus ou moins consciemment.

Je remercie, pour leurs blagues, leur soutien et pour leurs railleries, l'ensemble des collègues du LSIIT qui m'ont côtoyé. J'ai une pensée particulière pour Gael Vaubaisse, Claude Referria, Luigi Gumelleta, Eugène van Jaxivadd, Bruno Jauxline, Anthony Khevene, Étienne Rillourne, Salman Camun, Émile Vanut, Romaric Funer, Isaac Gondumal, Stanislas Copa, Xavier Louvigene, Karim « MC » Eperrere, Garcia Kreperins, Yvan Hylierst et — last but not least — Sam Jodhunt, collègues de travail, compagnons de délires et pour certains amis, avec lesquels tant de projets fantastiques et révolutionnaires ont vécu l'espace de quelques minutes dans le

« bureau de la fête ». Merci également à Marie-Claire « Patron » Hantsch et à Pierre Tellier pour avoir fait de mon second lieu de travail un endroit où il fait bon vivre.

Parce que sans eux j'aurais pu réaliser ces travaux plus rapidement, je me dois de remercier tous les jeunes chercheurs de l'Addal et de la CJC qui, en plus d'être des amis fidèles, ont été tour à tour moteurs et soutiens actifs dans les différents chantiers qui ont rythmé ma vie para-doctorale et tout spécialement Benoît Maugis, Caroline Pénicaud, Cécile Frolet, Julien Henninger, Maïwenn Corrignan, Sylvain Collonge, Victoire Goust et Vincent Reillon. J'ai une pensée toute particulière pour Delphine Quénet et Elise Fouquerel, qui ont supporté plus que tous les autres mon humour lorsque nous partagions le thé, pour Olivier Habrylo, sans qui je me serais noyé, pour Emmanuelle Ebel, qui a partagé mes alternances de déprime et de travail acharné et enfin pour Geoffroy Belhenniche et Morgane Gorria, qui m'ont impressionné et inspiré à tout point de vue.

Je remercie également tous ceux qui par leur amitié fidèle et nos expériences communes ont participé à forger ma personnalité et, en particulier, les lalquiens et associés : Djam, Xurk et son Srilankon, Toupitov, Jojo, Mikougrill, Anne-Lise, Superpopo, Dwight, Mikouboy. Ça fait nunuche mais je le dis quand-même : vous aurez beau partir aussi loin que vous voudrez, vous aurez toujours une place de choix dans mon cœur. D'autres amis tiennent une place particulière parce que j'ai passé avec eux des moments formidables dans des univers incroyables et parce que sans eux, j'aurais probablement craqué avant la fin de mes travaux de recherche : mes compagnons de scène. Merci donc à toute la troupe de « J'Te Paye Impro » !

Il y a tant d'autres gens que j'aurais voulu remercier ici : mes compagnons de la Croix-Rouge, les danseurs, mes autres amis qui ne rentrent pas dans les cases artificielles de ces quelques paragraphes, ceux qui y rentraient mais qui n'ont pas été cités. Qu'ils veuillent bien se reconnaître dans ces remerciements et accepter mes excuses pour la blessure à leur ego de n'apparaître pas nommément ici, mais ce chapitre est déjà bien trop long au vu des habitudes académiques.

Parce qu'ils ont plus que quiconque fait de moi qui je suis, je tiens à remercier ma famille, qui m'a inconditionnellement soutenu tout au long de ma courte vie et en particulier, ces dernières années, pendant les rares moments que je leur accordais. Cousins et cousines, neveux et nièces, oncles et tantes, feu grands-parents, beaux-parents, (beaux-)frères et (belles-)sœurs : qu'ils soient tous remerciés pour n'avoir jamais mis trop de pression et toujours assez d'encouragement. En particulier, je remercie mon infaillible, toujours présente et désespérément trop gentille génitrice et mon irresponsable, toujours en retard et désespérément trop flemmard frère.

Je pense qu'il faut avoir mené un projet de l'ampleur d'un doctorat et avoir connu le stress final de ceux qui, comme le veut la tradition, sont en retard, pour comprendre

à quel point je suis redevable à celle qui m'a aidé et soutenu, qui s'est pliée en quatre pour m'assurer les meilleures conditions de travail pour la conclusion du manuscrit. Parce que sans elle, la vie n'aurait probablement pas le même goût (et aussi parce qu'elle ne trouvera pas ce remerciement amusant) je remercie celle qui me soutient au moins autant qu'elle me prend du temps, ma compagne Marie.

Enfin, je tiens à remercier, du fond du cœur, l'inséparable duo des Pascaux : Pascal Schreck, mon directeur de thèse et maître à penser, pour m'avoir montré avec fermeté et sympathie que mes forteresses n'étaient que des châteaux de cartes mais aussi pour avoir su avec une impressionnante modestie être un modèle de chercheur ; Pascal Mathis, mon co-encadrant et chef en toutes choses, pour m'avoir proposé d'une part des pistes qui ne me plaisaient pas jusqu'à ce que je réalise qu'elles étaient la meilleure solution à mon problème et d'autre part des solutions excellentes dont il oubliait qu'il les avait lui-même rejetées deux mois plus tôt quand je les lui soumettais. Ils ont tous deux su encadrer mes travaux en me laissant la liberté nécessaire pour les mener comme je l'entendais, faisant de moi non pas un technicien de la recherche mais un chercheur, même si le temps passé sur mes travaux était un bien maigre remerciement pour leur investissement. Pour ces quatre fabuleuses années durant lesquelles vous m'avez fait confiance, merci !

À mon père : pas parfait, mais il s’en est fallu de peu...

Scientiam patior gaudiumque meum est

S

I

Il n'est pas de problème dont une absence de solution ne finisse par venir à bout
– Henri Queuille, homme politique français

1 Contexte

L'ordinateur est aujourd'hui indispensable dans tous les travaux de conception technique, de l'architecture à la mécanique. Les technologies de l'informatique graphique permettent ainsi de s'immerger dans un environnement en relief, de numériser des objets planaires ou tridimensionnels ou encore de simuler des phénomènes lumineux complexes. En particulier, les outils de modélisation géométrique et de Conception Assistée par Ordinateur (CAO) permettent à l'utilisateur de concevoir sur l'ordinateur des objets en donnant sa topologie ou en modélisant interactivement la forme au moyen de courbes et de surfaces.

La manipulation directe des courbes et surfaces peut s'avérer délicate, en particulier dans les cas où l'utilisateur conçoit un objet dont il connaît des caractéristiques techniques mais ignore la forme. Afin de rendre la conception plus intuitive, les méthodes de modélisation déclarative sont donc apparues, permettant à l'utilisateur de décrire l'objet et d'attendre de l'ordinateur qu'il lui fournisse des représentations graphiques satisfaisant cette description. Ainsi, les problèmes de résolution

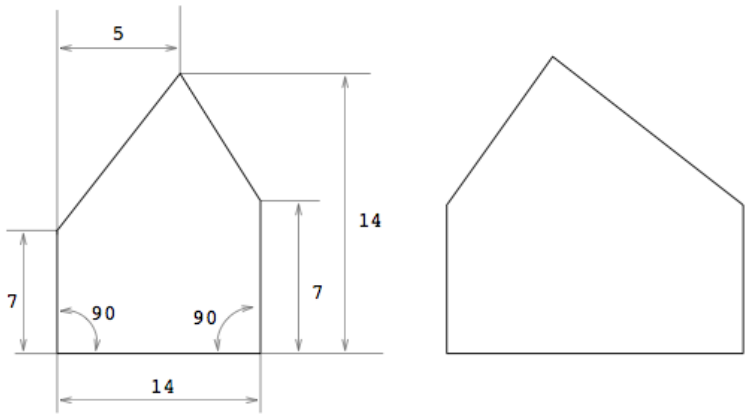


Figure 1 - Une esquisse cotée (gauche) et une figure à l'échelle (droite) sans cotation.

de Systèmes de Contraintes Géométriques (GCS) consistent à prendre en entrée un dessin esquissé, sur lequel l'utilisateur impose un dimensionnement fonctionnel par un système de cotes et à automatiquement produire une figure à l'échelle respectant la cotation.

La figure 1 illustre la finalité de la résolution de systèmes de contraintes géométriques. À gauche, sur l'esquisse cotée en deux dimensions, le concepteur a indiqué des contraintes d'angle entre droites et de distance entre points (les unités ne sont pas précisées). Certaines de ces contraintes ont une orientation implicite : la distance de 5, par exemple, indique la distance horizontale, c'est-à-dire la distance entre les projetés des points sur l'axe des abscisses. À droite est représentée une figure à l'échelle, sans cotation. Comme nous le voyons plus loin, il s'agit d'une solution particulière, mais il en existe d'autres.

Si l'idée des logiciels de résolution de systèmes de contraintes géométriques est apparue au début des années 60 avec les travaux de S [Sut63], les logiciels de dessin technique se sont, depuis, considérablement enrichis en fonctionnalités. Pour autant, le principe général est resté globalement inchangé :

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

Dans le cadre de la CAO, les méthodes rencontrées dans la littérature peuvent être regroupées en deux catégories :

– les méthodes numériques, qui traduisent le GCS en un système d'équations et travaillent sur ces équations, soit au moyen de manipulations algébriques soit au moyen de calculs itératifs ;

– les méthodes géométriques, qui opèrent en deux phases : une phase de planification et une phase de résolution numérique.

Bien évidemment, comme avec toute tentative de classification avec un faible nombre de classes, certaines méthodes sont à considérer comme hybrides, ayant des approches à la fois numériques et géométriques.

Dans tous les cas et quelle que soit la nature de l'approche, un point commun à tous les travaux menés récemment est l'utilisation de la décomposition. En effet, les méthodes de décomposition permettent à la fois de réduire la complexité et de faire collaborer des solveurs, éventuellement de nature différente. Elles consistent, comme toutes les méthodes « diviser pour régner » en informatique, à identifier des sous-systèmes résolubles et à assembler les solutions de ces sous-systèmes pour produire la solution du système global.

En outre, un autre point commun à (presque) toutes les méthodes de la littérature est de s'intéresser aux objets rigides et de considérer les GCS décrivant des objets non rigides comme des systèmes à corriger. Dans la littérature, ces objets sont assimilés aux systèmes décrivant une infinité de solutions, dits sous-contraints, auxquels on préfère les systèmes bien contraints, décrivant un nombre fini non nul de solutions (une dernière catégorie étant celle des systèmes sur-contraints, n'admettant aucune solution).

Il est connu que les systèmes rigides invariants par déplacement, admettant une infinité de solutions de par la possibilité d'appliquer à une solution une translation et/ou une rotation sans violer de contraintes, sont formellement sous-contraints. Dans la littérature, l'invariance des systèmes rigides par déplacement est souvent prise en compte en fixant deux points (en 2D) ou trois points (en 3D). Cependant, considérer explicitement les groupes de transformations permet d'être plus général dans la résolution de systèmes de contraintes géométriques, notamment en considérant d'autres groupes de transformations que les déplacements. Ainsi, les travaux décrits notamment dans [SS06, SM06] ont permis de formaliser la notion de *repère*, qui correspond aux éléments à fixer pour se ramener au cas bien contraint (dans le cas des déplacements planaires, un point et une direction).

2 Motivations

L'objectif principal de nos travaux est de rendre les logiciels de modélisation par contraintes accessibles à des utilisateurs non experts. Plusieurs défis sont à relever pour cela, au nombre desquels la mise au point d'outils spécifiques d'interaction 3D pour les contraintes géométriques [dRvdMB08, FMS04, FSSB05], l'utilisation de connaissances sémantiques et topologiques sur l'objet à résoudre [HJA98,

OSMA01, SOZA06] pour faciliter la pose des contraintes, la cinématique [Kra92, AKC96] ou encore le parcours de l'espace des solutions à la recherche de la solution voulue par l'utilisateur [EVSD00b, LSRGJA05, vdMB05, SAZK06].

Nous nous intéressons pour notre part à un autre défi : celui de l'incrémentalité. Nous pensons en effet que le processus de résolution classique mentionné plus haut (étapes 1 à 3) n'est pas adapté à des utilisateurs non experts, car il part du principe que le dessin esquissé ne sera pas modifié. Lorsque l'utilisateur ajoute une contrainte, le dessin est considéré comme différent et la résolution démarre à nouveau de zéro, sans utiliser les connaissances acquises lors de la résolution (ou tentative de résolution) du système de contraintes géométriques pré-changement.

Nous pensons que le processus de modélisation par contraintes devrait être le suivant :

1. l'utilisateur dessine une esquisse initiale, éventuellement vide ;
2. l'utilisateur impose un dimensionnement avec des outils spécifiques ;
3. un solveur fournit les solutions du système de contraintes géométriques ;
4. tant que l'utilisateur n'est pas satisfait par les solutions
   - il/elle modifie l'esquisse en ajoutant ou retirant des contraintes et des éléments d'esquisse ;
   - le solveur modifie les solutions calculées précédemment pour construire les solutions de la nouvelle esquisse.

Bien entendu, pour réaliser cela, il est nécessaire d'être capable de résoudre les systèmes intermédiaires, de l'esquisse initiale jusqu'à l'esquisse finale. Sans cela, il est impossible à l'utilisateur non expert de savoir s'il doit modifier l'esquisse. Nous nous intéressons donc aux systèmes qui sont sous-contraints, même avec le point de vue de l'invariance par des groupes de transformations, c'est-à-dire décrivent une infinité de solutions. Il s'agit, intuitivement, des GCS décrivant des objets articulés. Nos travaux visent à traiter de manière homogène ces systèmes et les systèmes bien contraints.

Nous voulons enfin, pour rester dans la lignée des travaux conduits dans l'équipe antérieurement au début du projet de recherche, que nos algorithmes prennent en compte l'information de l'invariance de sous-systèmes par des groupes de transformations connus et que la modification de l'esquisse par le concepteur amène à la découverte des modifications des groupes d'invariance des différents sous-systèmes.

3 Démarche

Afin de répondre aux motivations évoquées plus haut, le but de nos travaux est d'offrir au concepteur des outils lui donnant une intuition de la déformabilité de l'objet qu'il a décrit. Les outils que nous proposons sont au nombre de deux :
– nous montrons à l'utilisateur quels sont les éléments du GCS qui se déplacent ensemble en effectuant une décomposition en ses sous-systèmes bien contraints *modulo* un groupe de transformations connu ;
– nous montrons à l'utilisateur comment s'articulent les éléments du GCS en lui fournissant un repère pour le GCS.

La démarche que nous avons adoptée consiste à rechercher une paramétrisation du système de contraintes géométriques, c'est-à-dire un ensemble d'objets géométriques à fixer pour se ramener à un nombre fini de solutions. Cette paramétrisation fournit une bonne intuition du niveau de constriction[1] du GCS et est facilement représentable graphiquement.

À partir de cette paramétrisation, nous calculons un plan de construction par blocs indiquant dans quel ordre les différentes entités géométriques et les différents sous-systèmes doivent être construits et comment leur position peut être calculée à partir des éléments construits antérieurement.

Enfin, des valeurs numériques sont fournies pour les différents constituants de la paramétrisation du GCS et le plan de construction est numériquement évalué avec ces valeurs pour fournir les solutions.

L'ensemble des algorithmes mis au point durant nos travaux sont incrémentaux : ils consistent en l'étude des modifications apportées par un changement dans le GCS aux structures de données considérées.

4 Contributions

Nous proposons un algorithme incrémental de calcul des entités géométriques à fixer pour se ramener à un nombre fini de solutions. Cet algorithme est basé sur un calcul de flux, en extension des travaux de L et M [LM96b]. Nous

[1] Le terme de « constriction » n'est théoriquement pas approprié puisqu'il évoque une compression ou un resserrement. Il faudrait normalement parler du niveau de contrainte. Toutefois, de par la polysémie du mot « contrainte », nous préférons donner un nouveau sens à « constriction », en nous rassurant par le fait que contrainte et constriction ont la même racine et avaient il y a bien des années la même signification. Nous ne faisons donc que réunir deux cousins qui s'étaient perdus de vue.

proposons un nouveau type de graphe de flux, spécifique à nos travaux, que nous appelons graphe de $\mathcal{R}$-flux. À chaque étape de la conception du système de contraintes géométriques, l'algorithme fournit une paramétrisation combinatoire sous la forme d'une valuation des arcs du graphe de $\mathcal{R}$-flux et nous montrons comment l'interpréter géométriquement. Nous proposons un algorithme de modification à la volée de la paramétrisation si elle ne donne pas à l'utilisateur un retour satisfaisant sur les libertés de mouvement de l'objet.

Notre algorithme de paramétrisation fonctionne sous l'hypothèse que le système de contraintes géométriques n'est pas redondant, c'est-à-dire qu'aucune contrainte n'est une conséquence de la satisfaction des autres contraintes. Pour assurer le respect de cette hypothèse, nous proposons une extension incrémentale de la méthode du témoin proposée par M et F [MF09] nous permettant de détecter efficacement une contrainte redondante au moment de son ajout.

Nous proposons, toujours en nous basant sur la méthode du témoin, une méthode d'identification des sous-systèmes bien contraints maximaux, c'est-à-dire n'étant pas sous-systèmes d'un système lui-même bien contraint. Nous proposons des pistes pour faire cela directement dans le graphe de $\mathcal{R}$-flux et montrons les limites de cette approche.

Nous déduisons de l'algorithme d'identification des sous-systèmes bien contraints maximaux un algorithme de décomposition plus général que les méthodes de la littérature, appelé $\mathcal{W}$-décomposition. Nous montrons quelles sont les propriétés de la $\mathcal{W}$-décomposition.

Nous proposons également des ajouts à une formalisation des systèmes de contraintes géométriques réalisée précédemment à nos travaux et en déduisons une démonstration des conditions de correction et de complétude des algorithmes de décomposition. Cette démonstration existait déjà dans le cas de systèmes rigides mais notre volonté de traitement homogène de tous les systèmes non sur-contraints imposait une généralisation de la preuve. Notre démonstration se base sur la notion de bord d'un sous-système, c'est-à-dire intuitivement l'ensemble des informations que l'on peut déduire dans ce sous-système et qui concernent des entités géométriques qu'il partage avec le reste du système. Nous proposons un algorithme de calcul du bord assurant que le bord ne soit pas sur-contraint.

5 Organisation

Le présent mémoire se décompose en trois parties.

La partie I traite du contexte de nos travaux en présentant les systèmes de contraintes géométriques et les approches de leur résolution. Elle est composée de trois chapitres. Le chapitre 1 fournit une formalisation des systèmes de contraintes géométriques, de leurs opérations et de leur résolution. Dans le chapitre 2 nous effectuons un état de l'art des méthodes de résolution de systèmes de contraintes géométriques et évoquons notamment les algorithmes de décomposition ainsi que les différentes approches de la sous-constriction. Le chapitre 3, complémentaire du chapitre 1, exprime et prouve une série de théorèmes dont le dernier permet de démontrer quelles sont les conditions de correction et de complétude des méthodes de décomposition, en commençant par formaliser le point de vue multi-groupe [SM06].

La partie II traite de la paramétrisation de GCS sous-contraints et des modalités de leur résolution. Elle est composée de trois chapitres. Le chapitre 4 détaille notre algorithme de paramétrisation combinatoire basé sur les graphes de $\mathcal{R}$-flux. Le chapitre 5 décrit comment interpréter géométriquement la paramétrisation combinatoire, comment déduire un plan de construction (éventuellement par blocs) à partir de cette paramétrisation et il explicite l'interprétation géométrique du flux obtenu. Le chapitre 6 traite du problème spécifique de la détection de la sur-constriction lors de l'ajout de contraintes ou lors de l'évaluation numérique de la paramétrisation : il montre que ces deux problèmes ne sont pas trivialement résolus en adaptant les méthodes existantes puis propose des extensions de la méthode du témoin pour les résoudre.

La partie III traite de la décomposition de systèmes de contraintes géométriques. Elle est composée de deux chapitres. Le chapitre 7 propose une extension de la méthode du témoin permettant d'identifier les sous-systèmes bien contraints maximaux et donne des pistes pour réaliser cette identification dans le graphe de $\mathcal{R}$-flux sans le recours au témoin. Il montre également comment modifier les algorithmes de paramétrisation pour tenir compte de la connaissance de la bonne constriction de certains sous-systèmes. Le chapitre 8 détaille le fonctionnement et l'analyse de l'algorithme de $\mathcal{W}$-décomposition.

Enfin, une conclusion fait la synthèse de nos contributions et apporte des perspectives.

P `

L `

´ ´

Cette partie traite de la résolution de systèmes de contraintes géométriques en général et des fondements de leur décomposition. Elle définit formellement ce que sont les systèmes de contraintes géométriques et en quoi consiste leur résolution. Elle fait un survol des méthodes de résolution de la littérature. Elle met en évidence les conditions pour que des méthodes de décomposition soient correctes et complètes.

Motivation

À de nombreux points de vue, la communauté des contraintes géométriques travaille sur des fondations dont la solidité est une conjecture, car elles sont issues de la topologie structurale et que leur généralisation n'est pas triviale.

Tout d'abord, les définitions même des systèmes de contraintes géométriques, de leurs opérations, de leurs modes de résolutions, si elles sont informellement les mêmes pour tout le monde, varient toutefois beaucoup d'un auteur à l'autre. Des concepts identiques portent des noms différents d'un article à l'autre, ce qui rend leur compréhension et leur comparaison délicate. L'univers géométrique sur lequel ces systèmes sont basés est là encore souvent implicite dans la littérature.

De plus, une approche quasi-systématique à l'heure actuelle dans les méthodes de résolution est la décomposition des systèmes de contraintes géométriques à résoudre, c'est-à-dire leur découpage en plusieurs petits systèmes, la résolution de ces sous-systèmes et l'assemblage des solutions. La correction et la complétude de cette approche, c'est-à-dire l'assurance que l'assemblage des solutions forme bien des solutions d'une part, l'assurance que l'on peut ainsi obtenir toutes les solutions d'autre part, restent pourtant à prouver.

Démarche

Afin de nous attaquer aux éléments de problématique cités ci-dessus, nous avons participé à des travaux, initiés par Pascal S et Pascal M et visant à formaliser les systèmes de contraintes géométriques et leurs opérations. Ils ont ainsi formalisé en quoi consiste un univers géométrique, comment se définit syntaxiquement un système de contraintes géométriques à partir d'un univers géométrique, quelles sont les opérations permises. Ils ont formalisé le pendant sémantique des systèmes de contraintes géométriques, à savoir les figures solutions et, là encore, quelles sont les opérations permises sur des figures. Ils ont étendu ces définitions à une approche prenant en compte des groupes de transformation, formalisant ainsi le

point de vue multi-groupe [SM06].

Dans le cadre de ce projet de recherche, nous avons participé à l'expression des conditions qui permettent d'assurer la correction et la complétude des méthodes de décomposition et nous avons démontré qu'elles étaient nécessaires et suffisantes dans le cas général. Nous avons ainsi montré que l'on ne peut retirer un sous-système sans modifier les solutions du système résultant que si le système retiré est remplacé par l'ensemble des informations calculables dans ce système concernant les entités géométrique qu'il partage avec le reste du système. Dans le cas où cet ensemble contient des redondances, une base de cet ensemble suffit.

Le chapitre 1 fournit une formalisation des systèmes de contraintes géométriques, de leurs opérations et de leur résolution.

Dans le chapitre 2 nous effectuons un état de l'art des méthodes de résolution de systèmes de contraintes géométriques et évoquons notamment les algorithmes de décomposition ainsi que les différentes approches de la sous-constriction.

Le chapitre 3, complémentaire au chapitre 1, exprime et prouve une série de théorèmes dont le dernier permet de démontrer quelles sont les conditions de correction et de complétude des méthodes de décomposition.

C 1

T ` ´ ´

Sommaire

> *Il y a bien moins de difficultés à résoudre un problème qu'à le poser*
> – Comte Joseph de Maistre, philosophe français

Dans ce chapitre, nous formalisons les différentes notions utilisées dans le domaine de la résolution de Systèmes de Contraintes Géométriques (GCS). Il est important de noter, en particulier pour la bonne compréhension du chapitre 2, qui passe en revue les travaux classiques du domaine, que la formalisation présentée dans ce chapitre n'est pas un standard reconnu par l'ensemble de la communauté des contraintes géométriques. La résolution de contraintes géométriques est une science jeune, disposant de ce fait de terminologies différentes pour des notions équivalentes. La formalisation que nous donnons ici est issue de travaux de Pascal M et Pascal S [Sch02] et a été finalisée durant nos travaux.

Rappelons que l'objectif de la résolution de systèmes de contraintes géométriques est, à partir de la description technique d'un objet, le calcul de dessins correspondant à cette description. Un énoncé consiste en un ensemble de propriétés que le dessin doit satisfaire. Une façon habituelle d'exprimer un énoncé en Conception Assistée par Ordinateur (CAO) réside dans la donnée d'une esquisse cotée, c'est-à-dire un dessin approximatif de l'objet que l'on décrit, sur lequel sont dessinés – avec des traits différents du reste du dessin [ISO04] – des représentations graphiques des contraintes. Plus formellement, un énoncé peut aussi être donné sous la forme d'un ensemble de termes en logique du premier ordre. À chaque terme est associée une signification géométrique. Ainsi, les énoncés ont deux aspects : la syntaxe et la sémantique. Les *systèmes de contraintes géométriques* expriment l'aspect syntaxique tandis que les *figures* géométriques capturent l'aspect sémantique.

Dans ce chapitre, nous proposons une formalisation des GCS. Après un rappel concernant les spécifications algébriques (section 1.1), nous commençons par expliquer ce qu'est un univers géométrique (section 1.2) puis détaillons les aspects syntaxiques (section 1.3) et sémantiques (section 1.4) de la résolution de systèmes de contraintes géométriques. Nous définissons ensuite formellement les différents degrés de constriction[1] d'un GCS (section 1.5) puis les notions de bord et de décomposition de GCS (section 1.6).

Les définitions données dans ce chapitre sont l'occasion de démontrer certaines propriétés des systèmes de contraintes géométriques et de leurs opérations. Nous évoquons toutefois un certain nombre de résultats sans en faire ici la démonstration. Il s'agit de théorèmes qu'il est important d'avoir en tête parce que certaines des définitions qui les suivent ne sont valables que de par ces théorèmes. Au risque de redites, leur démonstration n'est donnée qu'au chapitre 3. En effet, ils sont utiles à la démonstration des conditions de correction et de complétude des méthodes de décomposition de GCS.

1.1 Spécifications algébriques

La formalisation que nous présentons ici est faite à la manière des spécifications algébriques [EM85, Wir90, BKL+91] : des sortes spécifient les domaines considérés, l'appartenance des inconnues aux domaines en question est assuré par un typage des symboles. Nous rappelons ici quelques définitions des spécifications algébriques.

La notion de signature capture la syntaxe de l'univers que l'on décrit.

Définition 1. Signature hétérogène : *Une signature hétérogène est un triplet* $\Sigma =$

[1] *Cf.* note 1 en page 5

$\langle S, F, P\rangle$ *où*
– S *est un ensemble fini de symboles de type appelés* sortes *;*
– F *est un ensemble de symboles fonctionnels muni de deux fonctions :*
  • *l'arité,* $\mathtt{ar}$: $F \to S^*$
  • *la coarité,* $\mathtt{coar}$: $F \to S$
– P *est un ensemble de symboles prédicatifs muni d'une fonction nommée également arité et notée* $\mathtt{ar}$: $P \to S^*$.

Les indications de typage sont notées de manière usuelle par :
– $f : s_1 \dots s_n \to s$ pour dire que $\mathtt{ar}(f) = s_1 \dots s_n$ et $\mathtt{coar}(f) = s$, avec $f \in F$. Si $n = 0$, on a $\mathtt{ar}(f) = \varepsilon$ (on note $f :\to s$) et f est appelé une constante ;
– $p : s_1 \dots s_n \to$ pour dire que $\mathtt{ar}(p) = s_1 \dots s_n$, pour $p \in P$.
Une *variable typée* est un symbole associé à une sorte. On note $x : s$ pour indiquer que x est de sorte s, avec $s \in S$. On appelle *ensemble sorté* de variables une famille $X = (X_s)_{s\in S}$ de symboles de variables : chaque ensemble X_s contient des variables de la sorte s.

À la syntaxe d'une signature on fait correspondre une sémantique, qui explicite l'interprétation faite des sortes et des symboles fonctionnels et prédicatifs.

Définition 2. Σ-algèbre : *Soit* $\Sigma = \langle S, F, P\rangle$ *une signature hétérogène. Une* Σ*-algèbre* E *consiste en la donnée*
– *d'un ensemble* E_s *par symbole de sorte* $s \in S$ *;*
– *d'une fonction* $\tilde{f} : E_{s_1} \times \dots \times E_{s_n} \to E_s$ *par symbole fonctionnel* $f : s_1 \dots s_n \to s$ *;*
– *d'un sous-ensemble* $\tilde{p}$ *de* $E_{s_1} \times \dots \times E_{s_n}$ *par prédicat* $p : s_1 \dots s_n \to$.

Lorsque $(o_1, \dots, o_n) \in \tilde{p}$, on dit que $\tilde{p}(o_1, \dots, o_n)$ est vérifiée.

Le passage de Σ à E est appelée une interprétation ou un *modèle* de la signature. De même, l'ensemble E_s est appelé une interprétation de s, $\tilde{f}$ une interprétation de f et $\tilde{p}$ une interprétation de p. L'interprétation d'une variable est appelée *valuation*.

Définition 3. Valuation de variables : *Soient* $\Sigma = \langle S, F, P\rangle$ *une signature,* E *une* Σ*-algèbre et* $X = \bigcup_{s\in S} X_s$ *un ensemble de variables* Σ*-sorté. Une valuation de variables est une fonction* $v : \bigcup_{s\in S} X_s \to \bigcup_{s\in S} E_s$ *telle que si* $x : s$*, alors* $v(x) \in E_s$.

Pour simplifier les notations, on peut écrire $v : \bigcup_{s\in S} X_s \to E$.

1.2 Univers géométrique

Pour pouvoir décrire formellement des objets au moyen de GCS, il est nécessaire d'établir :

1. un langage formel : un moyen de décrire syntaxiquement les systèmes de contraintes géométriques ;
2. une sémantique : un domaine d'interprétation des variables, dans lequel sont exprimées les solutions.

Ces deux éléments, avec la relation d'interprétation de l'un à l'autre, forment un *univers géométrique*.

Définition 4. Univers géométrique : *Un univers géométrique est une paire* $(\Sigma, \mathcal{E})$*, où*
– Σ *est une signature hétérogène ;*
– $\mathcal{E}$ *est un modèle de* Σ *(une* Σ*-algèbre).*

Exemple 1. Signature et modèle 2D

Considérons la signature suivante :
sortes
angle, longueur, point, droite, cercle
prédicats

```
dist_pp : point point longueur     # distance entre deux points
angle_ll : droite droite angle     # angle entre deux droites
center : cercle point              # centre du cercle
tangency_cl : cercle droite        # droite tangente au cercle
inc_pl : point droite              # incidence point–droite
```

Un modèle de cette signature pourrait être le plan euclidien, $\mathbf{E_2}$, avec les interprétations classiques :
– le domaine support de la sorte *point* est $\mathbb{R}^2$;
– celui de la sorte *droite* est l'ensemble $[0, \pi[\times\mathbb{R}$, c'est-à-dire le couple (angle par rapport à l'axe des abscisses, distance signée à l'origine) ;
– celui de la sorte *cercle* est l'ensemble $\mathbb{R}^2 \times \mathbb{R}^{+*}$, c'est-à-dire le couple (coordonnées du centre, rayon non nul).

Par exemple, on associe le symbole prédicatif `tangency_cl` à l'ensemble

$$\{(xc, yc, r), (a, d) | dist\_pl(xc, yc, a, d) = r\}$$

où *dist_pl* est la distance d'un point à une droite.

Un tel cadre est souvent étendu au moyen de symboles prédicatifs liés aux contraintes usuelles rencontrées en CAO. Le modèle considéré est généralement l'espace euclidien $\mathbf{E_3}$.

Sauf mention contraire explicite, les exemples du reste du présent mémoire sont construits sur la signature de l'exemple 1 avec le plan euclidien comme modèle. Les résultats du présent chapitre et du chapitre 3 ne sont pas limités à cet univers géométrique.

1.3 Systèmes de contraintes géométriques

1.3.1 Systèmes et sous-systèmes

Un système de contraintes géométriques est une conjonction de termes relationnels construit sur une signature et un ensemble de variables sortées. Les variables logiques sont soit des inconnues soit des paramètres de l'énoncé géométrique. Les paramètres sont les éléments géométriques (métriques ou entités géométriques) dont les valeurs sont fournies par l'utilisateur. Les inconnues sont les éléments dont le solveur doit trouver la valeur.

Définition 5. Système de contraintes géométriques : *Soit* Σ *une signature multi-sorte. Un système de contraintes géométriques est un triplet* $\mathcal{S} = (C, X, A)$ *où*
– *C est un ensemble de termes prédicatifs construit sur* Σ *;*
– *X et A sont deux ensembles disjoints de variables dont les sortes sont dans* Σ *;*
– *les arguments apparaissant dans les termes de C sont des termes fonctionnels de* Σ *construits sur* $X \cup A$*.*

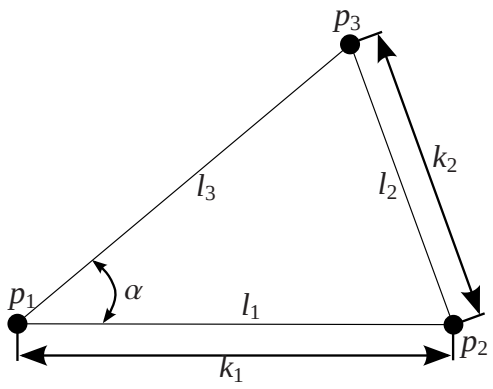
Exemple 2. Énoncé sous forme graphique et textuelle

La figure 1.1 représente une esquisse cotée et le système de contraintes géométriques qui la décrit[2]. Le GCS est composé de trois points p_1 à p_3 et trois droites l_1 à l_3, six contraintes d'incidence point-droite, deux contraintes de distance entre deux points et une contrainte d'angle entre deux droites.

Étant donnés un GCS $\mathcal{S} = (C, X, A)$ et un sous-ensemble C' de C on note
– $vars_{\mathcal{S}}(C')$ l'ensemble des variables logiques impliquées dans C' ;
– $parameters_{\mathcal{S}}(C')$ l'ensemble des paramètres apparaissant dans C' ;
– $unknowns_{\mathcal{S}}(C')$ l'ensemble des inconnues apparaissant dans C'.
On a donc $parameters_{\mathcal{S}}(C') = vars_{\mathcal{S}}(C') \cap A$ et $unknowns_{\mathcal{S}}(C') = vars_{\mathcal{S}}(C') \cap X$.

Symétriquement, on note $constraints_{\mathcal{S}}(i)$ l'ensemble $C' \subseteq C$ des contraintes dans lesquelles l'inconnue $i \in X$ apparaît. Pour un ensemble de variables $I \subseteq X$, on

[2] La syntaxe de description utilisée est proche de celle du GCML (*cf.* annexe A.1).



unknowns
p_1, p_2, p_3, l_1, l_2, l_3
parameters
k_1, k_2, α
constraints
on_pl(p_1, l_1) on_pl(p_2, l_1)
on_pl(p_2, l_2) on_pl(p_3, l_2)
on_pl(p_3, l_3) on_pl(p_1, l_3)
dist_pp(p_1, p_2, k_1)
dist_pp(p_2, p_3, k_2)
angle_ll(l_1, l_3, α)

Figure 1.1 - Énoncé sous forme graphique (à gauche) et textuelle (à droite). *Cf.* exemple 2.

note $span_{\mathcal{S}}(I)$ l'ensemble des contraintes concernant *uniquement* des variables de I, c'est-à-dire l'ensemble C' tel que $vars_{\mathcal{S}}(C') = I$. Sans avoir nécessairement égalité entre les deux, on a donc $span_{\mathcal{S}}(I) \subseteq \bigcup_{i \in I} constraints_{\mathcal{S}}(i)$.

Quand il n'y a pas de confusion possible quant à $\mathcal{S}$, on note ces cinq opérations de sélection en omettant de préciser $\mathcal{S}$ en indice, *e.g.* $vars(C)$.

Afin de formaliser les principes de la décomposition, nous définissons les sous-systèmes :

Définition 6. Sous-système : *Étant donné un système de contraintes géométriques $\mathcal{S} = (C, X, A)$, un sous-système $\mathcal{S}'$ de $\mathcal{S}$, noté $\mathcal{S}' \subset \mathcal{S}$, est un système de contraintes géométriques (C', X', A') tel que*
– $C' \subset C$;
– $X' = \text{unknowns}_{\mathcal{S}}(C')$;
– $A' = \text{parameters}_{\mathcal{S}}(C')$.

Par définition des opérations de sélection *unknowns* et *parameters*, si on a l'inclusion $(C', X', A') \subset (C, X, A)$ alors on a $X' \subseteq X$ et $A' \subseteq A$.

1.3.2 Opérations

L'addition et la soustraction de deux systèmes (avec pour condition qu'ils soient basés sur la même signature) correspondent à l'union et la différence des ensembles de contraintes. Dans le cas de l'addition de deux systèmes, si une variable est une

inconnue dans un système et un paramètre dans l'autre, elle est une inconnue dans l'union[3].

Définition 7. Addition de systèmes de contraintes géométriques : *Étant donnés deux GCS* $\mathcal{S}_1 = (C_1, X_1, A_1)$ *et* $\mathcal{S}_2 = (C_2, X_2, A_2)$*, le système* $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$ *est le système* (C, X, A) *défini par*
– $C = C_1 \cup C_2$;
– $X = X_1 \cup X_2$;
– $A = (A_1 \cup A_2) \backslash (X_1 \cup X_2)$.

$\mathcal{S}_1 + \mathcal{S}_2$ est également noté $\mathcal{S}_1 \cup \mathcal{S}_2$ et peut être appelé indifféremment addition ou union de $\mathcal{S}_1$ et de $\mathcal{S}_2$.

Définition 8. Soustraction de systèmes de contraintes géométriques : *Étant donnés deux GCS* $\mathcal{S} = (C, X, A)$ *et* $\mathcal{S}_1 = (C_1, X_1, A_1)$ *avec* $\mathcal{S}_1 \subset \mathcal{S}$*, le système* $\mathcal{S}_2 = \mathcal{S} - \mathcal{S}_1$ *est le système* (C_2, X_2, A_2) *défini par*
– $C_2 = C \backslash C_1$;
– $X_2 = X \cap \text{vars}(C_2)$;
– $A_2 = A \cap \text{vars}(C_2)$.

$\mathcal{S} - \mathcal{S}_1$ est également noté $\mathcal{S} \backslash \mathcal{S}_1$ et peut être appelé indifféremment soustraction de $\mathcal{S}_1$ de $\mathcal{S}$ ou bien $\mathcal{S}$ privé de $\mathcal{S}_1$.

L'intersection de GCS correspond quant à elle à l'intersection des trois ensembles composant les GCS.

Définition 9. Intersection de systèmes de contraintes géométriques : *Étant donnés deux GCS* $\mathcal{S}_1 = (C_1, X_1, A_1)$ *et* $\mathcal{S}_2 = (C_2, X_2, A_2)$*, le système* $\mathcal{S} = \mathcal{S}_1 \cap \mathcal{S}_2$ *est le système* (C, X, A) *défini par*
– $C = C_1 \cap C_2$;
– $X = X_1 \cap X_2$;
– $A = A_1 \cap A_2$.

Contrairement aux opérations classiques sur les ensembles, nous ne définissons pas de complément d'un GCS : on pourrait en effet considérer que le complément $\overline{\mathcal{S}'}$ d'un GCS $\mathcal{S}' \subset \mathcal{S}$ est $\mathcal{S} - \mathcal{S}'$. Toutefois, de par la définition de la soustraction, on aurait alors l'intersection $\overline{\mathcal{S}'} \cap \mathcal{S}'$ non vide, car les deux systèmes peuvent avoir des entités géométriques en commun.

[3] Ce choix est utile à la décomposition : lorsqu'on extrait un sous-système et qu'on le considère résolu, certaines des contraintes qu'il partage avec le reste du système peuvent être considérées comme des paramètres car provenant de la résolution du sous-système extrait. Pour autant, dans l'union des deux systèmes, les entités géométriques en question doivent rester des inconnues.

Théorème 1. Non vacuité de l'intersection d'un GCS et du reste de sa soustraction : *Soient $\mathcal{S} = (C, X, A)$, $\mathcal{S}_1 = (C_1, X_1, A_1)$ et $\mathcal{S}_2 = (C_2, X_2, A_2)$ trois GCS avec $\mathcal{S}_1 \subset \mathcal{S}$ et $\mathcal{S}_2 = \mathcal{S} - \mathcal{S}_1$. L'intersection de $\mathcal{S}_1$ et $\mathcal{S}_2$ est vide si et seulement si les variables de $\mathcal{S}_1$ n'apparaissent pas dans les contraintes de $\mathcal{S}_2$ et que les variables de $\mathcal{S}_2$ n'apparaissent pas dans les contraintes de $\mathcal{S}_1$.*

Démonstration:

Soit $\mathcal{S}' = (C', X', A')$ l'intersection de $\mathcal{S}_1$ et $\mathcal{S}_2$: $C' = C_1 \cap C_2$, $X' = X_1 \cap X_2$ et $A' = A_1 \cap A_2$.

Par définition de la soustraction de GCS, on a $C' = \emptyset$.

Toujours par définition de la soustraction, on a $X_1 = X \cap vars(C_1)$ et $X_2 = X \cap vars(C_2)$ et donc $X' = X \cap vars(C_1) \cap vars(C_2)$. Une inconnue peut apparaître dans plusieurs contraintes de $\mathcal{S}$. Une entité géométrique $x \in X$ apparaît dans X_1, par définition de la soustraction, ssi elle apparaît dans une contrainte de C_1.

Si une inconnue x apparaît dans une contrainte de $\mathcal{S}_1$ et dans une contrainte de $\mathcal{S}_2$, on a donc $x \in (X_1 \cap X_2)$ et donc $X' \neq \emptyset$.

En revanche, si les inconnues de $\mathcal{S}_1$ n'apparaissent pas dans $\mathcal{S}_2$ et inversement, on a $X \cap vars(C_1) \cap vars(C_2) = \emptyset$. X' est vide ssi les inconnues de $\mathcal{S}_1$ n'apparaissent pas dans $\mathcal{S}_2$ et que les inconnues de $\mathcal{S}_2$ n'apparaissent pas dans $\mathcal{S}_1$.

On tient le même raisonnement pour un paramètre. □

Nous définissons également la notion de sous-système engendré. Il s'agit, intuitivement, du sous-système obtenu en ne considérant qu'un sous-ensemble des inconnues et les contraintes les concernant.

Définition 10. Sous-système engendré : *Soit un GCS $\mathcal{S} = (C, X, A)$. Pour un sous-ensemble X' de X, le sous-système de $\mathcal{S}$ engendré par X' est le GCS $\mathcal{S}' = (C', X', A')$ avec*
– $C' = \text{span}(X')$;
– $A' = \text{parameters}(C')$.

La définition de système engendré peut être étendue à un ensemble X' qui n'est pas inclus dans X en considérant le système engendré par $X' \cap X$.

1.4 Solutions d'un système de contraintes géométriques

1.4.1 Figures

La notion de figure donne une sémantique aux systèmes de contraintes géométriques. Intuitivement, une figure est la donnée des coordonnées des différentes entités géométriques qui composent un GCS. Formellement, étant donné un univers $(\Sigma, \mathcal{E})$ et un système de contraintes géométrique, une figure est une fonction de Σ vers $\mathcal{E}$ définie pour toutes les variables du système.

Définition 11. Figure : *Étant donnés un univers* $(\Sigma, \mathcal{E})$ *et un ensemble de variables* X *dont les sortes sont dans* Σ*, une figure est une valuation* $f : X \to \mathcal{E}$.

Avec cette définition, une figure est vue comme un vecteur de couples (u_i, x_i) où
– $u_i \in X$ est une inconnue du GCS ;
– x_i est l'ensemble des coordonnées de l'interprétation de x_i.
Une figure paramétrée est définie pour une valuation des paramètres du GCS.

Définition 12. Figure paramétrée : *Étant donnés un univers* $(\Sigma, \mathcal{E})$*, une figure* $\rho : A \to \mathcal{E}$ *et un ensemble de variables* X *disjoint de* A *et dont les sortes sont dans* Σ*, une figure paramétrée est une fonction* $f : \rho \to (X \to \mathcal{E})$*. On note* $f_r ho : X \to \mathcal{E}$ *la figure* $f(\rho)$.

Une figure correspondant à un système de contraintes géométriques n'est pas nécessairement une solution du système. L'esquisse, par exemple, est une représentation graphique dans laquelle toutes les entités géométriques sont représentées : il s'agit donc d'une figure du système.

Définition 13. Figure solution : *Étant donnés un univers* $(\Sigma, \mathcal{E})$*, un système de contraintes géométriques* $\mathcal{S} = (C, X, A)$ *construit sur* Σ *et une valuation* $\rho : A \to \mathcal{E}$*, une figure* $f_\rho : X \to \mathcal{E}$ *est une solution de* $\mathcal{S}$ *si toutes les contraintes de* C *sont satisfaites lorsqu'elles sont interprétées dans* $\mathcal{E}$ *en prenant* $\rho(a)$ *comme interprétation de tout* $a \in A$ *et* $f_\rho(x)$ *comme interprétation de tout* $x \in X$.

Sauf mention contraire explicite, nous effectuons un abus de langage en appelant *figure* une *figure solution* dans la mesure où nous ne traitons que rarement de figures qui ne sont pas solution du GCS qu'elles représentent.

L'ensemble des figures solutions de $\mathcal{S}$ étant donné une valuation ρ des paramètres est noté $\mathcal{F}_\rho(\mathcal{S})$. Dans la suite du présent mémoire, pour la clarté des notations, si les

valeurs des paramètres ne sont pas importantes, il est noté $\mathcal{F}(\mathcal{S})$ voire simplement $\mathcal{F}$ si aucune confusion n'est possible. De même, on peut noter f un élément de $\mathcal{F}$.

Dans la suite du présent mémoire, parmi toutes les valeurs possibles des paramètres, nous ne considérons, sauf mention contraire, que celles qui maximisent la dimension de la variété sous-jacente, c'est-à-dire que les cas *dégénérés* où deux entités géométriques non explicitement égales sont confondues (*e.g.* métrique d'une distance égale à 0) ne sont pas considérés.

La restriction d'une fonction (solution ou non) à une partie de son domaine permet de définir la notion de *sous-figure*.

Définition 14. Restriction d'une figure : *Soient deux figures $f : X \to \mathcal{E}$ et $f' : X' \to \mathcal{E}$ avec $X' \subset X$ et telles que pour tout $x \in X'$, $f(x) = f'(x)$. f' est la restriction de f à X', notée $f' = f|_{X'}$.*

f' est aussi appelée *sous-figure* de f

La notion de restriction d'une figure d'un GCS (C, X, A) peut-être étendue à des ensembles X' qui ne sont pas inclus dans X en considérant que $f|_{X'} = f|_{X' \cap X}$. Bien sûr, $f|_X = f$.

En outre, un ensemble de figures d'un système peut être restreint à un sous-ensemble des inconnues. Il s'agit de l'ensemble contenant les sous-figures restreintes à ce sous-ensemble des inconnues.

Définition 15. Restriction d'un ensemble de figures : *Étant donné un GCS $\mathcal{S} = (C, X, A)$, la restriction de $\mathcal{F}(\mathcal{S})$ à un sous-ensemble X' de X, notée $\mathcal{F}(\mathcal{S})|_{X'}$ est l'ensemble $\{f|_{X'} \mid f \in \mathcal{F}(\mathcal{S})\}$.*

Il est important de noter que la restriction d'un ensemble de figures solutions à un sous-ensemble des inconnues n'est pas le pendant dans $\mathcal{E}$ du système engendré par ce sous-ensemble. Autrement dit[4], pour un système $\mathcal{S} = (C, X, A)$ et son sous-système $\mathcal{S}' = (C', X', A')$ engendré par X' (d'où $X' \subset X$), on a $\mathcal{F}(\mathcal{S})|_{X'} \subseteq \mathcal{F}(\mathcal{S}')$. En effet, les figures de $\mathcal{F}(\mathcal{S})|_{X'}$ respectent toutes les contraintes de C' – et font donc partie de $\mathcal{F}(\mathcal{S}')$ – mais respectent également les contraintes de $C \backslash C'$ qui concernent des entités géométriques de X'.

Exemple 3. Lien entre solutions d'un sous-système et sous-figures solutions

Les figures 1.2 et 1.3 illustrent le fait que la restriction d'un ensemble de figures à un sous-ensemble des inconnues est incluse dans

[4] *Cf.* théorème 4 p. 75

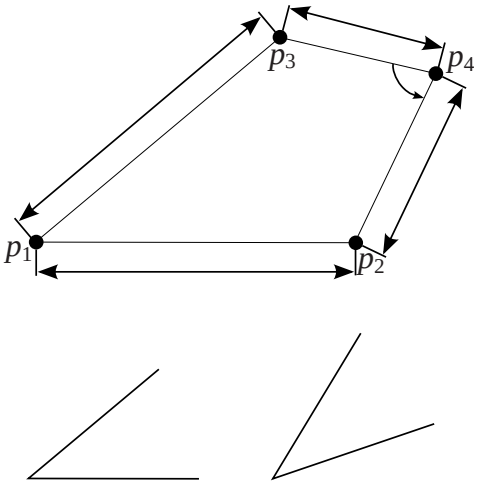


Figure 1.2 - GCS rigide et deux sous-figures restreintes à $\{p_1, p_2, p_3\}$

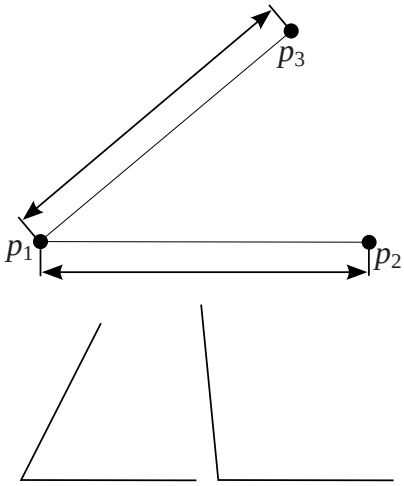


Figure 1.3 - Sous-système engendré par $\{p_1, p_2, p_3\}$ du GCS de la figure 1.2 et deux de ses solutions

l'ensemble des solutions du système engendré par ce sous-ensemble, mais que la réciproque n'est pas toujours vraie.

La figure 1.2 représente, en haut, un GCS constitué de 4 points p_1 à p_4, avec quatres contraintes de distance et une contrainte d'angle entre trois points. Le lecteur pourra se persuader que ce GCS est rigide : le triangle $p_3p_4p_2$ est contraint par deux distances et un angle et est donc aisément constructible. Le point p_1 est alors constructible par intersection de deux cercles, de centres p_3 et p_2 respectivement. Le bas de la figure 1.2 représente deux sous-figures restreintes à $\{p_1, p_2, p_3\}$. Comme le GCS est rigide, la distance entre p_2 et p_3 est la même dans toutes les sous-figures.

La figure 1.3 représente, en haut, un sous-système du GCS de la figure 1.2 : le système engendré par $\{p_1, p_2, p_3\}$. Ce GCS est composé des trois points p_1, p_2 et p_3 et de deux contraintes de distance. En bas de la figure sont représentées deux solutions de ce GCS. On notera qu'ici, la distance entre p_2 et p_3 est variable : en effet, rien dans le GCS ne contraint cette distance. Les solutions représentées sur la figure 1.3 ne sont ainsi pas des solutions du GCS de la figure 1.2.

On voit donc bien que l'inclusion est valide dans un sens (les solutions représentées à la figure 1.2 sont des solutions du GCS de la figure 1.3) mais pas dans l'autre.

Cette relation d'inclusion non symétrique indique que pour résoudre un GCS, il ne suffit pas d'assembler des figures de ses sous-systèmes.

1.4.2 Jointures

Nous définissons une opération de *jointure* qui explicite les conditions d'assemblage de deux sous-figures. Il s'agit d'une des opérations centrales en relation avec la décomposition en sous-systèmes. Cette opération ne peut être effectuée que sous certaines conditions de compatibilité.

Définition 16. Compatibilité de figures : *Soient deux figures* $f_1 : X_1 \to \mathcal{E}$ *et* $f_2 : X_2 \to \mathcal{E}$. *On pose* $X_e = X_1 \cap X_2$. f_1 *et* f_2 *sont dites compatibles si* $f_1|_{X_e} = f_2|_{X_e}$. *On note ceci par* $f_1 \equiv_{X_e} f_2$.

Autrement dit, deux figures sont compatibles si elles associent les mêmes coordonnées aux inconnues qu'elles partagent. Deux telles figures peuvent être jointes. Cette opération sémantique est le pendant de l'addition syntaxique de GCS.

Définition 17. Jointure de figures : *Soient* $f_1 : X_1 \to \mathcal{E}$ *et* $f_2 : X_2 \to \mathcal{E}$ *deux figures compatibles. La jointure de* f_1 *et* f_2, *notée* $f_1 \otimes f_2$ *est définie par*

$$\begin{array}{rcl} f_1 \otimes f_2 : & X_1 \cup X_2 \to & \mathcal{E} \\ & x \quad \to & \begin{cases} f_1(x) \text{ si } x \in X_1 \\ f_2(x) \text{ si } x \in X_2 \end{cases} \end{array}$$

Le théorème 6[5] montre que la restriction préserve l'opération de jointure pour deux figures, c'est-à-dire que la jointure de deux figures restreintes est égale à la restriction de la jointure des deux figures. Ce qui s'écrit :

> Si $f = f_1 \otimes f_2$, avec $f_1 : X_1 \to \mathcal{E}$ et $f_2 : X_2 \to \mathcal{E}$, alors pour tout sous-ensemble X' de $X_1 \cup X_2$, $f|_{X'} = f_1|_{X'} \otimes f_2|_{X'}$.

L'opération de jointure peut être étendue à des ensembles de figures, en opérant la jointure de tous les couples compatibles. S'il n'existe pas de couple de figures compatibles dans $\mathcal{F}_1 \times \mathcal{F}_2$, la jointure de $\mathcal{F}_1$ et $\mathcal{F}_2$ est vide.

Définition 18. Jointure d'ensemble de figures : *Soient* $\mathcal{F}_1$ *et* $\mathcal{F}_2$ *deux ensembles de figures, avec* $\mathcal{F}_i$ *un ensemble de figures* $f_i : X_i \to \mathcal{E}$. *On pose* $X_e = X_1 \cap X_2$. *La jointure de* $\mathcal{F}_1$ *et* $\mathcal{F}_2$ *est l'ensemble* $\mathcal{F}_1 \otimes \mathcal{F}_2 = \{f = f_1 \otimes f_2 \mid f_1 \in \mathcal{F}_1, f_2 \in \mathcal{F}_2, f_1 \equiv_{X_e} f_2\}$

L'opération de jointure de figures correspond à la recombinaison de sous-systèmes après un processus de décomposition. Le chapitre 3 montre la correction de cette démarche.

[5] *Cf.* p. 77

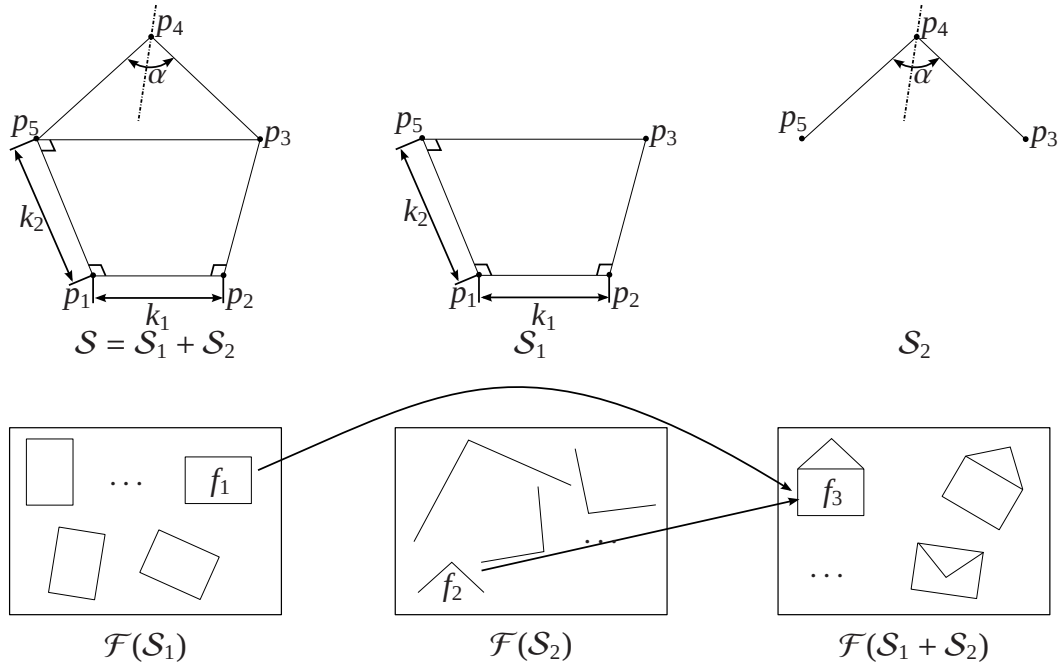


Figure 1.4 - Jointure de deux sous-figures

Le théorème 5[6] montre le lien étroit entre l'addition de GCS et la jointure des solutions de ces GCS : il exprime le fait que l'ensemble des solutions d'un GCS est la jointure des solutions de ses sous-systèmes. Formulé autrement :

> Soient $\mathcal{S}_1$ et $\mathcal{S}_2$ deux GCS construits sur la même signature. $\mathcal{F}(\mathcal{S}_1 + \mathcal{S}_2) = \mathcal{F}(\mathcal{S}_1) \otimes \mathcal{F}(\mathcal{S}_2)$.

Exemple 4. Jointure d'ensembles de figures

> Dans le plan euclidien, la figure 1.4 montre un GCS $\mathcal{S}$ coupé en deux parties $\mathcal{S}_1$ et $\mathcal{S}_2$. La ligne pointillée exprime la symétrie dans le triangle supérieur. L'ensemble infini $\mathcal{F}(\mathcal{S}_2)$ contient tous les triangles $p_3p_4p_5$ où $\widehat{p_3p_4p_5} = \alpha$ et où les segments p_3p_4 et p_4p_5 sont de même longueur. L'ensemble infini $\mathcal{F}(\mathcal{S}_1)$ contient tous les rectangles dont les côtés ont pour longueur k_1 et k_2.
>
> La figure f_3 est une solution de $\mathcal{S}$ et on a $f_3 = f_1 \otimes f_2$.

La restriction ne préserve pas forcément l'opération de jointure pour des ensembles de figures, à l'exception d'un cas spécifique qui sera utile pour la décomposition. Le théorème 7[7] montre les conditions de préservation de la jointure d'ensembles de figures par la restriction. Autrement dit :

[6] *Cf.* p. 76
[7] *Cf.* p. 77

Soient $\mathcal{S}_1 = (C_1, X_1, A_1)$ et $\mathcal{S}_2 = (C_2, X_2, A_2)$ deux GCS construits sur la même signature. On pose $X_e = X_1 \cap X_2$ et $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$. Pour tout ensemble $X' \subseteq (X_1 \cup X_2)$, $\mathcal{F}(\mathcal{S})|_{X'} = \mathcal{F}(\mathcal{S}_1)|_{X'} \otimes \mathcal{F}(\mathcal{S}_2)|_{X'}$ si et seulement si $X_e \subseteq X'$.

Exemple 5. Conditions de préservation de la jointure d'ensembles de figures par la restriction

Considérons à nouveau la figure 1.4, que nous avons déjà mentionnée dans l'exemple 4. On y voit aisément que la relation d'égalité n'est pas valable quand $X_e \not\subseteq X$.

En effet, le point p_1 appartient à $\mathcal{S}_1$ et le point p_4 à $\mathcal{S}_2$. $\mathcal{F}(\mathcal{S}_1)|_{\{p_1,p_4\}} = \mathcal{F}(\mathcal{S}_1)|_{\{p_1\}} = \mathbf{E_2}$ c'est-à-dire tout le plan euclidien. Il en va de même pour $\mathcal{F}(\mathcal{S}_2)|_{\{p_1,p_4\}}$. La jointure de $\mathcal{F}(\mathcal{S}_1)|_{\{p_1,p_4\}}$ et $\mathcal{F}(\mathcal{S}_2)|_{\{p_1,p_4\}}$ est donc l'ensemble des couples de points du plan, $\mathbf{E_2}^2$, tandis que quel que soit le couple de points dans $\mathcal{F}(\mathcal{S})|_{\{p_1,p_4\}}$, la distance entre les deux points est la même.

1.5 Niveaux de constriction

Dans l'ensemble des définitions que nous avons vues jusqu'ici, nous ne prêtons pas attention à la taille de l'ensemble des solutions d'un système de contraintes géométriques. Il est pourtant primordial, d'un point de vue applicatif, que le nombre de solutions soit tout d'abord non nul, mais aussi qu'il ne soit pas trop grand, sans quoi il sera difficile de les présenter à l'utilisateur. Sachant cela, la communauté a classé les GCS en trois catégories : sous-contraints, sur-contraints et bien contraints. Nous appelons *niveau de constriction* ou *degré de constriction* d'un GCS son appartenance à l'une de ces trois catégories.

Un GCS sur-contraint est un GCS qui, par exemple de par une contradiction des contraintes, ne possède aucune solution.

Définition 19. Sur-constriction : *Un système de contraintes géométriques* $\mathcal{S} = (C, X, A)$ *est sur-contraint pour une valuation des paramètres* ρ *si* $|\mathcal{F}_\rho(\mathcal{S})| = 0$.

Exemple 6. GCS sur-contraint

La figure 1.5 représente un GCS sur-contraint de par une contradiction des contraintes. En effet, les trois côtés de ce triangle sont contraints d'avoir la même longueur, ce qui interdit[8] l'angle droit.

La figure 1.6 représente un GCS sur-contraint de par le non respect de l'inégalité triangulaire. En effet, le segment $[p_1 p_2]$ a une longueur

[8] En géométrie euclidienne

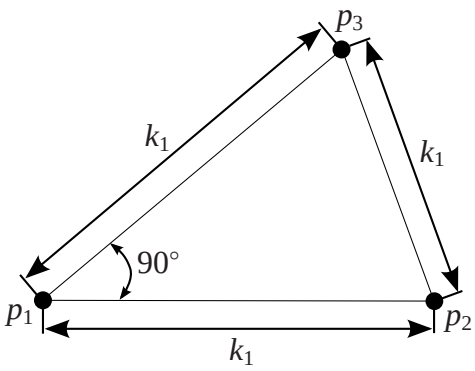


Figure 1.5 - Triangle sur-contraint du plan euclidien de par une contradiction des contraintes : l'angle vaut 60°(triangle équilatéral) et 90°(contrainte) à la fois

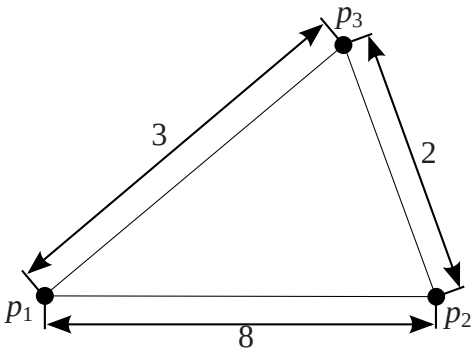


Figure 1.6 - Triangle sur-contraint du plan euclidien de par le non respect de l'inégalité triangulaire : la somme des longueurs p_1–p_3 et p_3–p_2 est inférieure à la longueur p_1–p_2.

de 8. Pour que le triangle existe, il est dès lors nécessaire que la somme des longueurs des deux autres segments soit supérieure ou égale à 8, sans quoi on ne peut construire le point p_3[8].

Un exemple célèbre (pour des raisons que nous voyons plus bas) de système sur-contraint est celui représenté à la figure 1.14, dit « double-banane ». Il s'agit d'un GCS en 3D composé de quatre tétraèdres, arrangés en deux paires de tétraèdres connectés par une face. Les deux sommets non connectés de chaque paire sont communs aux deux paires (p_1 et p_2). Les seules contraintes (non explicitement représentées sur la figure 1.14) sont les distances entre les sommets d'un même tétraèdre. Ce GCS est sur-contraint car chaque paire de tétraèdre est rigide : on peut donc calculer la valeur de la distance entre p_1 et p_2 à partir de chacune des deux « bananes » et lorsqu'elles ne sont pas cohérentes il n'existe aucune solution.

Un GCS sous-contraint est un GCS qui, par exemple de par le trop faible nombre de contraintes par rapport au nombre d'entités géométriques, possède une infinité de solutions.

Définition 20. Sous-constriction : *Un système de contraintes géométriques $\mathcal{S}$ est sous-contraint pour une valuation des paramètres ρ si $|\mathcal{F}_\rho(\mathcal{S})| = \infty$.*

Exemple 7. GCS sous-contraint

La figure 1.7 montre un GCS sous-contraint très simple, que nous appelons le « papillon ». Il consiste en deux sous-systèmes partageant un point commun, chaque sous-GCS correspondant à l'un des

triangles et étant constitué de trois distances. Comme aucun angle n'est contraint entre des côtés des deux triangles, on peut transformer une figure solution en appliquant une rotation de centre p_1 sans violer de contrainte, c'est-à-dire obtenir ainsi une autre solution[9]. Pour revenir à un nombre fini de solutions, il faut donc d'une part fixer un des deux triangles dans le plan, mais aussi donner une direction dans le second triangle pour empêcher la rotation autour de p_1. La figure 1.9 montre deux solutions du GCS pour illustrer cette liberté de mouvement.

Un autre exemple est donné à la figure 1.8. Il s'agit d'une *chaîne fermée*[10] constituée de 4 sous-systèmes engendrés chacun par une contrainte de distance. Chacun de ces sous-systèmes est classiquement appelé une barre rigide. Si l'une des barres est totalement fixée, les 3 autres sont toujours susceptibles de se mouvoir (sauf cas de valeurs de distance très particulières : l'une des valeurs est égale à la somme des trois autres). Par exemple, si le segment $[p_1p_2]$ de la figure 1.8 est fixé, le segment $[p_2p_3]$ est encore en libre rotation autour de p_2. La figure 1.10 montre deux solutions du GCS pour illustrer cette liberté de mouvement.

Nous avons déjà vu dans l'exemple 6 que la « double-banane » était un GCS sur-contraint. Toutefois, si la valeur de la distance entre p_1 et p_2 est la même dans chacune des deux paires de tétraèdres et qu'il n'y a donc pas sur-constriction, alors le système est sous-contraint, car chacune des deux paires de tétraèdres est en libre rotation autour de l'axe p_1p_2.

Enfin, un GCS bien contraint est un GCS qui n'est ni sur-contraint ni sous-contraint.

Définition 21. Bonne constriction : *Un système de contraintes géométriques $\mathcal{S}$ est bien contraint pour une valuation des paramètres ρ si $|\mathcal{F}_\rho(\mathcal{S})|$ est fini non nul.*

Notons que la bonne constriction est parfois définie [Jer02] comme la dénombrabilité des solutions d'un GCS, ce qui autorise certains systèmes ayant une infinité de solutions.

Dans la plupart des méthodes de résolution, on considère comme bien contraint un GCS définissant un objet rigide[11]. On se ramène donc à un nombre fini de solutions

[9] On dit que les sous-systèmes $p_1p_2p_3$ et $p_1p_4p_5$ sont en libre rotation autour de p_1

[10] Nous définissons formellement les chaînes fermées plus loin, à la définition 33, mais on peut ici le comprendre intuitivement avec l'idée d'un mécanisme fermé, créant un objet de genre non nul.

[11] C'est-à-dire, intuitivement, un objet pour lequel toutes les distances et tous les angles entre deux parties sont fixes

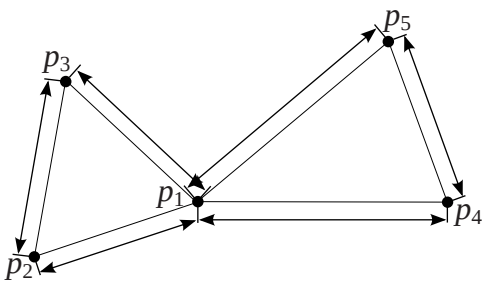


Figure 1.7 - Système sous-contraint : le « papillon », fait de deux triangles rigides en libre rotation autour de leur point commun p_1.

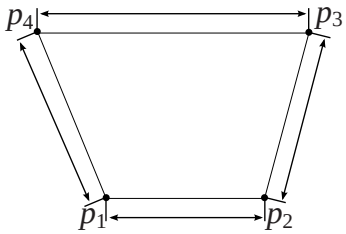


Figure 1.8 - Système sous-contraint : chaîne fermée constituée de 4 barres rigides. Deux solutions de ce GCS sont présentées à la figure 1.10

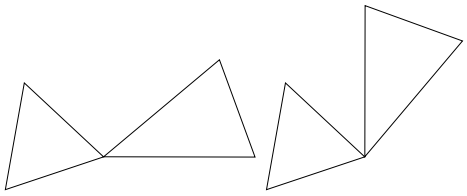


Figure 1.9 - Deux solutions du « papillon » représenté à la figure 1.7

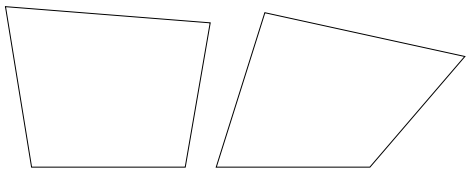


Figure 1.10 - Deux solutions du « 4 barres » représenté à la figure 1.8

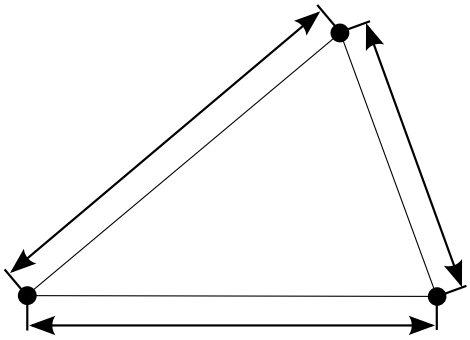


Figure 1.11 - Triangle rigide contraint par trois distances

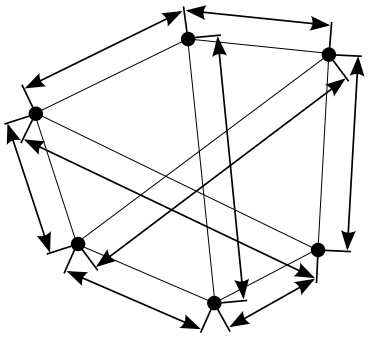


Figure 1.12 - Hexagone rigide indécomposable

en positionnant le GCS : on fixe un point et une droite (en 2D) ou un point et deux droites (en 3D) dans le système.

Exemple 8. GCS rigide

Nous donnons ici deux exemples de GCS décrivant des objets rigides. La figure 1.11 illustre un triangle contraint par trois distances. Ses solutions sont aisément constructibles [Car89], si on le fixe dans le plan en donnant les coordonnées d'un de ses points et la direction d'une droite passant par ce point.

La figure 1.12 représente un GCS rigide mais considéré indécomposable[12] fait de 6 points et 9 distances. Dans la suite, nous faisons référence à ce système en l'appelant $K_{3,3}$: le graphe construit en remplaçant chaque entité géométrique par un nœud et en insérant une arête entre deux nœuds s'il existe une contrainte entre les deux points correspondants[13] est le graphe biparti complet $K_{3,3}$.

Les trois définitions des niveaux de constriction que nous venons de donner sont sémantiques : elles déterminent trois classes de GCS basées sur la taille de l'ensemble des solutions. Parce qu'il est nécessaire de pouvoir déterminer le niveau de constriction durant la phase de résolution, c'est-à-dire avant de calculer l'ensemble des solutions, la littérature contient des définitions basées sur une analyse du GCS, qui déterminent des classes de GCS proches de celles recherchées mais pas tout à fait égales.

Nous détaillons plus avant ces différentes caractérisations de la constriction au chapitre 2. Nous en donnons ici une, découlant de la définition de la rigidité générique, introduite par L [Lam70] pour des graphes. Nous l'adaptons à notre forma-

[12] Au sens où les seuls sous-systèmes rigides qu'il contient sont engendrés par une contrainte.
[13] *Cf.* définition 32 p. 37 du graphe de contrainte d'un GCS

lisme. Elle requiert tout d'abord la définition des notions de *degrés de liberté* et de *degrés de restriction*.

Définition 22. Degrés de liberté d'une entité géométrique : *Soit* $(\Sigma, \mathcal{E})$ *un univers géométrique. Le nombre de degrés de liberté d'une entité géométrique* x *de sorte* $s \in \Sigma$*, noté* $\mathrm{ddl}(x)$*, est le nombre minimal de paramètres qu'il faut pour fixer un élément interprétant la sorte* s*.*

Exemple 9. Degrés de liberté d'une entité géométrique

En deux dimensions, le nombre minimum de paramètres pour fixer un point est de 2 (*e.g* abscisse et ordonnée), de même pour une droite (*e.g.* coefficient directeur et ordonnée à l'origine ou coordonnées du projeté de l'origine sur la droite). En trois dimensions, un point a 3 degrés de liberté et une droite en a 4.

Définition 23. Degrés de restriction d'une contrainte : *Soit* $(\Sigma, \mathcal{E})$ *un univers géométrique. Le nombre de degrés de restriction d'une contrainte* c*, noté* $\mathrm{ddr}(c)$ *est le nombre de degrés de liberté qu'elle retire à l'une des variables qu'elle contraint si toutes les autres sont fixées.*

Exemple 10. Degrés de restriction d'une contrainte

Une contrainte fixant la distance entre deux points (`dist_pp()` dans la signature de l'exemple 1) possède 1 degré de restriction, quelle que soit la dimension du modèle géométrique considéré :

- en 1D, un point possède 1 degré de liberté et lorsque l'on connaît un second point et sa distance à ce point, il est fixé ;
- en 2D, un point possède 2 degrés de liberté et lorsque l'on connaît un second point et sa distance à ce point, il se retrouve sur un cercle. Il lui reste alors un degré de liberté (positionnement sur le cercle) ;
- en 3D, un point possède 3 degrés de liberté et lorsque l'on connaît un second point et sa distance à ce point, il se retrouve sur une sphère. Il lui reste alors deux degrés de liberté (positionnement sur la sphère).

Le degré de restricition d'une contrainte paramétrée dépend de la valeur de son paramètre. Ainsi, l'exemple 10 mentionne un degré de restriction de 1 pour une contrainte de distance entre deux points. Toutefois, si la valeur associée à cette contrainte est nulle, la contrainte de distance devient une contrainte d'égalité de points, qui retire donc n degrés de liberté en dimension n.

Toutefois, pour toutes les autres valeurs de paramètres, le degré de restriction est 1. On dit alors que le paramètre 0 est un cas dégénéré et on considérera que le degré de restriction d'une contrainte de distance est 1.

Définition 24. Degrés de liberté et de restriction d'un GCS : *Soit $\mathcal{S} = (C, X, A)$ un GCS. Le nombre de degrés de liberté de $\mathcal{S}$ est la somme des degrés de liberté de ses entités géométriques. Le nombre de degrés de restriction de $\mathcal{S}$ est la somme des degrés de restriction de ses contraintes.*

$$\mathrm{ddl}(\mathcal{S}) = \sum_{x \in X} \mathrm{ddl}(x) \qquad \mathrm{ddr}(\mathcal{S}) = \sum_{c \in C} \mathrm{ddr}(c)$$

Exemple 11. Degrés de liberté d'un GCS

Si l'on revient à la figure 1.3 (p. 23), on a un GCS où la somme des degrés de liberté des entités est 6 (3 points en 2D, 2 degrés de liberté chacun) et la somme des degrés de restriction des contraintes est 2 (2 contraintes de distance). Il reste donc 4 degrés de liberté à fixer dans le GCS. En effet, en fixant le point p_1, une des coordonnées de p_2 et une des coordonnées de p_3 on a retiré toutes ses libertés au système.

À partir des définitions des degrés de liberté et de restriction, on peut définir les niveaux de rigidité structurels. Ils correspondent à la caractérisation de la rigidité de L [Lam70].

Définition 25. Niveau de rigidité structurel : *Soit $(\Sigma, \mathbf{E_2})$ un univers géométrique avec une interprétation en deux dimensions et $\mathcal{S} = (C, X, A)$ un GCS construit sur Σ. $\mathcal{S}$ est dit*

- *structurellement sur-rigide si $\exists X' \subseteq X$ tel que le système $\mathcal{S}'$ engendré par X' satisfait* $\mathrm{ddr}(\mathcal{S}') > \mathrm{ddl}(\mathcal{S}') - 3$ *;*
- *structurellement sous-rigide si* $\mathrm{ddr}(\mathcal{S}) < \mathrm{ddl}(\mathcal{S}) - 3$ *et qu'il n'est pas structurellement sur-rigide ;*
- *structurellement rigide s'il n'est ni structurellement sur-rigide ni structurellement sous-rigide.*

Autrement dit, $\mathcal{S}$ est structurellement bien rigide si $ddr(\mathcal{S}) = ddl(\mathcal{S}) - 3$ et que pour tout sous-système $\mathcal{S}' \subset \mathcal{S}$ on a $ddr(\mathcal{S}') \leq ddl(\mathcal{S}') - 3$. De nombreux solveurs 2D recherchent donc à obtenir un système à n entités et $2 \times n - 3$ contraintes ayant un degré de restriction (et n'ayant pas de sous-système à n' entités géométriques et plus de $2 \times n' - 3$ contraintes).

De par l'importance accordée aux systèmes rigides, de nombreux articles de la littérature parlent de systèmes structurellement bien contraints (respectivement structurellement sur-contraints et structurellement sous-contraints) pour désigner des systèmes structurellement rigides (respectivement structurellement sur-rigides et structurellement sous-rigides). Le retrait de 3 degrés de liberté dans la définition correspond aux trois degrés de liberté d'un objet rigide dans le plan : deux degrés en translation et un degré en rotation.

Il est important de noter que la définition de la bonne constriction structurelle ne permet pas de détecter les cas où des valeurs spécifiques des paramètres empêchent d'avoir des solutions (cas de la figure 1.6) ou produisent des cas dégénérés (*e.g.* contrainte de distance avec une métrique de 0). Elle permet de définir des systèmes qui sont rigides *dans le cas général*. Plus formellement, elle est valable uniquement dans le cas générique [GSS93, Gra02], c'est-à-dire quand les valeurs des paramètres n'amènent pas un cas dégénéré.

Définition 26. Généricité : *Soit $\mathcal{S} = (C, X, A)$ un GCS. On note Δ l'espace des valuations ρ des paramètres. $\mathcal{S}$ est dit génériquement bien contraint (resp. génériquement sur-contraint et génériquement sous-contraint) si la mesure du sous-espace de Δ tel que $\mathcal{S}$ n'est pas bien contraint (resp. sur-contraint et sous-contraint) est nulle par rapport à Δ.*

On étend bien entendu cette définition à n'importe quelle classe de GCS : un GCS est génériquement rigide, par exemple, si le sous-espace des valuations des paramètres tel qu'il n'est pas rigide est de dimension nulle.

Intuitivement, la généricité est une définition probabiliste : une propriété est génériquement vrai quand son contraire est de probabilité nulle.

Notons que dans les applications classiques au plan euclidien (resp. à l'espace euclidien), la définition de la généricité s'entend au sens du nombre de figures solutions dans $\mathbb{C}^2$ (resp. $\mathbb{C}^3$) : le GCS de la figure 1.11 est génériquement rigide car pour la quasi-totalité des valuations des paramètres, il existe des solutions, même si ces solutions sont dans les complexes quand les valuations des paramètres ne permettent pas de respecter l'inégalité triangulaire.

La définition 25 a d'autres limites que la restriction au cas générique : elle ne fonctionne que pour des systèmes ne contenant que des points et des contraintes de distance et a été abusivement généralisée à tous les systèmes 2D. Dans de nombreux systèmes avec des contraintes d'angle, la définition tient toujours, mais il existe des contre-exemples, comme celui de la figure 1.13 où un triangle est contraint par trois angles : structurellement, le système est bien contraint mais dans la pratique il est soit sur-contraint (à partir de deux angles, on peut calculer le troisième, il peut donc y avoir contradiction) soit sous-contraint (si les valeurs sont cohérentes, alors le système n'est pas rigide, car on peut appliquer une homothétie à une solution sans violer de contrainte). Elle doit donc être étendue en ajoutant que dans n'importe quel ensemble de n droites, il doit y avoir au maximum $n - 1$ angles.

Il a été suggéré dans un premier temps d'étendre la définition de la bonne constriction structurelle en 3D aux GCS contenant n entités géométriques et $3 \times n - 6$ contraintes et ne contenant pas de sous-système à n' entités géométriques ayant

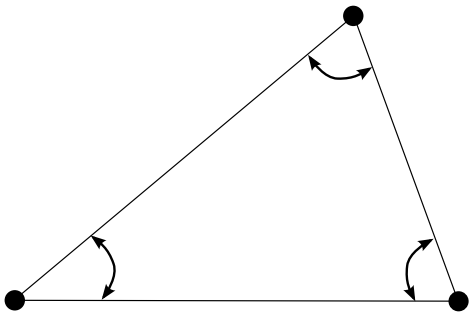


Figure 1.13 - Contre-exemple 2D de l'extension de la caractérisation de Laman à des GCS ne contenant pas uniquement des points et des distances : le système est soit sur-contraint soit sous-contraint, alors que $ddr(\mathcal{S}) = ddl(\mathcal{S}) - 3$.

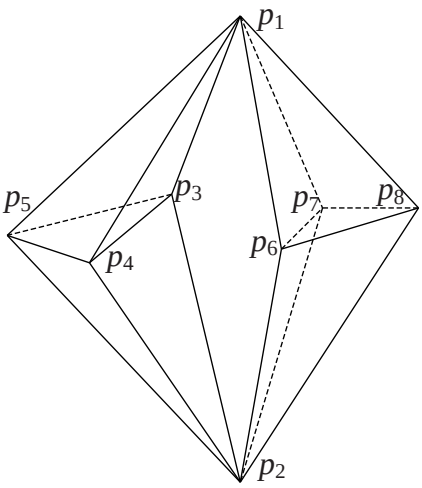


Figure 1.14 - La double-banane : chaque trait représente une contrainte de distance. Selon les métriques associées aux contraintes, le système peut n'avoir aucune solution ou une infinité de solutions.

plus de $3 \times n' - 6$ contraintes. Cette approche suivait la logique de la définition 25 : chaque entité géométrique a 3 degrés de liberté et un objet rigide a 6 degrés de liberté (3 en translation, 3 en rotation). Toutefois, des contre-exemples ont rapidement été trouvés, parmi lequel la « double-banane » déjà rapidement évoquée dans les exemples 6 et 7. Des versions étendues de cet exemple existent pour augmenter la connectivité du graphe [MS04]. D'autres configurations beaucoup plus simples utilisant des contraintes d'angle existent, comme par exemple les configurations d'O [MF06].

1.6 Bord et décomposition

Dans un système $\mathcal{S}$, un sous-système $\mathcal{S}'$ a des variables *internes* et des variables *de bord*. Les variables de bord sont celles qui apparaissent dans des contraintes où apparaissent également des variables de $\mathcal{S} - \mathcal{S}'$. Ainsi, par exemple, dans la figure 1.4, pour le sous-système $\mathcal{S}_1$, les variables de bord sont $\{p_3, p_5\}$ et les variables internes sont $\{p_1, p_2\}$.

Nous avons rencontré les variables de bord lorsque nous avons défini la compati-

bilité de figures[14]. Nous en avons aussi précisé les conditions d'existence au théorème 1. Le système qu'elles induisent joue un rôle prépondérant dans la décomposition en sous-systèmes car le *système de bord* d'un système $\mathcal{S}_1$, sous-système de $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$, contient toutes les informations nécessaires pour récupérer $\mathcal{F}(\mathcal{S})|_{vars(\mathcal{S}_2)}$.

Définition 27. Système de bord : *Soient $\mathcal{S}_1 = (C_1, X_1, A_1)$ et $\mathcal{S}_2 = (C_2, X_2, A_2)$ deux GCS construits sur la même signature. Le système de bord de $\mathcal{S}_1$ par rapport à $\mathcal{S}_2$ est le système $\mathcal{B}_{\mathcal{S}_2}(\mathcal{S}_1) = (Ce, Xe, Ae)$ tel que*
– $Xe = X_1 \cap X_2$;
– $Ae = A_1 \cap A_2$;
– $\mathcal{F}(\mathcal{B}_{\mathcal{S}_2}(\mathcal{S}_1)) = \mathcal{F}(\mathcal{S}_1)|_{(X_1 \cap X_2)}$.

Pour améliorer la clarté des notations, quand le système de bord de $\mathcal{S}_1$ est calculé par rapport à $\mathcal{S}_2$ et en l'absence d'ambiguïté, $\mathcal{B}_{\mathcal{S}_2}(\mathcal{S}_1)$ sera noté simplement $\mathcal{B}(\mathcal{S}_1)$. De même, on parlera du *bord* de $\mathcal{S}_1$.

Il est important de noter que la définition du système de bord est sémantique puisque fondée sur une égalité d'ensembles solutions. Il n'y a donc pas de garantie que la signature permette d'exprimer les contraintes correspondantes. Ceci est notamment discuté au chapitre 3, où nous évoquons les conditions de correction des méthodes de décomposition.

Le théorème 8[15] montre que retirer un sous-système $\mathcal{S}_1$ d'un GCS $\mathcal{S}$ ne change pas les solutions du système résultant si le bord de $\mathcal{S}_1$ est ajouté :

> Soit $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$ un GCS avec $\mathcal{S}_2 = (C_2, X_2, A_2)$. La restriction de $\mathcal{F}(\mathcal{S})$ aux variables de $\mathcal{S}_2$ est l'ensemble des solutions du système obtenu en ajoutant le bord de $\mathcal{S}_1$ à $\mathcal{S}_2$:
> $$\mathcal{F}(\mathcal{S})|_{X_2} = \mathcal{F}(\mathcal{S}_2 + \mathcal{B}(\mathcal{S}_1))$$

En d'autres termes, il prouve la validité des méthodes de décomposition ascendantes : si les solveurs des sous-systèmes sont corrects (*i.e.* ne fournissent que des figures qui satisfont les contraintes) alors l'assemblage des sous-figures fournira des solutions valides.

Exemple 12. Ajout du bord

> Dans l'exemple 4, X_2 est l'ensemble $\{p_3, p_4, p_5\}$ et $\mathcal{F}(\mathcal{S}_2)$ contient tous les triangles isocèles dont l'angle en le sommet principal vaut α. On peut observer que $\mathcal{F}(\mathcal{S})|_{X_2}$ est le sous-ensemble de $\mathcal{F}(\mathcal{S}_2)$ où la distance entre p_3 et p_5 est k_1. $\mathcal{F}(\mathcal{S}_2)$ comporte une infinité de triangles qui ne sont des restrictions d'aucune solution de $\mathcal{S}$: ceux pour lesquels la distance entre p_3 et p_5 ne vaut pas k_1.

[14] *Cf.* définition 16 p. 24
[15] *Cf.* p. 78

L'ensemble X_1 est $\{p_1, p_2, p_3, p_5\}$. Les variables du bord sont donc $X_e = X_1 \cap X_2 = \{p_3, p_5\}$ et $\mathcal{F}(\mathcal{B}(\mathcal{S}_1))$ contient tous les segments du plan euclidien dont la longueur est k_1. En considérant la signature de l'exemple 1, cet ensemble peut être syntaxiquement exprimé par le système $\mathcal{B}(\mathcal{S}_1) = (\{\texttt{dist\_pp}(p_3, p_5, k_1)\}, \{p_3, p_5\}, \{k_1\})$.
Ainsi, $\mathcal{S}_2 + \mathcal{B}(\mathcal{S}_1)$ restreint $\mathcal{S}_2$ aux triangles isocèles où le segment opposé à l'angle principal a pour longueur k_1. On a bien $\mathcal{F}(\mathcal{S}_2 + \mathcal{B}(\mathcal{S}_1)) = \mathcal{F}(\mathcal{S})|_{X_2}$.
Notons que $\mathcal{F}(\mathcal{B}(\mathcal{S}_2))$ est l'ensemble des segments du plan – *i.e.* $\mathbf{E_2}^2$ – puisque les inconnues du bord de $\mathcal{S}_2$ sont $\{p_3, p_5\}$ et que la distance entre ces deux points, dans $\mathcal{S}_2$, n'est contrainte ni directement (par l'énoncé) ni indirectement (par un théorème permettant de calculer cette distance à partir des contraintes de $\mathcal{S}_2$). Ici, la relation est $\mathcal{F}(\mathcal{S})|_{X_1} = \mathcal{F}(\mathcal{B}(\mathcal{S}_2) + \mathcal{S}_1)$ mais le système de bord $\mathcal{B}(\mathcal{S}_2)$ n'amène pas d'informations pertinentes puisque $\mathcal{F}(\mathcal{S})|_{X_1} = \mathcal{F}(\mathcal{S}_1)$. Cela signifie que retirer $\mathcal{S}_2$ de $\mathcal{S}$ n'a aucun impact sur l'espace des solutions valides des inconnues de X_1.

Le bord d'un GCS peut être génériquement sur-contraint : par exemple, si un système rigide partage trois points avec le reste du système, son bord contient notamment les trois distances et les trois angles du triangle. Nous définissons donc la notion de base d'un GCS, à partir de laquelle l'ensemble des informations du GCS peuvent être calculées.

Définition 28. Base d'un système de contraintes géométriques : *Soit $\mathcal{S}$ un système de contraintes géométriques, avec $\mathcal{S} = (C, X, A)$. Une base de $\mathcal{S}$ est un GCS $\mathcal{S}' = (C', X, A)$ minimal non génériquement sur-contraint tel que $\forall c \in C, (C' \cup \{c\}, X, A)$ est génériquement sur-contraint.*

Nous précisons maintenant ce que signifie décomposer un système de contraintes géométriques. La décomposition d'un GCS repose sur la stratégie « diviser pour régner ». Le GCS est coupé en sous-systèmes qu'un solveur peut gérer (*i.e.* soit résoudre directement, soit décomposer récursivement). L'objectif de la décomposition est la résolution du système global, la résolution directe d'un GCS de grande taille étant souvent impossible. La décomposition a aussi pour avantage de réduire la complexité du processus de résolution, même si dans bien des cas il ne s'agit que de division par une constante, la complexité de la résolution d'un sous-système étant la même que la complexité de résolution du système global. Pour des méthodes avec une complexité exponentielle, comme nous en voyons au chapitre 2, le gain est substantiel.

Définition 29. Décomposition d'un GCS : *Une décomposition d'un GCS $\mathcal{S} = (C, X, A)$ est une suite de GCS $\mathcal{S}_1, \ldots, \mathcal{S}_n$ telle que $\mathcal{F}(\mathcal{S}) = \mathcal{F}(\mathcal{S}_1 + \cdots + \mathcal{S}_n)|_{X \cup A}$ et $\mathcal{S} \subset (\mathcal{S}_1 + \cdots + \mathcal{S}_n)$.*

Autrement dit, une décomposition n'est pas une partition des contraintes de $\mathcal{S}$ dans la mesure où de nouvelles contraintes peuvent apparaître dans les systèmes $\mathcal{S}_i$. Toutefois, s'il y a des contraintes supplémentaires créées, elles doivent nécessairement être redondantes avec celles de $\mathcal{S}$, car dans le cas contraire, $\mathcal{F}(\mathcal{S}) \neq \mathcal{F}(\mathcal{S}_1 + \cdots + \mathcal{S}_n)$. De manière évidente, une décomposition doit être guidée par la sémantique, sa définition reposant sur le fait que l'interprétration d'une séquence $\mathcal{S}_1 + \cdots + \mathcal{S}_n$ est la même que celle de $\mathcal{S}$.

1.7 Méthodes de résolution

Nous formalisons ici diverses notions utiles à la résolution. Tout d'abord, définissons ce qu'est un solveur.

Définition 30. Solveur : *Soient* $(\Sigma, \mathcal{E})$ *un univers géométrique et* $\mathcal{S} = (C, X, A)$ *un système de contraintes géométriques non génériquement sur-contraint. Un solveur est une fonction qui à* $\mathcal{S}$ *associe un ensemble de figures de* $\mathcal{S}$.

Notons qu'un solveur, d'après cette définition, doit associer à chaque inconnue des coordonnées, mais que les figures qu'il fournit ne sont pas nécessairement des solutions. On dit d'un solveur qu'il est *correct* si toutes les figures qu'il fournit sont des solutions du GCS et qu'il est *complet* s'il fournit toutes les solutions.

Une approche répandue pour la conception d'un solveur est la donnée d'un plan de construction.

Définition 31. Plan de construction : *Étant donné un ensemble de solveurs* P, *un plan de construction est une décomposition de* $\mathcal{S}$ *en* $\mathcal{S}_1 \ldots \mathcal{S}_n$ *telle que :*
- $\mathcal{S}_1$ *est un sous-système de* $\mathcal{S}$ *;*
- $\forall i \in [1, \ldots, n]$, $\mathcal{S}_i$ *est résoluble par un solveur de* P *;*
- $\forall i \in [2, \ldots, n]$, *les entités géométriques communes à* $\mathcal{S}_i$ *et à la somme des* $\mathcal{S}_j$, $j < i$ *sont des paramètres de* $\mathcal{S}_i$.

On dit d'un plan de construction qu'il est strict si aucun système $\mathcal{S}_i$ n'a plus d'une inconnue. Sinon, on dit qu'il s'agit d'un plan de construction par blocs.

Une autre approche répandue (et non incompatible) est d'abstraire le GCS sous forme d'un graphe de contrainte.

Définition 32. Graphe de contrainte : *Le graphe de contrainte d'un GCS* $\mathcal{S} = (C, X, A)$ *est le graphe* (V, E) *défini par :*
- *il y a une bijection entre* V *et les entités géométriques de* $\mathcal{S}$ *;*

– *il y a une bijection entre E et les contraintes de $\mathcal{S}$;*
– *si $x \in X \cup A$ est une variable de c, l'arête correspondant à c est incidente au nœud correspondant à x.*

Notons que dans le cas de contraintes dont l'arité est supérieure à 2, il s'agit d'un hyper-graphe. Selon la topologie du graphe de contraintes, on peut dire que le GCS possède une chaîne fermée.

Définition 33. Chaîne fermée : *Une chaîne fermée d'un GCS est un sous-système dont le graphe de contraintes est un cycle.*

Par opposition, un système est une chaîne ouverte s'il ne contient pas de chaîne fermée. Dans le domaine de la cinématique, la définition de chaîne fermée est considérée dans le cadre d'un assemblage d'objets rigides. Le graphe de contraintes contient alors un nœud par objet rigide et les arêtes représentent les contraintes d'incidence. Pourtant, en modélisation par contraintes, chaque objet rigide serait représenté par un sous-système à plusieurs entités, formant plusieurs chaînes fermées. Dès lors, un objet que l'on modéliserait naturellement comme une chaîne fermée peut être considérée comme une chaîne ouverte si les cycles du graphe correspondent en réalité à des sous-systèmes rigides.

C 2

É ’

Sommaire

Chaque problème résolu en fait naître d’autres, en général plus difficiles
– Georges Pompidou, homme politique français

Dans ce chapitre, nous effectuons un aperçu des principales méthodes de résolution de GCS décrites dans la littérature. Nous commençons par décrire globalement quelles sont les grandes approches de la résolution de systèmes de contraintes géométriques et montrons quels sont actuellement les grands algorithmes de décomposition. Nous mettons ensuite l’accent sur une étude des articles traitant de la caractérisation du niveau de constriction d’un système et de la manipulation ou de la résolution de systèmes de contraintes géométriques sous-contraints.

2.1 Résolution de systèmes de contraintes géométriques

La section 2.1.1 offre un aperçu des grandes familles de méthodes de résolution et donne leurs forces et faiblesses respectives. La section 2.1.2 effectue un survol des méthodes de décomposition de GCS.

2.1.1 Grandes approches de la résolution

La résolution de systèmes de contraintes géométriques est une science jeune et, à ce titre, le recul peut manquer encore pour effectuer une classification satisfaisante des différentes approches de la résolution de GCS. Nous donnons ici une classification proche de la plupart de celles que l'on retrouve dans les articles récents.

On peut tout d'abord classer les méthodes de résolution en deux grandes familles, selon qu'elles traduisent les contraintes sous forme d'équations ou non. Les premières forment la famille des méthodes de résolution algébriques, les secondes la famille des méthodes de résolution géométriques.

Les méthodes algébriques sont générales et peuvent fonctionner dans tout univers géométrique dans lequel les résolutions d'équations sont formalisées et axiomatisées. Elles traduisent le système de contraintes géométriques sous la forme de systèmes d'équations et travaillent sur ces équations en oubliant le caractère géométrique de ce que représentent les inconnues. Les méthodes algébriques peuvent elles-mêmes se différencier en deux familles :
– les méthodes symboliques, qui manipulent formellement ces équations ;
– les méthodes numériques, qui réalisent des calculs itératifs approchant les solutions.

Les méthodes géométriques ne peuvent fonctionner que dans le cadre des raisonnements pris en compte par le programme et ne sont donc pas complètes. Elles se basent en général sur des raisonnements génériques. Leur objectif est la mise au point d'un plan de construction paramétré à évaluer numériquement. Les méthodes géométriques peuvent elles aussi se subdiviser en
– les méthodes à base de règles, qui opèrent des déductions géométriques à partir de règles de déduction explicites ;
– les méthodes à base de graphe, qui abstraient et approchent les connaissances géométriques sous la forme de règles combinatoires.

Nous donnons ici des détails sur les différentes familles de méthodes. Le lecteur désireux d'en savoir plus pourra se référer à quelques papiers réalisant un état de l'art de ces questions : [BFH+95, Doh95, HJA05, Hof06, JMS07, JA09]

2.1.1.1 Méthodes symboliques

Les méthodes symboliques consistent à manipuler directement les équations. Leur intérêt réside dans leur capacité à résoudre de manière exacte toute une classe de problèmes avec des paramètres. Ces méthodes sont proches des preuves en géométrie.

Le principe de ces méthodes est de trianguler le système d'équations puis de résoudre chacune des équations obtenues. Parmi les méthodes de triangularisation, on peut citer les bases de G ¨ [BCK88, Buc85, Kon90, Kon92] et la pseudo-division de R -W [Cho88, CG90, GC98a, GC98b, Wu86].

Leur inconvénient principal est leur coût algorithmique, au moins exponentiel, qui les rend inutilisables dans la pratique pour des logiciels de modélisation interactive.

2.1.1.2 Méthodes numériques

Les méthodes numériques, contrairement aux symboliques, ont l'avantage de leur efficacité : elles consistent à profiter de la puissance de calcul des ordinateurs pour effectuer des calculs itératifs approchant les solutions. Elles consistent à considérer le système de contraintes géométriques comme un problème d'optimisation.

La plus connue d'entre elles est la méthode de N -R [Ypm95], qui consiste à dériver la matrice correspondant au système de contraintes et à approcher la courbe de la fonction f à optimiser par sa dérivée f' (la matrice Jacobienne dans notre cas). À partir d'une valeur x_n à l'étape n (initialement celle de l'esquisse) on prend comme valeur à l'étape $n+1$ une racine de l'hyperplan tangent à la courbe en x_n : $x_{n+1} = x_n - f'(x_n)^{-1} f(x_n)$. On itère ainsi jusqu'à être suffisamment proche d'une solution. Puisqu'elle passe par une inversion de matrices, la méthode ne fonctionne que sur des matrices carrées et donc des systèmes bien contraints.

Cette méthode est bien sûr sensible aux optimums locaux et est bien connue pour manquer de stabilité. L'homotopie ou continuation [AG93] a été appliquée aux systèmes de contraintes géométriques par L et M [LM95, LM96a] puis utilisée par différents auteurs [Dur98, DH00]. C'est une méthode plus stable opérant un suivi de courbe paramétrée : elle consiste à faire l'interpolation linéaire pour t de 0 à 1 de la courbe définie par le système d'équations $H(X,t) = tF(X) + (1-t)(F(X) - F(x_0))$. Elle est un peu plus coûteuse mais a, contrairement à N -R , des bassins d'attraction intuitifs : si la figure initiale x_0 est proche d'une solution x, c'est vers x que l'homotopie convergera.

Des approches globales par algorithmes génétiques ont été proposées [GCG99, CC02, CLC06, VEG06] mais donnent des résultats peu satisfaisants en terme de ratio temps/précision. L'arithmétique par intervalles [Moo87] fournit de bons résultats mais son coût l'a longtemps rendue inutilisable dans la pratique. L'ajout de techniques de programmation par contraintes améliore leur efficacité [NTC10]. La littérature contient d'autres méthodes [Bor81, LLG81, Nel85].

Outre les problèmes de stabilité, les méthodes numériques souffrent aussi de leur incomplétude : N -R , par exemple, ne fournit qu'une seule solution. L'homotopie peut fournir toutes les solutions, mais son coût en est alors augmenté.

2.1.1.3 Méthodes à base de graphes

Les méthodes à base de graphe traduisent le système de contraintes géométriques sous la forme d'un graphe de contrainte. Elles fonctionnent par décompte de degrés de liberté et sont donc limitées au cas générique. La connaissance géométrique est compilée sous la forme de règles combinatoires.

Une approche classique est celle de la propagation de degrés de liberté [Sut63, LK98, FBMB90, VA92] : en fixant un segment, on regarde itérativement quelles entités sont considérées comme constructibles. La rétro-propagation consiste, à l'inverse, à retirer itérativement du graphe les nœuds qui ont autant de contraintes que de degrés de liberté, jusqu'à atteindre un système que l'on doit savoir résoudre, dans l'idéal limité à un segment (en 2D, deux segments ayant un point commun en 3D). Les méthodes de (rétro-)propagation échouent à résoudre des systèmes dont le graphe de contrainte et triconnexe, comme celui de la figure 8.2. A -A *et al.* [AAHMS99] proposent une méthode de constructions de clusters par propagation et d'assemblage de ces clusters, permettant d'augmenter la classe des systèmes résolubles, sans pour autant permettre de traiter n'importe quel système à graphe triconnexe.

Une autre façon de procéder, avec des résultats similaires, consiste à considérer le systèmes de contraintes géométriques sous la forme d'un graphe de flux et à chercher à maximiser le flot y circulant. Le théorème de K¨ -H indique que l'existence d'un couplage parfait est équivalente à la bonne constriction générique du GCS. L et M [LM96b] sont les premiers à utiliser cette approche sur des systèmes de contraintes géométriques, approche reprise et étendue par H - *et al.* [HLS98]. J *et al.* [JNT03] y ajoutent des connaissances géométriques explicites pour améliorer la correction.

Enfin, les méthodes de décomposition, descendantes ou ascendantes, sont nom-

breuses avec des graphes. Parmi les plus importantes, on peut citer les travaux d'O [Owe91], qui effectue une décomposition descendante en découpant des graphes en composantes triconnexes par la recherche de paires d'articulations. Il coupe le graphe au niveau d'une paire d'articulation, résout (récursivement) l'une des deux parties, et ajoute une contrainte virtuelle dans la seconde pour palier au retrait du reste du système[1]. F et H [FH97] obtiennent une méthode de décomposition similaire à celle d'O en formant des *clusters*. J -A *et al.* [JASRVMVP04] montrent que les systèmes résolubles par ces méthodes sont les mêmes et propose une formalisation unifiée permettant de démontrer la congruence de ces méthodes.

2.1.1.4 Méthodes à base de règles

Les méthodes à base de règles utilisent des systèmes experts intégrant des raisonnements géométriques explicites. À partir d'un ensemble de règles de déductions et du système de contraintes géométriques, elles consistent à chercher à appliquer des règles de déduction, le cas échéant en construisant de nouvelles entités géométriques, pour calculer un plan de construction enchaînant des constructions géométriques simples : intersections de droites, de cercles, construction d'une droite passant par deux points connus, *etc.*

La plupart des approches de résolution de GCS à base de règles sont limités au plan [Ald88, AMRV91, Brü93]. Les premières méthodes traitant de systèmes 3D sont en réalité limitées à des cas très spécifiques et ont une puissance de résolution faible [Brü87]. Certaines méthodes se confondent en outre avec des prouveurs géométriques [Brü87].

S [Sun87] propose une méthode par agglomération en construisant deux types de sous-systèmes : les CA-sets, où tous les angles sont connus ; les CD-sets, où toutes les distances sont connues. Un raisonnement géométrique lui permet d'ajouter des entités à des CD-sets ou des CA-sets et d'opérer des fusions de ces sous-systèmes. Le mode de fusion ou d'ajout lui permet également de déduire un plan de construction.

V *et al.* [VSR92] montrent que l'ajout de nouvelles informations permet d'augmenter la puissance de résolution, en résolvant notamment le système de la figure 8.2, que les méthodes d'alors (opérant principalement par propogation des degrés de liberté) ne pouvaient traiter.

[1] Les travaux d'O seront décrits plus précisément dans la section 3.3, où nous en prouvons la correction et la complétude.

J -A et S -R [JASR97] formalisent un solveur à base de règles sous la forme d'un système de réécriture et prouvent ainsi la convergence de leur méthode.

Peu de nouveautés ont été proposées dans la littérature dans la dernière décennie.

2.1.1.5 Méthodes hybrides

Bien sûr, certaines méthodes ne se classent totalement dans aucune des quatre familles évoquées plus haut, car elles sont hybrides.

Parmi celles-ci, on peut citer notamment la méthode de reparamétrisation qui consiste à résoudre un système que l'on ne sait pas décomposer en retirant une contrainte puis en opérant une rétro-propagation. Si la rétro-propagation échoue à aboutir à une chaîne ouverte, on continue à retirer des contraintes et à recommencer la rétro-propagation. Ensuite, on rajoute autant de contraintes qu'on en a préalablement retirées, pour transformer la chaîne ouverte obtenue en un système bien contraint. On fait ensuite numériquement varier les valeurs des paramètres des contraintes ajoutées pour rattraper les contraintes que l'on a retirées. Cette approche a été introduite par G *et al.* [GHY02, GHY04] dans le cas de la suppression d'une unique contrainte et étendue par F et S [FS06, FS07, FS08].

L et K [LK96, LK98] introduisent une méthode considérant un raisonnement sur les graphes pour accélérer le raisonnement d'un système expert. De même, J -A et S -R [JASR99] étendent les travaux de H et J -A [HJA97] combinant une méthode constructive à base de graphes et une méthode symbolique d'analyse d'équations. Ils la formalisent notamment sous la forme d'un système de réécriture.

L *et al.* [LKLK03] améliorent une méthode à base de graphes en effectuant des calculs numériques pour des étapes que le raisonnement combinatoire ne peut effectuer.

2.1.2 Décomposition de systèmes de contraintes géométriques

Une tendance générale des travaux de résolution, depuis la fin des années 80, est la décomposition du système. Qu'elle soit ascendante (on identifie un sous-sytème résoluble, on décompose récursivement le reste du système, puis on assemble les solutions) ou descendantes (on donne un critère de découpage du système en plusieurs sous-systèmes nous assurant de savoir assembler les solutions des sous-systèmes et

on itère récursivement sur ces sous-systèmes avec comme critère d'arrêt l'identification de sous-systèmes que l'on sait résoudre) elles ont toutes pour objectif à la fois de simplifier le raisonnement géométrique et de faire diminuer le coût. Elles ont également l'avantage d'augmenter la puissance de résolution, par rapport par exemple aux méthodes à base de graphe fixant un segment puis opérant une propagation des degrés de liberté. Cette puissance accrue tient du fait que les méthodes consistent – même si ce n'est pas formalisé ainsi – à remplacer un sous-système résolu par son bord.

Nous donnons ici des détails sur certains travaux particuliers de décomposition. Le lecteur intéressé par les méthodes de décomposition pourra se référer à [JTNM06]. En outre, certaines méthodes citées plus haut[2] fonctionnent par décomposition.

H *et al.* [HLS97, HLS98, HLS99] développent[3] une méthode de décomposition à base de flux qui produit des sous-systèmes *denses minimaux*. Ces sous-systèmes correspondent aux plus petits sous-systèmes respectant le critère de L -[4].

Z et G [ZG04, ZG06a] proposent une décomposition de GCS 3D basée sur la connexité du graphe et les techniques d'assemblage correspondantes. Ils arrivent notamment à détecter la sur-constriction dans certains cas, comme celui de la double-banane, où l'extension du critère de L indique la bonne constriction. Ils proposent également [ZG06b] des méthodes de décomposition spécifiques pour des systèmes sous-contraints.

S [Sit06] introduit les graphes de couture[5] et des critères permettant de déterminer les conditions d'assemblage de systèmes 3D rigides ayant des points communs.

D *et al.* [DMS97, DMS98] introduisent la notion de bord pour généraliser la notion de lien virtuel d'O : lorsqu'un sous-système est résolu et retiré, ils ajoutent au reste du système les informations calculables dans le sous-système résolu concernant les entités géométriques partagées avec le reste du système.

Des travaux récents sur la décomposition cherchent à ne pas favoriser les systèmes rigides. S et S [SS06] montrent ainsi comment résoudre des systèmes que l'on peut mettre à l'échelle. S et M [SM06] en déduisent un algorithme de décomposition qui réussit à résoudre des systèmes rigides non construc-

[2] S [Sun87], O [Owe91], L et M [LM96b], F et H [FH97], A -A *et al.* [AAHMS99], J -A *et al.* [JASRVMVP04]

[3] En extension des travaux de L et M [LM96b]

[4] *Cf* définition 22 p. 31

[5] *Angl. :* seam graphs

tibles à la règle et au compas. van der M et B [vdMB10] montrent comment résoudre des systèmes rigides en les décomposant en des clusters non rigides.

2.2 Gestion des différents niveaux de constriction

La section 2.1 présentait un état de l'art des méthodes de résolution de GCS. Très générale, elle n'abordait pas spécifiquement la question de la gestion des différents niveaux de constriction d'un GCS. La présente section se centre sur la littérature décrivant des approches de la sous-constriction, de la sur-constriction et de quelques thématiques liées.

La section 2.2.1 fait le tour des différentes approches visant à caractériser le niveau de constriction d'un GCS ou à s'abstraire de l'hypothèse de rigidité ; la section 2.2.2 explique en détail la méthode du témoin ; la section 2.2.3 parle des travaux visant à s'abstraire de l'hypothèse de rigidité dans la résolution ; la section 2.2.4 explique quelles méthodes existent pour gérer le problème spécifique de la sur-constriction d'un GCS ; enfin, la section 2.2.5 détaille les travaux concernant la correction de GCS sous-contraints ou l'utilisation de la sous-constriction pour la résolution de systèmes.

2.2.1 Caractérisation du niveau de constriction

2.2.1.1 Topologie structurale

La topologie structurale est un domaine d'étude scientifique à part entière, plus vieux et plus vaste encore que celui des systèmes de contraintes géométriques. Il serait donc illusoire de vouloir en faire ici un état de l'art approchant l'exhaustivité. De par ses liens forts avec le sujet de nos travaux, nous en faisons ici un survol puis détaillons quelques travaux particuliers.

Le sujet d'étude de la topologie structurale est les assemblages de segments rigides (ou barres), reliés ensemble par leurs extrémités. Ces assemblages peuvent être représentés sous la forme de graphes, dont les sommets sont des points (du plan ou de l'espace selon la dimension) et où une arête entre deux sommets signifie que ces deux points sont reliés par une barre. La question posée en topologie structurale est de savoir si un assemblage donné est rigide, c'est-à-dire la distance entre n'importe quel couple de points est fixée.

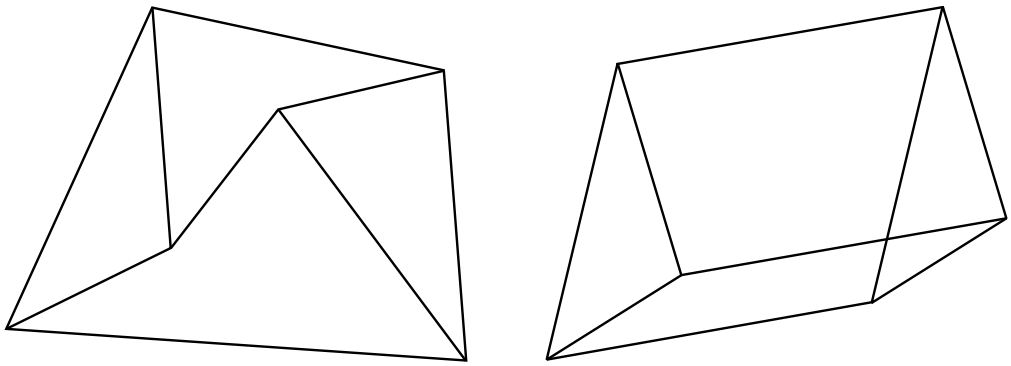


Figure 2.1 - Deux réalisations d'un même graphe 2D : réalisation générique (**a**) et dégénérée (**b**)

Le lien avec la résolution de GCS vient de la possibilité de considérer ce champ de recherche comme un cas spécifique de la caractérisation de la rigidité, limité à un univers où la seule entité géométrique est le point et les contraintes sont toutes des contraintes de distance de point à point. Une réalisation d'un graphe est une solution du GCS pour des valeurs connues des paramètres (ici les longueurs des barres).

Les travaux en topologie structurale sont basés sur l'hypothèse de généricité, qui affirme que si une réalisation d'un graphe est rigide, alors presque toutes les réalisations de ce graphe sont rigides. Autrement dit, les valeurs des paramètres importent peu pour étudier la rigidité d'un système à barres, puisque seules certaines valeurs spécifiques des paramètres, dont la probabilité d'apparition est proche de 0, génèrent des réalisations non rigides. Ces cas sont appelés des cas dégénérés et sont la plupart du temps des réalisations où les points vérifient des contraintes d'incidence ou d'alignement qui n'appartiennent pas au système. La figure 2.1 montre deux réalisations d'un même graphe : celle de gauche est générique (non dégénérée) et rigide ; celle de droite est un cas dégénéré de par des égalités de distance qui, induisant des parallélismes, génèrent un degré de liberté supplémentaire.

En terme de GCS, l'hypothèse de généricité exprime que s'il existe une valuation des paramètres ρ_0 telle que les solutions du système $\mathcal{S} = (C, X, A)$ sont rigides, alors pour presque toute valuation $\rho : A \rightarrow \mathcal{E}$ des paramètres, $\mathcal{F}_\rho(\mathcal{S})$ est bien-contraint *modulo* les déplacements.

Dans le plan et avec l'hypothèse de généricité, la caractérisation de la rigidité a été résolue par L [Lam70][6]. Basée sur le théorème de K¨ -H , la caractérisation de L est purement combinatoire : elle indique qu'un graphe planaire à n sommets est rigide s'il a $2n - 3$ arêtes et aucun sous-graphe de n' sommets avec plus de $2n' - 3$ arêtes. Des preuves alternatives du théorème de L ont été

[6] *Cf.* chapitre 1 et plus précisément la définition 25, p. 32

données [ST08, Tay99].

La caractérisation de L de la rigidité structurale est longtemps restée inapplicable de par le coût exponentiel d'une vérification de tous les sous-graphes. Les travaux de L ´ et Y [LY82] puis de J et J ´ [JJ05] ont permis de trouver des caractérisations basées sur la connexité du graphe et ainsi des algorithmes en temps polynomial [Con05]. De même, H [Hen92] fournit un algorithme permettant de décomposer le graphe en blocs rigides contenus les uns dans les autres (qui correspond à une triangularisation par blocs).

En trois dimensions, aucune caractérisation combinatoire n'est connue. Une extension naïve du critère de L consisterait à définir comme structurellement rigide un graphe à n sommets et à $3n - 6$ arêtes et n'ayant pas de sous-graphe à n' arêtes et plus de $3n' - 6$ arêtes[7]. Le contre-exemple classique de la double-banane[8] est constitué de deux systèmes rigides ayant deux points communs.

Le système 2-connexe résultant est à la fois sous-contraint et structurellement sur-contraint : il est structurellement sur-contraint car, sauf cas dégénéré, les valeurs de la distance entre les deux points communs, calculées dans chacun des deux systèmes rigides, ne sont pas identiques. Dans le cas où les valeurs sont cohérentes et où les deux systèmes rigides peuvent être assemblés, alors l'ensemble est flexible puisque la droite passant par les deux points communs constitue un axe autour duquel un système peut tourner sans que l'autre ne change.

Les tentatives de déjouer ce problème en s'attaquant à la connexité du graphe et en cherchant des coupes de graphe ont échoué, puisque M et S [MS04] ont exhibé des systèmes construits à partir de la double-banane, respectant combinatoirement l'extension 3D de la caractérisation de L mais avec une connexité plus elevée. De même, C [Cra79] montre une extension de la double-banane fait de plusieurs « bananes » interconnectées et propose d'analyser la topologie du système pour caractériser la rigidité.

Dans le domaine de l'étude de la rigidité de systèmes 3D, l'un des résultats les plus importants à ce jour est le théorème de C [Cau13] : il montre que les polyèdres convexes sont rigides si leurs faces sont rigides. Sa preuve était erronée, mais a été plus tard corrigée par S en 1928. A a affaibli les hypothèses de C en 1950 [Sab04]. Le théorème de C a été démontré différemment (et sans erreur connue) par S [Sto68].

[7] De manière générale, on peut étendre ce critère en dimension d en considérant structurellement rigide un graphe à n sommets et $dn - \frac{d(d+1)}{2}$ arêtes et n'ayant pas de sous-système à n' sommets et plus de $dn' - \frac{d(d+1)}{2}$ arêtes.

[8] *Cf.* fig. 1.14 p. 34

Ce résultat a été partiellement étendu par G [Glu74] qui montre que presque tous les polyèdres (convexes ou non) sont rigides. Il fait l'hypothèse que cela est vrai pour tout polyèdre triangulé, mais C [Con79] exhibe des contre-exemples.

H [Hen92] pose la question des conditions nécessaires pour qu'un graphe ait une unique réalisation : il ne s'intéresse pas seulement à la rigidité du graphe (nécessaire pour avoir une unique réalisation) mais aux conditions dans lesquelles, *modulo* les déplacements, il n'existe qu'une seule solution. Il conclut notamment que si un graphe en dimension d a une unique réalisation, alors soit c'est un graphe complet avec au plus $d+1$ nœuds, soit c'est un graphe $d+1$ connexe génériquement sur-contraint. Il propose des algorithmes (différents pour le plan et pour l'espace) pour identifier les graphes rigides et non génériquement sur-contraints.

F [Fek06] pose quant à lui la question des éléments qu'il faut fixer dans un graphe 2D quelconque pour qu'il soit rigide et, plus précisément, de l'ensemble minimal de nœuds du graphe à fixer pour cela. Il montre que les deux problèmes peuvent être résolus en temps polynomial, sans pour autant fournir d'algorithme.

Le lecteur désireux d'en savoir plus sur la topologie structurale pourra se référer au livre de G , S et S [GSS93] ou au plus récent ouvrage de G [Gra02].

2.2.1.2 Applications de la topologie structurale aux GCS

Les travaux de L et M [LM96b] visent à découper un système de contraintes 2D fait de n points et de $2n-3$ distances en trois parties : une partie rigide, une partie flexible et une partie sur-contrainte, chacune des parties n'étant pas nécessairement connexe. Ils considèrent le graphe biparti entités – contraintes et cherchent un flux maximal. Afin de prendre en compte l'invariance par les déplacements, ils sélectionnent deux points reliés par une contrainte de distance et ajoutent une contrainte virtuelle, liée à ces deux points, reliée au nœud puits par un arc de capacité 3. Si un couplage parfait est trouvé, le graphe correspond à un GCS rigide. Quand ce n'est pas le cas, ils identifient trois sous-graphes maximaux : le sous-graphe qui admet un couplage parfait (qui correspond à la partie rigide), le sous-graphe dans lequel il n'est pas possible de saturer toutes les contraintes (qui correspond à la partie sur-contrainte) et le sous-graphe dans lequel il n'est pas possible de saturer tous les points (qui correspond à la partie flexible). Ils proposent des moyens d'identifier des contraintes à retirer dans le second et à ajouter dans le troisième afin de rendre le système rigide. Classiquement, ils traduisent également le graphe de flux en graphe réduit, en orientant les arcs de valuation nulle du point vers la contrainte et les autres arcs dans les deux sens, puis en fusionnant les nœuds

appartenant à une même composante connexe de ce graphe.

L et M [LM98] ont proposé une méthode de décomposition trouvant une décomposée améliorée par rapport à celle de D -M . La méthode est basée sur une étude du graphe bi-parti correspondant soit au GCS soit à un système d'équations. Le calcul d'un couplage maximal en ancrant une arête donnée pouvant mener à des erreurs, comme c'était le cas dans l'algorithme de L - et M [LM96b], ils proposent de tester chaque couplage maximal obtenu en fixant tour à tour les différentes arêtes du graphe. Ils peuvent ainsi détecter des graphes (et, par extension, des GCS) structurellement sur-contraints.

Les travaux de H *et al.* [HLS97] se basent sur la rigidité structurelle en définissant la densité d'un graphe comme la différence de la somme des degrés de liberté des entités géométriques et de la somme des degrés de restriction des contraintes et en recherchant des sous-graphes denses, c'est-à-dire des sous-graphes dont la densité est supérieure ou égale à $-d$ en dimension d.

Pour trouver des sous-graphes denses, ils se basent sur un calcul de couplage maximal dans le graphe biparti. Ils « distribuent » les degrés de restriction des contraintes dans le graphe de flux en cherchant des chemins améliorants, jusqu'à ne pas pouvoir distribuer totalement les degrés de restriction d'une contrainte. Lorsque cela arrive, ils identifient un sous-graphe dense contenant cette contrainte.

Ils proposent également un algorithme pour trouver le sous-graphe dense minimal d'un graphe structurellement sur-contraint dans le cas où les valeurs des paramètres font qu'il a tout de même des solutions. Toutefois, l'ensemble des algorithmes proposés dans [HLS97] se basent sur l'heuristique de la rigidité structurelle et échouent donc à caractériser correctement des systèmes construits sur un univers géométrique plus vaste que des points et des distances du plan.

Des tentatives ont été menées [Sit00] pour trouver des heuristiques améliorant l'algorithme de recherche des sous-graphes denses, en ne considérant que les sous-graphes contenant au moins $d+1$ sommets en dimension d ou en ajoutant des règles *ad hoc* pour reconnaître certains sous-graphes précis. Il est facile toutefois de montrer [JNT02] que ces deux approches n'améliorent que très peu la caractérisation, voire créent de nouveaux problèmes.

2.2.1.3 Rigidité structurelle étendue

Pour palier à cela, les travaux de J *et al.* [JNT02, JNT03] donnent une nouvelle définition de la rigidité structurelle, appelée *es-rigidité* (*angl.* : exten-

ded structural rigidity), qui contrairement aux caractérisations précédentes, prend en compte des connaissances géométriques. Ils introduisent la notion de Degré De Rigidité (DOR) d'un sous-système qui est le nombre de déplacements indépendants qu'il peut admettre.

À partir de cette notion, l'*es*-rigidité se définit comme suit : un GCS $\mathcal{S}$ est
– sur-*es*-rigide si $\exists \mathcal{S}' \subseteq \mathcal{S}$, $\mathrm{DOR}(\mathcal{S}') > ddl(\mathcal{S}') - ddr(\mathcal{S})$;
– sous-*es*-rigide si $\mathrm{DOR}(\mathcal{S}) < ddl(\mathcal{S}) - ddr(\mathcal{S})$ et que $\mathcal{S}$ ne contient pas de sous-système sur-*es*-rigide ;
– *es*-rigide si $\mathrm{DOR}(\mathcal{S}) = ddl(\mathcal{S}) - ddr(\mathcal{S})$ et que $\mathcal{S}$ ne contient pas de sous-système sur-*es*-rigide.
J *et al.* prouvent que l'*es*-rigidité est une meilleure approximation de la rigidité que la rigidité structurelle classique en montrant qu'il n'y a pas plus de faux positifs (systèmes non rigides classés comme étant rigides) en utilisant l'*es*-rigidité, mais qu'il y a moins de faux négatifs.

De par la forte similarité entre la rigidité structurelle classique et l'*es*-rigidité[9], ils conseillent donc une modification de tous les algorithmes basés sur la rigidité structurelle classique afin de prendre en compte l'*es*-rigidité. À titre d'exemple, ils montrent quelles modifications à apporter à l'algorithme de recherche de sous-graphe dense de H *et al.* [HLS97].

2.2.1.4 Autres caractérisations de la rigidité

D'autres caractérisations de la rigidité ont été proposées : S et Z [SZ04, Zho06] introduisent la rigidité modulaire[10], une caractérisation approximative combinatoire basée sur un calcul de flux. Ils montrent que les systèmes rigides sont module-rigides et que leur caractérisation permet de ne pas tomber dans certains pièges de la caractérisation de L .

H *et al.* [HLSJS+09] proposent une caractérisation combinatoire étudiant le graphe de contraintes mais pour un univers géométrique incluant des contraintes d'angles et d'incidence.

[9] Il suffit de remplacer $\frac{d(d+1)}{2}$ par $\mathrm{DOR}(\mathcal{S})$
[10] *Angl. :* module-rigidity

2.2.2 Interrogation de témoins

Nous proposons, aux chapitres 6 et 7 des extensions de la méthode du témoin. Nous effectuons donc ici une explication détaillée de ce en quoi elle consiste. L'interrogation de témoins [MF06, MFLM06, MF07, MF09] se classe dans les méthodes algébriques et permet d'identifier les degrés de liberté d'un GCS. Elle se base sur une généralisation de l'hypothèse de généricité à d'autres univers géométriques que l'univers classique de la topologie structurale et, comme toute méthode algébrique, sur une traduction du GCS $\mathcal{S} = (C, X, A)$ sous la forme d'un système d'équations $F(A, X)$.

Classiquement, la recherche de solutions consiste à trouver les valeurs des inconnues de X telles que $F(A, X) = 0$. Une fois une solution de ce système connue, la méthode consiste à analyser les mouvements infinitésimaux permis, c'est-à-dire trouver quelles perturbations de la solution ne violent pas les contraintes, afin de trouver quels sont les degrés de liberté du système. Le nouveau système d'équations à résoudre est $F(A, X + \varepsilon v) = 0$, où v représente le vecteur vitesse des entités géométriques.

Un développement limité nous permet d'établir que

$$F(A, X + \varepsilon v) = F(A, X) + \varepsilon F'(A, X) \times v + O(\varepsilon^2)$$

Comme nous étudions les possibilités de mouvement d'une solution du système, nous avons $F(A, X) = 0$. De même, l'élément $O(\varepsilon^2)$ est négligeable. Nous posons donc la question des conditions dans lesquelles $F'(A, X) \times v = 0$. Une solution triviale – et non intéressante – est celle où v est le vecteur nul : si la solution n'est pas modifiée, elle reste une solution. En excluant ce cas, nous cherchons à trouver les solutions de $F'(A, X) = 0$.

La méthode consiste donc à étudier la matrice des dérivées partielles de F, la matrice Jacobienne J, et à en chercher le noyau. Or rechercher le noyau de cette matrice est un calcul très coûteux dans le cas général.

L'originalité de la méthode du témoin est la généralisation de l'hypothèse de généricité : sauf dans des cas dégénérés, dont la probabilité d'apparition est 0 dans l'espace des paramètres, les degrés de liberté d'un système (C, X, A) sont les mêmes pour n'importe quelle valuation de A. Par conséquent, plutôt que d'étudier le système $F(A, X)$ en général, nous pouvons étudier un témoin, c'est-à-dire un système $F(A_t, X)$, dont les solutions X_t sont connues. X_t est donc la solution d'un système similaire, puisque les valeurs des paramètres ne sont pas les mêmes. Dans la plupart des cas (nous détaillons cela plus bas), le témoin est fourni par l'esquisse que

l'utilisateur a tracée. Notons que le calcul de la matrice Jacobienne est inchangé, puisque la dérivée de l'équation $f(X) = a$ est $f'(X) = 0$ quel que soit a.

Ainsi, nous recherchons le noyau de la matrice Jacobienne de $F(A_t, X)$ évaluée en le témoin, notée J_t dans la suite. Dans la mesure où les éléments de J_t sont des nombres et non des expressions formelles, leur manipulation est bien moins coûteuse.

L'interrogation du témoin permet dès lors :

1. de calculer le rang de J_t et ainsi de détecter une redondance (donc, une sur-constriction générique) si le rang est inférieur au nombre de lignes ;
2. de vérifier l'invariance du GCS par un groupe de transformations élémentaire (translation, rotation autour d'un point, homothétie) donné en testant que le vecteur des mouvements correspondant à ce groupe fait partie de ker J_t ;
3. de détecter, en combinant les deux éléments ci-dessus, la bonne-constriction du GCS *modulo* un groupe donné.

Pour le premier élément, la méthode est simple : le calcul du rang de J_t se fait de façon classique, par exemple par une élimination de G -J . Pour le second élément, l'idée est de calculer les vecteurs correspondant au mouvement infinitésimal de chaque entité géométrique si on leur applique la transformation à tester. Ainsi, si l'on veut tester si le système est invariant par translation selon l'axe des abscisses, on va tester si le vecteur $v_{T_x} = (1, 0, 1, 0, \ldots, 1, 0)$ fait partie de ker J_t, *i.e.* si $J_t \times v_{T_x} = 0$. Nous faisons dans cet exemple l'hypothèse que le GCS est en 2D et composé uniquement de points. Pour d'autres exemples (autres entités géométriques, autres groupes de transformations), le lecteur pourra se référer par exemple à [MF09][11]. Pour vérifier qu'un GCS est invariant par un groupe composé de plusieurs groupes élémentaires, on vérifie son invariance par les différents groupes qui le composent : ainsi pour tester l'invariance par les déplacements, on teste[12] l'invariance par les translations en x, les translations en y et les rotations autour de l'origine.

Enfin, le troisième élément, *i.e.* tester si le GCS est G-bien-contraint se fait en deux étapes : tout d'abord vérifier que la taille du noyau de J_t correspond bien au nombre de degrés de liberté d'un système G-bien-contraint ; ensuite, en vérifiant l'invariance du GCS par G. Ainsi, par exemple, pour vérifier qu'un GCS 2D est **D**-bien-contraint, on vérifie que la taille du noyau est 3 et, le cas échéant, on vérifie que le système est **D**-invariant.

Les limites de la méthode du témoin sont :
– l'exactitude du calcul : tant que les valeurs de la matrice sont dans $\mathbb{Q}$, il est pos-

[11] En particulier la table 1, p. 345
[12] Par exemple, car d'autres combinaisons seraient possibles

sible de faire efficacement des manipulations exactes mais autrement[13] il faut choisir entre calcul exact et calcul approché ;
– l'hypothèse que l'on dispose d'un témoin typique, c'est-à-dire une figure ayant les mêmes propriétés combinatoires que les solutions du système[14] : il arrive, particulièrement quand le GCS contient des contraintes non paramétrées (incidence, parallélisme, *etc.*), que l'esquisse ne soit pas un témoin. Il est possible de générer un témoin dans le cas de contraintes non paramétrées, en résolvant le système engendré par ces contraintes, par exemple au moyen d'un solveur par intervalles [FMF09] ;
– sa complexité, en $O(n^3)$ pour n éléments géométriques : cette complexité est élevée comparée à d'autres méthodes de résolution actuelles, en $O(n^2)$. Toutefois, l'augmentation de la puissance de calcul des ordinateurs semble ne pas s'être accompagnée d'une augmentation de la taille des GCS à résoudre, ce qui a pour conséquence que les temps de calcul sont à l'heure actuelle tout à fait raisonnables.

Notons que, si les seuls aspects de l'interrogation de témoins auxquels nous faisons référence dans ce manuscrit sont ceux concernant la caractérisation des flexions d'un GCS, ils s'intègrent dans le cadre beaucoup plus général d'une approche probabiliste consistant à tester un prédicat sur un (ou des) témoin(s) et à déduire de sa validité qu'il est génériquement valide. Cette approche est par exemple utilisée dans Cabri-Géomètre [Cab] pour vérifier des propriétés géométriques non évidentes à prouver formellement, comme l'alignement de points ou l'orthogonalité de droites dans le cas général.

2.2.3 Abstraction de l'hypothèse de rigidité

La grande majorité des travaux sur les systèmes de contraintes géométriques font l'hypothèse de la rigidité du GCS à résoudre. Ils considèrent donc comme fixés trois degrés de liberté en 2D et 6 en 3D. Certains travaux cherchent à avoir une approche plus générale et à ne pas favoriser la rigidité.

Plusieurs travaux cherchent à ne pas utiliser les coordonnées cartésiennes des entités géométriques du GCS, ce qui leur permet de ne pas privilégier les déplacements par le retrait de trois degrés de liberté. Pour cela, S ´ *et al.* [Ser00, LLS00] traduisent le GCS sous la forme d'un squelette, en remplaçant les distances point à point par des vecteurs et en considérant des contraintes d'angle, d'incidence et de mesure sur ces vecteurs. Ils identifient alors des boucles de vecteur et d'angle indépendantes et,

[13] Il existe des systèmes simples pour lesquels il n'existe pas de témoin rationnel, par exemple un pentagone régulier

[14] *I.e.* si les contraintes et leurs paramètres décrivent des solutions dégénérées, le témoin doit présenter les mêmes dégénérescences.

à chaque boucle, associent une équation. Ils cherchent alors à résoudre un système d'équations dont les inconnues sont les angles entre les vecteurs et les normes des vecteurs. Y [Yan02, Yan03, Yan06] ainsi que M et F utilisent les déterminants de C -M pour disposer de systèmes d'équations plus simples, dans lesquels les inconnus sont les distances et ne dépendent donc pas d'un repère donné[15].

Les travaux de S *et al.* consistent à redéfinir la bonne constriction en considérant un système rigide comme sous-contraint s'il n'est pas fixé : la possibilité d'appliquer un déplacement sans violer de contraintes permet en effet de générer une infinité de solutions. Leurs premiers travaux [SS02, SS03, SS06] consistent à considérer des GCS invariants par homothétie, contraints par des angles, des ratios de distance et des incidences. Dans [SM06], ils en déduisent une généralisation multi-groupe de l'algorithme de décomposition de M [Mat97] : ils cherchent à fixer des éléments géométriques dans le système, en fonction du groupe considéré (pour les déplacements, un point et une direction en 2D, un point et trois directions en 3D ; pour les similitudes[16], deux points en 2D, deux points et une direction en 3D) et exhibent des systèmes rigides difficilement constructibles directement mais contenant un système invariant par homothétie facile à résoudre et qui, mis à l'échelle, permet la construction du système global.

2.2.4 Gestion de la sur-constriction

Certains travaux se servent de la sur-constriction générique dans la résolution : P - *et al.* [PŽD08], en s'appuyant sur leurs travaux précédents de formulation intrinsèque des contraintes [Pod02] (similaires à l'approche de Y), améliorent leur capacité de résolution en ajoutant de l'information redondante pour arriver à des motifs qu'ils savent résoudre, à la manière d'un système expert. Leur méthode revient à peu près à ajouter des contraintes de bord lorsqu'un sous-système est résolu, sans toutefois retirer le sous-système. Nous avons également déjà cité H [Hen92] à la section 2.2.1.1, qui utilise des contraintes génériquement redondantes mais de métriques cohérentes avec les solutions pour assurer l'unicité de la réalisation d'un graphe.

La grande majorité des travaux, aussi bien en topologie structurale qu'en résolution de systèmes de contraintes géométriques, considèrent la sur-constriction générique comme source d'erreur. Nous avons déjà évoqué les travaux de caractérisation de la rigidité : ils incluent en général l'identification des sous-systèmes sur-contraints, comme le font par exemple L et M [LM96b]. G et C [GC98b]

[15] Voir aussi [Hav91] pour plus de détails.

[16] Composition des translations, rotations et homothéties

utilisent la méthode de calcul symbolique de R -W sur le système d'équations correspondant à un GCS pour décider si les contraintes sont indépendantes. Les équations étant quelconques, le coût de cette approche la rend inutilisable dans la pratique.

Pendant longtemps, l'approche de la correction de la sur-constriction générique a été de laisser l'utilisateur retirer des contraintes. Toutefois, en l'absence d'une excellente connaissance des mécanismes de la résolution de systèmes de contraintes géométriques, la tâche n'est pas aisée. N *et al.* [NDB98] cherchent donc à corriger automatiquement la partie sur-contrainte d'un système fait de blocs rigides liés par des contraintes d'indicence, au moyen d'un graphe de dépendance. Après une phase de rétro-propagation dans le graphe pour obtenir un squelette du graphe initial, ils détectent les parties sur-contrainte et sous-contrainte et peuvent alors retirer une contrainte dans l'une et en ajouter une dans l'autre. Ils recommencent alors le processus avec ce nouveau graphe jusqu'à ce que toutes les contraintes d'incidence soient satisfaites.

H *et al.* [HSY04] réutilisent les plans de décomposition-assemblage[17] et l'algorithme de nœud de frontière[17] pour trouver automatiquement quelle contrainte retirer dans un GCS avec une unique contrainte redondante. Ils proposent des pistes pour étendre cette approche à des GCS avec plus d'une contrainte redondante.

La méthode du témoin[18] permet d'obtenir les mêmes résultats que [GC98b] pour une complexité plus faible, sous l'hypothèse de typicalité.

2.2.5 Gestion de la sous-constriction

De même que pour la sur-constriction, la sous-constriction est dans la majorité des travaux considérée comme une erreur à corriger. Nous ne revenons pas sur les travaux visant à détecter la sous-constriction : les systèmes sous-contraints étant assimilés aux systèmes non rigides, les méthodes caractérisant la rigidité incluent une caractérisation des systèmes sous-contraints. Nous détaillons ici les principaux travaux concernant la manipulation de systèmes sous-contraints : l'ajout de contraintes pour arriver à un système bien contraint, ainsi que les travaux concernant la résolution ou l'utilisation, dans la résolution, de systèmes sous-contraints.

[17] *Angl. :* « decomposition-recombination plans » ou DR-plans [HLS01a] et « frontier vertex algorithm » [HLS01b]

[18] *Cf.* section 2.2.2

2.2.5.1 Ajout de contraintes

L'approche la plus courante de la sous-constriction est sa correction, par ajout de contraintes. J -A *et al.* [JASRVMVP03, JASRVM03] ont formalisé les deux problèmes que posent les systèmes sous-contraints :

1. complétion bien contrainte : trouver automatiquement un ensemble de contraintes à ajouter à un GCS structurellement sous-contraint pour qu'il devienne structurellement rigide ;
2. complétion : trouver automatiquement un ensemble de contraintes à ajouter à un GCS structurellement sous-contraint pour qu'il devienne résoluble au moyen des méthodes de résolution géométriques dont on dispose[19].

Ils introduisent la notion de décomposition par ensemble[20] qui consiste à découper un GCS $\mathcal{S}$ en trois sous-systèmes $\mathcal{S}_1$, $\mathcal{S}_2$ et $\mathcal{S}_3$ tels que $\mathcal{S}_1 + \mathcal{S}_2 + \mathcal{S}_3 = \mathcal{S}$ et que $\mathcal{S}_i \cap \mathcal{S}_j$ soit limité à une unique entité géométrique : ce découpage permet de séparer un système dont le graphe de contrainte n'est pas 3-connexe sans « casser » de contrainte. En opérant récursivement, ils construisent un *s-arbre*[21]. Ils montrent qu'un GCS décomposable en un s-arbre n'est pas sur-contraint et que si les feuilles du s-arbre correspondent chacune à une des contraintes du système, il est structurellement rigide. Ils en déduisent un algorithme pouvant fournir les différentes complétions bien contraintes d'un système sous-contraint en ajoutant des contraintes entre deux entités géométriques d'une feuille lorsqu'il n'y en a pas. Puisque plusieurs décompositions en s-arbres sont possibles, plusieurs complétions bien contraintes existent. Ils proposent donc des méthodes pour trouver, parmi les possibilités, celles qui répondent au problème 2.

Z et G [ZG06b] rajoutent à cela un troisième problème, celui de la complétion bien contrainte optimale : trouver automatiquement un ensemble de contraintes à ajouter à un GCS structurellement sous-contraint pour qu'il devienne structurellement rigide et que l'ensemble d'équations à résoudre simultanément soit de taille minimale. Ils proposent également un algorithme pour le problème 1, fondé sur la recherche de sous-systèmes résolubles une fois un segment (deux points contraints par une distance) fixé. Le sous-système résolu est retiré et les entités correspondantes marquées, dans un Graphe Orienté Acyclique (DAG) comme dépendant des deux points du segment. Ils fixent alors un segment dans le système restant, et recommencent, jusqu'à avoir résolu l'ensemble du système. Ils étudient alors le DAG pour invalider la fixation de certains points, jusqu'à ce qu'il ne reste plus qu'un segment fixé. Ils évoquent des modifications de l'algorithme pour tendre vers une solution du problème de complétion bien contrainte optimale.

[19] C'est-à-dire des méthodes à base de graphe, de règles, *etc.* qui sont incomplètes mais plus efficaces que des méthodes algébriques complètes

[20] *Angl. :* set decomposition

[21] *Angl. :* s-tree, pour set decomposition tree

Avant les travaux de J -A *et al.*, F et H [FH97] ont proposé une méthode par formation de *clusters* [FH96] qui concerne les problèmes de complétion et de complétion bien contrainte optimale. Les travaux de T et W [TW06] introduisent un algorithme fondé sur un catalogue de règles de construction, à partir desquelles ils appliquent une rétro-propagation jusqu'à arriver à un squelette du GCS et à pouvoir fournir alors une paramétrisation du système. Nous avons déjà parlé à la section 2.2.4 des travaux de N *et al.* [NDB98] qui corrigent un système contenant une partie sous-contrainte et une partie sur-contrainte.

2.2.5.2 Résolution et utilisation de systèmes non rigides

Les systèmes de contraintes géométriques non rigides sont très étudiés dans le domaine de la robotique et, plus précisément, de la cinématique : l'objectif est d'étudier quelles sont les libertés d'un assemblage et d'exprimer l'espace des coordonnées possibles de certaines entités géométriques [ADG00, PCRT07]. Parmi les travaux utilisant vraiment une approche de résolution de systèmes de contraintes géométriques, citons ceux de K *et al.* [Kra90, Kra91, Kra92, AKC96], qui considèrent un assemblage d'objets rigides dans l'espace, connectés entre eux par des contraintes d'incidence. Initialement, aucune des contraintes d'incidence n'est considérée comme satisfaite, et chaque objet dispose de 3 degrés de liberté en translation et 3 degrés de liberté en rotation. Ensuite, les contraintes d'incidence sont consommées les unes après les autres et, à l'aide d'un catalogue des configurations possibles, K retire des degrés de liberté aux différents objets.

Nous avons déjà évoqué les travaux de S *et al.* [SS06, SM06] généralisant la notion de bonne constriction à la bonne constriction *modulo* un groupe de transformations. Ils montrent qu'ils peuvent non seulement résoudre des GCS bien contraints *modulo* les similitudes mais aussi aboutir à un algorithme de décomposition plus général permettant de résoudre facilement et efficacement des systèmes que les méthodes de la littérature ne savent résoudre que numériquement.

Les travaux de van der M et B [vdMB08, vdMB10] consistent en la résolution de GCS rigides, mais en considérant des clusters non rigides. Ils considèrent trois types de clusters :
– les clusters rigides ;
– les clusters dont on peut changer l'échelle[22], qui sont, dans la terminologie de S *et al.* des systèmes bien contraints *modulo* les similitudes ;
– les clusters radiaux, qui sont des assemblages faits de plusieurs droites passants par un même point, avec un autre point incident à chaque droite.

[22] *Angl. :* scalable clusters

Avec une approche classique de réécriture et un catalogue de règles de réécriture spécifiques à ces clusters, ils améliorent la puissance de résolution des méthodes de clusters existantes.

G *et al.* [GHY02, GHY04] proposent une méthode de résolution hybride d'un GCS rigide lorsqu'aucune méthode de décomposition n'est connue pour ce système : ils retirent une contrainte c et la remplacent par une autre contrainte c'. Ils expriment alors les solutions de ce nouveau système comme une fonction de la valeur de la métrique associée à c' et en déduisent l'équation exprimant la métrique de c en fonction de celle de c'. F et S *et al.* [FS06, FS07, FS08] étendent cette méthode à la possibilité de remplacer plus d'une contrainte : ils retirent une contrainte, appliquent une méthode de rétro-propagation et recommencent jusqu'à ce que toutes les entités géométriques restantes soient liées à moins de contraintes que le nombre de leurs degrés de liberté. Ils ajoutent alors autant de contraintes qu'ils en ont retirées, de sorte à pouvoir continuer la rétro-propagation. Ils font alors varier les métriques associées aux contraintes ajoutées par la méthode de N - R pour rattraper les valeurs associées aux contraintes qu'elles remplacent.

2.2.6 Parcours de l'espace des paramètres

Non formellement liés à la sous-constriction, les travaux portant sur le parcours de l'espace des solutions sont importants lorsque un GCS a un très grand nombre de solutions ce qui, de par les constructions géométriques dont les résultats sont des multi-fonctions, comme une simple intersection de cercles, est fréquent.

La question posée dans les travaux de E *et al.* [EVSD00b, EVSMD03] est de trouver des solutions « proches » de l'esquisse. Pour cela, ils parcourent un plan de construction (considéré comme une liste de définitions) et construisent un arbre des solutions. Chaque niveau de l'arbre correspond à une étape du plan de construction, la racine étant la première construction. À un nœud donné on associe comme fils les différentes constructions possibles. Ainsi, un nœud correspondant à une intersection de cercles aura deux successeurs dans le cas général, un ou zéro pour certaines valeurs des coordonnées des centres des cercles et de leur rayon. E *et al.* proposent alors des méthodes permettant d'effectuer des coupes dans l'arbre, qu'ils appellent « gel de branche » pour ne parcourir qu'une partie des solutions et sélectionner celles qui ressemblent le plus à l'esquisse, en terme d'orientations relatives des entités. Ils proposent aussi une méthode de parcours interactif des solutions en sélectionnant un nœud de l'arbre et en observant quelles sont les possibilités pour la construction suivante.

Van der M et B [vdMB05, vdMB06] étendent ces travaux en pro-

posant une méthode constructive de détermination des domaines de validité des paramètres. S *et al.* [SAZK06] proposent également une méthode de navigation dans l'arbre des solutions pour passer d'une solution à l'autre. J -A *et al.* [JALSR02, JALSR03, LBYJA04, LSRGJA05] proposent d'utiliser des algorithmes génétiques pour parcourir l'espace des paramètres et résoudre le problème d'identification d'une racine [BFH+95].

C 3

C ´ ´

Sommaire

> *On a tort d'apprendre aux enfants que tous les problèmes n'ont qu'une et une seule solution*
> – Alice Ferney, écrivaine française

Nous avons vu au chapitre 1 que la décomposition d'un GCS en sous-systèmes et l'assemblage de leurs solutions était une pratique désormais systématique dans toutes les méthodes de résolution. Malgré le nombre de telles méthodes présentées

dans la littérature, la correction et la complétude du principe de la décomposition sont toujours restées des conjectures.

Autrement dit, il n'a jamais été montré, avant le début de nos travaux, sous quelles conditions l'assemblage des figures solutions des sous-systèmes $\mathcal{S}_1 \ldots \mathcal{S}_n$ d'un système $\mathcal{S}$ (ces conditions pouvant inclure des modifications des systèmes $\mathcal{S}_i$ n'en faisant plus à proprement parler des sous-systèmes de $\mathcal{S}$) conduisait à un ensemble de figures égal à l'ensemble des solutions de $\mathcal{S}$. Cette démonstration a uniquement été faite dans le cas des GCS rigides par S et M [Mat97]. Pourtant, l'importance de la robustesse des algorithmes est connue et affirmée dans la littérature [Sch01]. H [Hof01] ainsi que J -A et S -R [JASR97] montrent chacun la convergence d'une méthode de résolution et non sa correction.

L'un des éléments essentiels de notre démarche étant le traitement homogène des GCS sous-contraints et bien contraints, nous proposons ici d'étendre aux cas non rigides la démonstration de [Mat97] en explicitant les conditions de la correction et de la complétude de la décomposition et en prouvant qu'elles sont nécessaires et suffisantes. Pour cela, nous commençons par formaliser le point de vue multi-groupe de S et M [SM06] (section 3.1). Nous pouvons alors effectuer la démonstration des conditions de correction et de complétude des méthodes de décomposition (section 3.2) de façon générale : nos démonstrations s'appliquent aux méthodes privilégiant la rigidité aussi bien qu'aux méthodes prenant en compte d'autres groupes de transformations. Enfin, nous appliquons ces théorèmes à la méthode d'O [Owe91] (section 3.3).

3.1 Le point de vue multi-groupe

Comme nous l'avons vu dans les deux chapitres précédents, les GCS sont généralement considérés comme bien contraints lorsqu'ils sont rigides. La rigidité d'un GCS est bien souvent prise pour hypothèse par les solveurs. Pourtant, une solution d'un système rigide peut être déplacée sans violer de contraintes et le système a donc une infinité de solutions. Formellement, il est sous-contraint. Plus généralement, des systèmes non rigides peuvent être considérés comme correctement conçus si c'est ce que l'utilisateur recherche, par exemple pour un système que l'on peut mettre à l'échelle [SS06].

Pour tenir compte de cela, une définition plus générale de la bonne constriction a été informellement donnée par Pascal S et Pascal M précédemment au début de nos travaux [SM06] consistant à partitionner l'ensemble des solutions selon des groupes de transformationset à considérer le nombre de partitions pour déterminer le niveau de constriction. Dans ce chapitre, nous définissons formellement ces

notions.

Nous définissons l'invariance par un groupe de transformations (section 3.1.1) et étendons l'opération de jointure à une opération de jointure par transformation (section 3.1.2). Nous définissons ensuite ce qu'est un repère pour un GCS (section 3.1.3) puis nous étondons la notion classique de bonne constriction à la bonne constriction *modulo* un groupe de transformations(section 3.1.4).

3.1.1 Invariance par un groupe de transformations

En CAO, l'ensemble des solutions $\mathcal{F}(\mathcal{S})$ d'un GCS est habituellement stable par certaines transformations géométriques.

Définition 34. Transformation géométrique : *Une transformation géométrique est une fonction qui à une figure d'un GCS associe une autre figure de ce GCS.*

La stabilité d'un ensemble de solutions par des transformations géométriques conduit à considérer un groupe d'invariance et son action sur l'ensemble des solutions. Nous définissons l'invariance d'un groupe de figures par un groupe de transformations, qui est informellement le fait qu'on puisse appliquer une transformation à une figure sans violer de contraintes.

Définition 35. Invariance par un groupe de transformations : *Un ensemble de figures $\mathcal{F}$ est invariant par un groupe de transformations G (ou G-invariant) si* $\forall(f,\varphi) \in \mathcal{F} \times G, \varphi(f) \in \mathcal{F}$.

Par extension, on dit d'un GCS $\mathcal{S}$ qu'il est invariant par le groupe G si $\mathcal{F}(\mathcal{S})$ est invariant par G. Ceci amène une notion sémantique (invariance d'un ensemble de figures) au niveau syntaxique (invariance de systèmes de contraintes). De même, on dira qu'une contrainte est G-invariante.

Un groupe de transformations définit une relation d'équivalence entre deux figures : f et f' sont équivalentes s'il existe une transformation $\varphi \in G$ telle que $f = \varphi(f')$. Nous appelons *orbites* pour le groupe G les classes d'équivalence de cette relation.

Définition 36. Orbites : *Soit G un groupe de transformations, $\mathcal{F}$ un ensemble de figures G-invariant et f une figure appartenant à $\mathcal{F}$. L'orbite de f par G est l'ensemble* $G.f = \{f' \mid \exists\varphi \in G, \varphi(f') = f\}$.

L'ensemble des orbites de $\mathcal{F}$ par G, noté $\mathcal{F}/G$, forme une partition de $\mathcal{F}$. $|\mathcal{F}/G|$ est donc le nombre d'orbites de $\mathcal{F}$ par G. Notons que si $\mathcal{F}$ est invariant par un groupe

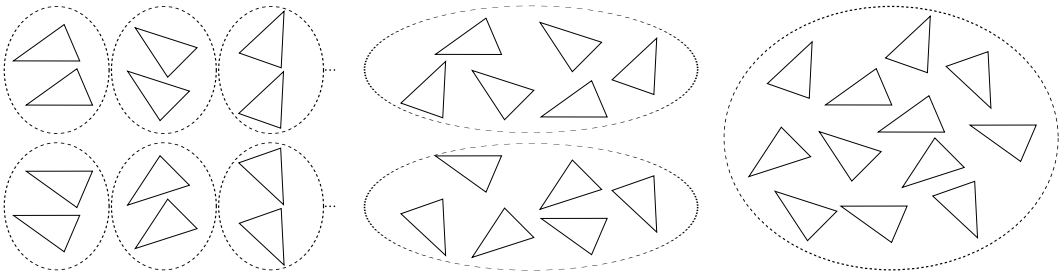


Figure 3.1 - Orbites du GCS de la figure 1.11 pour les translations (à gauche), les déplacements (au milieu) et pour les isométries (à droite)

G, il est invariant par tout $G' \subset G$. De plus, si $\mathcal{F}$ est invariant par les groupes G_1 et G_2, alors il est également invariant par $\langle G_1 \cup G_2 \rangle$, le groupe engendré par G_1 et G_2.

Exemple 13. Orbites d'un GCS

Considérons le GCS de la figure 1.11 qui consiste en un triangle rigide contraint par trois distances. La figure 3.1 montre les orbites des solutions de ce système pour trois groupes de transformations : les translations, les déplacements (translations et rotations) et les isométries (déplacements et symétries). Il n'y a qu'une orbite pour les isométries, mais deux pour les déplacements : on peut passer d'une orbite à l'autre en opérant une symétrie. Pour les translations, il y a une infinité de solutions, comme l'indiquent les points de suspension sur la droite. On peut passer horizontalement d'une orbite à l'autre en opérant une rotation. Comme pour le cas des déplacements, on peut passer verticalement d'une orbite à l'autre par symétrie.

Quand un système $\mathcal{S}$ est G-invariant, $\mathcal{F}(\mathcal{S})$ peut être caractérisé par un ensemble de représentants $\mathcal{F}_r$ contenant une figure par orbite. En d'autres termes, $\mathcal{F}(\mathcal{S}) = G.\mathcal{F}_r$.

Définition 37. Représentants : *Soit $\mathcal{S}$ un système de contraintes géométriques G-invariant et $\mathcal{F}$ l'ensemble de ses solutions. Un ensemble de figures $\mathcal{F}_r$ est dit représentatif de $\mathcal{F}$ si*
– $|\mathcal{F}_r| = |\mathcal{F}/G|$
– $\forall (f_1, f_2) \in \mathcal{F}_r^2, \nexists \varphi \in G, f_1 = \varphi(f_2)$
Chaque figure de $\mathcal{F}_r$ est appelée un représentant de $\mathcal{F}$.

Exemple 14. Représentants

Dans l'exemple 13, l'ensemble des solutions peut être défini par $\mathcal{F}(\mathcal{S}) = G.\mathcal{F}_r$ avec G le groupe des déplacements et $\mathcal{F}_r$ un ensemble contenant une figure de chacune des deux orbites de $\mathcal{F}/G$.

3.1.2 Jointures par transformation

Avec le point de vue de l'invariance par des groupes de transformationsvient aussi une nouvelle définition de l'opération de jointure. Elle consiste à considérer comme compatibles pour la jointure deux figures pour lesquelles il existe des transformations des groupes considérés qui, appliquées sur les figures, les rendent compatibles au sens de la définition 16.

Définition 38. Compatibilité par transformation : *Soient* $\mathcal{S}_1 = (C_1, X_1, A_1)$ *et* $\mathcal{S}_2 = (C_2, X_2, A_2)$ *deux GCS construits sur la même signature. On pose* $X_e = X_1 \cap X_2$. *Deux figures* f_1 *et* f_2, *solutions respectives de* $\mathcal{S}_1$ *et* $\mathcal{S}_2$ *sont compatibles par transformation pour un ensemble de groupes* $\mathcal{G}$ *s'il existe* $(\varphi_1, \varphi_2) \in G_1 \times G_2$ *tels que* $G_1 \in \mathcal{G}$, $G_2 \in \mathcal{G}$ *et* $\varphi_1(f_1)|_{X_e} = \varphi_2(f_2)|_{X_e}$.

Dans la suite, étant données deux figures f_1 et f_2 et deux groupes G_1 et G_2, on note $(\Psi_1, \Psi_2)^{(G_1,G_2)}_{(f_1,f_2)} \subseteq G_1 \times G_2$ l'ensemble des couples (φ_1, φ_2) tels que $\varphi_1(f_1)|_{X_e} = \varphi_2(f_2)|_{X_e}$. En l'absence de confusion, G_1 et G_2 pourront être omis de la notation.

Chaque élément $(\Psi_1, \Psi_2)_{(f_1,f_2)}$ permet d'exprimer une solution de $\mathcal{S}$. Nous avons alors

$$\mathcal{F}(\mathcal{S}) = \bigcup_{(f_1,f_2)\in\mathcal{F}_{r_1}\times\mathcal{F}_{r_2}} \{f | f = \varphi_1(f_1) \otimes \varphi_2(f_2) \wedge (\varphi_1, \varphi_2) \in (\Psi_1, \Psi_2)_{(f_1,f_2)}\}$$

Dans le cas où $G_2 \subseteq G_1$, la définition de la compatibilité par transformation pourrait être « simplifiée » en montrant que s'il existe $(\varphi_1, \varphi_2) \in (\Psi_1, \Psi_2)_{(f_1,f_2)}$ tel que $\varphi_1(f_1)|_{X_e} = \varphi_2(f_2)|_{X_e}$, alors il existe $\varphi \in G_1$ tel que $\varphi(f_1)|_{X_e} = f_2|_{X_e}$, en posant $\varphi = \varphi_2^{-1} \circ \varphi_1$. Autrement dit, les deux figures sont compatibles par transformation s'il existe $\varphi \in G_1$ tel que $\varphi(f_1) \equiv_{X_e} f_2$.

À partir de cette définition de compatibilité, nous étendons la définition de la jointure.

Définition 39. Jointure par transformation : *Soient* f_1 *et* f_2 *deux figures compatibles par transformation pour un ensemble de groupes* $\mathcal{G}$. *La jointure par transformation de* f_1 *et* f_2 *est l'ensemble* $f_1 \otimes_{\mathcal{G}} f_2 = \{f \mid f = \varphi_1(f_1) \otimes \varphi_2(f_2), (\varphi_1, \varphi_2) \in (\Psi_1, \Psi_2)^{(G_1,G_2)}_{(f_1,f_2)}, G_1 \in \mathcal{G}, G_2 \in \mathcal{G}\}$

Cette définition est très différente de la jointure classique[1] en cela qu'elle définit un ensemble de figures. En outre, toutes les figures de $f_1 \otimes_{\mathcal{G}} f_2$ n'appartiennent

[1] *Cf.* définition 17

pas nécessairement à la même orbite selon le plus grand groupe de $\mathcal{G}$: en effet il peut exister plusieurs φ_1 telles que $\varphi_1(f_1) \equiv_{X_e} f_2$, ce qui signifie qu'il n'existe pas nécessairement de transformation globale permettant de passer d'une figure à l'autre.

Étant donnés deux groupes G_1 et G_2, en considérant un ensemble de figures $\mathcal{F}_{r_1}$ représentatif de $\mathcal{S}_1$ pour G_1 (*i.e.* contenant une figure de chaque orbite de $\mathcal{F}(\mathcal{S}_1)$ pour G_1) et un ensemble $\mathcal{F}_{r_2}$ représentatif de $\mathcal{S}_2$ pour G_2, on peut montrer que l'ensemble des jointures par transformation de deux figures appartenant à $\mathcal{F}_{r_1} \times \mathcal{F}_{r_2}$ est égal à $\mathcal{F}(\mathcal{S}_1 + \mathcal{S}_2)$.

Considérons deux systèmes $\mathcal{S}_1 = (C_1, X_1, A_1)$, G_1-invariant, et $\mathcal{S}_2 = (C_2, X_2, A_2)$, G_2-invariant. Soit $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$ et posons $X_e = X_1 \cap X_2$ l'ensemble des inconnues communes aux deux GCS. $\mathcal{F}(\mathcal{S}_1)$ et $\mathcal{F}(\mathcal{S}_2)$ peuvent chacun être décrits par un ensemble de représentants, que l'on nomme respectivement $\mathcal{F}_{r_1}$ et $\mathcal{F}_{r_2}$: $\mathcal{F}(\mathcal{S}_1) = G_1.\mathcal{F}_{r_1}$ et $\mathcal{F}(\mathcal{S}_2) = G_2.\mathcal{F}_{r_2}$. On peut ainsi écrire n'importe quelle solution $f \in \mathcal{F}(\mathcal{S})$ comme $f = \varphi_1(f_1) \otimes \varphi_2(f_2)$ avec $f_i \in \mathcal{F}_{r_i}$ et $\varphi_i \in G_i$ pour chaque i.

Notons que si, pour une figure $f = \varphi_1(f_1) \otimes \varphi_2(f_2)$, avec X_e l'ensemble des entités géométriques communes à f_1 et f_2, on connaît une transformation $\varphi'_1 \in G_1$ telle que $\varphi'_1(f|_{X_e}) = f|_{X_e}$, alors $\varphi_1(\varphi'_1(f_1)) \otimes \varphi_2(f_2)$ est également solution.

Exemple 15. Résolution du « papillon » par jointure de solutions des deux triangles

La figure 3.2 montre le GCS du « papillon » fait de deux triangles rigides $\mathcal{S}_1$ et $\mathcal{S}_2$. Ces sous-systèmes sont bien contraints *modulo* les déplacements et partagent un point commun P. Soient f_1 et f_2 des représentants de $\mathcal{F}(\mathcal{S}_1)$ et de $\mathcal{F}(\mathcal{S}_2)$ respectivement.

Une solution f de $\mathcal{S}$ peut être construite en déplaçant f_1 et f_2 de telle sorte que les coordonnées de P dans chaque figure soient cohérentes. Cela revient à trouver un couple (φ_1, φ_2) tel que $f = \varphi_1(f_1) \otimes \varphi_2(f_2)$ et $\varphi_1(f_1)|_{\{P\}} = \varphi_2(f_2)|_{\{P\}}$. Un exemple est donné à la figure 3.2b.

N'importe quel déplacement φ'_1 ne modifiant pas les coordonnées de P permet de trouver une nouvelle solution $f' = \varphi_1(\varphi'_1(f_1)) \otimes \varphi_2(f_2)$ comme l'illustre la figure 3.2c.

Cette façon de trouver de nouvelles solutions à partir d'une solution spécifique revient à trouver les stabilisateurs pour chaque groupe des inconnues partagées. On note $G^x = \{\varphi \mid \varphi(x) = x\}$ le sous-groupe stabilisateur de x pour G. Si $f = \varphi_1(f_1) \otimes \varphi_2(f_2)$ avec X_e les inconnues communes à f_1 et à f_2, alors pour tout $g_1 \in G_1^{f|_{X_e}}$ et $g_2 \in G_2^{f|_{X_e}}$, on a $(g_1\varphi_1, g_2\varphi_2) \subset (\Psi_1, \Psi_2)_{f_1,f_2}$. Notons que $G_1^{f|_{X_e}}$ est isomorphe à $G_1^{f_1|_{X_e}}$ puisque tous les stabilisateurs d'une orbite spécifique sont conjugués. Cela

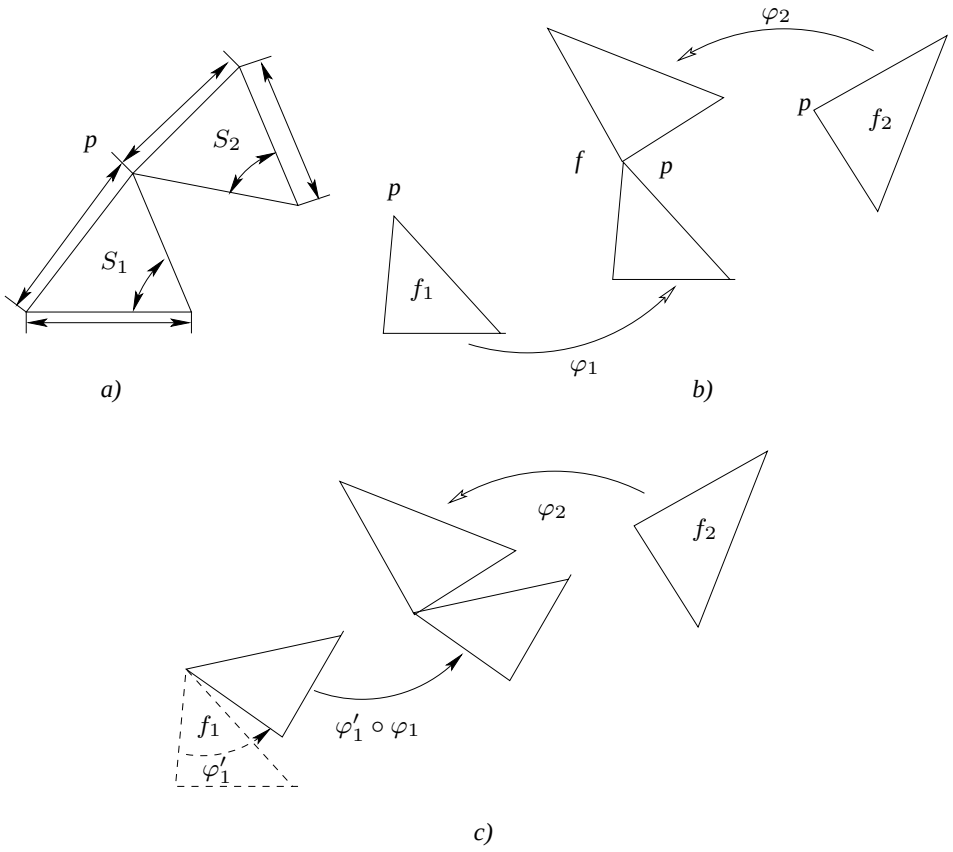


Figure 3.2 - Résolution du « papillon » par jointure de solutions des deux triangles : a) Esquisse ; b) Deux solutions sont assemblées ; c) Le déplacement de f_1 n'empêche pas l'assemblage

amène à dire, dans l'exemple 15, que la rotation autour du point P peut être effectuée avant ou après application de φ_1.

Le théorème suivant montre que si l'on connaît un sous-système $\mathcal{S}_1$ d'un système $\mathcal{S}$ et que le groupe de bonne constriction de $\mathcal{S}_1$ inclut celui de $\mathcal{S}$, alors l'ensemble obtenu par jointure par transformation d'un représentant de $\mathcal{F}(\mathcal{S}_1)$ et de $\mathcal{F}(\mathcal{S}-\mathcal{S}_1+\mathcal{B}(\mathcal{S}_1))$ permet de représenter toutes les solutions de $\mathcal{S}$.

Théorème 2. Jointure et bord : *Soit $\mathcal{S}=\mathcal{S}_1+\mathcal{S}_2$ un système G-invariant vérifiant $\mathcal{F}(\mathcal{S}_1)=G'.\mathcal{F}_{r_1}$ et $\mathcal{F}(\mathcal{B}(\mathcal{S}_1)+\mathcal{S}_2)=G.\mathcal{F}_{r_2}$, avec $G\subseteq G'$. On pose*

$$\mathcal{F}=\bigcup_{(f_{r_1},f_{r_2})\in\mathcal{F}_{r_1}\times\mathcal{F}_{r_2}} f_{r_1}\otimes_{G'} f_{r_2}$$

On a $\mathcal{F}(\mathcal{S})=G.\mathcal{F}$.

Démonstration:

La preuve se fait par inclusion mutuelle : $G.\mathcal{F}\subseteq\mathcal{F}(\mathcal{S})$ et $\mathcal{F}(\mathcal{S})\subseteq G.\mathcal{F}$.

1. $G.\mathcal{F}\subseteq\mathcal{F}(\mathcal{S})$: il s'agit de montrer que la jointure par transformation ne produit pas de figures qui ne sont pas des solutions.
   Si $f\in\mathcal{F}$ alors $f=\varphi(f_{r_1})\otimes f_{r_2}$ avec $\varphi\in G'$, $f_{r_1}\in\mathcal{F}_{r_1}$ et $f_{r_2}\in\mathcal{F}_{r_2}$. Puisque $\varphi(f_{r_1})\in\mathcal{F}(\mathcal{S}_1)$, $f\in\mathcal{F}(\mathcal{S})$.
2. $\mathcal{F}(\mathcal{S})\subseteq G.\mathcal{F}$: il s'agit de montrer que toutes les solutions sont dans $G.\mathcal{F}$.
   Considérons une figure $f=f_1\otimes f_2$ et soit f_{r_2} le représentant de l'orbite de f_2. Il existe $\varphi\in G$ tel que $\varphi(f_2)=f_{r_2}$. Puisque $\mathcal{F}(\mathcal{S})$ est G-invariant, $\varphi(f)\in\mathcal{F}(\mathcal{S})$. On a donc $\varphi(f)=\varphi(f_1)\otimes\varphi(f_2)$. Puisque $G\subseteq G'$, on a $\varphi(f_1)\in\mathcal{F}(\mathcal{S}_1)$ et il existe donc un représentant $f_{r_1}\in\mathcal{F}_{r_1}$ de l'orbite contenant $\varphi(f_1)$.

□

3.1.3 Repères

Informellement, un repère d'un GCS $\mathcal{S}$ est un sous-système R de $\mathcal{S}$ tel que si les coordonnées des entités géométriques de R sont fixées, alors le nombre de solutions de $\mathcal{S}$ est fini. Les repères servent à désigner des figures spécifiques dans un ensemble de figures.

Rappelons tout d'abord qu'on dit de l'action d'un groupe G qu'elle est *libre* (ou que G agit librement) sur un ensemble $\mathcal{F}$ si pour tout g_1 et g_2 distincts dans G et tout élément $f\in\mathcal{F}$, on a $g_1.f\neq g_2.f$.

Dans une orbite de figures $G.f$, une figure spécifique f' peut être identifiée en donnant une de ses sous-figures $f'|_X$ à condition que G agisse librement sur $G.f'|_X$. Si ce n'est pas le cas, alors plusieurs figures de $G.f$ peuvent contenir $f'|_X$.

Exemple 16. Repère

Considérons un système $\mathcal{S}$ décrivant un triangle en 2D ABC contraint par les longueurs de ses trois côtés. Il est invariant par le groupe **D** des déplacements (entre autres).

Étant données des coordonnées du point A, il y a une infinité de solutions, toutes en rotation libre autour de A. En matière de transformations, cela vient du fait que les déplacements n'agissent pas librement sur un point.

Si en revanche on donne les coordonnées du point A et la direction de la droite (AB), seules deux figures satisfont ces paramètres, chaque figure étant le symétrique de l'autre par rapport à (AB). Dans chacune des deux orbites de $\mathcal{F}(\mathcal{S})/\mathbf{D}$, **D** agit librement sur les figures.

Définition 40. Repère : *Un GCS R tel que G agit librement sur les éléments de $\mathcal{F}(R)/G$ est un repère pour le groupe G, ou G-repère.*

Par abus de langage, on appelle aussi repère pour G un type de systèmes qui sont des repères pour G. Ainsi, on dit qu'un point et une direction sont un repère pour les déplacements, car tout système consistant en un point et une direction est un **D**-repère.

Notons que la fixation d'un G-repère n'assure pas l'unicité des orbites des figures solutions pour G. Ainsi, par exemple, un système admettant des symétries aura plusieurs orbites pour les déplacements. D'autres cas sont possibles [Hen92] de par le fait que les opérations de construction géométriques (intersection de cercles, par exemple) sont des multi-fonctions.

3.1.4 Constriction *modulo* un groupe de transformations

En CAO, l'hypothèse est généralement faite que les GCS à résoudre sont rigides. La plupart des algorithmes de résolution ne traitent donc que ce type de systèmes et échouent pour les autres. Les systèmes rigides sont donc considérés comme bien contraints. Toutefois, dans la mesure où ils sont invariants par déplacement, une infinité de solutions $\mathcal{F}$ peut être construite en appliquant des déplacements à une solution particulière $f : \mathcal{F} = \mathbf{D}.f$. En considérant l'invariance par des groupes de transformations, il est possible de proposer d'autres définitions des niveaux de constriction pour résoudre ce paradoxe des systèmes bien contraints ayant une infinité de solutions.

Définition 41. Constriction *modulo* : *Un GCS $\mathcal{S}$ est bien contraint* modulo *un groupe G, ou G-bien-contraint, si $|\mathcal{F}(\mathcal{S})/G|$ est fini non nul.*

La définition de la sous-constriction peut également être étendue en considérant comme G-sous-contraint un système dont $|\mathcal{F}(\mathcal{S})/G|$ est infini. En revanche, la définition de la sur-constriction ne dépend pas d'un groupe : un système est sur-contraint quand il n'a pas de solutions, quel que soit le groupe considéré.

On peut en revanche définir la sous-G-définition et la sur-G-définition, comme suit :

Définition 42. Sous-G-définition : *Un système $\mathcal{S}$ est dit sous-G-défini s'il est G-invariant et que pour tout couple de figures f_1 et f_2 solutions de $\mathcal{S}$ tel qu'il existe $\varphi \in G, \varphi(f_1) = f_2$, il existe une infinité de transformations $\varphi' \in G$ telles que $\varphi'(f_1) = f_2$.*

Exemple 17. GCS sous-G-défini

Un système sous-G-défini est un système G-invariant qui est plus petit qu'un G-repère. Par exemple, un système composé d'un point incident à une droite est invariant par similitude, mais il existe une infinité de similitudes permettant de passer d'une solution de ce système à une autre.

Définition 43. Sur-G-définition : *Un système $\mathcal{S}$ est dit sur-G-défini s'il n'est ni sur-contraint ni G-invariant.*

Exemple 18. GCS sur-G-défini

Un système sur-G-défini est un système qui contient des contraintes non G-invariantes. Par exemple, un système bien contraint *modulo* les déplacements est sur-défini pour les similitudes.

Un GCS peut être bien contraint *modulo* plusieurs groupes. Dans l'exemple 3.1, le système est bien contraint *modulo* les déplacements (deux orbites) mais également *modulo* les isométries (une seule orbite).

La définition 41 amène quelques propriétés naturelles :

- pour un GCS $\mathcal{S} = (C, X, A)$ et $X' \subset X$, si $\mathcal{S}$ est G-bien-contraint alors $|\mathcal{F}(\mathcal{S})|_{X'}/G|$ est fini
- si un GCS $\mathcal{S}$ est à la fois G_1-bien-contraint et G_2-bien-contraint, alors $\mathcal{S}$ est G-bien-contraint, avec $G = G_1 \cap G_2$ et $|\mathcal{F}(\mathcal{S})/G| \leq |\mathcal{F}(\mathcal{S})/G_1| \times |\mathcal{F}(\mathcal{S})/G_2|$

Notons que pour n'importe quel système $\mathcal{S}$ il existe toujours un groupe G tel que $\mathcal{S}$ soit G-bien-contraint : le groupe des permutations de l'ensemble des solutions.

Bien évidemment, exprimer ce groupe revient à connaître toutes les solutions de $\mathcal{S}$ et n'a donc aucune utilité pour la résolution ou la décomposition de $\mathcal{S}$.

La décomposition d'un système G-bien-contraint est basée sur le théorème 2. Considérons un GCS G-bien-contraint $\mathcal{S}$. Supposons que l'on peut trouver des solutions d'un sous-système G_1-bien-contraint $\mathcal{S}_1 \subset \mathcal{S}$ tel que $G_1 \subseteq G$.

Le système résultant est $\mathcal{S}_2 = \mathcal{S} - \mathcal{S}_1 + \mathcal{B}(\mathcal{S}_1)$. Il est G-bien-contraint puisqu'il a les mêmes propriétés que $\mathcal{S}$ de par l'ajout du bord. En posant que $\mathcal{F}_{r_1}$ et $\mathcal{F}_{r_2}$ sont respectivement des ensembles de représentants de $\mathcal{F}(\mathcal{S}_1)/G_1$ et de $\mathcal{F}(\mathcal{S}_2)/G$, le théorème 2 nous indique que $\mathcal{F}(\mathcal{S}) = G.\mathcal{F}$ avec

$$\mathcal{F} = \bigcup_{(f_{r_1}, f_{r_2}) \in \mathcal{F}_{r_1} \times \mathcal{F}_{r_2}} f_{r_1} \otimes_{G_1} f_{r_2}$$

Étant donnés deux représentants $f_{r_1} \in \mathcal{F}_{r_1}$ et $f_{r_2} \in \mathcal{F}_{r_2}$ avec X_e les inconnues communes à f_{r_1} et à f_{r_2}, $f_{r_1} \otimes_{G_1} f_{r_2}$ est soit vide soit isomorphe à $G_1^{f_{r_1}|_{X_e}}/G$. Comme $\mathcal{S}$ est G-bien-contraint, $G_1^{f_{r_1}|_{X_e}}/G$ est fini.

Deux cas sont à considérer, selon que $G_1^{f_{r_1}|_{X_e}}$ est fini ou non. Si $\mathcal{S}_1$ et $\mathcal{S}_2$ partagent un repère, étant donnée la définition d'un repère, $G_1^{f_{r_1}|_{X_e}}$ est fini. S'ils partagent « moins » qu'un repère, c'est-à-dire un sous-système sur lequel le groupe considéré n'agit pas librement, $G_1^{f_{r_1}|_{X_e}}$ peut être infini, c'est-à-dire qu'un nombre infini de transformations peuvent être appliquées pour effectuer la jointure de f_{r_1} et f_{r_2}. Mais dans ce cas, puisque $\mathcal{S}$ est G-bien-contraint, $G_1^{f_{r_1}|_{X_e}}/G$ est fini et $X_2 = X_e$.

Théorème 3. Bord d'un sous-système : *Soit $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$ un GCS G-bien-contraint, avec $\mathcal{S}_1$ un sous-système G_1-bien-contraint, $G \subseteq G_1$. G_1 agit finiment transitivement sur $\mathcal{B}_{\mathcal{S}_2}(\mathcal{S}_1)$*

Démonstration:

Nous raisonnons par l'absurde. Supposons que G_1 n'agisse pas finiment transitivement sur $\mathcal{B}_{\mathcal{S}_2}(\mathcal{S}_1)$: cela signifie qu'il existe une infinité d'orbites pour G_1 de jointures d'une figure solution de $\mathcal{S}_1$ avec une figure solution de $\mathcal{B}_{\mathcal{S}_2}(\mathcal{S}_1)$. Par définition du bord, il existe donc une infinité d'orbites pour G_1 de jointures d'une figure solution de $\mathcal{S}_1$ avec une figure solution de $\mathcal{S}_2$.

Soient f_1 une solution de $\mathcal{S}_1$ et f_2 une solution de $\mathcal{S}_2$. L'ensemble des figures obtenues par jointure de f_1 et de f_2 est sous-contraint *modulo* G_1. Comme $G \subseteq G_1$, il est aussi sous-contraint *modulo* G. Puisque cet ensemble est un sous-ensemble des solutions de $\mathcal{S}$, $\mathcal{S}$ est sous-contraint *modulo* $\mathcal{S}$. □

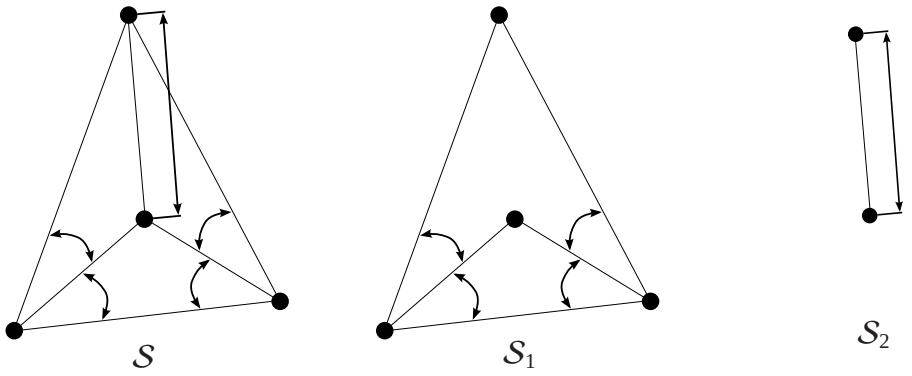


Figure 3.3 - Décomposition multi-groupe : $\mathcal{S}$ est rigide ; $\mathcal{S}_1 \subset \mathcal{S}$ est **S**-bien-contraint ; $\mathcal{S}_2 = \mathcal{S} - \mathcal{S}_1$ est rigide

Exemple 19. Décomposition par similitude et déplacement

Considérons un exemple où deux sous-systèmes partagent un repère à la fois pour les similitudes et les déplacements, comme celui de la figure 3.3, extraite de [SM06]. Ce système est **D**-bien-contraint. Si le sous-système $\mathcal{S}_1$ est la figure 3.3b, $\mathcal{F}(\mathcal{S}_1)$ peut être $\mathbf{S}.\mathcal{F}_{r_1}$ où **S** est le groupe des similitudes et $\mathcal{F}_{r_1}$ un ensemble contenant un unique représentant ($|\mathcal{F}(\mathcal{S}_1)/\mathbf{S}| = 1$). Si f_{r_1} est une solution de $\mathcal{S}_1$ et f_2 une solution de $\mathcal{S}_2$, il existe une unique similitude φ telle que $\varphi(f_{r_2}|_{X_e}) = f_{r_1}|_{X_e}$ où X_e sont les points communs.

Considérons ensuite un exemple où les entités géométriques communes ne forment pas un repère : soit un système $\mathcal{S}_1$ **D**-bien-contraint et un système $\mathcal{S}_2$ G-bien-contraint avec G le groupe des rotations autour du point P avec $X_e = \{P\}$. Le système global $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$ est G-bien-contraint. Étant donné deux représentants f_{r_1} pour $\mathcal{S}_1$ et f_{r_2} pour $\mathcal{S}_2$, il y a une infinité de déplacements φ tels que $\varphi(f_{r_2})|_{X_e} = f_{r_1}|_{X_e}$. Toutefois, toutes les figures obtenues par jointures sont équivalentes *modulo* G à condition que les inconnues de $\mathcal{S}_2$ se limitent au point P, sinon il y a une articulation libre de f_{r_1} et f_{r_2} autour de P et $\mathcal{S}$ n'est pas G-bien-contraint.

Dans le cas bien contraint, joindre des sous-systèmes implique le calcul d'un nombre fini de transformations. Ce calcul est immédiat à condition que les deux sous-systèmes partagent un repère et que, pour chaque groupe et pour chaque type de repère, la transformation symbolique d'un repère en un autre soit connue. Si le sous-système commun est moins qu'un repère, une transformation aléatoire parmi l'infinité des transformations possibles peut être choisie.

Avec l'ensemble des notions définies ci-dessus, nous pouvons étendre la notion de décomposition à une décomposition multi-groupe. Généralement, dans le domaine

de la résolution de GCS, les décompositions sont centrées sur les déplacements, c'est-à-dire que l'on recherche des sous-systèmes rigides. Avec une approche multi-groupe, la définition 29 peut être étendue pour que les déplacements ne soient pas explicitement considérés.

Définition 44. Décomposition multi-groupe : *Étant donné un ensemble $\mathcal{G}$ de groupes de transformations, une $\mathcal{G}$-décomposition d'un GCS $\mathcal{S}$ est une décomposition de $\mathcal{S}$ en systèmes $\mathcal{S}_1, \ldots, \mathcal{S}_n$ telle que chaque $\mathcal{S}_i, i \in [1, n]$ soit G_i-invariant, avec $G_i \in \mathcal{G}$.*

Cette définition ajoute une condition sémantique (invariance) à la définition syntaxique de la décomposition. Dans un univers géométrique donné, il y a de nombreuses décompositions possibles d'un système. En matière de complexité de résolution elles ne sont pas équivalentes mais, par définition, mènent toutes aux mêmes solutions.

3.1.5 Groupes considérés

Dans le reste du présent mémoire, nous considérons sauf mention contraire les groupes obtenus à partir des rotations autour d'un point, des translations et des homothéties. Ces groupes induisent une structure de treillis : la composition de n'importe quel couple de groupes amène un sur-groupe engendré. Pour des raisons applicatives, en l'occurence le fait que certaines des compositions ne nous paraissent pas intéressantes pour la résolution de GCS, nous nous limitons seulement à certains des groupes. La figure 3.4 montre les groupes considérés et leurs inclusions.

3.2 Conditions de correction et complétude de la décomposition

Comme pour toute méthode de résolution de problème, les questions de correction et de complétude se posent :

1. correction : les figures obtenues par décomposition et assemblage des figures sont-elles des solutions du système global ?
2. complétude : les solutions du système global sont-elles toutes obtenues par décomposition et assemblage ?

Nous allons démontrer que la soustraction d'un sous-système ne modifie pas les solutions du système global si on lui ajoute le bord du système soustrait. Autrement

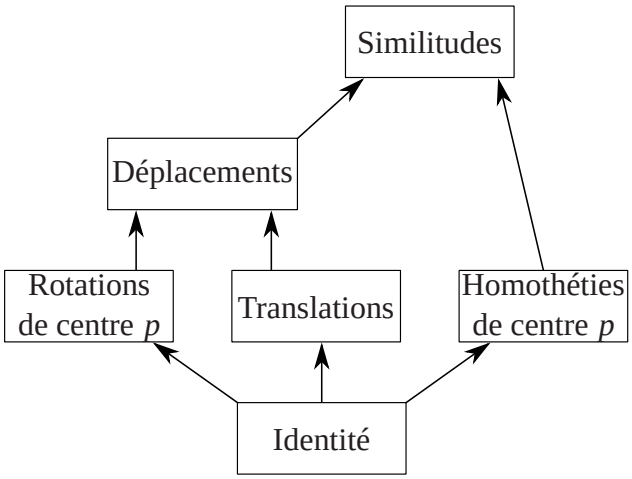


Figure 3.4 - Treillis des groupes de transformations considérés dans notre implantation ; les arcs indiquent l'inclusion

dit, si on a $\mathcal{S}$ le GCS global, $\mathcal{S}_1$ un sous-système de $\mathcal{S}$ et $\mathcal{S}' = \mathcal{S} - \mathcal{S}_1$, nous allons montrer que $\mathcal{F}(\mathcal{S}' + \mathcal{B}_{\mathcal{S}'}(\mathcal{S}_1)) = \mathcal{F}(\mathcal{S})|_{unknowns(\mathcal{S}')}$.

Cette démonstration avait déjà été faite par M et S [Mat97] pour le cas particulier de la décomposition d'un GCS rigide en sous-systèmes rigides. Dans le cadre de notre travail, il était important de déterminer les conditions de correction et de complétude des méthodes de décomposition dans le cas général, afin d'assurer l'homogénéité de traitement de nos algorithmes, que nous voulons capables d'agir sur n'importe quel GCS non sur-contraint.

Notre démonstration est donc indépendante de l'univers géométrique considéré[2]. Elle s'applique aussi bien aux méthodes de décomposition classiques qu'aux méthodes multi-groupes.

Pour arriver à notre résultat, nous avons besoin de plusieurs théorèmes intermédiaires :
- il nous faut montrer que l'ensemble des solutions du sous-système engendré par un sous-ensemble X des inconnues contient l'ensemble des figures solutions du système global restreintes à X (théorème 4) ;
- il nous faut montrer que l'ensemble des solutions d'une addition de deux systèmes est la jointure des ensembles des solutions de ces systèmes (théorème 5) ;
- il nous faut enfin déterminer les conditions de préservation de la jointure d'ensembles de figures par la restriction (théorème 7), ce pour quoi nous aurons besoin de prouver la préservation de la jointure de deux figures par restriction (théorème 6).

[2] À la condition près de possibilité d'expression du bord dans l'univers géométrique

L'ensemble de ces théorèmes nous permet de démontrer qu'en retirant un sous-système puis en rajoutant son bord, les solutions ne changent pas et donc qu'à condition de pouvoir calculer et exprimer le bord d'un système, les méthodes de décomposition sont correctes et complètes.

3.2.1 Lien entre solutions d'un sous-système et sous-figures solutions

Nous commençons par montrer que la restriction d'un ensemble de figures à un sous-ensemble des inconnues n'est pas le pendant dans $\mathcal{E}$ du système engendré par ce sous-ensemble. Autrement dit, pour un système $\mathcal{S} = (C, X, A)$ et son sous-système $\mathcal{S}' = (C', X', A')$ engendré par X' (avec donc $X' \subset X$), on a $\mathcal{F}(\mathcal{S})|_{X'} \subseteq \mathcal{F}(\mathcal{S}')$. En effet, les figures de $\mathcal{F}(\mathcal{S})|_{X'}$ respectent toutes les contraintes de C' – et font donc partie de $\mathcal{F}(\mathcal{S}')$ – mais respectent également les contraintes de $C \backslash C'$ qui concernent des entités géométriques de X'.

Théorème 4. Lien entre solutions d'un sous-système et sous-figures solutions : *Soit $\mathcal{S} = (C, X, A)$ un GCS et $X' \subset X$ un sous-ensemble des inconnues. On pose $\mathcal{S}' = (C', X', A')$ le système engendré par X'. On a $\mathcal{F}(\mathcal{S})|_{X'} \subseteq \mathcal{F}(\mathcal{S}')$ mais pas nécessairement $\mathcal{F}(\mathcal{S}') \subseteq \mathcal{F}(\mathcal{S})|_{X'}$.*

Démonstration:

Montrons que $\mathcal{F}(\mathcal{S})|_{X'} \subseteq \mathcal{F}(\mathcal{S}')$. Soit f une solution de $\mathcal{S}$. Comme $C' \subset C$, toutes les contraintes de C' sont satisfaites dans f et donc *a fortiori* dans $f|_{X'}$. On a donc $\mathcal{F}(\mathcal{S})|_{X'} \subseteq \mathcal{F}(\mathcal{S}')$.

Montrons que l'inclusion réciproque n'est pas forcément vraie. Les contraintes de C' sont satisfaites par toutes les figures de $\mathcal{F}(\mathcal{S}')$. Si l'on considère une contrainte $c \notin C'$, elle ne sera satisfaite par toutes les figures de $\mathcal{F}(\mathcal{S}')$ que si c est redondante avec C', c'est-à-dire qu'à partir d'une preuve de satisfaction des contraintes de C' on peut apporter la preuve de c. Si c n'est pas redondante avec C', alors il existe des figures qui satisfont toutes les contraintes de C' et qui ne satisfont pas c. Autrement dit, il existe des figures de $\mathcal{F}(\mathcal{S}')$ qui ne sont pas des solutions du système obtenu en ajoutant c à $\mathcal{S}'$. Or $C' \subset C$. Si les contraintes de $C \backslash C'$ ne sont pas toutes redondantes avec C', on a montré ce qu'il fallait démontrer. Exhiber un système de plus d'une contrainte dont toutes les contraintes ne sont pas linéairement dépendantes est aisé. □

Le lecteur pourra se référer à l'exemple 3[3] pour un exemple des implications de ce théorème.

[3] *Cf.* p. 22

3.2.2 Lien entre addition et jointure

Nous montrons ensuite le lien étroit entre l'addition de GCS et la jointure des solutions de ces GCS : l'ensemble des solutions d'un GCS est la jointure des solutions de ses sous-systèmes.

Théorème 5. Lien entre addition et jointure : *Soient $\mathcal{S}_1$ et $\mathcal{S}_2$ deux GCS construits sur la même signature.* $\mathcal{F}(\mathcal{S}_1 + \mathcal{S}_2) = \mathcal{F}(\mathcal{S}_1) \otimes \mathcal{F}(\mathcal{S}_2)$.

Démonstration:

Soient $\mathcal{S} = (C, X, A)$, $\mathcal{S}_1 = (C_1, X_1, A_1)$ et $\mathcal{S}_2 = (C_2, X_2, A_2)$ trois GCS tels que $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$.
Démontrons que $\mathcal{F}(\mathcal{S}) \subseteq \mathcal{F}(\mathcal{S}_1) \otimes \mathcal{F}(\mathcal{S}_2)$ et $\mathcal{F}(\mathcal{S}_1) \otimes \mathcal{F}(\mathcal{S}_2) \subseteq \mathcal{F}(\mathcal{S})$

1. $\mathcal{F}(\mathcal{S}) \subseteq \mathcal{F}(\mathcal{S}_1) \otimes \mathcal{F}(\mathcal{S}_2)$
   Une figure $f \in \mathcal{F}(\mathcal{S})$ peut s'exprimer comme $f|_{X_1} \otimes f|_{X_2}$. D'après le théorème 4, on a $f|_{X_1} \in \mathcal{F}(\mathcal{S}_1)$ puisque $\mathcal{F}(\mathcal{S})|_{X_1} \subseteq \mathcal{F}(\mathcal{S}_1)$ et $f|_{X_1} \in \mathcal{F}(\mathcal{S})|_{X_1}$. Similairement, $f|_{X_2} \in \mathcal{F}(\mathcal{S}_2)$. On a donc $\mathcal{F}(\mathcal{S}) \subseteq \mathcal{F}(\mathcal{S}_1) \otimes \mathcal{F}(\mathcal{S}_2)$.
2. $\mathcal{F}(\mathcal{S}_1) \otimes \mathcal{F}(\mathcal{S}_2) \subseteq \mathcal{F}(\mathcal{S})$
   Considérons une figure $f \in \mathcal{F}(\mathcal{S}_1) \otimes \mathcal{F}(\mathcal{S}_2)$. Elle est définie pour tous les éléments de X. Admettons que $f \notin \mathcal{F}(\mathcal{S})$, c'est-à-dire qu'il existe une contrainte $c \in C$ telle que l'interprétation de c dans $\mathcal{E}$ n'est pas satisfaite avec les valeurs associées aux entités géométriques par f. Comme $C = C_1 \cup C_2$, on a soit $c \in C_1$ soit $c \in C_2$ et donc c apparaît dans $\mathcal{S}_1$ ou dans $\mathcal{S}_2$. Comme $f \in \mathcal{F}(\mathcal{S}_1) \otimes \mathcal{F}(\mathcal{S}_2)$, les interprétations des contraintes de C_1 et C_2 sont satisfaites pour les valeurs de f. La contrainte c est donc satisfaite. Il y a contradiction, l'hypothèse $f \notin \mathcal{F}(\mathcal{S})$ est donc fausse : on a $f \in \mathcal{F}(\mathcal{S})$ et donc $\mathcal{F}(\mathcal{S}_1) \otimes \mathcal{F}(\mathcal{S}_2) \subseteq \mathcal{F}(\mathcal{S})$.

On a l'inclusion mutuelle : $\mathcal{F}(\mathcal{S}) \subseteq \mathcal{F}(\mathcal{S}_1) \otimes \mathcal{F}(\mathcal{S}_2)$ et $\mathcal{F}(\mathcal{S}_1) \otimes \mathcal{F}(\mathcal{S}_2) \subseteq \mathcal{F}(\mathcal{S})$. On a donc $\mathcal{F}(\mathcal{S}_1 + \mathcal{S}_2) = \mathcal{F}(\mathcal{S}_1) \otimes \mathcal{F}(\mathcal{S}_2)$. □

Le lecteur pourra se référer à l'exemple 4[4] pour un exemple des implications de ce théorème.

3.2.3 Préservation de la jointure de deux figures par la restriction

Nous montrons maintenant que la restriction préserve l'opération de jointure pour deux figures, c'est-à-dire que la jointure de deux figures restreintes est égale à la restriction de la jointure des deux figures.

[4] *Cf.* p. 25

Théorème 6. Préservation de la jointure de deux figures par la restriction : *Si* $f = f_1 \otimes f_2$*, avec* $f_1 : X_1 \to \mathcal{E}$ *et* $f_2 : X_2 \to \mathcal{E}$*, alors pour tout sous-ensemble* X' *de* $X_1 \cup X_2$*,* $f|_{X'} = f_1|_{X'} \otimes f_2|_{X'}$*.*

Démonstration:

Par définition de la jointure, on a $f : X_1 \cup X_2 \to \mathcal{E}$ avec $f(x) = f_1(x)$ si $x \in X_1$ et $f(x) = f_2(x)$ si $x \in X_2$. La restriction de f à X' est donc :

$$f|_{X'} : \begin{array}{l} X' \to \mathcal{E} \\ x \to \left\{ \begin{array}{l} f_1(x) \text{ si } x \in X_1 \cap X' \\ f_2(x) \text{ si } x \in X_2 \cap X' \end{array} \right. \end{array}$$

De même, par définition de la restriction, on a $\forall i \in \{1, 2\}$, $f_i|_{X'} : X_i \cap X' \to \mathcal{E}$ et $f_i|_{X'}(x) = f_1(x)$.

$$f_1|_{X'} \otimes f_2|_{X'} : \begin{array}{l} (X_1 \cap X') \cup (X_2 \cap X') \to \mathcal{E} \\ x \qquad\qquad \to \left\{ \begin{array}{l} f_1(x) \text{ si } x \in X_1 \cap X' \\ f_2(x) \text{ si } x \in X_2 \cap X' \end{array} \right. \end{array}$$

Or $(X_1 \cap X') \cup (X_2 \cap X') = (X_1 \cup X_2) \cap X'$. Comme $X' \subseteq (X_1 \cup X_2)$, le domaine de définition de $f_1|_{X'} \otimes f_2|_{X'}$ est X' et on a $f_1|_{X'} \otimes f_2|_{X'} = (f_1 \otimes f_2)|_{X'}$. □

3.2.4 Conditions de préservation de la jointure d'ensembles de figures par la restriction

En revanche, la restriction ne préserve pas forcément l'opération de jointure pour des ensembles de figures. Nous montrons les conditions de préservation de la jointure d'ensembles de figures par la restriction, à savoir le fait que l'intersection des ensembles d'inconnues des deux systèmes soit incluse dans le sous-ensemble de restriction.

Théorème 7. Conditions de préservation de la jointure d'ensembles de figures par la restriction : *Soient* $\mathcal{S}_1 = (C_1, X_1, A_1)$ *et* $\mathcal{S}_2 = (C_2, X_2, A_2)$ *deux GCS construits sur la même signature. On pose* $X_e = X_1 \cap X_2$ *et* $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$*. Pour tout ensemble* $X' \subseteq (X_1 \cup X_2)$*,* $\mathcal{F}(\mathcal{S})|_{X'} = \mathcal{F}(\mathcal{S}_1)|_{X'} \otimes \mathcal{F}(\mathcal{S}_2)|_{X'}$ *si et seulement si* $X_e \subseteq X'$*.*

Démonstration:

Montrons l'égalité de $\mathcal{F}(\mathcal{S})|_{X'}$ et de $\mathcal{F}(\mathcal{S}_1)|_{X'} \otimes \mathcal{F}(\mathcal{S}_2)|_{X'}$ par inclusion mutuelle.

1. $\mathcal{F}(\mathcal{S})|_{X'} \subseteq \mathcal{F}(\mathcal{S}_1)|_{X'} \otimes \mathcal{F}(\mathcal{S}_2)|_{X'}$
   Soit $f \in \mathcal{F}(\mathcal{S})$. On peut décomposer f en $f_1 \otimes f_2$ avec $f_1 \in \mathcal{F}(\mathcal{S})|_{X_1}$ et $f_2 \in \mathcal{F}(\mathcal{S})|_{X_2}$. Le théorème 4 nous dit que $f_1 \in \mathcal{F}(\mathcal{S}_1)$ et $f_2 \in \mathcal{F}(\mathcal{S}_2)$. Par définition, on a $f_1|_{X'} \in \mathcal{F}(\mathcal{S}_1)|_{X'}$ et $f_2|_{X'} \in \mathcal{F}(\mathcal{S}_2)|_{X'}$. Le théorème 6 nous indique que $f|_{X'} = f_1|_{X'} \otimes f_2|_{X'}$, on peut en déduire que $\mathcal{F}(\mathcal{S}_1 + \mathcal{S}_2)|_{X'} \subseteq \mathcal{F}(\mathcal{S}_1)|_{X'} \otimes \mathcal{F}(\mathcal{S}_2)|_{X'}$. Notons que cette relation est valide même si $X' \not\subseteq X_e$;
2. $\mathcal{F}(\mathcal{S}_1)|_{X'} \otimes \mathcal{F}(\mathcal{S}_2)|_{X'} \subseteq \mathcal{F}(\mathcal{S})|_{X'}$
   La démonstration vient de la définition de la jointure. Soit $f = f_1 \otimes f_2$, avec $f_1 \in \mathcal{F}(\mathcal{S}_1)|_{X'}$ et $f_2 \in \mathcal{F}(\mathcal{S}_2)|_{X'}$. Par définition, $f \in \mathcal{F}(\mathcal{S}_1)|_{X'} \otimes \mathcal{F}(\mathcal{S}_2)|_{X'}$. La figure f n'appartient à $\mathcal{F}(\mathcal{S}_1 + \mathcal{S}_2)|_{X'}$ que si $\mathcal{F}(\mathcal{S}_1)|_{X'}$ et $\mathcal{F}(\mathcal{S}_2)|_{X'}$ respectent les contraintes de X_e. Pour cela, il faut que le test de compatibilité des figures ait inclus les entités géométriques concernées par les contraintes de X_e, ce qui est le cas quand et seulement quand $X_e \subseteq X'$.

Il y a donc inclusion mutuelle si et seulement si $X_e \subseteq X'$. □

Le lecteur pourra se référer à l'exemple 5[5] pour un exemple des implications de ce théorème.

3.2.5 Correction de l'assemblage de figures

Nous pouvons maintenant en arriver au résultat central de cette section, essentiel pour la décomposition. Il montre que retirer un sous-système $\mathcal{S}_1$ d'un GCS $\mathcal{S}$ ne change pas les solutions du système résultant si le bord de $\mathcal{S}_1$ (dont l'existence a été prouvé au théorème 1) est ajouté. En d'autres termes, il prouve la validité des méthodes de décomposition ascendantes : si les solveurs des sous-systèmes sont corrects et complets (*i.e.* ne fournissent que des figures qui satisfont les contraintes et les fournissent toutes) alors l'assemblage des sous-figures fournira toutes les solutions valides et uniquement des solutions.

Théorème 8. Correction et complétude de l'assemblage de figures : *Soit $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$ un GCS avec $\mathcal{S}_2 = (C_2, X_2, A_2)$. La restriction de $\mathcal{F}(\mathcal{S})$ aux variables de $\mathcal{S}_2$ est l'ensemble des solutions du système obtenu en ajoutant le bord de $\mathcal{S}_1$ à $\mathcal{S}_2$:*

$$\mathcal{F}(\mathcal{S})|_{X_2} = \mathcal{F}(\mathcal{S}_2 + \mathcal{B}(\mathcal{S}_1))$$

Démonstration:

[5] *Cf.* p. 26

Soient $\mathcal{S} = (C, X, A)$, $\mathcal{S}_1 = (C_1, X_1, A_1)$ et $\mathcal{S}_2 = (C_2, X_2, A_2)$ trois GCS avec $\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$. Les variables de bord de $\mathcal{S}_1$ apparaissent dans les contraintes de $\mathcal{S}_1$ et les contraintes de $\mathcal{S}_2$, *i.e.* elles constituent l'ensemble $X_e = X_1 \cap X_2$. On sait grâce au théorème 4 que la relation $\mathcal{F}(\mathcal{S})|_{X_2} \subseteq \mathcal{F}(\mathcal{S}_2)$ n'est pas symétrique de par le fait que certaines contraintes qui concernent les inconnues de X_e peuvent être présentes dans C_1. Ainsi, soustraire $\mathcal{S}_1$ de $\mathcal{S}$ peut retirer des contraintes agissant sur une partie de X_2.

Le système de bord de $\mathcal{S}_1$ par rapport à $\mathcal{S}_2$ est le GCS tel que $\mathcal{F}(\mathcal{B}(\mathcal{S}_1)) = \mathcal{F}(\mathcal{S}_1)|_{X_e}$. Comme $X_e \subseteq X_2$, on peut déduire des théorèmes 5 et 7 que

$$\begin{aligned}\mathcal{F}(\mathcal{S})|_{X_2} &= \mathcal{F}(\mathcal{S}_1)|_{X_2} \otimes \mathcal{F}(\mathcal{S}_2)|_{X_2} \\ &= \mathcal{F}(\mathcal{S}_1)|_{X_e} \otimes \mathcal{F}(\mathcal{S}_2) \\ &= \mathcal{F}(\mathcal{B}(\mathcal{S}_1) + \mathcal{S}_2)\end{aligned}$$

□

Le lecteur pourra se référer à l'exemple 12[6] pour un exemple des implications de ce théorème.

Ce théorème nous permet de voir qu'en retirant le système $\mathcal{S}_2$ de $\mathcal{S}$ et en rajoutant le bord de $\mathcal{S}_2$ à $\mathcal{S}$, on n'a pas modifié les solutions de $\mathcal{S} - \mathcal{S}_2$. On peut donc résoudre les deux systèmes séparément et les assembler, le théorème 7 nous assurant que les deux systèmes sont compatibles.

Ce théorème est basé sur une définition sémantique du bord. Il n'est donc vérifié que si la signature considérée permet d'exprimer l'ensemble des contraintes du bord. Par exemple, le bord du système représenté à la figure 1.3 par rapport au reste du système dont il est extrait contient une inégalité puisque l'on sait que la distance entre p_2 et p_4 doit être inférieure ou égale à la somme des distances p_1–p_2 et p_1–p_4. La signature de l'univers géométrique de l'exemple 1 ne permet pas d'exprimer cette contrainte et on ne peut donc pas, simplement avec le théorème 8, prouver que décomposer le système de la figure 1.2 en commençant par soustraire le système de la figure 1.3 mènera à un résultat correct.

Le théorème 8 peut être affaibli en montrant qu'il n'est pas nécessaire d'ajouter l'ensemble du système de bord dans le cas où celui-ci est redondant. Il suffit alors d'ajouter une base du bord, c'est-à-dire un sous-ensemble minimal du bord tel que toutes les autres contraintes sont redondantes.

Ce faisant, il est possible de montrer la correction et la complétude d'un algorithme de résolution donné. En effet, pour le cas de systèmes **D**-bien-contraints par exemple, il est possible de montrer que la connaissance de l'ensemble des distances entre deux points et des angles entre deux droites est suffisant pour caractériser les

[6] *Cf.* p. 35

solutions d'un système. Autrement dit, si l'on sait calculer ces informations, toutes les autres informations du bord sont redondantes. Il suffit donc de savoir calculer une base de cet ensemble. On peut montrer, de même, que l'ensemble des angles et des bi-rapports de distance entre deux points est suffisant à caractériser les solutions d'un système **S**-bien-contraint.

3.3 Démonstration de la correction de la méthode d'Owen

Rappelons[7] que la méthode d'O [Owe91] est une méthode de décomposition descendante d'un GCS génériquement rigide utilisant une analyse du graphe de contrainte. Son principe est de rechercher une paire d'articulation[8] et de séparer le graphe en deux au niveau de cette paire d'articulation. L'un des deux sous-graphes, que nous appelons g_1, a désormais un nœud d'articulation. L'autre sous-graphe, g_2, correspond à un sous-système rigide. Si g_2 n'est pas un graphe triconnexe, O opère une récursion sur g_2. Puis une arête, appelée lien virtuel[9] est ajoutée entre les deux nœuds de la paire d'articulation dans g_1 et on opère récursivement sur g_1. Les systèmes triconnexes doivent être résolus et leurs solutions sont assemblées.

La méthode d'O fonctionne dans un univers géométrique 2D dont les sortes sont les points et les droites et les contraintes sont des distances entre points, des angles entre droites et des incidences point-droite. Elle ne fonctionne que sur des systèmes dont le graphe de contrainte n'est pas triconnexe : ceux-ci doivent être résolus au moyen d'un solveur extérieur. Faisons ici l'hypothèse que l'on dispose d'un solveur correct et complet pour ces systèmes et montrons que la décomposition d'O est alors correcte et complète.

Il est trivial de montrer que si le graphe admet un nœud d'articulation ou est déconnecté, le GCS correspondant n'est pas rigide : dans le cas déconnecté, il est évident que le GCS est coupé en deux parties qui ne partagent pas un repère pour les déplacements ; dans cas du nœud d'articulation, on montre que quelle que soit l'entité géométrique correspondant au nœud d'articulation, les deux sous-GCS partageant cette entité ne partagent pas un repère pour les déplacements. L'ensemble n'est donc pas rigide.

On sait donc que le graphe est un graphe biconnexe. Considérons maintenant une paire d'articulation en deux nœuds n_1 et n_2 et décomposons le GCS en les deux sous-systèmes situés de part et d'autre des entités géométriques correspondant à ces

[7] *Cf.* chapitre 2

[8] *I.e.* une paire de nœuds du graphe telle que si on retire ces nœuds du graphe ainsi que leurs arêtes incidentes, le graphe résultant a 2 composantes connexes

[9] *Angl. :* virtual bond

nœuds. S'il y a une arête entre n_1 et n_2, on décompose en n'incluant la contrainte correspondante que dans un seul des sous-systèmes.

Considérons maintenant que l'un des deux sous-systèmes est rigide. Il ne peut s'agir, comme on l'a montré plus haut, que de celui dont le graphe n'a pas de nœud d'articulation, g_2. Il peut s'agir soit d'un système triconnexe, soit d'un système sur lequel on va opérer une récursion. Son bord par rapport au reste du système est l'ensemble des informations que l'on peut calculer, dans un système rigide, concernant les deux entités géométriques partagées. Trois cas sont possibles :
– les deux entités géométriques sont des points : une base du bord est la distance entre ces points ;
– les deux entités géométriques sont des droites : une base du bord est l'angle entre ces droites ;
– une entité est une droite, l'autre un point : une base de bord est la contrainte d'incidence ou la distance entre le point et la droite, selon le cas.
Dans les trois cas, une base du bord est faite d'une contrainte avec un degré de restriction, qui se traduit en terme de graphes par une arête entre les deux entités. L'ajout du lien virtuel correspond donc bien à l'ajout du bord. Les deux systèmes partagent un repère pour les déplacements et peuvent être assemblés.

Nous avons montré que la méthode de décomposition d'O revenait, en terme de graphe, à décomposer un système en deux sous-systèmes $\mathcal{S}_1$ et $\mathcal{S}_2$ dont l'un, $\mathcal{S}_1$, est **D**-bien-contraint et à ajouter le bord de $\mathcal{S}_1$ dans $\mathcal{S}_2$. Le théorème 8 nous permet donc d'assurer que cette méthode est correcte et complète, sous l'hypothèse de disposer de solveurs corrects et complets pour les systèmes triconnexes.

Notons que l'une des configurations possibles de paire d'articulation (un point et une droite) peut amener un bord contenant une contrainte non exprimable dans l'univers géométrique d'O (distance entre un point et une droite).

La figure 3.5 montre un exemple de GCS compatible avec l'univers géométrique d'O et où cette configuration apparaît. Le sous-système engendré par l'ensemble $\{p_1, p_3, p_4, l_1\}$ est rigide. Considérer la paire d'articulation constituée de p_1 et de l_1 mène donc à remplacer ce système par un lien virtuel entre p_1 et l_1, dont l'interprétation géométrique est la distance de p_1 à l_1.

Une seconde hypothèse pour la correction et la complétude de la méthode d'O est donc que ce type de configuration n'intervient jamais.

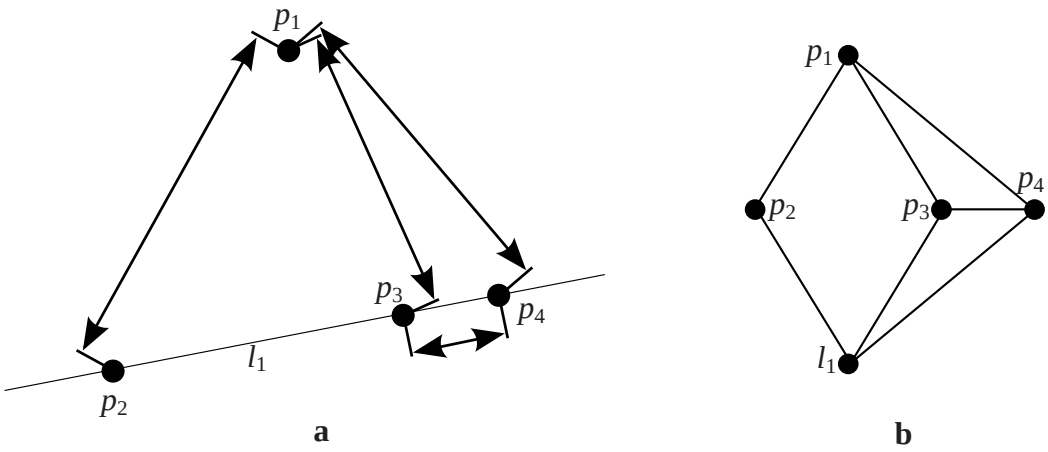


Figure 3.5 - Système de contrainte géométrique mettant en défaut la méthode d'O : esquisse (**a**) et graphe ((**b**)

B I

Nous avons formalisé, avec l'approche des spécifications algébriques, les systèmes de contraintes géométriques et leurs opérations, leurs niveaux de constriction, leur décomposition et leur résolution, ainsi que le point de vue multi-groupe. Nous avons réalisé un aperçu de la littérature concernant la résolution de systèmes de contraintes géométriques en mettant l'accent sur les méthodes traitant de systèmes sous-contraints. Nous avons montré quelles étaient les conditions de correction et de complétude des méthodes de décomposition et ainsi pu prouver que la méthode d'O était correcte et complète sous certaines hypothèses.

Bien sûr, la démonstration que nous avons effectué n'aboutira pas à la preuve de complétude d'un solveur de GCS d'une famille géométrique : seuls les solveurs algébriques peuvent être complets. Mais notre démonstration fournit les conditions suffisantes et nécessaires pour que la décomposition n'amène pas de facteur d'incomplétude supplémentaire.

Les développements récents dans le domaine de la preuve interactive ont montré qu'il pouvait être dangereux de se contenter de démonstrations non vérifiées par l'ordinateur. M et F [MF03] montrent par exemple que la formalisation de la géométrie par H n'est pas autant exempte d'hypothèses qu'H l'affirme et qu'il s'est probablement appuyé sur des figures pour raisonner, bien qu'il prétende le contraire. De même, H *et al.* [HHM+10] proposent une démonstration au moyen d'un assistant de preuves de la preuve de la conjecture de K ,

proposée 12 ans plus tôt par H sous la forme d'une démonstration mathématique d'environ 250 pages et publiée seulement en 2006 de par l'incapacité des rapporteurs à certifier la correction de la démonstration. Ils montrent notamment que l'utilisation d'un assistant de preuves a permis de déceler des inexactitudes dans la preuve originale.

Une suite logique des travaux présentés dans cette partie serait donc de formaliser les systèmes de contraintes géométriques dans un assistant de preuves et d'y effectuer les démonstrations des différents théorèmes menant à la démonstration des conditions de correction et de complétude des méthodes de décomposition à partir de la définition du bord.

En outre, il serait intéressant de reprendre, comme nous l'avons fait pour la méthode d'O , les différentes méthodes de décomposition de la littérature et de traduire leurs opérations dans notre formalisme afin de vérifier si elles sont correctes et complètes et, le cas échéant, de préciser sous quelles hypothèses elles le sont.

D `

P ´ `

´ ´

Nous avons vu dans la partie I quelles étaient les grandes approches de la résolution de Systèmes de Contraintes Géométriques (GCS). Nous avons notamment vu que la rigidité des objets conçus était souvent une hypothèse des algorithmes de résolution actuels. Dans cette partie, nous allons nous abstraire de cette hypothèse et étudier le cas spécifique et encore relativement peu traité des systèmes de contraintes géométriques sous-contraints.

Au chapitre 3 de la partie I, nous avons indiqué qu'il était inadéquat d'opposer rigide et sous-contraint dans la mesure où un système rigide a généralement une infinité de solutions de par son invariance par déplacement. Dans cette partie, nous nous intéressons aux systèmes sous-contraints dans leur ensemble, c'est-à-dire que nous cherchons à traiter de manière homogène des systèmes bien-contraints *modulo* des groupes de transformations globaux autres que l'identité et des GCS pour lesquels il n'existe pas de groupe de bonne constriction agissant globalement, c'est-à-dire intuitivement des systèmes articulés.

Motivation

Nous considérons, contrairement à ce qui a été fait jusqu'ici dans la littérature, que les solveurs devraient être capables de résoudre des GCS sous-contraints. Comme nous l'avons vu à la section 2.2, la plupart des articles traitant de la sous-constriction visent à la détecter, la considérant comme un cas d'erreur, ou à se ramener au cas rigide par un ajout de contraintes au GCS.

Pourtant, la résolution de GCS sous-contraints a des intérêts multiples : d'une part, pour la résolution d'un GCS rigide difficilement (voire non) décomposable, les solutions d'un sous-système sous-contraint peuvent être un point de départ pour un rattrapage numérique, de même qu'une décomposition en plusieurs systèmes sous-contraints (bien contraints *modulo* des groupes globaux ou non) peut augmenter la classe des systèmes rigides résolubles ; d'autre part, les GCS sous-contraints peuvent très bien être des objets finaux correspondant aux attentes de l'utilisateur : une paire de ciseaux, une lampe de bureau ou encore une voiture sont autant d'exemples de systèmes classiquement considérés comme sous-contraints même si, du point de vue de l'utilisateur, ils ont été contraints correctement.

Enfin – et cet argument a eu un impact déterminant sur l'orientation donnée à nos travaux – la capacité à résoudre des GCS sous-contraints est un impératif pour la mise au point de logiciels efficaces de Conception Assistée par Ordinateur (CAO) basés contraintes permettant aux utilisateurs de concevoir incrémentalement leur esquisse. Cette fonctionnalité, très peu envisagée jusqu'alors, est pourtant essentielle si l'on veut permettre à des utilisateurs non experts un accès aux logiciels de CAO

par contrainte : ceux-ci sont susceptibles de concevoir des GCS avec trop ou pas assez de contraintes par rapport à l'objet voulu et ont besoin de pouvoir procéder par essai/erreur en ayant un retour sur le niveau de constriction du système.

L'ensemble des algorithmes que nous proposons dans cette partie sont donc incrémentaux : ils fonctionnent sur la base de la mise-à-jour des données lors de l'ajout d'une entité géométrique ou d'une contrainte.

Démarche

Pour répondre aux besoins d'incrémentalité et de retour visuel sur le niveau de constriction, la démarche que nous abordons passe par la définition du schéma de résolution suivant :

1. trouver incrémentalement une paramétrisation du GCS,
2. trouver un plan de construction,
3. donner des valeurs aux paramètres et évaluer le plan de construction.

La paramétrisation du GCS évoquée à l'étape 1 est un repère pour le GCS : elle consiste à fournir, pour chaque entité géométrique du GCS, le nombre de degrés de liberté qu'il faut retirer afin que le GCS soit bien contraint *modulo* l'identité. Si le système est rigide, le résultat de l'algorithme de calcul d'une paramétrisation doit donc être un repère pour les déplacements.

Il existe un grand nombre de paramétrisations différentes pour un même système et toutes ne se valent pas : certaines mènent à des plans de construction susceptibles d'échouer selon les valeurs des paramètres alors que, pour le même système, d'autres permettent de trouver un plan de construction toujours valide.

L'étape 1 est abordée au chapitre 4, où nous étendons les travaux de L et M - en abstrayant le GCS sous forme d'un graphe biparti entités-contraintes et en calculant un flux traversant ce graphe. Les algorithmes que nous proposons partent de l'hypothèse que le GCS n'est pas génériquement sur-contraint.

Les étapes 2 et 3 sont abordées dans le chapitre 5, qui décrit comment interpréter géométriquement la paramétrisation combinatoire et comment déduire un plan de construction par blocs à partir de cette paramétrisation. Il explicite également l'interprétation géométrique du flux obtenu.

Le chapitre 6 traite du problème spécifique de la détection de la sur-constriction lors de l'ajout de contraintes ou lors de l'évaluation numérique de la paramétrisation : il montre que ces deux problèmes ne sont pas trivialement résolus en adaptant les

méthodes existantes, puis propose des extensions de la méthode du témoin [10] pour les résoudre.

[10] *Cf.* section 2.2.2 et [MF06, MFLM06, MF07, MF09]

C 4

P ´

Sommaire

Pour chaque problème, il y a une solution qui est simple, claire et fausse
– Henri Louis Mencken, journaliste américain

Dans ce chapitre, nous présentons un algorithme combinatoire, basé sur une méthode de flux, en extension des travaux de L et M [LM96b]. Il est conçu incrémentalement : à partir d'une paramétrisation d'un système de contraintes géométriques, une nouvelle paramétrisation est calculée en mettant à jour le flux, lorsqu'une contrainte ou une entité géométrique est ajoutée par l'utilisateur. Nous proposons également un algorithme – en réalité une modification mineure de l'algorithme utilisé à l'ajout d'une contrainte – permettant à l'utilisateur de changer de paramétrisation si celle qui lui est proposée ne lui convient pas[1].

[1] Par exemple parce qu'elle ne lui donne pas une bonne intuition des libertés du système

Nous avons vu, au chapitre 2[2], comment fonctionnait l'algorithme de flux de L - et M . Leur algorithme, limité à la 2D, utilise l'hypothèse de rigidité du GCS à résoudre pour déduire que trois degrés de liberté doivent être retirés pour ancrer une solution dans le plan. À cet effet, une contrainte virtuelle avec trois degrés de restriction est ajoutée et connectée à deux nœuds du graphe de flux.

Afin de s'abstraire de l'hypothèse de rigidité, notre algorithme n'utilise pas cette contrainte virtuelle obligatoirement satisfaite. En revanche, le nœud de chaque entité géométrique est liée à un nouveau nœud, qui dispose d'autant de degrés de restriction que l'entité géométrique a de degrés de liberté. Ces nœuds ne seront pas nécessairement saturés et représentent les éléments du repère que l'on calcule. Ainsi, nous déterminons quels degrés de liberté doivent être fixés.

Nous commençons, avec la section 4.1, par des définitions, notations et représentations spécifiques que nous utilisons dans la suite de ce manuscrit. Nous décrivons ensuite l'algorithme en lui-même à la section 4.2 et en donnons quelques exemples en section 4.3. Nous réalisons enfin une analyse de l'algorithme à la section 4.4.

4.1 Définitions et notations

Le but des algorithmes décrits dans ce chapitre est le calcul d'une paramétrisation combinatoire d'un GCS.

Définition 45. Paramétrisation combinatoire d'un GCS : *Soit* $\mathcal{S} = (C, X, A)$ *un GCS non génériquement sur-contraint. Une paramétrisation combinatoire est une fonction* $p : X \to \mathbb{N}$ *telle qu'il est possible de rendre* $\mathcal{S}$ *génériquement bien contraint* modulo *l'identité en fixant* $p(x)$ *degrés de liberté pour chaque entité géométrique* $x \in X$.

Pour une paramétrisation combinatoire p donnée, nous appelons les entités géométriques x telles que $p(x) > 0$ les entités de l'ancre.

Le graphe de flux spécifique que nous construisons est appelé un graphe de $\mathcal{R}$-flux. Un flot traversant un graphe de $\mathcal{R}$-flux est appelé un $\mathcal{R}$-flot.

Dans les graphes de $\mathcal{R}$-flux, on note
– $u \to t$ l'arc reliant les nœuds u et t et orienté de u vers t ;
– $u \xrightarrow{i} t$ l'arc de u à t ayant une valuation i ;
– $u \underset{j}{\to} t$ l'arc de u à t ayant une capacité de i ;

[2] *Cf.* section 2.2.1.2 p. 49

– $u \xrightarrow[j]{i} t$ l'arc de u à t valué i et de capacité j ;
– $\{\rightarrow t\}$ l'ensemble des arcs entrants de t ;
– $\{u \rightarrow\}$ l'ensemble des arcs sortants de u ;
– Cap(x) la capacité de l'arc x ;
– Val(x) la valuation de l'arc x.

La définition précise d'un graphe de $\mathcal{R}$-flux est la suivante.

Définition 46. Graphe de $\mathcal{R}$-flux d'un GCS : *Soit $\mathcal{S} = (C, X, A)$ un GCS. Le graphe de $\mathcal{R}$-flux de $\mathcal{S}$ est un graphe orienté g dont les arcs ont des valuations dans $\mathbb{N}$ tel que*
– *g a un nœud source n_s et un nœud puits n_p ;*
– *à chaque entité géométrique $x \in X \cup A$ correspond un nœud n_x, avec $n_s \xrightarrow{\text{ddl}(x)} n_x$;*
– *à chaque contrainte $c \in C$ correspond un nœud n_c avec $n_c \xrightarrow{\text{ddr}(c)} n_p$;*
– *si x est une variable de c, alors g inclut un arc $n_x \xrightarrow[k]{} n_c$, avec k le nombre de degrés de libertés que c peut génériquement ôter à x ;*
– *pour chaque entité géométrique x, g inclut un nœud r_x, appelé* nœud de repère *de x, avec les arcs $n_x \xrightarrow[\text{ddl}(x)]{} r_x$ et $r_x \xrightarrow[\text{ddl}(x)]{} n_p$.*

Il est bien entendu important que le flot traversant le graphe de $\mathcal{R}$-flux soit valide, au sens usuel de la validité d'un flot, que nous rappelons.

Définition 47. Graphe de $\mathcal{R}$-flux valide : *Un graphe de $\mathcal{R}$-flux g est valide si*
– *pour tout arc a de g, $0 \leq \text{Val}(a) \leq \text{Cap}(a)$;*
– *pour tout nœud n excepté n_s et n_p, $\Sigma_{i \in \{\rightarrow n\}}(\text{Val}(i)) = \Sigma_{i \in \{n \rightarrow\}}(\text{Val}(i))$*

L'objectif de notre algorithme de paramétrisation est de calculer un $\mathcal{R}$-flot *valide maximal* au sens de la définition suivante.

Définition 48. $\mathcal{R}$-flot valide maximal : *Un $\mathcal{R}$-flot valide est dit maximal si*
– *chaque nœud n_x correspondant à une entité géométrique est saturé, c'est-à-dire qu'on a $\Sigma_{i \in \{n_x \rightarrow\}}(\text{Val}(i)) = k$ avec $n_s \xrightarrow{k} n_x$;*
– *chaque nœud n_c correspondant à une contrainte est saturé : on a $\Sigma_{i \in \{\rightarrow n_c\}}(\text{Val}(i)) = k$ avec $n_c \xrightarrow{k} n_p$;*
– *les nœuds de repère ne sont pas nécessairement saturés : on a $n_x \xrightarrow[j]{i} r_x$ et $r_x \xrightarrow[j]{i} n_p$ avec $0 \leq i \leq j$.*

La définition d'un $\mathcal{R}$-flot valide maximal correspond à la définition classique d'un flot valide maximal, à la différence près que les nœuds de repère n'ont pas à être saturés.

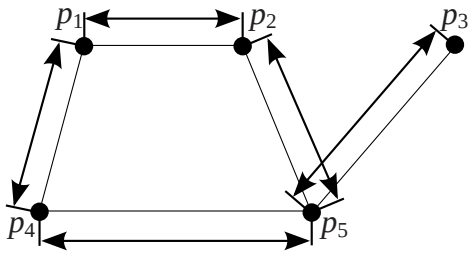


Figure 4.1 - Système articulé fait d'une chaîne fermée de 4 barres rigides et d'une barre en libre rotation autour d'un point de la chaîne

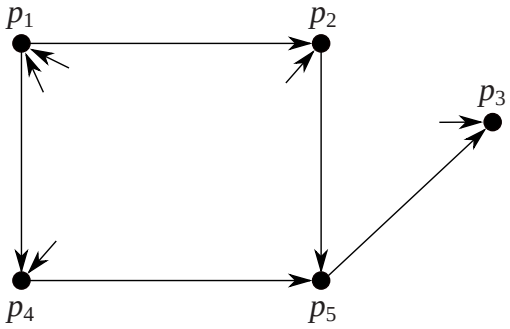


Figure 4.2 - Représentation de C du $\mathcal{R}$-flot de la figure 4.3

Les valuations des arcs allant d'un nœud d'entité à son nœud de repère nous permettront de définir une paramétrisation combinatoire.

Définition 49. Degré de repère d'un nœud du graphe de $\mathcal{R}$-flux : *Dans un graphe de $\mathcal{R}$-flux, on appelle* degré de repère *d'un nœud n_x correspondant à une entité géométrique la valuation de l'arc $n_x \rightarrow r_x$ le reliant à son nœud de repère.*

Par abus de langage, nous parlons également du degré de repère de r_x pour parler du degré de repère de n_x.

Dans un $\mathcal{R}$-flot valide maximal, les degrés de repère non nuls indiquent les degrés de liberté du système et, donc, les entités géométriques à fixer ou à restreindre[3] pour que le GCS ne soit pas sous-contraint. Dans le cas d'un système décrivant un objet rigide, la somme des degrés de repère est donc 3 en dimension 2 et 6 en dimension 3. L'interprétation géométrique de ces valuations, comme nous le voyons à la section 5.1, est un repère minimal pour le système à paramétrer.

La figure 4.3 montre un exemple d'un $\mathcal{R}$-flot valide maximal pour le GCS sous-contraint de la figure 4.1. L'orientation des arcs n'y est pas indiquée car c'est l'orientation classique des graphes de flux bi-partis : de la source vers le puits. Parce que cette représentation n'est pas très intuitive, nous lui substituons deux autres représentations, selon les cas :

– la représentation de C [Chy85], dans laquelle les contraintes ne sont pas explicitement représentées sous la forme de nœuds, mais uniquement sous la forme d'arcs entre les nœuds correspondant aux entités. L'orientation des arcs est faite

[3] Dans la suite, nous disons de l'entité géométrique x qu'elle est *fixée* si le degré de repère de n_x est égal $ddl(x)$ et qu'elle est *restreinte* si le degré de repère de n_x est strictement positif et inférieur à $ddl(x)$.

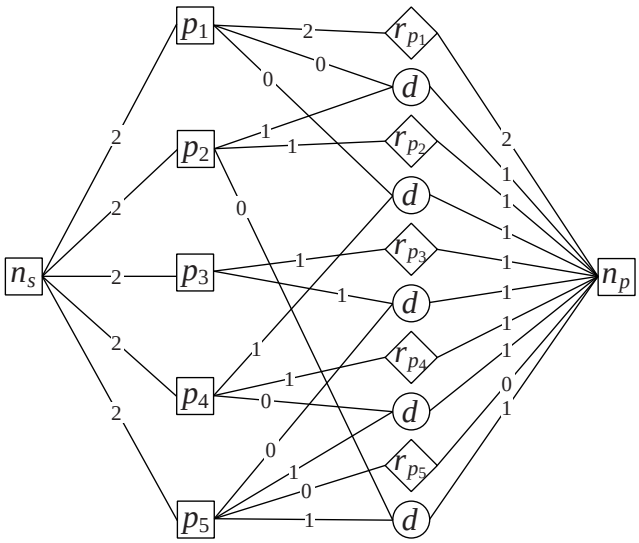


Figure 4.3 - $\mathcal{R}$-flot valide maximal correspondant au GCS de la figure 4.1 : les nœuds p_1 à p_5 représentent les entités géométriques du GCS ; les nœuds r_{p_1} à r_{p_5} sont leurs nœuds de repère ; les nœuds d représentent les contraintes de distance ; les arcs sont valués par la valeur du flux les traversant (les capacités ne sont pas indiquées). Les arcs sont orientés de la gauche vers la droite.

de telle sorte que l'arc soit entrant pour une entité si la valuation de l'arc entre cette entité géométrique et la contrainte est positive. Les nœuds de repère sont quant à eux représentés par des arcs sans origine, chaque arc figurant un degré de valuation. La figure 4.2 donne la représentation de C du $\mathcal{R}$-flot de la figure 4.3 ;

– la représentation de C n'est pas intuitive dans tous les cas : dans le cas de contraintes avec une arité supérieure à 1, la représentation des arcs devient un peu moins compréhensible ; lorsque plusieurs contraintes de même profil portent sur un même ensemble de contraintes, la représentation de C ne permet pas de différencier les contraintes ; surtout, les contraintes et la valuation des arcs liés à leur nœud sont implicites dans la représentation de C , ce qui la rend inappropriée pour des illustrations du fonctionnement de nos algorithmes lorsqu'il s'agit de montrer comment un $\mathcal{R}$-flot est modifié. C'est pourquoi dans certains cas, nous utilisons une représentation explicite du graphe de $\mathcal{R}$-flux, en oubliant simplement les nœuds n_s et n_p ainsi que les nœuds de repère dont le degré de repère est nul. L'orientation des arcs n'est pas représentée graphiquement car elle n'apporte pas d'information utile : les arcs du graphe de flux vont toujours des nœuds d'entité vers les nœuds de contrainte. De même, les capacités des arcs ne sont pas indiquées car elles sont égales au degré de restriction générique des contraintes. La figure 4.5, par exemple, montre les étapes de calcul du $\mathcal{R}$-flot de la figure 4.3 en utilisant cette représentation.

L'ajout des nœuds r_x permet de se passer de l'hypothèse de rigidité du GCS. Elle a toutefois le désavantage d'empêcher la détection de systèmes sur-contraints. On peut ainsi voir qu'à la figure 1.5, notre algorithme associe un $\mathcal{R}$-flot indiquant qu'il faut fixer un point pour que le système soit bien contraint, alors que 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 montrons au chapitre 6 comment la méthode du témoin[4] nous permet de détecter la sur-constriction. Les méthodes disponibles dans la littérature [LM96b, HLS98, JNT03] ne souffrent pas de ce désavantage grâce à l'hypothèse de rigidité, qui leur permet de savoir que le nombre total de degrés de repère doit être 3 ou 6, selon la dimension.

4.2 Algorithmes pour la paramétrisation combinatoire

Nous décrivons dans cette section les différents algorithmes nécessaires au calcul incrémental d'une paramétrisation combinatoire. Il nous faudra pour cela définir les opérations d'ajout d'entité géométrique et de contrainte. Nous commençons par donner l'algorithme général calculant une paramétrisation combinatoire d'un GCS de manière incrémentale, puis détaillons l'opération d'ajout d'une contrainte.

4.2.1 Principe général

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. À cet effet, nous définissons les trois opérations suivantes :

1. l'initialisation du $\mathcal{R}$-flux correspondant à un GCS vide ;
2. la modification d'un $\mathcal{R}$-flux lors de l'ajout d'une entité géométrique ;
3. la modification d'un $\mathcal{R}$-flux lors de l'ajout d'une contrainte.

À partir de ces trois opérations, on peut trivialement définir l'algorithme qui calcule une paramétrisation combinatoire d'un GCS. Le pseudo-code est donné à l'algorithme 4.1.

L'initialisation du graphe de $\mathcal{R}$-flux (ligne 1 de l'algorithme) est faite en considérant le graphe sans arc contenant les nœuds n_s et n_p.

L'ajout d'une entité géométrique x (ligne 5 de l'algorithme) est fait en ajoutant deux nœuds : le nœud n_x correspondant à x et son nœud de repère r_x. On ajoute trois arcs :

[4] Que nous avons évoquée dans le chapitre 2, *cf.* p.52

Entrées :
$\mathcal{S} = (C, X, A)$: un GCS
Résultat :
Graphe de $\mathcal{R}$-flux maximal valide correspondant à $\mathcal{S}$

1 $\mathcal{R} \leftarrow$ graphe de $\mathcal{R}$-flux initial
pour chaque *contrainte* $c(x_1, \ldots, x_n) \in C$ **faire**
  pour chaque *entité* $x_i \in \{x_1, \ldots, x_n\}$ **faire**
    si $\mathcal{R}$ *ne contient pas le nœud correspondant à* x_i **alors**
5   Ajouter x_i dans $\mathcal{R}$
6 Ajouter c dans $\mathcal{R}$
retourner $\mathcal{R}$

Algorithme 4.1: Calcul d'une paramétrisation combinatoire d'un GCS

$n_s \xrightarrow{ddl(x)} n_x$, $n_x \xrightarrow{ddl(x)} r_x$ et $r_x \xrightarrow{ddl(x)} n_p$.

Lors de l'ajout d'une contrainte, enfin (ligne 6 de l'algorithme), le graphe de $\mathcal{R}$-flux est modifié en recherchant un chemin améliorant [FJF56]. Nous décrivons l'opération en détail à la section 4.2.2.

4.2.2 Modification d'un $\mathcal{R}$-flot à l'ajout d'une contrainte

L'ajout d'une nouvelle contrainte c se fait lorsque toutes les entités géométriques liées à cette contrainte sont présentes dans le graphe, puis en baissant les valuations d'entrée de certains nœuds de repère afin de pouvoir saturer n_c. Cette opération est une adaptation de la recherche de chemin améliorant dans l'algorithme de F - F [FJF56]. Pour effectuer l'opération, nous recherchons un chemin sans cycle de n_c vers un nœud r_x tel que :
– r_x est au moins partiellement saturé : $\text{Val}(n_x \rightarrow r_x) > 0$;
– pour chaque nœud n_x du chemin, la valuation de l'arc lié à n_x et situé du côté de n_c est augmentable (inférieure à sa capacité) ;
– de même, pour chaque somme n_x du chemin, la valuation de l'arc lié à n_x et situé du côté de r_x est diminuable (non nulle).
Une fois un tel chemin trouvé, il faut « retourner » ce chemin : pour chaque nœud de contrainte $n_{c'}$ du chemin (sauf n_c), la valuation de l'arc situé du côté de n_c est diminuée de 1 et celle de l'arc situé du côté de r_x est augmentée de 1. Ensuite, la valuation de l'arc lié à r_x est réduite de 1 et celle de l'arc lié n_c est augmentée de 1. Si le chemin ne contient aucune contrainte, son retournement consiste uniquement à diminuer le degré de repère de r_x et à augmenter la valuation de l'arc $n_c \rightarrow n_x$. La

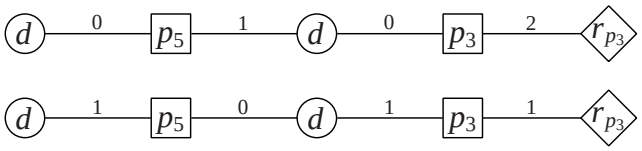


Figure 4.4 - Retournement d'un chemin, commençant par une contrainte de distance et se terminant par le nœud de repère r_{p_3}. Le chemin est celui qui est retourné à la figure 4.5**e** pour obtenir le $\mathcal{R}$-flot de la figure 4.5**f**.

figure 4.4 illustre le retournement d'un chemin (elle représente la transition entre les figures 4.5**e** et 4.5**f**).

On répète cette opération (recherche de chemin, retournement de chemin) autant de fois que le degré de restriction de c[5], afin que n_c soit saturé. L'algorithme 4.2 donne le pseudo-code de l'opération d'ajout d'une contrainte, en faisant appel à l'algorithme 4.3 pour chercher un chemin. L'algorithme 4.3 recherche les chemins valides avec comme nœud initial une entité géométrique donnée. L'algorithme 4.2 utilise donc cet algorithme sur chacune des entités géométriques 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 sont discutés plus avant dans la suite, 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 géométrique en-dessous d'une valeur indiquée par l'utilisateur.

L'algorithme 4.2 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 $\mathcal{R}$-flux calculé.

4.2.3 Modification d'un $\mathcal{R}$-flot

Il est également possible d'adapter l'algorithme d'ajout d'une contrainte (4.2) afin de modifier le $\mathcal{R}$-flot sans ajouter de contrainte et ainsi chercher d'autres repères.

[5]Il serait également possible de chercher directement un chemin permettant, en un retournement, de satisfaire la nouvelle contrainte. Toutefois, dans la pratique, la plupart des contraintes rencontrées ont un degré de restriction de 1, aussi les chances de trouver un tel chemin sont faibles.

Entrées :
$\mathcal{R}$: Graphe de $\mathcal{R}$-flux actuel, valide maximal
$c(x_1, ...x_m)$: Contrainte à ajouter
$L = \{(x_i, d_i)\}$: Liste des obligations de repère
T : Stratégie de choix de repère
Résultat :
Graphe de $\mathcal{R}$-flux valide maximal après ajout de n_c

---

$E \leftarrow$ liste $(n_{x_1}, ..., n_{x_m})$ des nœuds d'entités liées à c
Ajouter un nœud n_c lié à l'ensemble des nœuds de E
Valuer les arcs de n_c par 0
pour *i de 0 à* ddr(c) **faire**
  $P \leftarrow$ liste vide des chemins trouvés
  pour chaque *entité* $x \in E$ **faire**
    $\mathcal{A} \leftarrow$ nouvel arbre avec x comme unique nœud
    $P \leftarrow P +$ `RechercheChemins(`$\mathcal{R}$`, x, L, `$\mathcal{A}$`, {c})`
    `// Utilisation de l'algorithme 4.3`
  $p \leftarrow$ meilleur chemin de P d'après T
  Retourner le chemin p dans $\mathcal{R}$
  Incrémenter la valuation de l'arc liant n_c au nœud initial de p
retourner $\mathcal{R}$

Algorithme 4.2: Mise-à-jour du $\mathcal{R}$-flux lors de l'ajout d'une contrainte

Entrées :
$\mathcal{R}$: Graphe de $\mathcal{R}$-flux valide maximal
x : Entité dont on veut augmenter le degré de repère
$L = \{(x_i, d_i)\}$: Liste des obligations de repère
$\mathcal{A}$: Arbre des chemins parcourus
I : Contraintes interdites dans le chemin
Résultat :
Chemin améliorant

$P \leftarrow$ liste vide des chemins trouvés
pour chaque *contrainte c liée à x qui n'est pas dans I* **faire**
  si *c ne relie pas x à son parent dans* $\mathcal{A}$ **alors**
    si *c est retournable* **alors**
      pour chaque *entité e liée à c sauf x* **faire**
        ajouter *e* dans $\mathcal{A}$ avec *x* pour parent
        si *l'on peut retirer un degré de repère à* n_e **alors**
          marquer *e* comme final dans $\mathcal{A}$
        $P \leftarrow P +$ `RechercheChemins(`$\mathcal{R}$`, e, L, `$\mathcal{A}$`, I)`
        `// Récursion`
retourner P

Algorithme 4.3: Recherche de chemin améliorant dans le graphe de $\mathcal{R}$-flux

En effet, en recherchant un chemin améliorant ayant pour nœud initial n_x et pour nœud final un nœud $r_{x'}$ totalement ou partiellement saturé, on peut augmenter de 1 la valuation de l'arc $n_x \to r_x$ et diminuer de 1 la valuation de l'arc $n_{x'} \to r_{x'}$.

Entrées :
$\mathcal{R}$: Graphe de $\mathcal{R}$-flux actuel, valide maximal
x : Entité dont on veut augmenter le degré de repère
$L = \{(x_i, d_i)\}$: Liste des obligations de repère
T : Stratégie de choix de repère
Résultat :
Graphe de $\mathcal{R}$-flux valide maximal après augmentation de $\text{Val}(n_x \to r_x)$
Nouvelle liste des obligations de repère

$\mathcal{A} \leftarrow$ nouvel arbre avec x comme unique nœud
$P \leftarrow$ `RechercheChemins(`$\mathcal{R}$`, x, L,` $\mathcal{A}$`,` $\emptyset$`)` `// Utilisation de l'algorithme 4.3`
$p \leftarrow$ meilleur chemin de P d'après T
Retourner le chemin p dans $\mathcal{R}$
Incrémenter la valuation de l'arc $n_x \to r_x$

si $\exists d, (x, d) \in L$ **alors**
  $L \leftarrow L - (x, d) + (x, d + 1)$
sinon
  $L \leftarrow L + (x, 1)$
retourner $\mathcal{R}, L$

Algorithme 4.4: Augmentation du degré de repère d'une entité

Grâce à l'utilisation de l'algorithme 4.4, 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 du degré de repère d'une entité géométrique x, le couple (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 géométrique x. Dans l'algorithme 4.3, 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 » (ligne 7) : il faut pour satisfaire cette condition que le nombre de degrés de repères de e, c'est-à-dire la valuation de l'arc de n_e vers r_e, soit non nulle et supérieure à d quand (e, d) apparaît dans la liste des obligations de repère. On pourrait remplacer le test de la ligne 7 par le test « **si** $((\exists (e, d) \in L$ **et** $\text{Val}(n_e \to r_e) > d)$ **ou** $(\nexists (e, d) \in L$ **et** $\text{Val}(n_e \to r_e) > 0))$ ».

4.3 Exemples d'application

Nous donnons ici plusieurs exemples d'application de notre algorithme de paramétrisation combinatoire. Nous commençons avec un exemple en 2D, constitué d'une chaîne fermée de quatre barres rigides, partageant un point avec une cinquième barre (section 4.3.1) ; nous donnons ensuite un exemple en 3D avec le calcul de deux paramétrisations d'une plate-forme de S (section 4.3.2) ; enfin, nous montrons les limites de cet algorithme avec le célèbre exemple sur-contraint de la « double-banane » (section 4.3.3).

4.3.1 Exemple 2D

Considérons l'exemple de la figure 4.1, représentant un GCS constitué d'une chaîne fermée non rigide de quatre barres et d'une cinquième barre en rotation libre autour d'un des points de la chaîne. Le graphe de $\mathcal{R}$-flux initial est représenté à la figure 4.5**a**, avec uniquement le point p_1 et le nœud de repère r_{p_1}. La figure 4.5**b** montre le graphe de $\mathcal{R}$-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 4.5**c**, p_3 et p_4 ont été ajoutés, ainsi que les contraintes concernant les quatre entités géométriques considérées. À la figure 4.5**d** 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 4.5**e** montre l'ajout de la dernière contrainte de distance (non encore saturée) et un chemin possible à retourner (en pointillés) 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 4.5**f** montre le $\mathcal{R}$-flot résultant après cette exécution de l'algorithme. La figure 4.7 montre une interprétation géométrique de ce $\mathcal{R}$-flot sous forme de repère.

La figure 4.6 donne un exemple d'application de l'algorithme 4.4, qui augmente le degré de repère d'une entité géométrique. En partant du $\mathcal{R}$-flot final de la figure 4.5, l'utilisateur a demandé à augmenter le degré de repère de p_3. Le repère qui correspond à ce $\mathcal{R}$-flot est illustré à la figure 4.8.

4.3.2 Exemple 3D : la plate-forme de S

Considérons maintenant un exemple complexe en 3D : une plate-forme de S - [Ste66]. Elle est constituée de deux hexagones rigides (*e.g.* des hexagones tels que celui représenté à la figure 8.2, auquel on ajoute une contrainte de copla-

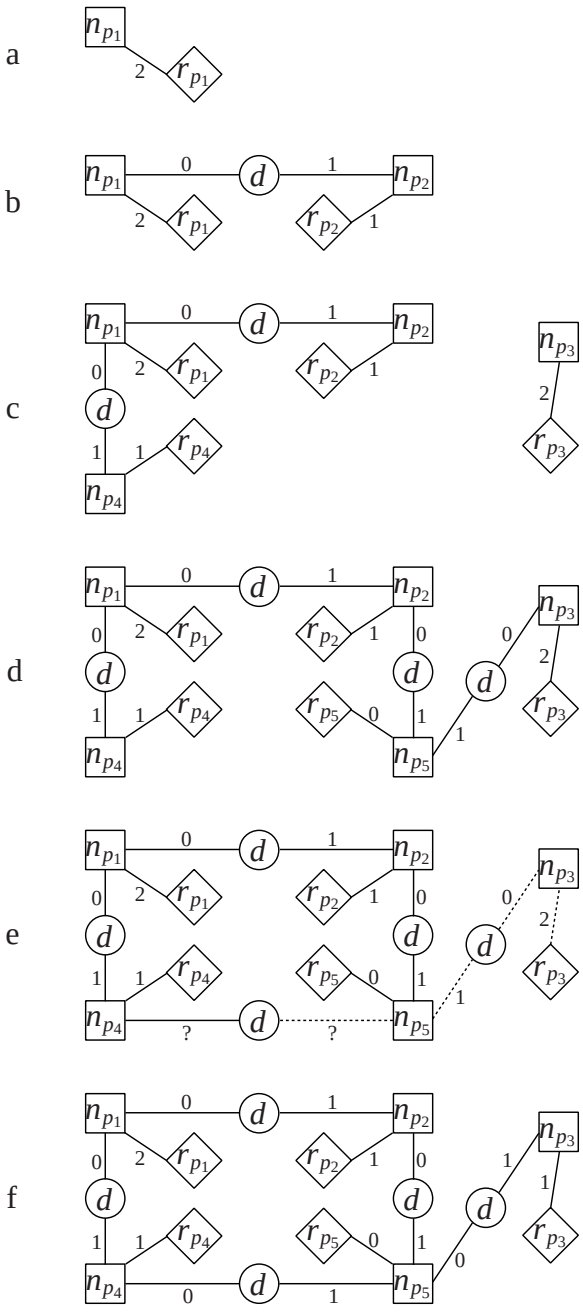


Figure 4.5 - Quelques étapes de l'algorithme 4.1 sur l'exemple de la figure 4.1 ; **a** : initialisation avec un point et son repère ; **b** : ajout d'un point et d'une contrainte ; **c** : ajout de deux points et des contraintes les concernant ; **d** : système entier sauf une contrainte ; **e** : un chemin possible à retourner en pointillé ; **f** : un $\mathcal{R}$-flot possible

a

b

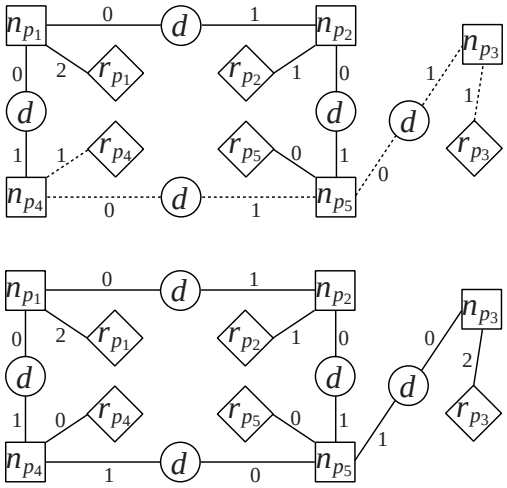


Figure 4.6 - Application de l'algorithme 4.4 sur le $\mathcal{R}$-flot de la figure 4.5**f** ;
a : un chemin améliorant possible pour ajouter un degré de repère à p_3 ;
b : $\mathcal{R}$-flot résultant

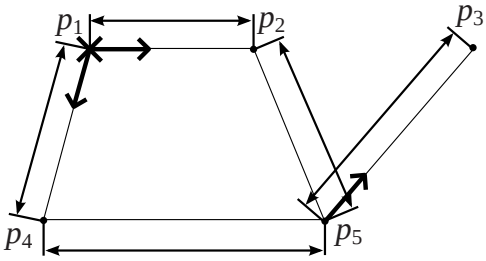


Figure 4.7 - Repère correspondant à l'interprétation du $\mathcal{R}$-flot final de la figure 4.5

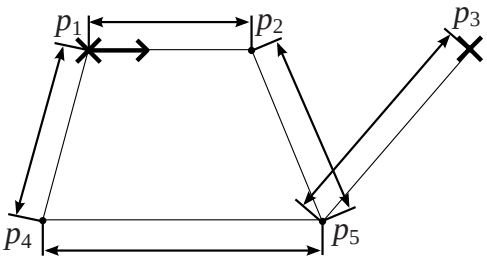


Figure 4.8 - Repère correspondant à l'interprétation du $\mathcal{R}$-flot final de la figure 4.6

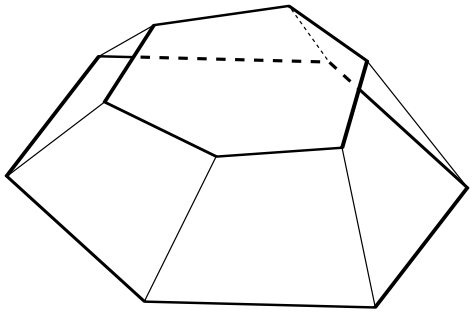


Figure 4.9 - Esquisse d'une plate-forme de S . Les deux hexagones en gras sont rigides ; les six autres segments indiquent des contraintes de distance.

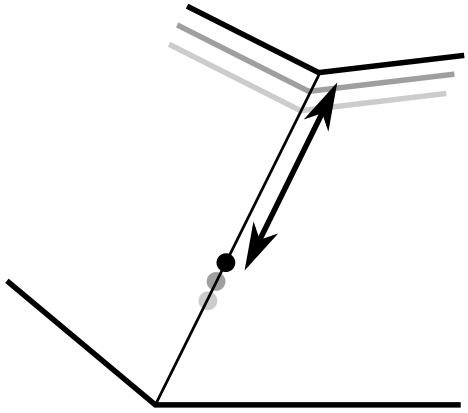


Figure 4.10 - Assemblage prismatique entre deux points d'une plate-forme de S : le point médian est incident à la ligne.

narité des six points) liés ensemble par six assemblages prismatiques, c'est-à-dire des barres dont la longueur est un paramètre fourni par l'utilisateur. Pour des valeurs données de ces paramètres, six barres rigides relient les deux hexagones, ce qui donne un système rigide, dont le graphe de contrainte est représenté à la figure 4.9. Pour représenter le fait que les barres sont des assemblages prismatiques, on peut remplacer chacune des six barres rigides par une droite, sur laquelle on place un point, à une distance fixe de l'hexagone supérieur. Comme la distance au second hexagone n'est pas connue, ce point est en libre translation sur la droite. Un exemple de cette représentation est donné à la figure 4.10.

La figure 4.11 donne certaines étapes du calcul d'une paramétrisation combinatoire de la plate-forme de S avec l'algorithme 4.1 ainsi que deux modifications du $\mathcal{R}$-flot avec l'algorithme 4.4.

La figure 4.11**a** montre sous forme de repère le $\mathcal{R}$-flot calculé pour le système composé des deux hexagones rigides sans qu'aucune contrainte ou entité géométrique ne les joigne. À la figure 4.11**b**, une droite a été ajoutée, contrainte à être incidente à l'un des points de l'hexagone inférieur. Comme cette contrainte a un degré de restriction de 2, la droite a encore deux degrés de liberté. La flèche barrée indique qu'il faut restreindre ces deux degrés de liberté. La figure 4.11**c** montre le $\mathcal{R}$-flot après ajout d'une contrainte d'incidence à un point de l'hexagone supérieur. Notre stratégie retire deux degrés de repère de l'hexagone supérieur.

À la figure 4.11**d**, les six droites ont été ajoutées et rendues incidentes chacune à deux points, un dans chacun des hexagones. Nous simulons alors une demande,

par l'utilisateur, de changement du $\mathcal{R}$-flot, afin de mieux comprendre dans quelle mesure l'hexagone supérieur est mobile. Six utilisations de l'algorithme 4.4 mènent au $\mathcal{R}$-flot représenté sous forme de repère à la figure 4.11**e**.

À la figure 4.11**f**, nous montrons l'ajout d'un point et de deux contraintes : une contrainte de distance par rapport à un point de l'hexagone supérieur (avec un degré de restriction) et une contrainte d'incidence à la droite passant par ce point (avec deux degrés de restriction). L'ensemble constitué de l'hexagone supérieur et du nouveau point est rigide. En ajoutant, similairement, cinq autres points, nous obtenons la figure 4.11**g**.

Enfin, la figure 4.11**h** est obtenue après une nouvelle utilisation de l'algorithme 4.4 : l'utilisateur demande à incrémenter le degré de repère de l'un des nouveaux points. Une fois que l'utilisateur a fait cela pour les six nouveaux points, nous obtenons la figure 4.11**i**. Elle indique que si l'hexagone inférieur est fixé, la position de l'hexagone supérieur dépend de la position, sur leur droite respective, de chacun des points glissants.

4.3.3 Exemple 3D : la « double-banane »

Considérons enfin le célèbre contre-exemple de la caractérisation de L [6] qu'est la« double-banane ». La figure 1.14 donne le graphe de contrainte du GCS de la double-banane. Il est constitué de deux paires de tétraèdres partageant une face (les « bananes »), les deux paires ayant deux points en commun. De manière plus générale, on peut considérer n'importe quel couple de systèmes rigides en 3D partageant deux points comme une forme de double-banane.

Ce GCS est génériquement sur-contraint : la distance entre p_1 et p_2 peut être calculée dans n'importe laquelle des deux bananes, la probabilité que les valeurs calculées soient cohérentes est donc quasi-nulle. S'il n'est pas sur-contraint parce que les distances sont cohérentes, alors il est sous-contraint : chacune des bananes est en libre rotation autour de l'axe (p_1p_2).

La figure 4.12 donne les étapes de calcul d'une paramétrisation combinatoire de ce système. Nous y représentons le graphe de contrainte du système déjà intégré au graphe de $\mathcal{R}$-flux et ajoutons à cela une interprétation sous forme de repère du $\mathcal{R}$-flot calculé.

La figure 4.12**a** montre le $\mathcal{R}$-flot calculé en ayant considéré une contrainte de distance entre deux points. Le $\mathcal{R}$-flot indique qu'un des points est fixé, l'autre est res-

[6] *Cf.* pp. 32 ou 46

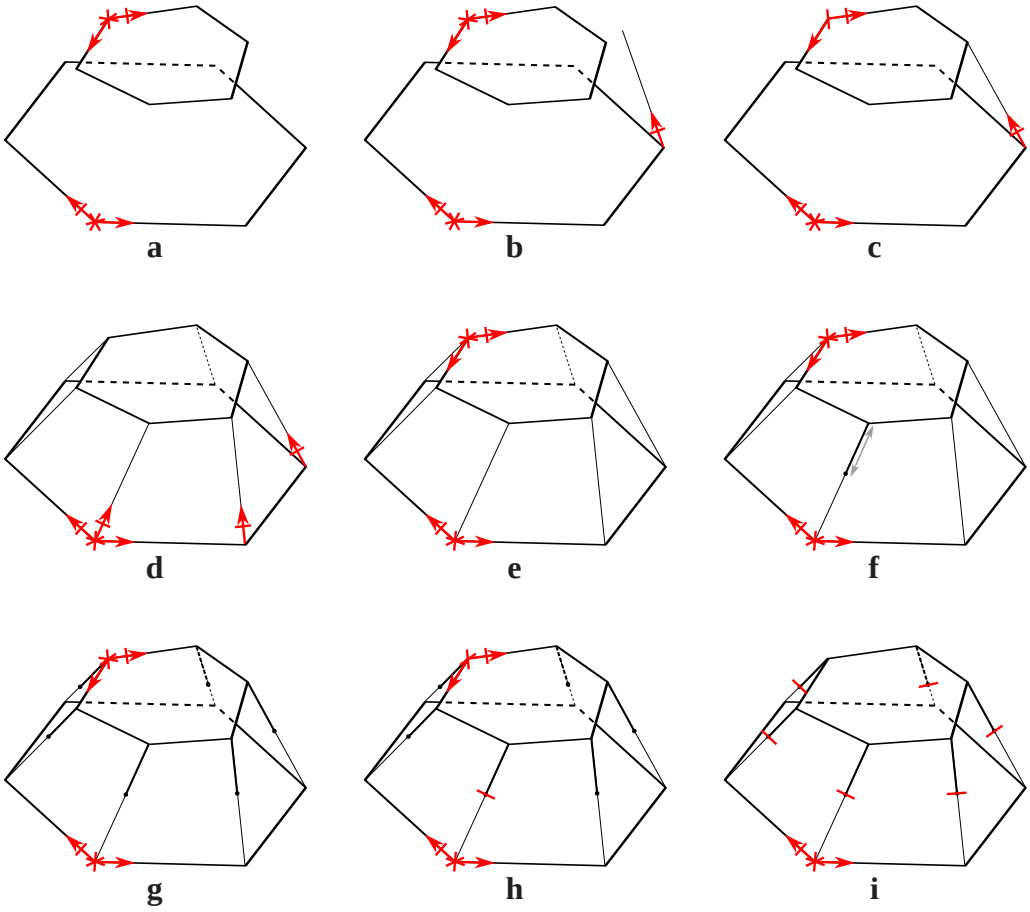


Figure 4.11 - Étapes du calcul d'une paramétrisation combinatoire d'une plate-forme de S avec l'algorithme 4.1. Voir texte

treint puisqu'il faut lui retirer deux degrés de liberté. Une interprétation géométrique de ce $\mathcal{R}$-flot est qu'il faut fixer un point et la direction de ce point à l'autre (la flèche barrée indique que cette direction correspond à deux degrés de liberté).

À la figure 4.12**b**, un troisième point a été ajouté, ainsi qu'une contrainte de distance. Il faut donc restreindre ce nouveau point, ce qui se traduit là encore par la donnée d'une direction. Puis une troisième contrainte de distance est ajoutée à la figure 4.12**c**, ce qui réduit le nombre de degrés de liberté du dernier point ajouté : il n'est plus sur une sphère mais sur un cercle, il suffit donc de le restreindre d'un degré de liberté. La figure 4.12**d** montre l'ajout d'un quatrième point, sans changement de la paramétrisation puisqu'il est construit à partir de trois contraintes de distance. On a construit un premier tétraèdre. De même, à la figure 4.12**e**, un point et trois contraintes de distance ont été ajoutés, complétant la première banane.

À la figure 4.12**f**, le premier tétraèdre de la seconde banane a été pris en compte dans le calcul du $\mathcal{R}$-flot. Il faut restreindre deux degrés de liberté de l'un de ses points et un degré de liberté d'un autre point, ce qui correspond bien à la pratique : le tétraèdre est en rotation autour du point qu'il partage avec la première banane. Les figures 4.12**g** et 4.12**h** montrent l'ajout des contraintes de distance permettant d'obtenir un système articulé qui n'est pas génériquement sur-contraint. Enfin, la figure 4.12**i** montre que l'ajout de la dernière contrainte finalise la seconde banane mais rend le système génériquement sur-contraint. Notre algorithme de paramétrisation ne détectant pas ce fait, il consomme un degré de repère et calcule donc un $\mathcal{R}$-flot indiquant qu'il faut fixer six degrés de liberté au total.

4.4 Analyse de l'algorithme

4.4.1 Correction

L'algorithme 4.1 produit incrémentalement une paramétrisation combinatoire d'un GCS. Nous montrons ici que le $\mathcal{R}$-flot qu'il calcule est un couplage parfait dans le graphe bi-parti $(N_X, N_C + R')$, où N_X est l'ensemble des nœuds d'entité, N_C l'ensemble des nœuds de contrainte et R' l'ensemble des nœuds de repère dont le degré est non nul.

Nous commençons par montrer que l'algorithme 4.1 produit bien un $\mathcal{R}$-flot valide maximal :

Théorème 9. Validité et maximalité des $\mathcal{R}$-flots calculés par l'algorithme 4.1 : *Soit $\mathcal{S} = (C, X, A)$ un GCS. Le $\mathcal{R}$-flot calculé en ajoutant toutes les entités géo-*

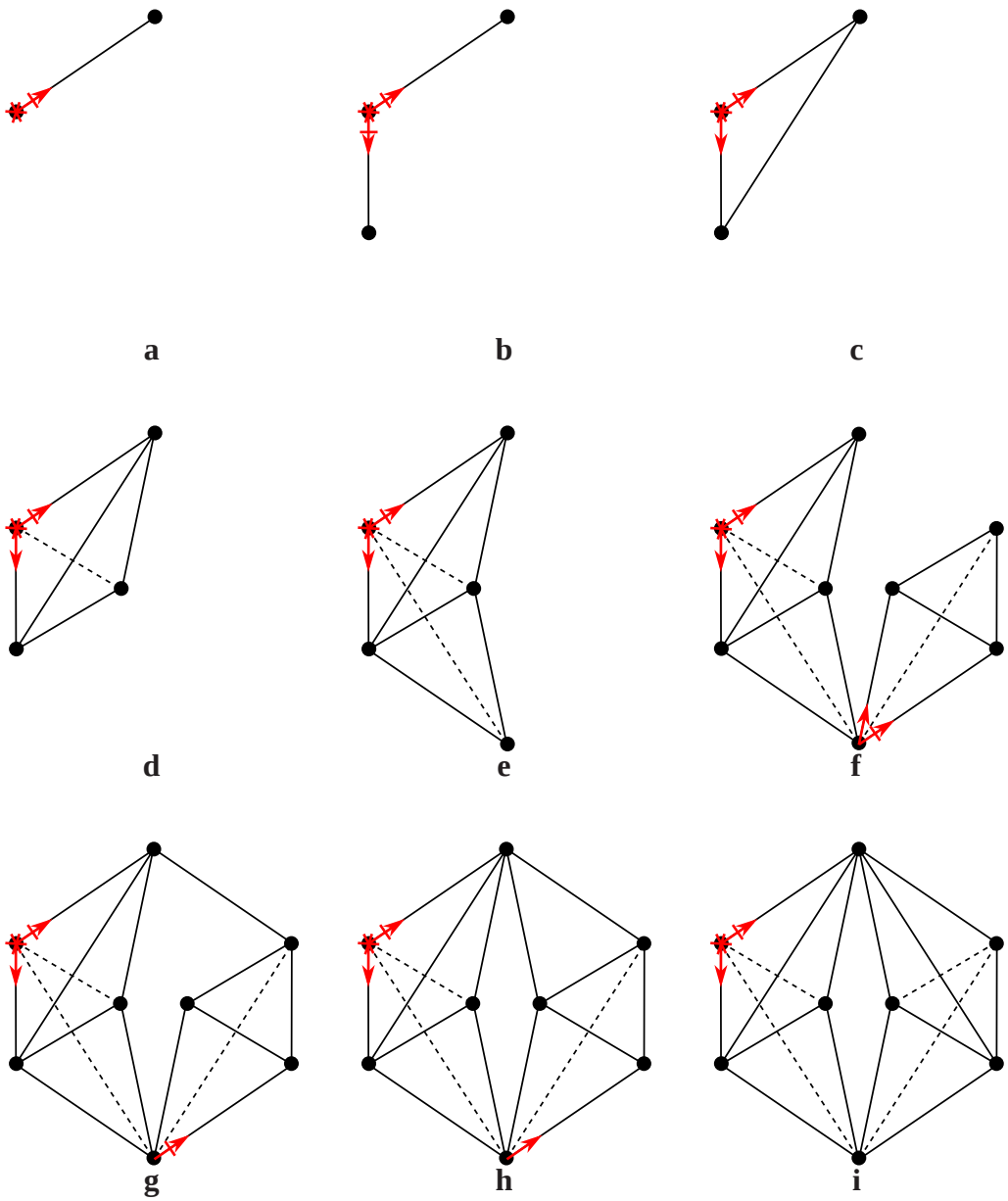


Figure 4.12 - Étapes du calcul d'une paramétrisation combinatoire de la double-banane avec l'algorithme 4.1. Voir texte

métriques de X et les contraintes de C en utilisant l'algorithme 4.1 est valide et maximal.

Démonstration:

Le $\mathcal{R}$-flot initial est obtenu en ayant deux nœuds : n_s et n_p. Il n'y a aucun arc entre ces nœuds, qui ne peuvent, par définition, pas être saturés. Le $\mathcal{R}$-flot est donc valide et maximal.

L'ajout d'une nouvelle entité géométrique est effectué en ajoutant le nœud d'entité et le nœud de repère et en saturant le nœud de repère. Si le $\mathcal{R}$-flot était maximal et valide avant ajout de l'entité géométrique, il est toujours maximal et valide après.

L'ajout d'une contrainte c consiste en la recherche de $ddr(c)$ chemins améliorants : la saturation des nœuds situés au milieu des chemins améliorants est inchangée lors du retournement du chemin[7] ; la saturation du dernier nœud d'entité du chemin est inchangée puisque son degré de repère est diminué ; après $ddr(c)$ retournements de chemins améliorants, le nouveau nœud de contrainte est saturé. Si le $\mathcal{R}$-flot était maximal et valide avant ajout de la contrainte, il est toujours maximal et valide après.

L'ajout de contraintes et d'entités géométriques préservent la validité et la maximalité du $\mathcal{R}$-flot, qui est valide et maximal à l'initialisation. Les $\mathcal{R}$-flots calculés avec l'algorithme 4.1 sont donc valides et maximaux. □

Nous montrons également que l'algorithme 4.4 préserve la validité et la maximalité d'un $\mathcal{R}$-flot.

Théorème 10. Préservation de la validité et de la maximalité d'un $\mathcal{R}$-flot par l'algorithme 4.4 : *Soit $\mathcal{R}$ un graphe de $\mathcal{R}$-flux valide et maximal. L'utilisation de l'algorithme 4.4 sur le graphe $\mathcal{R}$ pour une entité géométrique x dont le nœud de repère n'est pas saturé préserve la validité et la maximalité de $\mathcal{R}$.*

Démonstration:

L'algorithme 4.4 procède par recherche d'un chemin améliorant et retournement de ce chemin.

La saturation des nœuds situés au milieu du chemin est inchangée lors du retournement du chemin, puisque la valuation d'un arc est décrémentée et la valuation de l'autre arc incrémentée.

La saturation du dernier nœud d'entité du chemin est inchangée puisque la valuation de l'arc du côté initial est incrémentée et que la valuation de l'arc le reliant à son nœud de repère est décrémentée.

[7] Voir par exemple la figure 4.4 ou la démonstration du théorème 10

La saturation du nœud d'entité de x est inchangée puisque la valuation de l'arc du côté final est décrémentée et que la valuation de l'arc $n_x \rightarrow r_x$ est incrémentée.

Les valuations du $\mathcal{R}$-flot modifiées par l'algorithme 4.4 concernent uniquement des nœuds du chemin améliorant. Les saturations des nœuds de ce chemin sont inchangées. La validité et la maximalité du $\mathcal{R}$-flot sont donc préservées. □

Nous avons donc l'assurance que les $\mathcal{R}$-flots calculés avec nos algorithmes sont valides et maximaux.

Considérons le graphe bi-parti $B = (N_X, N_C \cup R_s \cup R_p)$ avec
- N_X l'ensemble des nœuds d'entité ;
- N_C l'ensemble des nœuds de contrainte ;
- R_s l'ensemble des nœuds de repère saturés ;
- R_p l'ensemble des nœuds partiellement saturés (avec une valuation non nulle inférieure à la capacité de leurs arcs) ; pour tout nœud $r_x \in R_p$, on a $\text{Cap}(r_x \rightarrow n_p) = \text{Val}(n_x \rightarrow r_x)$ dans B.

Autrement dit, B est le graphe bi-parti entités-contraintes et repères, en retirant simplement les nœuds de repère de degré nul et en baissant la capacité des arcs sortant des nœuds de repère non saturés afin de simuler le fait qu'ils n'ont pas d'obligation de saturation.

Nous pouvons alors prouver la correction de notre algorithme de paramétrisation.

Théorème 11. Correction de la paramétrisation combinatoire par $\mathcal{R}$-flot : *Soit $\mathcal{S}$ un GCS non génériquement sur-contraint. Un $\mathcal{R}$-flot de $\mathcal{S}$ calculé par l'algorithme 4.1 est un couplage parfait du graphe bi-parti B de $\mathcal{S}$.*

Démonstration:

Nous savons, grâce aux théorèmes 9 et 10 que les $\mathcal{R}$-flots calculés sont valides et maximaux.

On sait donc, par définition d'un $\mathcal{R}$-flot valide maximal[8], que
- les nœuds d'entité sont saturés : la somme des valuations de leurs arcs sortants est égale à la capacité de leur arc entrant ;
- les nœuds de contrainte sont saturés : la somme des valuations de leurs arcs entrants est égale à la capacité de leur arc sortant.

Les nœuds de repère, quant à eux, ne sont pas nécessairement saturés. Toutefois, la définition de B fait que les nœuds de repère de degré nul n'ont pas été inclus. Quant aux nœuds de repère partiellement saturés, la capacité de leur arc sortant dans B a été diminuée jusqu'à la valuation effective de leur

[8] *Cf.* déf. 48 p. 93

arc entrant. Par conséquent, la valuation de leur arc entrant est égale à la capacité de leur arc sortant : ils sont saturés dans B.

Tous les nœuds internes de B sont saturés : on a bien un couplage parfait. □

Pour un $\mathcal{R}$-flot valide maximal donné d'un GCS $\mathcal{S}$, considérons la fonction qui à toute entité géométrique de $\mathcal{S}$ associe son degré de repère dans le $\mathcal{R}$-flot. Sous l'hypothèse que le GCS n'est pas génériquement sur-contraint, cette fonction est une paramétrisation combinatoire de $\mathcal{S}$, d'après le théorème de K¨ -H [Die05] :

Théorème 12. *Un système de contraintes géométriques est structurellement bien contraint si et seulement si son graphe biparti admet un couplage parfait.*

4.4.2 Complexité

La complexité de nos algorithmes se calcule comme suit. Considérons que l'on dispose d'une méthode de comparaison de plusieurs chemins améliorants dont la complexité est en $O(P)$.

L'algorithme 4.3 réalise une recherche en largeur d'abord dans le graphe. La complexité d'une telle recherche est $O(|C| + |X|)$ [Knu98]. Dans le domaine spécifique de la résolution de GCS, $|C|$ et $|X|$ sont à peu près équivalents, de par l'impossibilité d'avoir plus de degrés de contrainte que de degrés de liberté. La complexité dans le pire des cas de l'algorithme 4.3 est donc $O(2 \times d \times |X|)$ – où d est le nombre moyen de degrés de liberté d'une entité géométrique – c'est-à-dire $O(|X|)$.

L'algorithme 4.4 fait appel à l'algorithme 4.3 à une reprise. Il fait également appel à la méthode de comparaison des chemins identifiés. La recherche dans la liste des obligations de repère comporte au maximum $|X|$ tests, dans la mesure où la liste ne peut pas contenir plus d'éléments qu'il n'y a d'entités géométriques. La complexité de l'algorithme 4.4 est donc en $O(P + 2|X|)$, c'est-à-dire en $O(P + |X|)$.

L'ajout d'une entité géométrique est réalisé en temps constant puisqu'il s'agit uniquement d'ajouter deux nœuds et de saturer leurs arcs.

L'algorithme 4.2 réalise $ddr(c)$ utilisations de l'algorithme 4.3 et de la comparaison de chemins pour ajouter la contrainte c. Il est toutefois facile, lors de la programmation effective, de le transformer en une recherche des chemins améliorants et en une sélection des $ddr(c)$ meilleurs chemins. La complexité de l'algorithme est donc en $O(|X| + ddr(c) \times P)$.

L'algorithme 4.1 réalise $|X|$ ajouts d'une entité géométrique et $|C|$ ajouts de contrainte. Sa complexité est donc en $O(|C|(|X| + dP))$, avec d le nombre moyen de

degrés de restriction d'une contrainte. Nous avons déjà vu que $|C|$ et $|X|$ étaient bornés par le nombre de degrés de liberté du GCS, qui est, à un facteur près, égal à $|X|$. La complexité dans le pire cas de l'algorithme 4.1 est donc $O(|X|^2 + |X|P)$.

Rappelons toutefois que notre algorithme fait l'hypothèse de non sur-constriction générique du GCS et qu'il s'agit d'une hypothèse forte. Dans notre implémentation, nous nous assurons du respect de cette hypothèse en utilisant la méthode du témoin[9], qui nécessite un calcul du rang de la matrice Jacobienne du GCS. La complexité de ce calcul étant $O(\min(m,n)mn)$ – avec m le nombre de lignes et n le nombre de colonnes – c'est-à-dire $O(m^2 n)$ et donc $O(|C|^2|X|)$ soit, dans le pire des cas, $O(|X|^3)$, il domine la complexité de l'algorithme 4.1.

[9] *Cf.* section 2.2.2

Chapitre 5

Interprétation d'un $\mathcal{R}$-flot

Sommaire

> *J'aime mes solutions depuis longtemps, mais ne sais pas encore comment je peux y arriver*
> – Extrait de *Le Cri d'Archimède*, Arthur Koestler, romancier hongrois

Au chapitre 4, nous avons montré comment calculer un $\mathcal{R}$-flot valide maximal à partir d'un GCS. Nous avons déjà évoqué[1] le fait qu'un $\mathcal{R}$-flot peut être interprété géométriquement sous la forme d'un repère. Dans ce chapitre, nous explicitons plus précisément les interprétations possibles d'un $\mathcal{R}$-flot : la section 5.1 détaille comment passer d'une paramétrisation sous forme de $\mathcal{R}$-flot à une paramétrisation sous forme de repère ; la section 5.2 explique comment, sous quelles conditions et avec quelles limites on peut déduire un plan de construction d'un $\mathcal{R}$-flot ; à partir de ces éléments, la section 5.3 présente des heuristiques de détermination de la qualité d'un $\mathcal{R}$-flot ; enfin, la section 5.4 détaille la problématique de l'interprétation numérique d'un tel plan de construction.

[1] Voir par exemple les figures 4.7, 4.8, 4.11 et 4.12.

5.1 Interprétation géométrique d'un $\mathcal{R}$-flot

Le $\mathcal{R}$-flot indique, par les degrés de repère, combien de degrés de liberté doivent être fixés pour les différentes entités géométriques. En revanche, il n'indique pas directement comment s'interprète géométriquement ce retrait de degrés de liberté.

Lorsqu'un nœud de repère est saturé, l'entité géométrique correspondante doit être entièrement fixée. L'interprétation est alors immédiate : les coordonnées de cette entité géométriques doivent être données comme paramètres. Lorsqu'une entité géométrique a un degré de repère nul, là aussi, l'interprétation est immédiate : les degrés de liberté de cette entité sont tous consommés par des contraintes, *i.e.* cette entité se construit à partir des contraintes sans qu'aucune de ses coordonnées n'ait à être explicitement donnée. Si tous les degrés de repère sont nuls, cela signifie que le système est bien contraint *modulo* l'identité.

Lorsqu'une entité a un degré de repère non nul mais qu'elle n'est pas saturée – elle est restreinte – plusieurs interprétations sont possibles. Par exemple, dans le plan, si le nœud n_p correspondant à un point p a un degré de repère de 1, on peut proposer, comme interprétation immédiate, de fournir l'abscisse x_p de p ou son ordonnée y_p. Si n_p est saturé de par un arc $n_c \xrightarrow{1} n_p$, avec c une contrainte de distance avec un point p', une interprétation du degré de repère peut être que la direction de p à p' doit être fournie : on est en effet dans le cas où le point p est sur un cercle de centre p'. Si c est une contrainte d'angle entre trois points, l'interprétation du degré de repère peut être de fournir la distance entre p et le point médian du triplet.

De nombreuses autres interprétations sont possibles, et leur nombre va croissant avec la taille de l'univers géométrique (dimension, contraintes considérées, entités géométriques considérées). Il est important, au moment de la définition de l'univers géométrique, de prendre en compte les différentes interprétations possibles de la restriction d'une entité.

En outre, certaines interprétations sont préférables à d'autres. Nous détaillons ces points plus tard[2], mais il est possible d'ores et déjà de réaliser que certaines interprétations peuvent mener à des échecs de construction géométrique. Ainsi, si l'on interprète la restriction d'un degré de repère d'un point p par la donnée de son abscisse et que p est contraint par une contrainte de distance de k avec le point p', alors toutes les valeurs de x_p plaçant p à une distance de p' supérieure à k mènent à un échec de la construction. À l'inverse, interpréter la restriction comme la donnée de la direction de la droite (pp') ne peut en aucun cas échouer.

[2] Voir section 5.4, p. 126

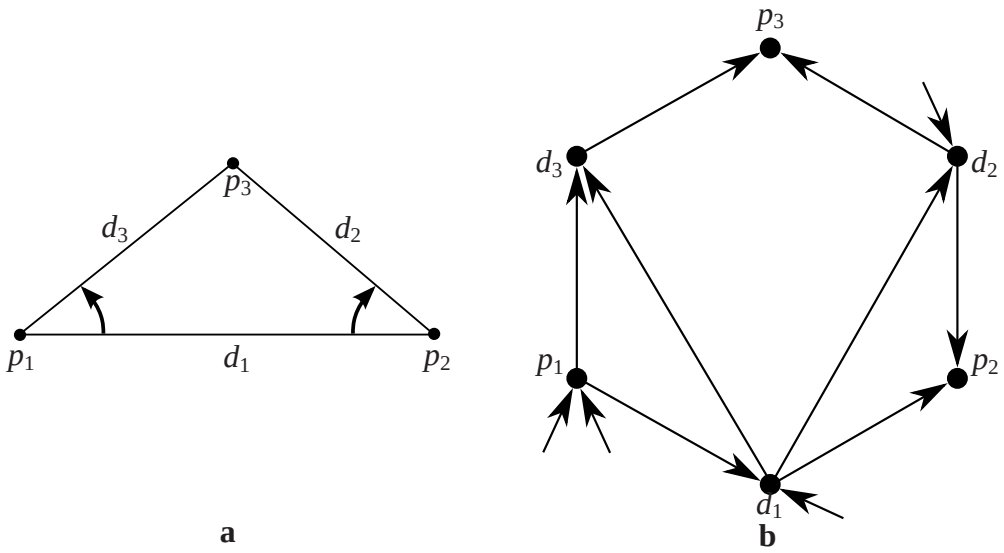


Figure 5.1 - Triangle invariant par similitude avec six contraintes d'incidence et deux contraintes d'angle droite-droite ; **a** : esquisse ; **b** : représentation de C d'un $\mathcal{R}$-flot valide maximal.

De plus – là encore, nous détaillons ce point plus loin[3] – une interprétation d'un $\mathcal{R}$-flot valide peut être un *repère sur-contraignant*.

Définition 50. Repère sur-contraignant : *Soit $\mathcal{S} = (C, X, A)$ et $\mathcal{S}' = (C', X', A')$ deux GCS tels que $\mathcal{S}'$ est bien contraint* modulo *l'identité et que $X' \cup A' \subseteq X \cup A$. $\mathcal{S}'$ est un repère sur-contraignant de $\mathcal{S}$ si $\mathcal{S}' + \mathcal{B}_{\mathcal{S}'}(\mathcal{S})$ est génériquement sur-contraint.*

Dans notre cas, il s'agit d'une paramétrisation qui introduit des informations redondantes et mène donc à un système génériquement sur-contraint (et sous-contraint dans le cas probabilistiquement nul où il n'est pas sur-contraint). Par exemple, considérons le GCS de la figure 5.1 : il représente un système bien-contraint *modulo* les similitudes avec trois points et trois droites, contraints par six contraintes d'incidence et deux contraintes d'angle. La figure 5.1**b** donne, en utilisant la représentation de C , un $\mathcal{R}$-flot valide maximal. Il indique qu'il faut fixer le point p_1 et restreindre les droites d_1 et d_2.

Une interprétation de ce $\mathcal{R}$-flot pourrait être de fournir les coordonnées de p_1 et les directions de d_1 et de d_2. Toutefois, l'angle entre d_1 et d_2 étant contraint à une valeur donnée, fournir les deux directions sur-contraint le GCS. Une autre interprétation – valide quant à elle – est de fournir les coordonnées de p_1, la direction de d_1 et la position, sur d_1, de l'intersection de d_1 et d_2.

Ce type de problèmes ne peut pas être détecté *a priori* par une analyse du graphe

[3] Voir section 6.1, p. 131

de $\mathcal{R}$-flux et requiert l'introduction de connaissances géométriques, par exemple à travers un système à base de règles. Même cette approche n'est pas sans défaut si les règles ne sont pas suffisantes. Nous expliquons à la section 6.3 comment il est possible d'étendre la méthode du témoin pour résoudre ces problèmes.

5.2 Déduction d'un plan de construction par blocs

De manière classique dans les travaux de résolution de systèmes de contraintes géométriques à base de graphe [Hen92, LM96b], il est possible d'opérer une triangularisation par bloc du GCS et d'en déduire un plan de construction par blocs, à partir du calcul d'un graphe réduit dont les arcs ne sont pas valués. Ce graphe réduit se calcule à partir d'un graphe orienté que l'on déduit du GCS.

Définition 51. Graphe orienté non valué d'un $\mathcal{R}$-flot : *Un graphe orienté non valué $H(r)$ d'un GCS $\mathcal{S}$ se construit à partir du graphe de $\mathcal{R}$-flux r de $\mathcal{S}$:*
- *les nœuds de $H(r)$ sont les nœuds de r, à l'exception de n_s, n_p et des nœuds de repère de degré 0 ;*
- *s'il y a un arc avec une valuation non nulle entre les nœuds n_1 et n_2 dans r, il y a un arc bi-directionnel entre n_1 et n_2 dans $H(r)$;*
- *s'il y a un arc avec une valuation nulle entre les nœuds n_x et n_c dans r, avec x une entité géométrique et c une contrainte, il y a un arc orienté de n_x vers n_c dans $H(r)$.*

La direction des arcs dans le graphe orienté non valué a une sémantique de dépendance pour la construction : s'il y a un arc du nœud d'entité n_x vers le nœud de contrainte n_c, cela signifie que l'entité géométrique x devra être construite avant de pouvoir utiliser la contrainte c pour construire la ou les entité(s) contraintes par c dont le nœud est lié à n_c par un arc à valuation non nulle dans le graphe de $\mathcal{R}$-flux. Notons que la direction des arcs de la représentation de C donne cette même sémantique.

À partir du graphe orienté non valué $H(r)$ d'un $\mathcal{R}$-flot r, on peut construire un *graphe réduit* de r en calculant les Composantes Fortement Connexes (CFCs) de $H(r)$.

Définition 52. Graphe réduit d'un $\mathcal{R}$-flot : *Un graphe réduit $H'(r)$ d'un GCS $\mathcal{S}$ se construit à partir du graphe de $\mathcal{R}$-flux r de $\mathcal{S}$:*
- *les nœuds de $H'(r)$ sont les CFCs de $H(r)$;*
- *s'il y a un arc du nœud n_1 vers le nœud n_2 dans $H(r)$ et que n_1 et n_2 appartiennent à deux CFCs différentes, alors il y a un arc du nœud n'_1 vers le nœud n'_2 de $H'(r)$, avec n'_i le nœud correspondant à la CFC de n_i.*

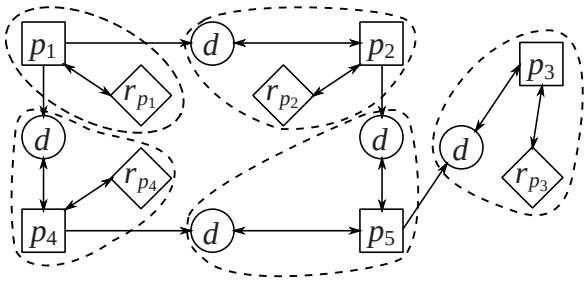


Figure 5.2 - Graphe orienté non valué du $\mathcal{R}$-flot de la figure 4.5**f** ; les pointillés indiquent les CFCs et le graphe réduit

Le graphe réduit est un Graphe Orienté Acyclique (DAG) : s'il y a un cycle dans le graphe réduit, cela veut dire qu'il y a une CFC plus grande qu'un nœud du graphe, ce qui est incompatible avec la définition du graphe réduit.

Là encore, les nœuds et les arcs ont une sémantique. La direction des arcs a toujours la même sémantique : celle de la dépendance pour la construction. Les CFCs de $H(r)$, qui deviennent des nœuds dans $H'(r)$, ont quant à elles la sémantique de la nécessité d'une construction atomique. En effet, l'existence d'un cycle passant par deux nœuds de $H(r)$ indique que les nœuds en question doivent être utilisés simultanément dans la construction : il n'est pas possible de déterminer un ordre de construction avec le $\mathcal{R}$-flot. Autrement dit, si les solveurs considérés ne sont pas capables de résoudre les CFCs, cela signifie que le graphe réduit ne permet pas le calcul d'un plan de construction satisfaisant. Lorsque plusieurs entités géométriques font partie de la même composante connexe, on obtient donc un plan de construction par blocs.

La figure 5.2 montre le graphe $H(r)$ avec r le $\mathcal{R}$-flot de la figure 4.5**f**. Les pointillés indiquent les CFCs et permettent donc de visualiser également $H'(r)$. Ce graphe réduit indique le plan de construction suivant :

1. fixer le point p_1
2. fixer la direction p_1p_2 (appelons-la v_1)
3. construire p_2 comme l'intersection d'un cercle de centre p_1 et de la droite de direction v_1 passant par p_1
4. fixer la direction p_1p_4 (appelons-la v_2)
5. construire p_4 comme l'intersection d'un cercle de centre p_1 et de la droite de direction v_2 passant par p_1
6. construire p_5 comme l'intersection d'un cercle de centre p_2 et d'un cercle de centre p_4
7. fixer la direction p_5p_3 (appelons-la v_3)
8. construire p_3 comme l'intersection d'un cercle de centre p_5 et de la droite de direction v_3 passant par p_5

5.3 Qualité d'un $\mathcal{R}$-flot

À partir des éléments fournis plus haut, nous pouvons établir certains critères de qualité d'un $\mathcal{R}$-flot, en fonction de ce que l'on peut en déduire. Nous explicitons ainsi la notion de stratégie rapidement abordée dans les algorithmes 4.2 et 4.4[4].

5.3.1 Plan de construction

À un même GCS on peut associer plusieurs $\mathcal{R}$-flux. Il n'y a donc pas un unique graphe réduit par système et donc plusieurs plans de construction sont calculables pour un même GCS.

Or, certains des plans de construction peuvent être par blocs, c'est-à-dire nécessiter la résolution atomique, par un solveur externe, d'un sous-système à plusieurs entités géométriques.

La figure 5.3 montre deux fois la même esquisse d'un pentagone rigide, avec deux repères différents (**a** et **b**). Elle montre également, avec la représentation de C , les graphes orientés non valués correspondants (**c** et **d**). On y voit que le repère de la figure 5.3**a** induit un plan de construction strict, puisque le graphe orienté non valué de la figure 5.3**c** n'a aucune CFC contenant plusieurs entités géométriques. En revanche, le repère de la figure 5.3**b** induit le graphe de la figure 5.3**d**, dans lequel un cycle créé une CFC contenant p_3, p_4 et p_5. Le plan de construction par blocs indique qu'après avoir fixé p_1 et la direction p_1p_2, on construit p_2 puis qu'il faut, d'un coup, construire p_3, p_4 et p_5 à partir des distances p_1-p_5 et p_2-p_3.

Un premier critère de qualité d'un $\mathcal{R}$-flot est donc la possibilité d'en déduire un plan de construction strict. Il existe toutefois des systèmes pour lesquels il n'existe aucun $\mathcal{R}$-flot permettant la déduction d'un plan de construction strict. Ainsi, si l'on considère l'hexagone $K_{3,3}$ de la figure 1.12[5], on peut lui associer quatre $\mathcal{R}$-flots différents[6], représentés à la figure 5.4, menant tous à l'existence d'un plan de construction par blocs. Il est possible de les ordonner par la taille du sous-système à résoudre atomiquement, mais il reste toujours un tel sous-système à au moins quatre entités géométriques.

[4] *Cf.* pp. 99 et 101

[5] *Cf.* p. 30

[6] Il y a bien évidemment plus de quatre $\mathcal{R}$-flots possibles, mais de par le fait que les triplets de points non connectés (dans le graphe de contrainte) ont les mêmes caractéristiques, les autres $\mathcal{R}$-flots sont tous identiques à l'un des quatre $\mathcal{R}$-flots de la figure 5.4, à un renommage près

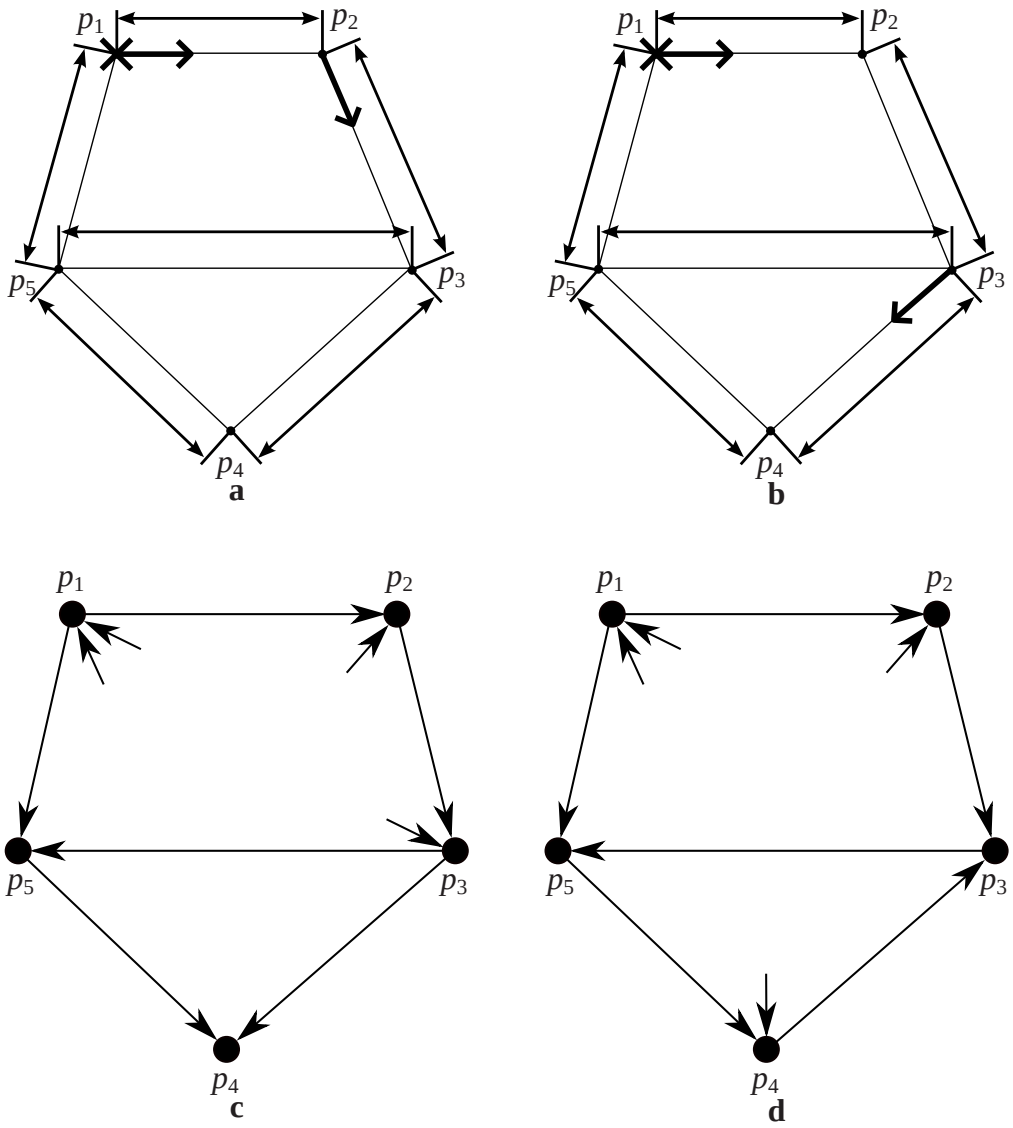


Figure 5.3 - Deux $\mathcal{R}$-flots pour un même GCS, l'un menant à un plan de construction strict (**c**) et l'autre à un plan de construction par blocs (**d**).

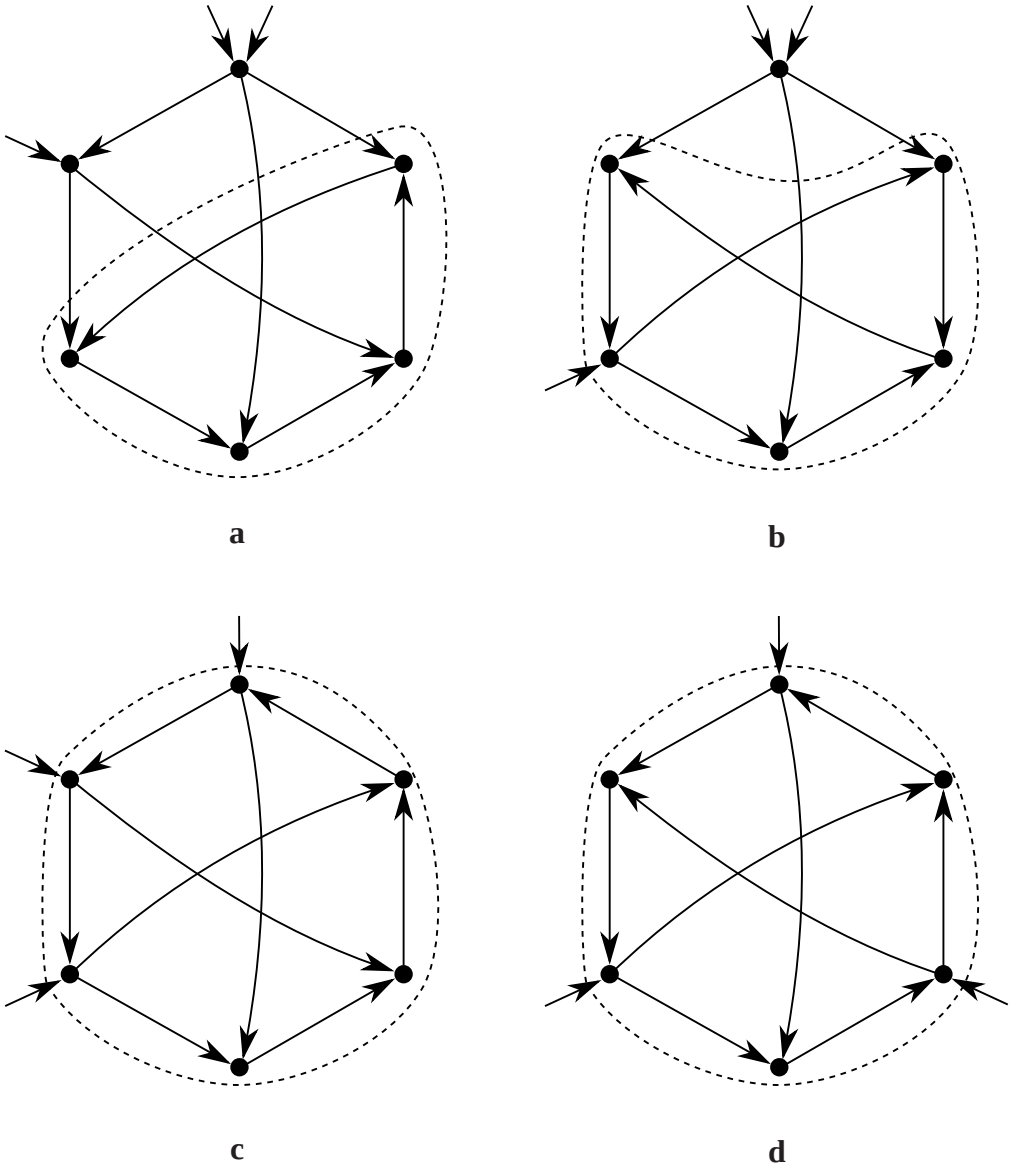


Figure 5.4 - Représentation de C des quatre $\mathcal{R}$-flots possibles pour le GCS de la figure 1.12 : les autres $\mathcal{R}$-flots possibles sont isomorphes à l'un de ces quatre $\mathcal{R}$-flots (*cf.* note 6) ; les pointillés indiquent les CFCs contenant plusieurs entités géométriques.

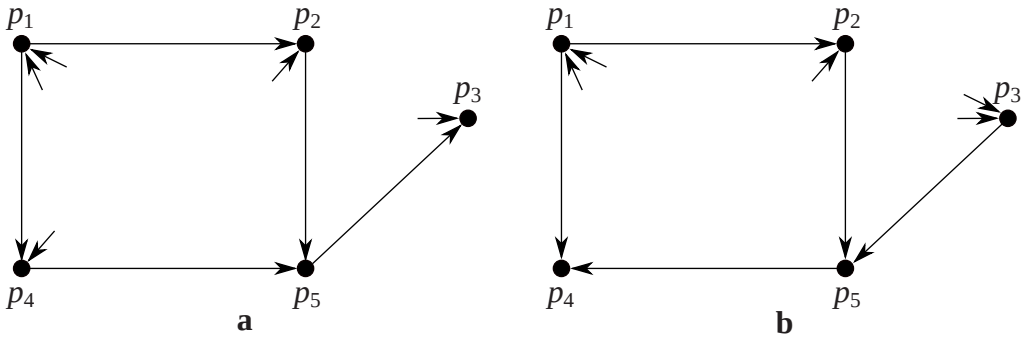


Figure 5.5 - Représentations de C des $\mathcal{R}$-flots de la figure 4.7 (**a**) et de la figure 4.8 (**b**)

5.3.2 Validité des paramètres

Les figures 4.7 et 4.8[7] donnent deux repères différents d'un même GCS. Nous donnons, à la figure 5.5, une représentation de C des $\mathcal{R}$-flots correspondants[8].

Dans un cas, le repère est constitué d'un point et de trois directions, dans l'autre de deux points et d'une direction. Bien que les deux repères soient géométriquement valides, les chances d'échouer à construire une solution numérique à partir des valuations des paramètres sont plus grandes avec le repère de la figure 5.5**b** : si les paramètres fournis font que les points fixés (p_1 et p_3) sont trop éloignés et/ou si la direction p_1p_2 fait que les points p_2 et p_3 sont trop éloignés, alors la construction du point p_5 est impossible et, donc, celle de p_4 également. Si cette construction échoue, l'ensemble des cinq paramètres (x_{p_1}, y_{p_1}, $v_{p_1p_2}$, x_{p_3}, y_{p_3}) doit être remis en question. Avec le $\mathcal{R}$-flot de la figure 5.5**a**, les points p_1, p_2 et p_4 peuvent être construits pour n'importe quelle valuation des paramètres du repère et un échec est uniquement possible à la construction de p_5 si les points p_2 et p_4 sont trop éloignés, ce qui signifie que seuls deux paramètres (la direction p_1p_2 et p_1p_4) sont à remettre en question.

En étudiant des exemples, nous avons mis au point plusieurs heuristiques pour éviter les $\mathcal{R}$-flots menant à des risques de valuations invalides des paramètres :

1. pénaliser les $\mathcal{R}$-flots avec plusieurs nœuds de repère saturés et, au contraire, favoriser les $\mathcal{R}$-flots où les degrés de repère peuvent être interprétés dans le contexte des contraintes terminant de saturer l'entité géométrique ;
2. favoriser les $\mathcal{R}$-flots qui induisent un plan de construction où les éléments de repère sont utilisés « tôt » dans la construction ;
3. favoriser les $\mathcal{R}$-flots menant à des constructions n'échouant que dans les cas dégénérés (*e.g.* la construction d'une droite passant par deux points n'échoue

[7] *Cf.* p. 104
[8] La figure 5.5**a** est en fait une copie de la figure 4.2

que si les deux points sont confondus) par rapport à ceux menant à des constructions susceptibles d'échouer de par les valeurs des paramètres (*e.g.* l'intersection d'un cercle et d'une droite qui n'est pas contrainte à passer par un point à l'intérieur du 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 l'avantage d'éviter les repères où plusieurs points sont fixés, ce qui implique virtuellement l'ajout de contraintes de distance entre ces points. Il faut y préférer des repères où des directions sont fixées, c'est-à-dire virtuellement ajouter des angles. Dans la mesure où les contraintes d'angle sont invariantes sous l'action des similitudes, elles réduisent moins les libertés du système que des contraintes de distance, dont le plus grand groupe d'invariance est les déplacements.

Concernant l'heuristique 2, nous définissons la précocité d'utilisation d'un élément de repère comme suit :

Définition 53. Degré de précocité d'un élément de repère dans un graphe réduit : *Soient r un $\mathcal{R}$-flot et D le DAG du graphe réduit de $H'(r)$. Le degré de précocité d'un nœud de repère r_x lié au nœud n_x est la longueur du plus long chemin dans D reliant un nœud de repère r_y à n_x, r_x et r_y n'étant pas nécessairement distincts.*

Le degré de précocité d'un nœud correspond à son rang dans le tri topologique du graphe dont les sommets sont ceux du DAG du graphe réduit de $H'(r)$ et dont les arcs sont les symétriques des arcs dudit DAG. Nous calculons ce degré de précocité au moyen de l'algorithme 5.1 qui propage le degré de précocité à partir des nœuds de repère. Dans cet algorithme, on considère que chaque nœud du DAG dispose de deux étiquettes entières : p, son degré de précocité et v, qui compte le nombre de parents du nœud qui ont obtenu leur degré de précocité définitif. On note $p[x]$) (resp. $v[x]$) l'étiquette p (resp. v) du nœud x. Rappelons que de par la présence dans le graphe réduit d'un nœud unique pour représenter une CFC du graphe orienté non valué, un nœud peut avoir plusieurs nœuds de repère parents.

Cet algorithme parcourt chaque nœud du graphe réduit une fois dans la première boucle (ligne 2), une seconde fois dans la seconde boucle (ligne 6). La boucle imbriquée (ligne 8) parcourt chaque arc du graphe réduit une et une seule fois. La complexité de l'algorithme est donc en $\mathcal{O}(|V| + |E|)$ avec V l'ensemble des nœuds du graphe réduit et E l'ensemble des arcs. Sachant que $|V| \leq ddl(\mathcal{S})$ et $|E| \leq ddr(\mathcal{S})$ et puisque dans le cas non sur-contraint, $ddr(\mathcal{S}) < ddl(\mathcal{S})$, cet algorithme est dans le pire des cas en $\mathcal{O}(ddl(\mathcal{S}))$.

Entrées :
D : plan de construction (DAG)`// strict ou par blocs`
Résultat :
D avec les nœuds étiquetés par leur précocité

```
P ← pile vide
pour chaque nœud n de D faire
    v[n] ← 0
    p[n] ← 0
    si n est un nœud de repère alors push(P, n)
tant que P n'est pas vide faire
    n ← pop(P)
    pour chaque nœud x fils de n dans D faire
        v[x] ← v[x] + 1
        p[x] ← max(p[x], p[n] + 1)
        si v[x] est égal au nombre de parents de x alors push(P, x)
pour chaque nœud de repère r faire
    p[r] ← p[n] − 1 avec n le nœud fils de r
```

Algorithme 5.1: Calcul du degré de précocité des nœuds de repère dans un plan de construction

5.3.3 Stratégies de calcul de $\mathcal{R}$-flots

Avec les éléments de qualité d'un $\mathcal{R}$-flot évoqués en 5.3.1 et 5.3.2, on peut élaborer des stratégies de calcul de $\mathcal{R}$-flots. Dans les algorithmes 4.2 et 4.4, ces stratégies sont utilisées pour le choix du chemin améliorant et, donc, dans le choix de la paramétrisation.

De par notre incapacité à donner plus d'importance à l'une des heuristiques de la section 5.3.2 qu'à une autre et donc à établir une mesure de qualité globale, nous avons opté pour une comparaison multi-critère utilisant les fronts de P [9], c'est-à-dire un ordre partiel dans lequel un $\mathcal{R}$-flot est meilleur qu'un autre s'il obtient un meilleur score sur l'ensemble des heuristiques considérées. S'il n'est pas possible d'ordonner deux $\mathcal{R}$-flots, ils sont considérés comme de même qualité.

Il n'y a qu'un critère de qualité qui, d'après nous, est prioritaire sur les autres : lorsqu'un $\mathcal{R}$-flot induit un plan de construction par blocs, il est considéré comme de moins bonne qualité qu'un $\mathcal{R}$-flot menant à un plan de construction strict.

[9] Ainsi nommés en l'honneur de V. P , qui en introduisit la notion dans le contexte de la théorie économique [Par09].

Notons que notre notion de stratégie de choix est loin d'être idéale, car elle peut mener à des *maxima* locaux : en effet, nous considérons à chaque ajout de contrainte le meilleur chemin possible à partir du $\mathcal{R}$-flot calculé au précédent ajout de contrainte. Il est possible que certains excellents $\mathcal{R}$-flots ne soient accessibles qu'en passant, dans les étapes intermédiaires, par des $\mathcal{R}$-flots de mauvaise qualité. Un calcul exhaustif de tous les $\mathcal{R}$-flots serait trop coûteux.

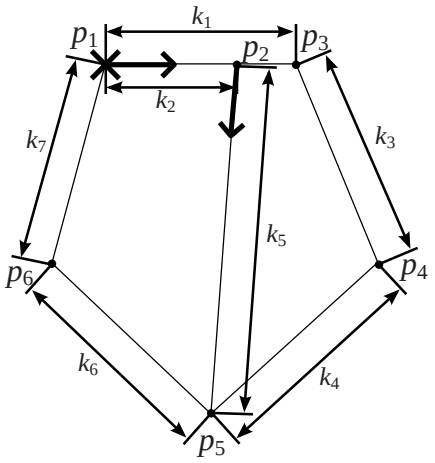
Il est important de garder en tête que notre notion de stratégie est basée sur des heuristiques et qu'il est possible de trouver des exemples où elles ne mènent pas à un $\mathcal{R}$-flot satisfaisant. Heureusement, dans la grande majorité des cas[10], les valeurs des paramètres prises sur l'esquisse mènent à des constructions valides, même avec des $\mathcal{R}$-flots considérés comme déconseillés par nos heuristiques. Dans les cas où les valeurs prises sur l'esquisse ne sont pas bonnes non plus, des méthodes numériques permettent de calculer de nouvelles valeurs à partir des valeurs de l'esquisse.

En outre, il faut également garder à l'esprit que la qualité sémantique du $\mathcal{R}$-flot, en tant que retour visuel pour l'utilisateur sur les libertés du système, est importante. Cette qualité est quant à elle difficile à mesurer sans ajouter d'information sur la sémantique des sous-systèmes, par exemple au moyen de *features* [HJA98, OSMA01, SOZA06]. Des paramétrisations considérées comme de bonne qualité par nos heuristiques sont en réalité inadaptées par rapport aux besoins de l'utilisateur. Ainsi, le $\mathcal{R}$-flot représenté à la figure 4.11**d**[11] est un excellent $\mathcal{R}$-flot d'après nos heuristiques, mais son utilité en tant que retour utilisateur est faible. Les figures 4.11**e** ou 4.11**i** donnent des $\mathcal{R}$-flots de moindre qualité d'après nos heuristiques, mais plus utiles pour l'utilisateur : le premier indique que l'hexagone du dessus a toujours 6 degrés de liberté, la seconde indique que la position de l'hexagone supérieur est entièrement déterminée par la position de l'hexagone inférieur et par les longueurs des six cylindres hydrauliques.

5.4 Interprétation numérique

Resituons tout d'abord l'interprétation numérique d'un $\mathcal{R}$-flot dans notre processus général de résolution[12]. L'algorithme 4.1 fournit un $\mathcal{R}$-flot duquel nous déduisons une paramétrisation du GCS (étape 1). Le GCS et sa paramétrisation sont fournis à un algorithme qui en déduit un plan de construction, strict ou par blocs (étape 2). Dans cette section, nous discutons l'étape 3 : l'interprétation numérique du plan de construction.

[10] ... que nous avons essayés ...
[11] *Cf.* p. 107
[12] *Cf.* p. 88



unknowns
p_1, p_2, p_3, p_4, p_5, p_6
parameters
k_1, k_2, k_3, k_4, k_5, k_6, k_7
constraints
dist_pp(p_1, p_3, k_1)
dist_pp(p_1, p_2, k_2)
dist_pp(p_3, p_4, k_3)
dist_pp(p_4, p_5, k_4)
dist_pp(p_2, p_5, k_5)
dist_pp(p_5, p_6, k_6)
dist_pp(p_6, p_7, k_7)
on_pl(p_2, line(p_1, p_3))

Figure 5.6 - Pentagone articulé fait de deux chaînes fermées entre-mêlées avec six points, sept distances et une incidence point-droite : esquisse (à gauche) et énoncé (à droite). Le repère indiqué sur l'esquisse est une interprétation géométrique du $\mathcal{R}$-flot de la figure 5.7.

5.4.1 Interprétation numérique du plan de construction

Considérons l'exemple 5.6 : il s'agit d'un pentagone articulé ($p_1p_3p_4p_5p_6$) avec un point supplémentaire (p_2) sur un des segments (p_1p_3), avec une barre rigide entre ce point supplémentaire et le point opposé (p_5). Cette barre crée deux chaînes fermées ($p_1p_2p_5p_6$ et $p_2p_3p_4p_5$). La figure 5.7 donne une représentation de C d'un $\mathcal{R}$-flot valide maximal pour ce GCS, dont une interprétation géométrique est le repère représenté sur la figure 5.6. À partir de ce $\mathcal{R}$-flot, on peut établir le plan de construction fourni à la table 5.1.

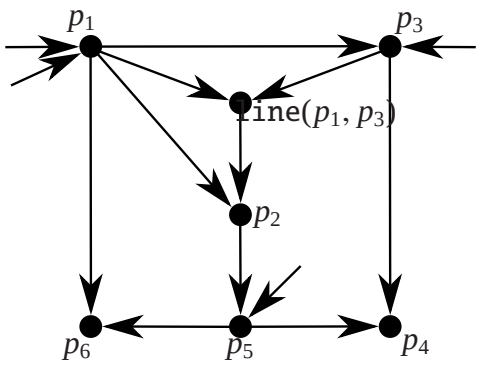


Figure 5.7 - Un $\mathcal{R}$-flot valide maximal pour le GCS de la figure 5.6

Tableau 5.1 - Plan de construction déduit du $\mathcal{R}$-flot de la figure 5.7

1. fixer p_1
2. fixer la direction p_1p_3
3. cercle c_1 = `mk-circle(`p_1`,` k_1`)`
4. droite d_1 = `mk-line(`p_1`,` $\overrightarrow{p_1p_3}$`)`
5. p_3 = `inter-cl(`c_1`,` d_1`)`
6. droite d = `inc-pp(`p_1`,` p_3`)`
7. cercle c_2 = `mk-circle(`p_1`,` k_2`)`
8. p_2 = `inter-cl(`c_2`,` d`)`
9. fixer la direction p_2p_5
10. droite d_2 = `mk-line(`p_2`,` $\overrightarrow{p_2p_5}$`)`
11. cercle c_3 = `mk-circle(`p_2`,` k_5`)`
12. p_5 = `inter-cl(`c_3`,` d_2`)`
13. cercle c_4 = `mk-circle(`p_3`,` k_3`)`
14. cercle c_5 = `mk-circle(`p_5`,` k_4`)`
15. p_4 = `inter-cc(`c_4`,` c_5`)`
16. cercle c_6 = `mk-circle(`p_1`,` k_7`)`
17. cercle c_7 = `mk-circle(`p_5`,` k_6`)`
18. p_6 = `inter-cc(`c_6`,` c_7`)`

Quatre paramètres doivent être fournis : les deux coordonnées du point p_1 pour l'étape 1, la direction p_1p_3 pour l'étape 2 et la direction p_2p_5 à l'étape 9. En prenant comme valeurs pour ces paramètres celles de l'esquisse, on a la direction p_1p_3 horizontale et à peu près orthogonale à la direction p_2p_5.

Certaines étapes du plan de construction ont plusieurs solutions. Ainsi, l'étape 8 consiste en l'intersection d'un cercle avec une droite passant par le centre de ce cercle, ce qui signifie qu'il y a deux solutions. De façon similaire, l'étape 15 consiste en l'intersection de deux cercles : il peut y avoir deux, une ou aucune solutions.

Interpréter numériquement le plan de construction revient donc à traverser un DAG, avec un niveau par étape, une unique racine à l'étape 1 et un fils par possibilité de construction pour chaque nœud. La question de la recherche d'un chemin dans ce DAG menant à des solutions proches visuellement de l'esquisse a été étudiée par B et S [BS03], E *et al.* [EV01, EVSD00a, EVSD00b, EVSMD03], S *et al.* [SAZK06] et van der M et B [vdMB05, vdMB06].

5.4.2 Échecs d'une construction

Selon la paramétrisation et son interprétation géométrique, certaines étapes de l'interprétation numérique du plan de construction peuvent échouer.

Ainsi, considérons les étapes 15 et 18 du plan de construction de la table 5.1. Les deux constructions dépendent directement des directions données en paramètre, puisque l'angle $\angle(\overrightarrow{p_1p_3}, \overrightarrow{p_2p_5})$ change les positions relatives des deux centres des cercles à intersecter. Ainsi, selon les valeurs associées aux contraintes de distance (k_1 à k_7 et en particulier les paramètres k_3 à k_7), certaines valeurs de cet angle peuvent mener à un échec de la construction, *e.g.* si $k_6 + k_7 < k_5$ et que les directions p_1p_3 et p_2p_5 sont orthogonales.

Dans un tel cas, la partie du solveur chargée de l'interprétation du plan de construction peut fournir une mesure d'erreur. Pour l'exemple mentionné ci-dessus, en cas d'erreur à l'étape 18 parce que les cercles c_6 (de centre p_1 et de rayon k_7) et c_7 (de centre p_5 et de rayon k_6) ne s'intersectent pas, la mesure d'erreur est la plus courte distance entre c_6 et c_7 (*i.e.* $|\overrightarrow{p_1p_5}| - (k_7 + k_6)$).

En traversant le DAG de construction à partir de l'étape qui échoue jusqu'à la racine, le solveur construit la liste des paramètres de repère $r_1, \ldots, r_n$ qui affectent la mesure d'erreur. Il est alors possible de considérer une fonction d'erreur $d(r_1, \ldots, r_n)$ et de chercher des valeurs telles que $d(r_1, \ldots, r_n) \leq 0$. Pour cela, une application de la méthode de N -R est possible.

Avec les nouvelles valeurs, nous réinterprétons le plan de construction. Si un nouvel échec apparaît, nous recommençons le processus de trouver les paramètres à modifier et de calculer de nouvelles valeurs pour modifier la valeur d'erreur. Il faut alors vérifier, au fur et à mesure des itérations de N -R , que les fonctions d'erreur considérées dans les étapes précédentes ne reprennent pas une valeur positive. Dans notre exemple, cela peut arriver par exemple si $k_7+k_6 < k_5$ et $k_3+k_4 < k_5$: corriger la direction p_2p_5 pour éviter un échec à l'étape 15 peut mener à un échec à l'étape 18 et inversement.

L'approche de rattrapage numérique à partir des valeurs de l'esquisse demande à être approfondie pour éviter ce type d'erreurs. Une approche plus globale afin d'éviter des *minima* locaux[13] serait également à envisager.

[13]Par exemple en s'inspirant des travaux de J -A *et al.* [JALSR02, JALSR03, LBYJA04, LSRGJA05] sur les algorithmes génétiques

C 6

G -

Sommaire

> *Ce qui fait d'un problème un problème, c'est de contenir une contradiction*
> – José Ortega y Gasset, philosophe et homme politique espagnol

L'algorithme de paramétrisation combinatoire incrémental décrit au chapitre 4 part de l'hypothèse forte que le GCS à résoudre n'est pas génériquement sur-contraint. Ce chapitre rappelle brièvement quels sont les différents cas où une sur-constriction peut apparaître dans nos algorithmes (section 6.1) et montre que les approches connues dans la littérature ne sont pas directement appliquables à ces problèmes (section 6.2). Enfin, nous proposons des extensions de la méthode du témoin (section 6.3) pour résoudre ces problèmes.

6.1 Éléments de problématique

Trois problèmes importants de sur-constriction se posent avec notre démarche :

1. l'ajout d'une contrainte peut mener à la sur-constriction de tout ou partie du GCS,
2. une mauvaise interprétation du $\mathcal{R}$-flot peut mener à sur-contraindre le système en fixant deux éléments dépendants,
3. certaines valuations des paramètres de repère peuvent empêcher l'existence de solutions.

Les deux premiers sont des problèmes de sur-constriction générique et nous proposons des solutions plus loin. Concernant le troisième, que nous venons d'évoquer en section 5.4.2, nous l'abordons à nouveau parmi les perspectives de nos travaux[1].

6.1.1 Contraintes redondantes

Au chapitre 4 et tout particulièrement à la section 4.3.3, nous avons montré que l'algorithme incrémental de paramétrisation[2] est sensible à la sur-constriction générique : il ne permet pas de détecter l'ajout d'une contrainte redondante avec les contraintes déjà connues.

Ce problème se pose avec des systèmes élémentaires. Par exemple, la figure 6.1**a** montre un GCS composé de deux points et de deux contraintes. Les figures 6.1**b** et 6.1**c** montrent comment se comporte notre algorithme de paramétrisation à l'ajout de la seconde contrainte. Une autre possibilité de modification du $\mathcal{R}$-flot serait de retourner la première contrainte de distance et d'obtenir une paramétrisation où les deux points sont restreints mais aucun n'est fixé.

6.1.2 Repère sur-contraignant

Les théorèmes 11 et 12 nous assurent qu'à tout $\mathcal{R}$-flot valide maximal d'un système non génériquement sur-contraint on peut associer une interprétation géométrique valide. Rappelons toutefois, comme nous l'avons indiqué à la section 5.1, sur l'exemple de la figure 5.1, que toutes les interprétations géométriques d'un $\mathcal{R}$-flot valide maximal ne sont pas valides. Dans cet exemple, un $\mathcal{R}$-flot pour un système bien-contraint *modulo* les similitudes consiste notamment à restreindre d'un degré de liberté chacune des deux droites. L'interprétation consistant à fixer la direction de ces deux droites sur-contraint génériquement le GCS, puisque dans un système bien-contraint *modulo* les similitudes, tous les angles entre deux droites sont connus.

[1] *Cf.* p. 191
[2] *Cf.* algorithmes 4.1 et 4.4 pp. 97 et 101

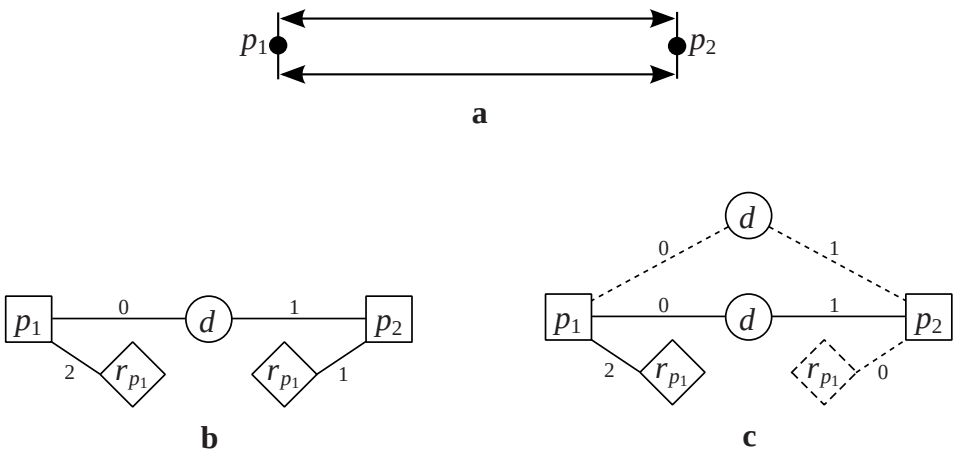


Figure 6.1 - Système génériquement sur-contraint à deux points et deux distances : esquisse (**a**) ; $\mathcal{R}$-flot avant ajout de la seconde contrainte (**b**) ; $\mathcal{R}$-flot après ajout de la seconde contrainte (**c** – les pointillés indiquent les modifications).

Un autre exemple est celui de la figure 6.2**b**, qui donne un $\mathcal{R}$-flot maximal valide pour un GCS représentant un triangle rigide dans le plan. Ce $\mathcal{R}$-flot indique qu'il faut restreindre chacun des trois points d'un degré de liberté. Une interprétation possible serait de fournir comme paramètre les abscisses des trois points. Toutefois, dans la mesure où le GCS est rigide, fournir l'abscisse de deux points rend le système bien-contraint *modulo* les translations selon l'axe *Oy*. L'information de la troisième abscisse est donc redondante.

6.2 Problématique nouvelle

La littérature regorge de méthodes visant à détecter la sur-constriction générique d'un GCS[3] et un grand nombre d'entre elles concernent des méthodes de graphe et de flux. Dans cette section, nous expliquons pourquoi ces techniques de détection de la sur-constriction générique ne fonctionnent pas dans un graphe de $\mathcal{R}$-flux.

Les méthodes de la littérature travaillant sur des graphes sont basées sur l'hypothèse de rigidité du GCS. Dans la mesure où le GCS doit être rigide, ces méthodes peuvent comptabiliser les degrés de liberté du système ou de ses sous-systèmes pour rechercher l'existence d'un sous-système ne satisfaisant pas la caractérisation de L [4]. Elles peuvent également chercher à poser des repères pour les déplacements en fixant une barre rigide en 2D ou deux barres avec un point commun en

[3] *Cf.* section 2.2.4 p. 55
[4] *Cf.* p. 32

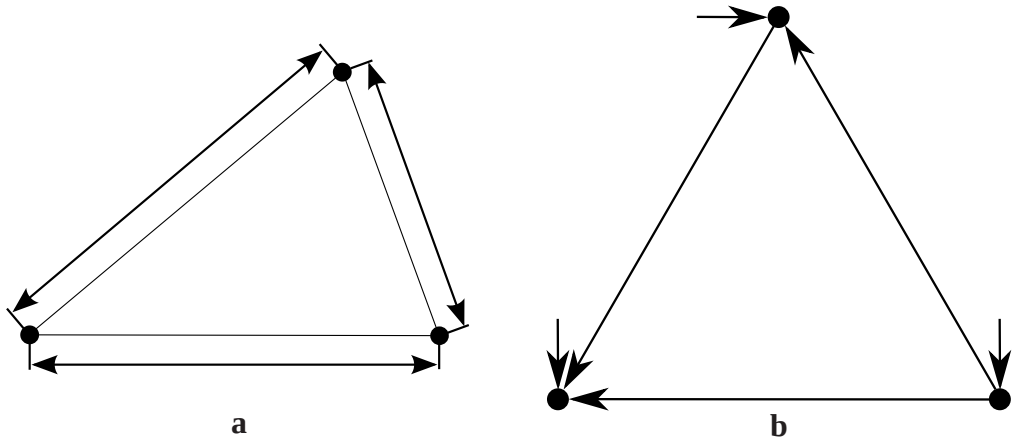


Figure 6.2 - Triangle rigide en 2D : esquisse (**a** – copie de la figure 1.11) et un $\mathcal{R}$-flot valide maximal sous représentation de C (**b**)

3D.

Avec la suppression de cette hypothèse, nous perdons la possibilité de facilement détecter la sur-constriction générique. Les méthodes de H *et al.* [HSY04], N *et al.* [NDB98], J *et al.* [JNT02, JNT03] sont inopérantes dans cette situation. En outre, contrairement à ce que l'on pourrait penser à première vue, une généralisation de l'approche de H [Hen92] ne fonctionne pas non plus. Nous détaillons ici les raisons de cet échec.

Le principe de la méthode de H est de simuler la donnée d'un repère pour les déplacements en fixant un point et en restreignant d'un degré de liberté un second point, lié au premier par une contrainte de distance. Une fois ce repère posé, on calcule un flux maximal. Par définition d'un repère pour les déplacements, si le système est rigide on doit pouvoir trouver un flux maximal valide pour n'importe quel repère. La méthode de H consiste à considérer que si le système est sur-contraint, il existe au moins un repère tel qu'on ne pourra pas trouver un flux maximal valide. La figure 6.3 illustre cette approche avec un exemple de repère qui mène à l'impossibilité de trouver un flux valide : l'arête pointillée ne peut être orientée sans sur-saturer un nœud du graphe.

Nous montrons que la généralisation de l'approche de H aux graphes de $\mathcal{R}$-flux se heurte à la fois à des faux positifs (systèmes considérés comme bien contraints alors qu'ils sont sur-contraints) et à des faux négatifs (systèmes considérés comme sur-contraints alors qu'ils sont bien contraints).

Une extension visant à garder l'idée de poser tous les repères pour les déplacements pourrait classer comme sur-contraint un système invariant par translation, par exemple. En effet, la pose d'un repère pour les déplacements sur une barre connec-

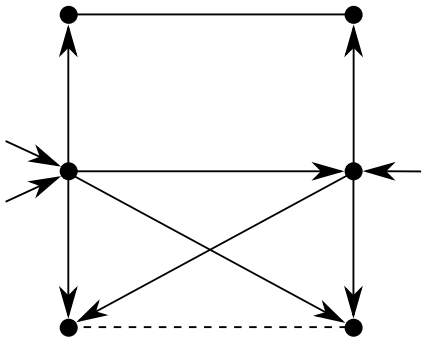


Figure 6.3 - Détection de la sur-constriction générique par la méthode de H : l'arête en pointillés ne peut être orientée sans sur-saturer un nœud du graphe.

tée à un point déjà restreint par l'utilisateur mènera à une sur-constriction de ce segment.

La généralisation en 3D de cette méthode conduit à des faux positifs, puisque la méthode permet la détection de sous-graphes à n nœuds et plus de $2n-3$ arêtes. En 3D, elle consisterait à chercher des sous-graphes à n nœuds et plus de $3n-6$ arêtes et est donc piégée par les autres cas de sur-constriction, comme la « double-banane[5] ».

Une généralisation de la méthode pourrait alors consister à ne pas se limiter à des repères pour les déplacements posés sur deux points connectés par une contrainte, mais à tester tous les repères possibles. Tout d'abord, ceci est infaisable dans la pratique : il s'agit en effet de tester toutes les possibilités de fixer $ddl(\mathcal{S}) - ddr(\mathcal{S}c)$ degrés de repère parmi les $ddl(\mathcal{S})$ degrés de liberté, c'est-à-dire qu'il y a[6] $C_{ddl(\mathcal{S})}^{ddl(\mathcal{S}c)-ddr(\mathcal{S})}$ combinaisons. Dans un exemple aussi petit que la « double-banane », cela représente déjà $\frac{24!}{6!}$ possibilités, soit un peu plus de $8,6 \times 10^{20}$ paramétrisations à tester, ce qui rend le test sur ce système infaisable.

En outre, avec la disparition de l'hypothèse de rigidité, cette extension de la méthode de H aux graphes de $\mathcal{R}$-flux ne fonctionne pas en 2D. En effet, en l'absence de cette hypothèse, le nombre de degrés de repère à fixer n'est plus nécessairement 3, il est quelconque. La figure 6.4 montre ainsi un système 2D génériquement sur-contraint pour lequel toutes les paramétrisations permettent de trouver un $\mathcal{R}$-flot valide maximal. Il s'agit là d'un cas de faux positif.

À l'inverse, il y a un risque de classer comme sur-contraint un GCS qui n'est pas génériquement sur-contraint (cas de faux négatif) si l'on place trop de degrés de

[5] *Cf.* figure 1.14 p. 34

[6] *Modulo* la possibilité d'identifier certaines symétries

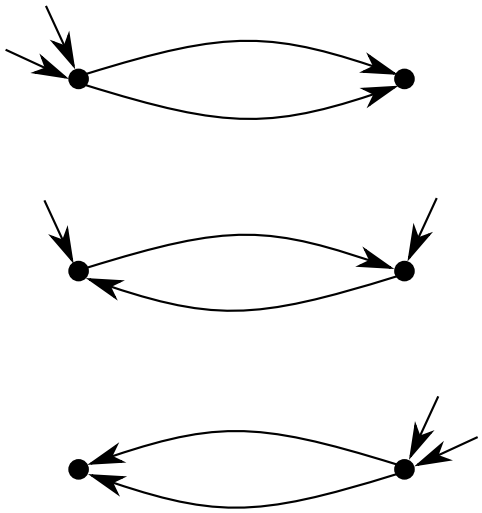


Figure 6.4 - Représentation de C des trois paramétrisations possibles du GCS génériquement sur-contraint de la figure 6.1**a** : elles sont toutes valides.

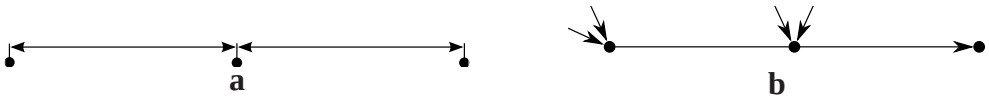


Figure 6.5 - Système 2D non sur-contraint avec des paramétrisations invalides ; **a** : esquisse ; **b** : représentation de C d'une paramétrisation invalide : les deux points de gauche sont saturés alors que la contrainte les liant n'est pas prise en compte.

repère dans un même sous-système. C'est le cas, par exemple, à la figure 6.5.

Une approche visant à corriger ce problème serait de considérer une construction incrémentale de la paramétrisation : après chaque ajout d'un degré de repère sur un nœud, si ce nœud est devenu saturé, on oriente tous ses arcs non encore orientés. On interdit alors d'ajouter un degré de repère sur un nœud déjà saturé. Cela permettrait notamment de faire baisser le nombre de combinaisons à essayer. La figure 6.6 montre toutefois que cette évolution de l'algorithme n'est pas suffisante : les étapes **b** et **c** montrent l'ajout de degrés de repère et l'orientation des arcs au fur et à mesure que des nœuds sont saturés. À la fin de l'étape **c**, les deux nœuds médians sont restreints mais non saturés, mais si on en sature un par un degré de repère supplémentaire, on arrive à un $\mathcal{R}$-flot non valide. La figure 6.6**d** montre l'une de ces deux paramétrisations invalides.

Là encore, on pourrait être tenté de penser que le passage de la figure 6.6**c** à la figure 6.6**d** pourrait être empêché en calculant un flux maximal à chaque ajout d'un degré de repère. En effet, on détecterait ainsi, comme on le voit à la figure 6.7, qu'à part la barre rigide de droite, tout le système est déterminé et qu'il est donc interdit

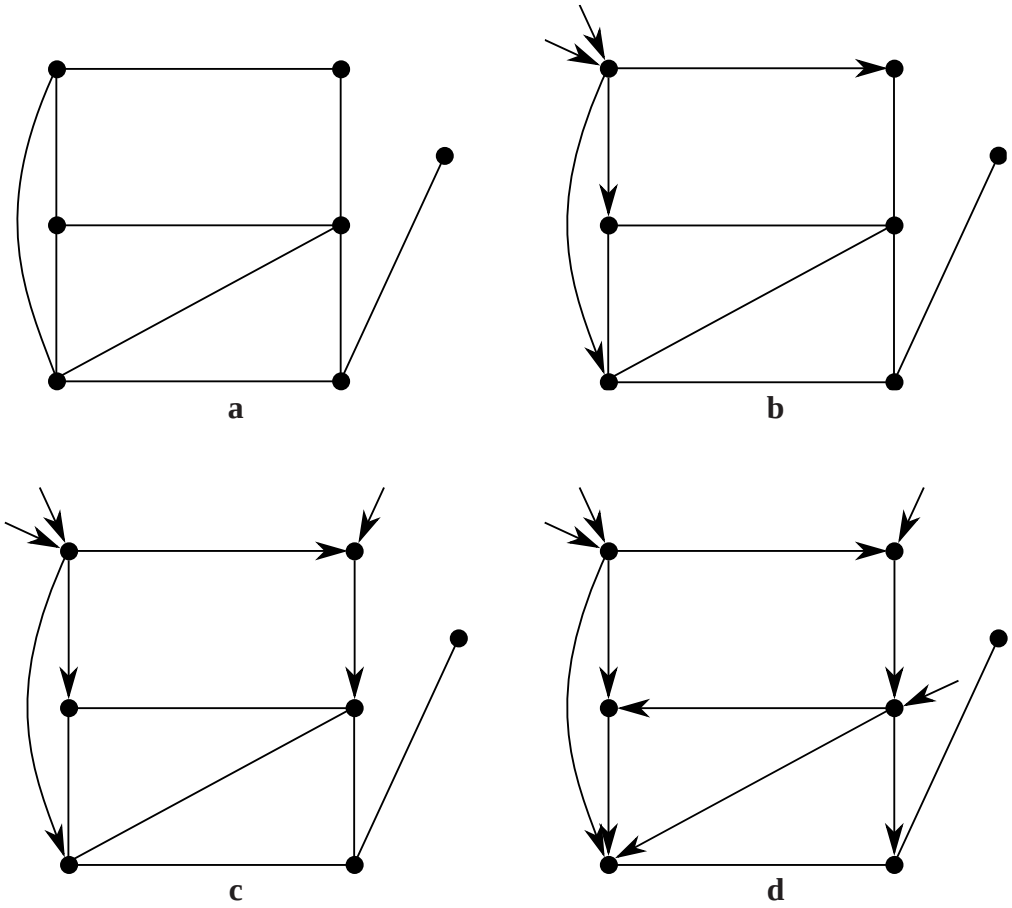


Figure 6.6 - Construction incrémentale d'une paramétrisation non valide d'un système 2D non sur-contraint ; **a** : graphe de contrainte (chaque trait représente une contrainte de distance ; **b** : fixation de 2 degrés de repère ; **c** : fixation d'un degré de repère supplémentaire ; **d** : la fixation d'un degré de repère supplémentaire mène à la sur-constriction du point inférieur gauche.

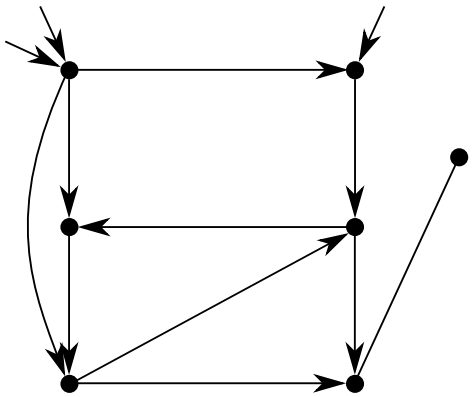


Figure 6.7 - Interdiction d'une paramétrisation sur-contraignante par calcul incrémental d'un flux maximal

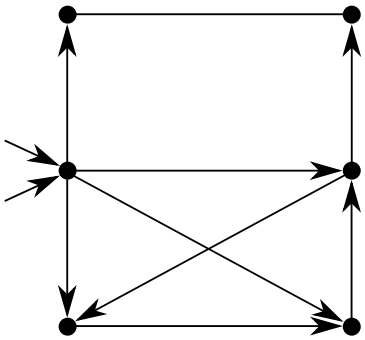


Figure 6.8 - Contre-exemple de la démarche illustrée à la figure 6.7

d'ajouter un degré de repère comme cela est fait à la figure 6.6**d**. La figure 6.8 montre toutefois qu'une telle approche ne permet pas de détecter la sur-constriction générique d'exemples simples qu'H sait détecter : une fois deux degrés de repère fixés, il est possible de calculer un flux menant à croire que la partie droite, en réalité sur-contrainte, est déterminée.

Dans ce dernier exemple, la partie sur-contrainte est limitée à une CFC. Malheureusement, ce n'est pas non plus un critère de détection des sous-systèmes sur-contraints : des exemples comme l'hexagone $K_{3,3}$ de la figure 1.12[7] existent, qui sont faits d'une CFC quelle que soit la paramétrisation et qui, pourtant, ne sont pas génériquement sur-contraints.

6.3 Extensions de la méthode du témoin

La section précédente ne prouve pas qu'il est impossible de détecter les systèmes génériquement sur-contraints combinatoirement dans un graphe de $\mathcal{R}$-flux, mais elle montre qu'une adaptation des approches classiques n'est pas triviale. En outre, les techniques de la littérature ne concernent absolument pas les problèmes d'interprétation géométrique sur-contraignante évoqués à la section 6.1.2.

Nous proposons ici deux extensions de la méthode du témoin[8] qui permettent de résoudre les problèmes des sections 6.1.1 et 6.1.2. Dans notre implémentation, nous faisons appel à cette extension lors du calcul d'un $\mathcal{R}$-flot et pour vérifier qu'une

[7] *Cf.* p. 30
[8] *Cf.* section 2.2.2 p. 52

interprétation géométrique donnée n'est pas sur-contraignante.

6.3.1 Détection incrémentale de contraintes redondantes

Comme nous l'avons vu au chapitre 2[9], l'interrogation du témoin permet de détecter des redondances dans un ensemble de contraintes en comparant le nombre de degrés de restriction et le rang de la matrice Jacobienne. La complexité algorithmique de cette interrogation est celle du calcul du rang d'une matrice, c'est-à-dire $O(\min(m,n)mn)$, avec m le nombre de lignes de la matrice (nombre total de degrés de restriction du GCS) et n le nombre de colonnes de la matrice (nombre total de degrés de liberté du GCS). Comme le nombre de degrés de restriction ne peut être supérieur au nombre de degrés de liberté, la complexité dans notre cas est en $O(m^2n)$.

Sous l'hypothèse que l'on dispose d'un témoin, il est donc possible de détecter l'ajout d'une contrainte redondante en interrogeant le témoin à chaque nouvelle contrainte ajoutée : si le rang n'a pas changé, la nouvelle contrainte est redondante et, par définition, il y a sur-constriction générique. La complexité d'une telle démarche serait toutefois élevée : $O(\Sigma_{i=0}^{m}(\min(i,n)in))$. Sachant que $m \leq n$, on a $\min(i,n) = i, \forall i \in [0,m]$ et donc $\Sigma_{i=0}^{m}(\min(i,n)in) = \Sigma_{i=0}^{m}(i^2n) = \frac{(2m+1)(m+1)m}{6}n$. La complexité est donc en $O(m^3n)$. Il est toutefois possible d'identifier incrémentalement l'ensemble des contraintes redondantes sans augmentation du coût par rapport à une unique interrogation du témoin.

En effet, considérons un GCS $\mathcal{S}$ non génériquement sur-contraint. Effectuer une élimination de G -J [10] sur la matrice Jacobienne évaluée en le témoin mène à une forme échelonnée réduite c'est-à-dire à une matrice $J' = (IP)$ avec
– I une matrice diagonale $m \times m$;
– P une matrice $m \times f$, avec $f = n - m$ le nombre de degrés de liberté du GCS.
Considérons maintenant un GCS $\mathcal{S}'$ avec $\mathcal{S} \subset \mathcal{S}'$. Afin de savoir si $\mathcal{S}'$ est génériquement sur-contraint, il suffit d'ajouter incrémentalement à $\mathcal{S}$ les entités géométriques et les contraintes[11] de $\mathcal{S}' - \mathcal{S}$ et d'effectuer une élimination de G -J à nouveau.

Dans la mesure où la matrice I avant ajout est diagonale, le nombre maximal d'opérations est $2 \times \min(m,n) \times f$: pour chaque ligne de I, chaque élément non nul de P

[9] *Cf.* section 2.2.2 p. 52

[10] Ou toute autre méthode permettant de calculer une base d'une matrice comme, par exemple, le procédé de G -S , qui a la même complexité que l'élimination de G -J .

[11] En s'assurant qu'une contrainte ne peut être ajoutée que si les entités géométriques qu'elle concerne sont toutes dans le GCS

doit être multiplié et ajouté à la nouvelle ligne. Le nombre réel d'opérations est en fait bien plus petit, dans la mesure où le nombre d'éléments nuls de la nouvelle ligne est élevé, une contrainte ne concernant en général qu'une faible partie des entités géométriques.

En procédant incrémentalement, le nombre d'opérations est inchangé : seul l'ordre des opérations est modifié. En effet, la méthode classique de l'élimination de G - J consiste en des opérations colonne par colonne pour mettre tous les éléments de la colonne à 0 sauf l'élément situé sur la diagonale :

pour chaque *colonne c de 0 à n* **faire**
  ligne $c \leftarrow$ (ligne c)$/J_{c,c}$
  pour chaque *ligne l de 0 à m sauf c* **faire**
    ligne $l \leftarrow$ ligne l – (ligne c) $\times J_{l,c}$

Algorithme 6.1: Calcul classique d'une forme échelonnée réduite par élimination de G -J

Le calcul incrémental d'une forme échelonnée réduite consiste simplement à faire ce calcul ligne par ligne :

pour chaque *ligne l de 0 à m* **faire**
  pour chaque *ligne c de 0 à l* – 1 **faire**
    ligne $l \leftarrow$ ligne l – (ligne l) $\times J_{c,c}$
  ligne $l \leftarrow$ (ligne l)$/J_{l,l}$

Algorithme 6.2: Calcul incrémental d'une forme échelonnée réduite par élimination de G -J

La complexité de l'algorithme incrémental de calcul d'une forme échelonnée réduite est donc également de $O(m^2n)$. Cette version de l'élimination de G -J a toutefois un énorme avantage : à chaque étape, quand une contrainte est ajoutée, elle nous permet de comparer le nouveau rang de la Jacobienne à l'ancien rang et ainsi de déterminer si la contrainte est redondante avec les contraintes déjà disponibles. Autrement dit, sans augmenter le nombre d'opérations, nous obtenons la forme échelonnée réduite de la matrice et un sous-système maximal non génériquement sur-contraint.

Exemple 20. Détection de la redondance dans un système 2D

Considérons l'exemple de la figure 6.9. Elle représente un système 2D à 4 points et 6 contraintes de distance. Le GCS correspondant est génériquement sur-contraint : la contrainte en pointillés est redondante avec les 5 autres. La matrice Jacobienne de ce GCS est donnée

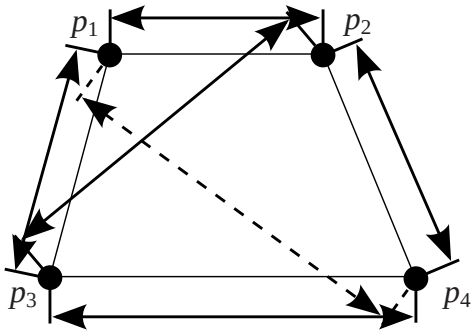


Figure 6.9 - Le « cerf-volant » : système 2D génériquement sur-contraint avec 4 points et 6 distances. Sans la contrainte pointillée, le GCS est rigide.

Tableau 6.1 - La matrice Jacobienne du GCS de la figure 6.9.

	$\dot{x_1}$	$\dot{y_1}$	$\dot{x_2}$	$\dot{y_2}$	$\dot{x_3}$	$\dot{y_3}$	$\dot{x_4}$	$\dot{y_4}$
l_1 : `dist_pp`(p_1, p_2)	x_1-x_2	y_1-y_2	x_2-x_1	y_2-y_1	0	0	0	0
l_2 : `dist_pp`(p_1, p_3)	x_1-x_3	y_1-y_3	0	0	x_3-x_1	y_3-y_1	0	0
l_3 : `dist_pp`(p_2, p_4)	0	0	x_2-x_4	y_2-y_4	0	0	x_4-x_2	y_4-y_2
l_4 : `dist_pp`(p_3, p_4)	0	0	0	0	x_3-x_4	y_3-y_4	x_4-x_3	y_4-y_3
l_5 : `dist_pp`(p_2, p_3)	0	0	x_2-x_3	y_2-y_3	x_3-x_2	y_3-y_2	0	0
l_6 : `dist_pp`(p_1, p_4)	x_1-x_4	y_1-y_4	0	0	0	0	x_4-x_1	y_4-y_1

à la table 6.1.

Considérons le témoin suivant (représenté graphiquement à la figure 6.10) :
- $p_1 = (2, 7)$;
- $p_2 = (5, 6)$;
- $p_3 = (1, 1)$;
- $p_4 = (6, 3)$.

La Jacobienne en ce témoin est donnée à la table 6.2, après une application partielle de l'élimination de G -J : cette table montre la matrice obtenue en utilisant la version incrémentale de l'élimination de G -J , après ajout de la sixième contrainte mais avant utilisation du pivot de G .

Il est aisé de voir que cette nouvelle contrainte sur-contraint génériquement le GCS : elle peut être obtenue en soustrayant la première ligne à la seconde.

De par la taille de la matrice associée[12], nous ne donnons pas ici de détails sur l'exemple de l'application de cette méthode de détection de la sur-constriction gé-

[12] Vingt-quatre colonnes et dix-huit lignes

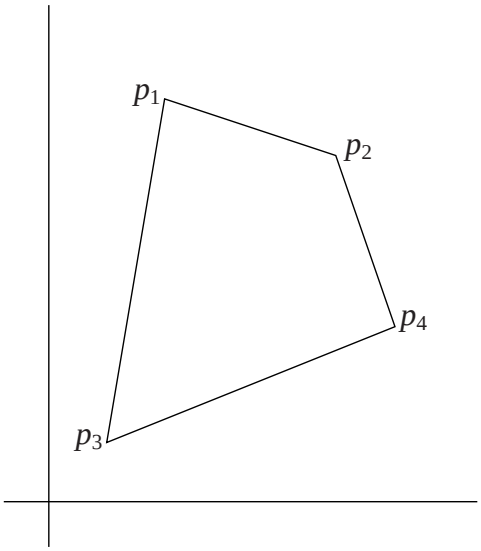


Figure 6.10 - Un témoin du GCS de la figure 6.9 avec $p_1 = (2, 7)$, $p_2 = (5, 6)$, $p_3 = (1, 1)$ et $p_4 = (6, 3)$.

Tableau 6.2 - Matrice Jacobienne de la table 6.1 évaluée en le témoin de la figure 6.10. L'élimination de G -J a été effectuée sur les cinq premières lignes. La sixième ligne est redondante ($l_6 = l'_2 - l'_1$)

	$\dot{x_1}$	$\dot{y_1}$	$\dot{x_2}$	$\dot{y_2}$	$\dot{x_3}$	$\dot{y_3}$	$\dot{x_4}$	$\dot{y_4}$
l'_1	1	0	0	0	0	$-\frac{4}{5}$	-1	$\frac{4}{5}$
l'_2	0	1	0	0	0	$-\frac{4}{5}$	0	$-\frac{1}{5}$
l'_3	0	0	1	0	0	$-\frac{3}{5}$	-1	$\frac{3}{5}$
l'_4	0	0	0	1	0	$-\frac{1}{5}$	0	$-\frac{4}{5}$
l'_5	0	0	0	0	1	$\frac{2}{5}$	-1	$-\frac{2}{5}$
l_6	-1	1	0	0	0	0	1	-1

nérique sur le GCS de la « double-banane[13] ». Elle permet toutefois d'ajouter les dix-sept premières contraintes sans détecter de sur-constriction générique et de détecter la redondance de la dix-huitième contrainte. La méthode fonctionne tout aussi bien pour des exemples avec un plus haut degré de connexité [MS04].

En outre, la méthode du témoin gère correctement la sur-constriction générique de GCS qui, pour des valeurs des paramètres le rendant non sur-contraint, est sous-contraint. Pour ces systèmes, les méthodes à base de graphes sont inutiles car elles ne peuvent prendre en compte tous les théorèmes géométriques.

Exemple 21. Conjugués harmoniques

Considérons le système 2D de la figure 6.11. Étant donnés deux points a et b, on peut considérer une fonction qui à tout point x incident à la droite (ab) associe un point y, conjugué harmonique de x. On construit y en suivant le plan de construction suivant :

- soit p un point non incident à (ab) ;
- soit d une droite incidente à x ;
- soit p_1 l'intersection des droites d et (ap) ;
- soit p_2 l'intersection des droites d et (bp) ;
- soit p' l'intersection des droites (bp_1) et (ap_2) ;
- y est l'intersection des droites (ab) et (pp').

On peut considérer le GCS $\mathcal{S} = (C, X, A)$ défini par[14] :

- $X = \{x, y, p, p_1, p_2, p', d, d_{ab}, d_{ap}, d_{bp}, d_{ap'}, d_{bp'}, d_{pp'}\}$ avec les six premiers éléments de sorte `point` et les autres de sorte `droite` ;
- $A = \{a, b\}$ avec tous les paramètres de sorte `point` ;
- $C = \{$
  $\texttt{inc\_pd}(a, d_{ab})$, $\texttt{inc\_pd}(b, d_{ab})$, $\texttt{inc\_pd}(x, d_{ab})$,
  $\texttt{inc\_pd}(y, d_{ab})$, $\texttt{inc\_pd}(x, d)$, $\texttt{inc\_pd}(p_1, d)$,
  $\texttt{inc\_pd}(p_2, d)$, $\texttt{inc\_pd}(a, d_{ap})$, $\texttt{inc\_pd}(p, d_{ap})$,
  $\texttt{inc\_pd}(p_1, d_{ap})$, $\texttt{inc\_pd}(b, d_{bp})$, $\texttt{inc\_pd}(p, d_{bp})$,
  $\texttt{inc\_pd}(p_2, d_{bp})$, $\texttt{inc\_pd}(a, d_{ap'})$, $\texttt{inc\_pd}(p', d_{ap'})$,
  $\texttt{inc\_pd}(p_2, d_{ap'})$, $\texttt{inc\_pd}(b, d_{bp'})$, $\texttt{inc\_pd}(p', d_{bp'})$,
  $\texttt{inc\_pd}(p_1, d_{bp'})$, $\texttt{inc\_pd}(p, d_{pp'})$, $\texttt{inc\_pd}(p', d_{pp'})$,
  $\texttt{inc\_pd}(y, d_{pp'})\}$.

Les solutions de $\mathcal{S}$ correspondent toutes à une figure obtenue en respectant le plan de construction ci-dessus. De par le grand nombre de contraintes d'incidence, l'hypothèse que l'on dispose d'un témoin serait, sans cela, une hypothèse forte : les contraintes d'incidence n'étant pas associées à une métrique, elles doivent être respectées par le témoin. Ici, un témoin est donc une solution du système.

[13] *Cf.* figure 1.14 p. 34

[14] Rappel : nous utilisons la signature de l'exemple 1, p. 16

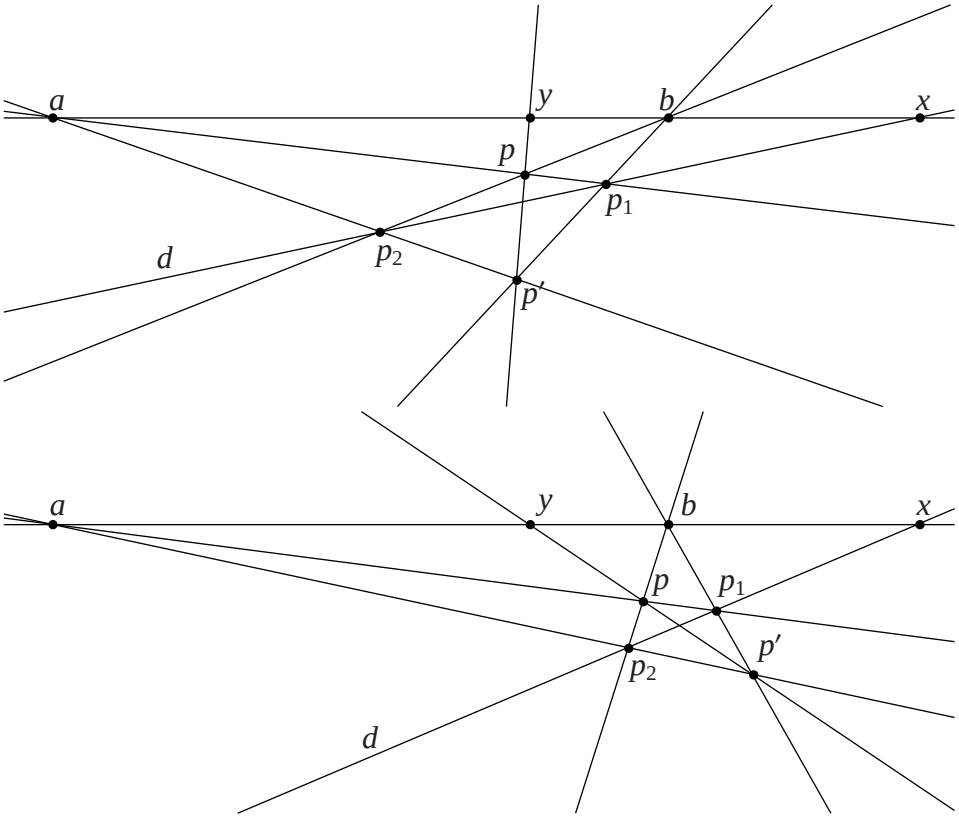


Figure 6.11 - y est le conjugué harmonique de x pour a et b : en 2D, étant donnés trois points a, b et x alignés et pour tout point p et toute droite d incidente à x, y est fixé : $p_1 = (ap) \cap d$, $p_2 = (bp) \cap d$, $p' = (ap_2) \cap (bp_1)$, $y = (ab) \cap (pp')$.

> Un comptage des degrés de liberté mène au résultat – correct – que le système a $13\times2-22=4$ degrés de liberté effectifs[15]. En revanche, aucune méthode combinatoire actuelle ne parvient à détecter que les positions de x et y sont liées et qu'il est impossible d'ajouter une contrainte de distance entre ces deux points. Notre algorithme de calcul d'un $\mathcal{R}$-flot, par exemple, acceptera l'ajout de cette nouvelle contrainte en retirant un degré de repère. L'interrogation du témoin, quant à elle, détecte une redondance.

Les contraintes redondantes identifiées peuvent être conservées et servir plus tard à trouver, parmi les différentes solutions satisfaisant les contraintes, celle que l'utilisateur recherche : la redondance des contraintes peut être nécessaire pour assurer l'unicité des solutions[16].

Notons que la méthode incrémentale d'interrogation du témoin permet également de résoudre un autre problème : celui de calcul d'une base du bord d'un système. Nous avons vu en effet[17] que l'ajout de l'ensemble des contraintes du bord pouvait mener à une sur-constriction générique. La plupart des méthodes de décomposition étant sensibles à la sur-constriction générique, il est important de pouvoir calculer une base de cet ensemble de contraintes, c'est-à-dire un sous-ensemble minimal tel que toutes les autres contraintes du bord sont redondantes avec ce sous-ensemble. Ici, cela est facile : il suffit de partir d'un système vide et d'ajouter une par une les contraintes du bord, en recalculant incrémentalement la forme échelonnée réduite de la matrice Jacobienne pour détecter les contraintes redondantes avec les contraintes déjà prises en compte.

6.3.2 Interprétations sur-contraignantes d'un $\mathcal{R}$-flot

Une interrogation du témoin permet de détecter une interprétation sur-contraignante d'un $\mathcal{R}$-flot sous forme de repère. Détaillons tout d'abord comment se traduit la fixation d'un repère dans la méthode du témoin : une fois la Jacobienne mise sous forme échelonnée réduite, l'équation $J\overrightarrow{V}=0$ est la suivante[18] :

[15] Rappelons que a et b sont des paramètres du GCS et n'ont donc aucun degré de liberté
[16] Voir [Hen92] et notamment la figure 2.
[17] *Cf.* section 3.2 (p. 73)
[18] On considère bien sûr que le GCS n'est pas génériquement sur-contraint.

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & \alpha_{1,1} & \cdots & \alpha_{n-m,1} \\ 0 & 1 & 0 & \cdots & 0 & \alpha_{1,2} & \cdots & \alpha_{n-m,2} \\ 0 & 0 & 1 & \cdots & 0 & \alpha_{1,3} & \cdots & \alpha_{n-m,3} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \alpha_{1,m} & \cdots & \alpha_{n-m,m} \end{pmatrix} \begin{pmatrix} \dot{v}_1 \\ \dot{v}_2 \\ \dot{v}_3 \\ \vdots \\ \dot{v}_m \\ \dot{v}_{r_{n-m}} \\ \vdots \\ \dot{v}_{r_1} \end{pmatrix} = 0$$

Dans cette notation, les $n-m$ derniers éléments du vecteur $\overrightarrow{V}$ sont notés $v_{r_{n-m}}$ à v_{r_1}. Il y en a autant que de degrés de liberté du GCS et donc autant que de degrés de repère dans les $\mathcal{R}$-flots associés. En effet, pour traduire dans la Jacobienne le fait que la coordonnée v_i fait partie du repère considéré, on effectue une permutation de colonnes (et la permutation de lignes correspondant dans le vecteur $\overrightarrow{V}$) pour que la colonne correspondant à $\dot{v}_i$ soit l'une des $n-m$ colonnes de droite de la matrice.

Imaginons maintenant qu'après la $i-1^{\text{ème}}$ étape de l'élimination de G -J (la matrice $(i-1)\times(i-1)$ supérieure gauche est diagonale), il soit impossible de trouver un pivot non nul. Cela signifie que la $i^{\text{ème}}$ colonne est combinaison linéaire des $i-1$ premières colonnes. Permuter cette colonne avec une des $i-1$ premières colonnes ne changera rien au problème : la nouvelle $i^{\text{ème}}$ colonne sera combinaison linéaire des $i-1$ nouvelles premières colonnes. De même, permuter la $i^{\text{ème}}$ colonne avec une colonne d'indice compris entre $i+1$ et m ne fera que retarder le problème : si la colonne est combinaison linéaire des $i-1$ premières colonnes, elle est également combinaison linéaire des $i-1+k$ premières colonnes si $k \geq 0$.

Il ne sera donc possible de diagonaliser la gauche de la matrice Jacobienne qu'à condition que la $i^{\text{ème}}$ colonne soit permutée avec une des $n-m$ dernières colonnes. Cette condition est nécessaire mais non suffisante, puisque certaines des $n-m$ dernières colonnes peuvent elles-mêmes être combinaison linéaire des $i-1$ premières colonnes.

Si l'on n'arrive pas à diagonaliser la matrice $m \times m$ gauche de la Jacobienne, c'est que la paramétrisation consistant à fixer les coordonnées correspondant aux $n-m$ colonnes de droite est non valable : cela signifie en effet qu'il n'est pas possible d'exprimer les variations des n premières coordonnées en fonction de celles considérées comme paramètres : une fois le produit effectué, la $i^{\text{ème}}$ ligne peut s'écrire

$$\dot{v}_i + \alpha_{1,i} \times \dot{v}_{r_{n-m}} + \cdots + \alpha_{n-m,i} \times \dot{v}_{r_1} = 0 \tag{6.1}$$

et donc

$$\dot{v}_i = -\alpha_{1,i} \times \dot{v}_{r_{n-m}} - \cdots - \alpha_{n-m,i} \times \dot{v}_{r_1} \tag{6.2}$$

Autrement dit, on peut exprimer les variations de v_i en fonction des variations des coordonnées $v_{r_{n-m}}$ à v_{r_1}. C'est la définition même d'un repère : s'il est fixé, les autres éléments sont fixés également. S'il n'est pas possible d'exprimer les m premières colonnes comme fonction des $n - m$ dernières colonnes, alors la paramétrisation correspondante ne forme pas un repère.

Il s'en déduit une méthode simple de détermination de la validité de l'interprétation géométrique d'un $\mathcal{R}$-flot : permuter les colonnes de la Jacobienne de sorte à placer à droite de la matrice celles correspondant à l'interprétation à tester et appliquer la méthode d'élimination de G -J .

Naïvement, la complexité d'une telle méthode, pour avoir l'assurance de trouver une interprétation valide, serait en $O(m^2 n \times \Pi_{x\in\mathcal{R}}(C_{ddl(x)}^{\mathrm{Val}(n_x \to r_x)}))$ avec $\mathcal{R}$ l'ensemble des entités de l'ancre : il faudrait, pour chaque interprétation géométrique possible, effectuer une élimination de G -J . Une méthode moins coûteuse consiste à tenter une première élimination de G -J , puis à permuter deux colonnes pour aboutir à une autre interprétation géométrique et à utiliser le pivot de G sur la colonne permutée : il y a 1 division de ligne à effectuer et $m - 1$ multiplications et additions de ligne. Avant permutation, la matrice est sous la forme (IP) avec I l'identité et P une matrice $m \times (n - m)$. Le nombre d'opérations à effectuer pour une multiplication ou addition de ligne est donc $n - m + 1$. La complexité de cette méthode est donc en $O(m^2 n + m \times (n - m) \times \Pi_{x\in\mathcal{R}}(C_{ddl(x)}^{\mathrm{Val}(n_x \to r_x)}))$.

B II

Nous avons défini une structure de données étendant les graphes de flux classiques, que nous appelons les graphes de $\mathcal{R}$-flux, et adapté la définition du couplage parfait à ces graphes. En nous basant sur les graphes de $\mathcal{R}$-flux, nous avons proposé des algorithmes incrémentaux de calcul d'une paramétrisation combinatoire d'un GCS, c'est-à-dire le nombre de degrés de liberté à fixer pour chaque entité géométrique pour se ramener à un nombre fini de solutions. Nous avons montré comment déduire de cette paramétrisation combinatoire un plan de construction par blocs. Nous avons également mis au point une extension incrémentale de la méthode du témoin permettant d'assurer qu'un GCS n'est pas génériquement sur-contraint, pour assurer la validité de nos algorithmes de paramétrisation.

Un travail important reste à effectuer concernant la gestion des valeurs associées aux éléments de repère. Le calcul de valeurs valides, c'est-à-dire telles que des solutions existent, pourrait être envisagé de plusieurs façons :

– notre méthode actuelle consiste à prendre les valeurs de l'esquisse, éventuellement mises à l'échelle. Dans le cas où cela ne suffit pas, un rattrapage numérique est possible en cherchant à minimiser une fonction d'erreur calculée pour la construction géométrique qui échoue. Des instabilités peuvent apparaître lorsque la modification de la valeur d'un paramètre amène l'échec d'une autre construction géométrique. Une étude spécifique des méthodes numériques permettant le calcul d'une valuation valide à partir d'une valuation initiale invalide est donc à mener, qui pourrait également servir à perturber les valeurs des paramètres pour

animer les solutions à l'écran ;
– l'utilisation d'une méthode itérative à partir d'une valuation initiale peut échouer non seulement de par les instabilités de telles méthodes mais aussi en raison de *minima* locaux. Des méthodes d'optimisation globale devraient donc également être essayées, par exemple au moyen d'algorithmes génétiques ;
– une approche plus constructive pourrait être une propagation d'intervalles : à partir d'un élément de repère donné, parcourir le plan de construction et, à chaque construction, calculer dans quels sous-espaces l'entité géométrique construite peut se trouver. Si cette entité géométrique est fixée ou restreinte, on en déduit quelles sont les valeurs possibles des éléments de repère correspondants.

Nous avons également fourni des pistes concernant la définition de la qualité d'un $\mathcal{R}$-flot. Il s'agit d'heuristiques dont l'intérêt reste à étudier plus profondément. La mise au point de nouvelles heuristiques, voire la donnée d'une définition précise de ce qu'est un bon $\mathcal{R}$-flot sont donc des perspectives naturelles de ce travail. En outre, notre approche d'une stratégie de construction d'un $\mathcal{R}$-flot en fonction des heuristiques peut mener vers des *maxima* locaux empêchant la découverte de $\mathcal{R}$-flots de bonne qualité. Là encore, une approche plus globale serait donc à envisager.

Enfin, la méthode itérative de N -R n'a jusqu'à maintenant jamais servi à trouver numériquement des solutions à un système sous-contraint. En effet, elle ne peut par définition fonctionner que sur des systèmes d'équations dont la matrice est carrée, c'est-à-dire correspondant à des systèmes bien-contraints *modulo* l'identité. Classiquement, la **D**-bonne-constriction est prise en compte en ajoutant trois équations, ce qui correspond à fixer un repère pour les déplacements. Notre algorithme de paramétrisation ouvre la porte à l'adaptation de la méthode de N -R à des systèmes de contraintes géométriques quelconques, sous l'hypothèse de non sur-constriction générique. Une autre hypothèse importante est la donnée d'une valuation initiale valide pour les éléments de la paramétrisation.

T `

D´ ` ´ ´

L'objectif général visé par nos travaux est de rendre les logiciels de modélisation par contraintes géométriques accessibles à des utilisateurs non experts. Les travaux de paramétrisation explicités à la partie II vont dans ce sens : ils fournissent à l'utilisateur une intuition des libertés de mouvement des entités géométriques du Système de Contraintes Géométriques (GCS) qu'il a esquissé et lui permettent de corriger l'esquisse si le résultat ne lui convient pas.

Une autre fonctionnalité importante de logiciels faciles d'accès est la possibilité de réutiliser des GCS déjà résolus. La conception d'une voiture en Conception Assistée par Ordinateur (CAO), par exemple, ne se fait pas en explicitant toutes les contraintes de distance, d'angle, *etc*. L'approche habituelle est plutôt de dessiner les différentes pièces séparément et de les assembler. Il est également important qu'en plus de donner une paramétrisation à l'utilisateur, le logiciel lui permette de visualiser quelles parties de l'objet sont rigides.

Dans cette partie, nous nous intéressons donc à la décomposition de systèmes de contraintes géométriques. Nous avons déjà montré, dans la partie I, quelles étaient les conditions de correction et de complétude des méthodes de décomposition.

Motivation

Les travaux relatés dans cette partie ont pour but de permettre la décomposition d'un GCS en ses sous-systèmes, en connaissant le groupe de bonne constriction de ces sous-systèmes. Nous poursuivons ainsi plusieurs objectifs.

Tout d'abord, nous voulons identifier, pour les différents groupes G considérés, les parties G-bien-contraintes d'un GCS, afin de pouvoir indiquer à l'utilisateur quels sous-systèmes sont rigides, quels sous-systèmes sont bien contraints *modulo* les similitudes, *etc*. Ces informations complètent les intuitions que donne la paramétrisation sous forme d'éléments de repère.

Inversement, nous voulons permettre à l'utilisateur d'insérer dans le GCS un objet atomique avec son groupe de bonne constriction, de façon à assurer d'une part la réutilisation de systèmes déjà résolus et d'autre part une diminution de la taille du système à résoudre.

Nous avons vu dans la partie II que nous pouvons déduire d'un $\mathcal{R}$-flot un plan de construction strict à la condition qu'aucun nœud de son graphe réduit ne corresponde à plus d'une entité géométrique. Dans le cas contraire, les sous-systèmes engendrés par ces entités et celles dont elles dépendent doivent être résolus par un solveur externe. Nous voulons donc également proposer un algorithme de décom-

position général qui pourra aider à résoudre ces systèmes.

Démarche

Nous proposons une extension de la méthode du témoin[19] qui permet d'identifier les sous-systèmes rigides maximaux, puis une extension multi-groupe permettant l'identification de sous-systèmes G-bien-contraints maximaux pour tout groupe G dont on connaît les différents types de repère.

Pour rester dans la perspective des travaux relatés dans la partie II, nous proposons une version incrémentale de ces algorithmes. Nous proposons également des pistes pour l'identification des sous-systèmes G-bien-contraint directement dans un graphe de $\mathcal{R}$-flux et montrons les limites de ces pistes. Nous montrons ensuite quelles modifications apporter aux algorithmes de paramétrisation du chapitre 4 pour prendre en compte la G-bonne-constriction de sous-systèmes.

Enfin, nous utilisons l'algorithme d'identification de sous-systèmes maximaux G-bien-contraints pour mettre au point un algorithme de décomposition général, non sensible à la connexité du graphe de contrainte, baptisé $\mathcal{W}$-décomposition.

Le chapitre 7 explicite nos algorithmes d'identification des sous-systèmes G-bien-contraints et l'adaptation des algorithmes de paramétrisation. Le chapitre 8 détaille le fonctionnement et l'analyse de la $\mathcal{W}$-décomposition.

[19] *Cf.* section 2.2.2 et [MF06, MFLM06, MF07, MF09]

C 7

I - ` G- -

Sommaire

Tous les progrès sont précaires, et la solution d'un problème nous confronte à un autre problème
– Martin Luther King, pasteur américain

Dans ce chapitre, nous nous attachons aux problématiques liées aux sous-systèmes bien contraints *modulo* un groupe connu. Nous commençons par proposer, à la section 7.1, de nouvelles extensions de la méthode du témoin qui permettent d'identifier les sous-systèmes *G*-bien-contraints maximaux d'un système. La section 7.2 fournit des pistes pour la détection des sous-systèmes *G*-bien-contraints sans avoir recours au témoin. Enfin, la section 7.3 explique comment adapter les algorithmes du chapitre 4 pour prendre en compte des sous-systèmes dont on connaît le groupe de bonne constriction.

7.1 Identification des sous-systèmes G-bien-contraints maximaux avec le témoin

Dans cette section, nous montrons comment l'interrogation du témoin peut être utilisée pour identifier efficacement tous les Sous-systèmes Rigides Maximaux (MRS) d'un GCS, même dans le cas de systèmes où des méthodes à base de graphes échoueraient à détecter la rigidité. L'idée de cette méthode avait déjà été esquissée dans la littérature [MF06, JTNM06]. Nous montrons ensuite comment étendre cette approche à l'identification des Sous-systèmes G-bien-contraints Maximaux (MGS) pour n'importe quel groupe G dont on connaît les différents types de repère.

7.1.1 Détection des sous-systèmes rigides maximaux

Nous détaillons ici une méthode permettant d'identifier les sous-systèmes rigides maximaux d'un GCS.

Définition 54. Sous-système Rigide Maximal : *Un MRS de $\mathcal{S}$ est un sous-système $\mathcal{S}_1 \subseteq \mathcal{S}$ tel que :*
- *$\mathcal{S}_1$ est* **D***-bien-contraint ;*
- *il n'existe pas de système $\mathcal{S}_2$* **D***-bien-contraint distinct de $\mathcal{S}_1$ tel que $\mathcal{S}_1 \subset \mathcal{S}_2$ et $\mathcal{S}_2 \subseteq \mathcal{S}$.*

Exemple 22. Sous-système Rigide Maximal

La figure 7.1 est constituée de quatre MRS : les segments $[p_2p_3]$, $[p_1p_2]$, $[p_1p_4]$ et le quadrilatère $p_3p_4p_5p_6$ sont rigides et ne sont pas sous-systèmes d'un GCS rigide.

L'ajout d'une contrainte de distance entre p_1 et p_5, par exemple, rendrait le système rigide et il ne serait donc plus constitué que d'un seul MRS.

L'idée de base de notre algorithme d'identification des MRS est d'étudier quelles entités géométriques sont fixées lorsque l'on fixe un repère pour les déplacements. Rappelons que nous avons vu à la section 6.3.2[1], qu'après que la Jacobienne a été mise sous forme échelonnée réduite, l'équation $J'\vec{V} = 0$ s'écrit ainsi :

[1] *Cf.* p. 145 et notamment les équations 6.1 et 6.2

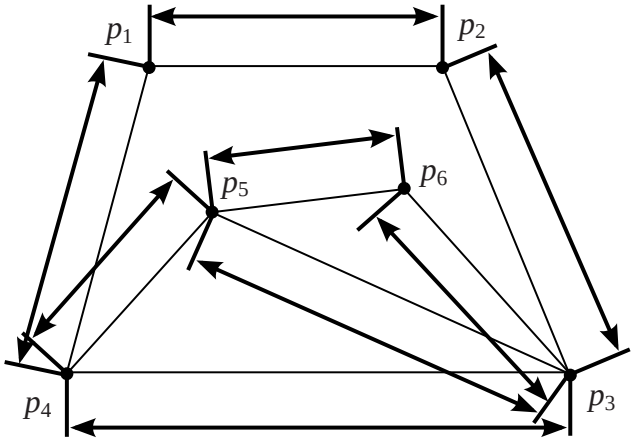


Figure 7.1 - GCS contenant 4 MRS

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & \alpha_{1,1} & \cdots & \alpha_{n-m,1} \\ 0 & 1 & 0 & \cdots & 0 & \alpha_{1,2} & \cdots & \alpha_{n-m,2} \\ 0 & 0 & 1 & \cdots & 0 & \alpha_{1,3} & \cdots & \alpha_{n-m,3} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \alpha_{1,m} & \cdots & \alpha_{n-m,m} \end{pmatrix} \begin{pmatrix} \dot{v}_1 \\ \dot{v}_2 \\ \dot{v}_3 \\ \vdots \\ \dot{v}_m \\ \dot{v}_{r_{n-m}} \\ \vdots \\ \dot{v}_{r_1} \end{pmatrix} = 0$$

Une fois le produit effectué, la $i^{\text{ème}}$ ligne peut s'écrire

$$\dot{v}_i + \alpha_{1,i} \times \dot{v}_{r_{n-m}} + \cdots + \alpha_{n-m,i} \times \dot{v}_{r_1} = 0$$

et donc

$$\dot{v}_i = -\alpha_{1,i} \times \dot{v}_{r_{n-m}} - \cdots - \alpha_{n-m,i} \times \dot{v}_{r_1}$$

Autrement dit, une fois la matrice mise sous forme échelonnée réduite[2], il est possible d'exprimer les variations de la coordonnée v_i en fonction des variations des coordonnées $v_{r_{n-m}}$ à v_{r_1}. Ces coordonnées forment donc un repère du GCS.

Si le GCS $\mathcal{S}$ n'est pas rigide, trois paramètres ne peuvent suffir pour ancrer toutes les entités géométriques et on a $n - m > 3$. Il est toutefois possible d'identifier un sous-ensemble des coordonnées qui dépend uniquement des trois paramètres v_{r_1}, v_{r_2} et v_{r_3} : on recherche l'ensemble V de tous les v_i tels que $\nexists e > 3, \alpha_{e,i} \neq 0$. Si les

[2]Sous l'hypothèse que le GCS ne soit pas génériquement sur-contraint (*cf.* section 6.3.1) et que les m premières colonnes soient linéairement indépendantes (*cf.* section 6.3.2)

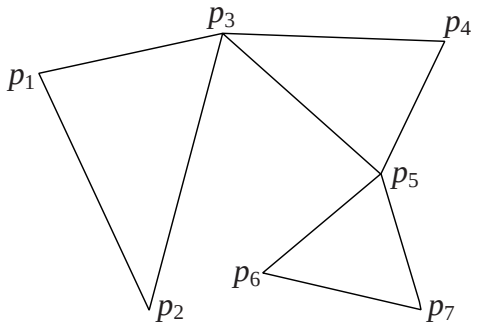


Figure 7.2 - Chaîne ouverte 2D faite de trois triangles rigides. Les contraintes de distance sont implicitement représentées par les segments.

Tableau 7.1 - Matrice Jacobienne du GCS de la figure 7.2

	$\dot{x_1}$	$\dot{y_1}$	$\dot{x_2}$	$\dot{x_3}$	$\dot{y_3}$	$\dot{x_4}$	$\dot{x_5}$	$\dot{y_5}$	$\dot{x_6}$	$\dot{y_2}$	$\dot{y_4}$	$\dot{y_6}$	$\dot{x_7}$	$\dot{y_7}$
l_1	x_1-x_2	y_1-y_2	x_2-x_1	0	0	0	0	0	0	y_2-y_1	0	0	0	0
l_2	x_1-x_3	y_1-y_3	0	x_3-x_1	y_3-y_1	0	0	0	0	0	0	0	0	0
l_3	0	0	x_2-x_3	x_3-x_2	y_3-y_2	0	0	0	0	y_2-y_3	0	0	0	0
l_4	0	0	0	x_3-x_4	y_3-y_4	x_4-x_3	0	0	0	0	y_4-y_3	0	0	0
l_5	0	0	0	x_3-x_5	y_3-y_5	0	x_5-x_3	y_5-y_3	0	0	0	0	0	0
l_6	0	0	0	0	0	x_4-x_5	x_5-x_4	y_5-y_4	0	0	y_4-y_5	0	0	0
l_7	0	0	0	0	0	0	x_5-x_6	y_5-y_6	x_6-x_5	0	0	y_6-y_5	0	0
l_8	0	0	0	0	0	0	x_5-x_7	y_5-y_7	0	0	0	0	x_7-x_5	y_7-y_5
l_9	0	0	0	0	0	0	0	0	x_6-x_7	0	0	y_6-y_7	x_7-x_6	y_7-y_6

paramètres v_{r_1} à v_{r_3} forment un repère pour les déplacements alors, par définition d'un repère[3], le système engendré par $V \cup \{v_{r_1}, v_{r_2}, v_{r_3}\}$ est **D**-bien-contraint. Il est maximal, c'est-à-dire qu'il n'existe aucun sous-système rigide $\mathcal{S}' \subseteq \mathcal{S}$ tel que le système engendré par $V \cup \{v_{r_1}, v_{r_2}, v_{r_3}\}$ est strictement inclus dans $\mathcal{S}'$.

Exemple 23. Identification d'un MRS

Considérons le GCS de la figure 7.2, qui représente un système 2D composé de trois triangles rigides formant une chaîne ouverte. Sa matrice jacobienne est fournie à la table 7.1[4], les lignes correspondant aux contraintes de distance entre :

- l_1 : p_1 et p_2
- l_2 : p_1 et p_3
- l_3 : p_2 et p_2
- l_4 : p_3 et p_4
- l_5 : p_3 et p_5
- l_6 : p_4 et p_5
- l_7 : p_5 et p_6
- l_8 : p_5 et p_7
- l_9 : p_6 et p_7

Considérons le témoin suivant :

- $p_1 = (0, 9)$;
- $p_2 = (4, 1)$;
- $p_3 = (7, 13)$;
- $p_4 = (14, 10)$;
- $p_5 = (12, 4)$;
- $p_6 = (7, 7)$;
- $p_7 = (13, 0)$.

La table 7.2 montre la mise sous forme échelonnée réduite de la Jacobienne évaluée en le témoin. On y constate que l'on peut exprimer $\dot{x_5}$, $\dot{y_5}$ et $\dot{x_6}$ en fonction de $\dot{y_6}$, $\dot{x_7}$ et $\dot{y_7}$. Le système engendré par p_5, p_6 et p_7 est donc le Sous-système Rigide Maximal fixé par le repère[5]

[3] *Cf.* définition 40 p. 69

[4] Les colonnes ont été permutées pour correspondre à une paramétrisation valide

[5] *Cf.* section 5.1 p. 116

Tableau 7.2 - Forme échelonnée réduite de la matrice Jacobienne de la table 7.1 évaluée en le témoin $p_1 = (0, 9)$, $p_2 = (4, 1)$, $p_3 = (7, 13)$, $p_4 = (14, 10)$, $p_5 = (12, 4)$, $p_6 = (7, 7)$, $p_7 = (13, 0)$

	$\dot{x}_1$	$\dot{y}_1$	$\dot{x}_2$	$\dot{x}_3$	$\dot{y}_3$	$\dot{x}_4$	$\dot{x}_5$	$\dot{y}_5$	$\dot{x}_6$	$\dot{y}_2$	$\dot{y}_4$	$\dot{y}_6$	$\dot{x}_7$	$\dot{y}_7$
r'_1	1	0	0	0	0	0	0	0	0	$\frac{4}{3}$	$\frac{101}{18}$	$-\frac{181}{108}$	-1	$-\frac{473}{108}$
r'_2	0	1	0	0	0	0	0	0	0	$-\frac{7}{3}$	$-\frac{40}{9}$	$\frac{28}{27}$	0	$\frac{140}{27}$
r'_3	0	0	1	0	0	0	0	0	0	4	$\frac{29}{2}$	$-\frac{15}{4}$	-1	$-\frac{59}{4}$
r'_4	0	0	0	1	0	0	0	0	0	0	$\frac{9}{2}$	$-\frac{17}{12}$	-1	$-\frac{37}{12}$
r'_5	0	0	0	0	1	0	0	0	0	0	$\frac{5}{2}$	$-\frac{7}{12}$	0	$-\frac{35}{12}$
r'_6	0	0	0	0	0	1	0	0	0	0	3	$-\frac{7}{6}$	-1	$-\frac{11}{6}$
r'_7	0	0	0	0	0	0	1	0	0	0	0	$-\frac{2}{3}$	-1	$\frac{2}{3}$
r'_8	0	0	0	0	0	0	0	1	0	0	0	$-\frac{1}{6}$	0	$-\frac{5}{6}$
r'_9	0	0	0	0	0	0	0	0	1	0	0	$-\frac{7}{6}$	-1	$\frac{7}{6}$

constitué par le point p_7 et la direction p_6-p_7.

À partir de ce principe, un algorithme naïf d'identification des MRS serait de fixer un repère pour les déplacements (en permutant les colonnes de la Jacobienne) et d'effectuer une élimination de G -J , puis de recommencer avec un repère pour les déplacements dans le reste du système. Le pseudo-code de cette approche est donné à l'algorithme 7.1.

À la ligne 4 de l'algorithme, la fixation d'un repère doit être guidée par la géométrie afin de s'assurer qu'on identifiera un système plus grand que le repère lui-même : si, dans l'exemple de la figure 7.2, on fixait les deux coordonnées du point p_7 et l'une des coordonnées de p_1, aucun MRS ne serait identifié. Pour les cas où le repère est mal choisi, il convient de complexifier un peu la ligne 9 de l'algorithme en n'effectuant de marquage que dans le cas où il y a au moins une entité géométrique n'appartenant pas totalement au repère qui est entièrement fixée. Il faut alors faire attention, dans la condition d'arrêt de la boucle de la ligne 2, aux cas où des sous-systèmes rigides maximaux sont plus petits qu'un repère pour les déplacements (les systèmes sous-**D**-définis[6]), par exemple une barre rigide en 3D ou un point concerné par aucune contrainte en 2D.

La complexité d'une telle approche dépend du nombre k de MRS : avec k éliminations de G -J , la complexité globale est en $O(km^2n)$[7]. Ce coût peut être réduit en ne redémarrant pas l'élimination de G -J de zéro pour chaque repère : à la fin de la boucle de la ligne 2, la matrice J' est sous forme échelonnée réduite. Pour changer de repère, il faut faire au maximum trois permutations et donc,

[6] *Cf.* définition 42 p. 70

[7] Rappelons que la complexité d'une application de l'élimination de G -J est $O(\min(m, n)mn)$ et qu'en l'absence de sur-constriction générique, $m \leq n$.

Entrées :
$\mathcal{S} = (C, X, A)$: Un GCS
J : la matrice Jacobienne de $\mathcal{S}$ évaluée en un témoin
Résultat :
Liste des MRS de $\mathcal{S}$

$i \leftarrow 0$
répéter
  $J' \leftarrow J$
  Permuter les colonnes de J' pour fixer un repère dans une partie non marquée de $\mathcal{S}$
  Effectuer une élimination de G -J sur J'
  $V \leftarrow$ ensemble des colonnes correspondant à des coordonnées exprimables en fonction du repère
  Marquer avec l'étiquette i les colonnes de J correspondant à des entités dont toutes les coordonnées sont
    – soit dans V
    – soit dans les trois dernières colonnes de J'
  $i \leftarrow i + 1$
jusqu'à *ce que toutes les colonnes de J soient marquées*
$L \leftarrow$ liste vide
tant que $i \geq 0$ **faire**
  $\mathcal{S}' \leftarrow$ sous-système de $\mathcal{S}$ engendré par les entités étiquetées i
  $L \leftarrow L + \mathcal{S}'$
  $i \leftarrow i - 1$
retourner L

Algorithme 7.1: Algorithme naïf d'identification des MRS basé sur la méthode du témoin

pour revenir à une forme échelonnée réduite, $3 \times m \times (n - m + 3)$ multiplications[8]. Le coût de cette version de l'algorithme est donc de $O(km(n - m) + m^2 n)$. Comme $km(n - m)$ est dominé par $m^2 n$, la complexité est donc en $O(m^2 n)$.

7.1.2 Extension à d'autres groupes de transformations

La définition de sous-système maximal peut être donnée pour toute classe de GCS. Nous définissons ainsi un Sous-système G-bien-contraint Maximal (MGS).

Définition 55. Sous-système G-bien-contraint Maximal : *Un MGS de $\mathcal{S}$ est un sous-système $\mathcal{S}_1 \subseteq \mathcal{S}$ tel que :*
– *$\mathcal{S}_1$ est G-bien-contraint ;*
– *il n'existe pas de système $\mathcal{S}_2$ G-bien-contraint tel que $\mathcal{S}_1 \subset \mathcal{S}_2$ et $\mathcal{S}_2 \subseteq \mathcal{S}$.*

Dans l'algorithme vu à la section précédente, l'identification d'un sous-système rigide maximal se fait en recherchant les coordonnées dont les variations peuvent s'exprimer en fonction des variations des coordonnées d'un repère pour les déplacements. Nous reprenons le même principe, mais l'utilisons pour identifier un MGS. Pour cela, il est uniquement nécessaire de connaître les types de repère des différents groupes considérés.

L'algorithme cité plus haut s'adapte alors sans souci : on réalise une première élimination de G -J après avoir permuté les colonnes pour placer un G-repère dans les colonnes de droite et si un MGS est identifié on le marque. Puis, jusqu'à ce que toutes les entités géométriques aient été marquées[9], on trouve une autre combinaison de $n - m$ colonnes contenant un G-repère non encore testé en utilisant l'algorithme 4.4 de modification d'un $\mathcal{R}$-flot.

Le pseudo-code de cette démarche est donné à l'algorithme 7.2.

7.1.3 Identification incrémentale des MGS

Dans l'optique d'une construction incrémentale du GCS, comme nous la concevons depuis le début de la partie II, il est intéressant de chercher à identifier les MGS au

[8] En réalité, $\Sigma_{i=1}^{3}(m \times (n - m + i))$: une fois les trois permutations effectuées, il faut faire les opérations de pivot nécessaires. Les $m - 3$ premières colonnes forment l'identité et ne coûtent donc rien. Lorsque la $m - 2^{\text{ème}}$ colonne a été traitée, les $m - 2$ premières colonnes forment l'identité, et ainsi de suite jusqu'à ce que les trois colonnes permutées aient été traitées.

[9] *Modulo* la détection des entités géométriques qui appartiennent à un système G-invariant maximal sous-G-défini

Entrées :
$\mathcal{S} = (C, X, A)$: Un GCS
$R \leftarrow \mathcal{R}$-flot maximal valide de $\mathcal{S}$
J : la matrice Jacobienne de $\mathcal{S}$ évaluée en un témoin
G : un groupe de transformation
Résultat :
Liste des MGS de $\mathcal{S}$

$i \leftarrow 0$
répéter
  Modifier R par l'algorithme 4.4 pour qu'il contienne un G-repère r non encore testé
  $E \leftarrow$ ensemble des entités qui sont entièrement dans r
  Permuter les colonnes de J pour correspondre à r
  Effectuer une élimination de G -J sur J
  $V \leftarrow$ ensemble des colonnes correspondant à des coordonnées exprimables en fonction de r
  si *$V \cup r$ contient des entités fixées qui ne sont pas dans E* **alors**
    Marquer avec l'étiquette i les colonnes de J correspondant à ces entités
    $i \leftarrow i + 1$
  sinon
    pour chaque *$e \in E$ non marqué* **faire**
      si *tous les G-repères contenant e ont été testés* **alors**
        `// Les MGS contenant e sont sous-G-définis`
        Marquer les colonnes de J correspondant à e avec l'étiquette i
        $i \leftarrow i + 1$
jusqu'à *ce que toutes les colonnes de J soient marquées*
$L \leftarrow$ liste vide
tant que $i \geq 0$ **faire**
  $\mathcal{S}' \leftarrow$ sous-système de $\mathcal{S}$ engendré par les entités étiquetées i
  $L \leftarrow L + \mathcal{S}'$
  $i \leftarrow i - 1$
retourner L

Algorithme 7.2: Algorithme d'identification des MGS basé sur la méthode du témoin

fur et à mesure de l'ajout des contraintes. Nous décrivons ici un algorithme pour ce faire.

À chaque ajout d'une contrainte c, nous effectuons la liste des groupes G, parmi les groupes pris en compte, tels que c est G-invariante[10]. Pour chaque groupe G satisfaisant cela, nous effectuons les permutations de colonnes appropriées pour simuler la fixation d'un G-repère fixant toutes les entités géométriques de $vars(c)$. Dans le cas où un G-repère ne peut être inclus dans $vars(c)$ (lorsque le système engendré est sous-G-défini), il faut tester les différents G-repères possibles. Pour chaque G-repère ainsi testé, nous identifions le MGS qu'il fixe. S'il inclut un MGS déjà identifié, ce dernier est retiré de la liste des MGS identifiés[11]. Il est donc nécessaire d'avoir la liste des configurations de G-repère fixant l'intégralité du système engendré par c, pour chaque type de contrainte.

Le pseudo-code de cette méthode est fourni à l'algorithme 7.3.

7.2 Approche combinatoire de l'identification de sous-systèmes G-bien-contraints

Dans cette section, nous donnons quelques pistes prometteuses pour identifier les sous-systèmes G-bien-contraints combinatoirement, dans le graphe de $\mathcal{R}$-flux, sans avoir recours à la méthode du témoin. Nous montrons que la méthode que nous proposons fonctionne bien dans de nombreux cas, mais se heurte aux mêmes limitations que toutes les méthodes à base de graphes.

7.2.1 Algorithmes

La méthode que nous détaillons ici est donnée, sous forme de pseudo-code, à l'algorithme 7.4. Ce pseudo-code parcourt les différents groupes G considérés : il exécute pour chaque groupe de transformations[12] l'algorithme 7.5.

Pour chaque groupe considéré, une rétro-propagation est possible pour éviter de raisonner sur des systèmes trop grands : pour chaque groupe G, nous commençons

[10] C'est-à-dire (*cf.* définition 35) les groupes G tels que les figures solutions du système engendré par c sont G-invariantes.

[11] Il serait également possible de le conserver et d'enregister la relation d'inclusion.

[12] À l'exception des groupes strictement inclus dans un groupe G' tel qu'il n'y a aucune contrainte qui ne soit pas G'-invariante. Pour des raisons de concision, ceci n'est pas exprimé à la ligne 2 de l'algorithme 7.4.

Entrées :
$\mathcal{S} = (C, X, A)$: Un GCS
$R \leftarrow \mathcal{R}$-flot maximal valide de $\mathcal{S}$
$c \in C$: la dernière contrainte ajoutée à $\mathcal{S}$
J : la matrice Jacobienne sous forme échelonnée réduite de $\mathcal{S}$ évaluée en un témoin
Résultat :
Liste des MGS contenant c

$i \leftarrow 0$
$\mathcal{G} \leftarrow$ liste des groupes d'invariance de c
pour chaque $G \in \mathcal{G}$ **faire**
  pour chaque *G-repère r fixant le sous-système engendré par c* **faire**
    $R \leftarrow R$ modifié par l'algorithme 4.4 incluant r
    Permuter les colonnes de J pour correspondre à R
    Mettre J sous forme échelonnée réduite
    Identifier le MGS $\mathcal{S}'$ fixé par r
    Marquer les colonnes de $\mathcal{S}'$ par i
    $i \leftarrow i + 1$
$L \leftarrow$ liste vide
tant que $i \geq 0$ **faire**
  $\mathcal{S}' \leftarrow$ sous-système de $\mathcal{S}$ engendré par les entités étiquetées i
  $L \leftarrow L + \mathcal{S}'$
  $i \leftarrow i - 1$
retourner L

Algorithme 7.3: Algorithme incrémental d'identification des MGS

Entrées :
$\mathcal{S} = (C, X, A)$: un GCS
Résultat :
Liste de sous-systèmes avec leur groupe de bonne constriction

---

$L \leftarrow$ liste vide
2 **pour chaque** *groupe G* **faire**
  $\mathcal{S}_G \leftarrow (C_G, X_G, A_G) \leftarrow$ sous-système de $\mathcal{S}$ engendré par ses contraintes G-invariantes
  Utiliser l'algorithme 7.5 pour $\mathcal{S}_G$ et G
  Ajouter le résultat à L
retourner L

Algorithme 7.4: Identification combinatoire des sous-systèmes bien contraints

donc par considérer le système $\mathcal{S}_G$ engendré par les contraintes G-invariantes. La rétro-propagation consiste tout simplement, à partir du graphe orienté non valué[13] d'un $\mathcal{R}$-flot de $\mathcal{S}_G$, à retirer récursivement tous les nœuds d'entité qui n'ont aucun arc sortant et un degré de repère nul. Ceci est réalisé par la boucle de la ligne 8.

Quand cela est fait, il y a deux possibilités :

1. les entités géométriques restantes ont toutes un degré de repère non nul,
2. il reste des entités géométriques avec un degré de repère nul.

Dans le premier cas, nous vérifions, pour chaque composante connexe[14] K restante, si les éléments de repère constituent un repère pour G. Si ce n'est pas le cas, mais que le nombre de degrés de repère de K pourrait être celui d'un G-repère, nous essayons de modifier les éléments de repère sur la composante connexe afin d'obtenir un G-repère. Si le nombre de degrés de repère de K est supérieur à celui d'un G-repère, nous essayons également de chercher un G-repère dans K. Si, directement ou *via* une modification, nous disposons d'un G-repère r, alors les entités géométriques de r, ainsi que toutes les entités géométriques qui ont été retirées et qui dépendaient uniquement de r, forment un système G-bien-contraint (ceci est testé aux lignes 20 à 23 de l'algorithme 7.5). S'il est impossible d'exhiber un G-repère dans K, on ne peut rien déduire.

Dans le second cas, il reste des entités géométriques avec un degré de repère nul : elles font partie d'une CFC du graphe orienté non valué. Dans ce cas, nous extrayons pour chaque CFC un graphe de $\mathcal{R}$-flux R' constitué de
– la CFC considérée ;
– les nœuds de contrainte liés à une entité géométrique de la CFC par un arc de valuation positive, ainsi que tous les nœuds d'entité liés à ces nœuds de contrainte.

Le nombre de degrés de repère d'un nœud d'entité x dans R' est égal à la somme du nombre de degrés de repère qu'il avait dans le graphe global et des valuations d'arc le reliant à des nœuds de contrainte extérieurs à R'. Cette extraction est réalisée par les lignes 15 à 16 de l'algorithme 7.5.

À la ligne 17, un appel est effectué à l'algorithme 7.6, qui traite R'. De toute évidence, le graphe orienté non valué de R' a une unique CFC. Il est possible, en revanche, de modifier R' afin de tenter de trouver une configuration du $\mathcal{R}$-flot qui permet de minimiser la taille de la CFC et de continuer à retirer certaines entités géométriques par rétro-propagation. Nous revenons alors aux deux cas évoqués plus haut. Si, en revanche, il n'est pas possible de diminuer la taille de la CFC, mais que le nombre de degrés de repère dans R' pourrait correspondre à un G-repère, nous testons tous les G-repères possibles pour R'. Si pour chaque G-repère il est possible de trouver un $\mathcal{R}$-flot valide maximal, nous faisons l'hypothèse que le GCS

[13] *Cf.* définition 51 p. 118
[14] Mais pas nécessairement fortement connexe

Entrées :
$\mathcal{S} = (C, X, A)$: un GCS ne contenant que des contraintes G-invariantes
R : $\mathcal{R}$-flot maximal valide de $\mathcal{S}$
G : un groupe de transformation
Résultat :
Liste de sous-systèmes G-bien-contraints de $\mathcal{S}$

$P \leftarrow$ pile vide
$D \leftarrow$ Graphe Orienté Acyclique (DAG) avec
– un nœud par entité de $\mathcal{S}$
– aucun arc
pour chaque *$x \in X$ de degré de repère nul dans R* **faire**
  si *aucun arc de valuation nulle ne relie n_x à un nœud de contrainte* **alors**
    Empiler x sur P
8 **tant que** *P n'est pas vide* **faire**
  Dépiler x de P
  Retirer x et les contraintes le concernant de $\mathcal{S}$ et mettre à jour R
  pour chaque *x' lié à une des contraintes retirées* **faire**
12 Ajouter un arc $x' \rightarrow x$ dans D
    si *plus aucun arc de valuation nulle ne relie $n_{x'}$ à un nœud de contrainte dans R* **alors** empiler x' sur P
tant que *des entités ont un degré de repère nul* **faire**
15 $x \leftarrow$ une entité de degré de repère nul
16 $R' \leftarrow$ Composante Fortement Connexe (CFC) « élargie » de $H(R)$ contenant `x// Voir texte`
17 Tenter de modifier R' par l'algorithme 7.6
  si *toutes les entités x ont été testées* **alors** sortir de la boucle
pour chaque *composante connexe K sans nœud de degré de repère nul* **faire**
20 **si** *K contient un G-repère r (éventuellement* via *l'algorithme 4.4)* **alors**
      tant que *D contient une entité dont les parents sont tous dans r* **faire**
        Rajouter à r les entités qui, dans D, ont tous leurs parents dans r
23   Le GCS engendré par les entités de r est G-bien-contraint
retourner *liste des GCS identifiés aux lignes 17 et 23*

Algorithme 7.5: Algorithme combinatoire d'identification de sous-systèmes G-bien-contraints

correspondant à R' est G-bien-contraint.

Entrées :
$\mathcal{S}' = (C', X', A')$: un GCS
R' : $\mathcal{R}$-flot maximal valide de $\mathcal{S}'$
G : groupe de transformation
Résultat :
Liste de sous-systèmes G-bien-contraints de $\mathcal{S}$

$K \leftarrow$ sous-graphe engendré par la CFC de $H(R')$ contenant les entités de degré de repère nul
$n \leftarrow$ somme des degrés de repère dans R'
4 **pour chaque** *G-repère r possible dans* R' **faire**
  Modifier R' par l'algorithme 4.4 pour inclure r
  si $n >$ *au nombre de degrés de repère d'un G-repère* **alors**
    tant que $H(R')$ *a une CFC de la même taille que* K **faire**
      si *tous les* $\mathcal{R}$*-flots incluant r ont été testés* **alors**
        Passer au prochain tour de la boucle de la ligne 4
      sinon
        Modifier R' pour tester un $\mathcal{R}$-flot incluant r non encore testé
    $L \leftarrow$ appel à l'algorithme 7.5
    retourner L
  sinon si *on ne peut pas trouver un* $\mathcal{R}$*-flot valide maximal incluant r* **alors**
15   **retourner** $\emptyset$

Algorithme 7.6: Tentative de cassage d'une CFC dans un $\mathcal{R}$-flot

À chaque fois qu'un sous-GCS $\mathcal{S}'$ est identifié comme G-bien-contraint, toutes ses entités géométriques internes[15] sont retirées, ainsi que les contraintes les concernant. Les autres entités géométriques, toujours liées à une contrainte, doivent alors être considérées comme fixées : si le retrait des entités internes et des contraintes associées a rendu un nœud d'entité insaturé, le nombre de ses degrés de repère est augmenté jusqu'à saturation.

[15] C'est-à-dire n'apparaissant dans aucune contrainte hors de $\mathcal{S}'$

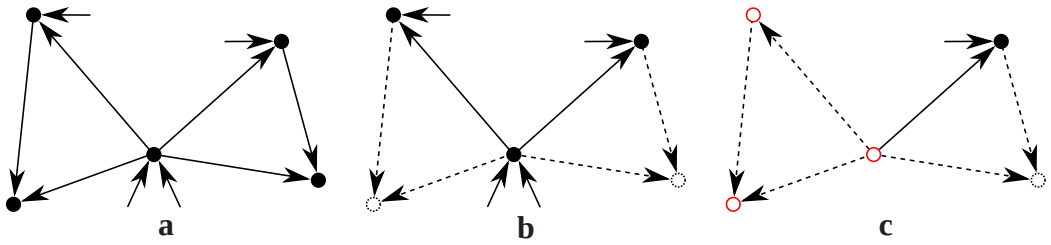


Figure 7.3 - Application de l'algorithme 7.4 sur le « papillon » ; **a** : $\mathcal{R}$-flot valide maximal ; **b** : les pointillés signalent les éléments retirés par rétro-propagation ; **c** : identification d'un **D**-repère et remplacement du GCS **D**-bien-contraint par ses entités fixées ; l'identification du triangle droit comme **D**-bien-contraint n'est pas illustrée.

7.2.2 Exemples d'application

7.2.2.1 Application sur le « papillon »

Considérons à nouveau le GCS de la figure 1.7[16], constitué de deux triangles rigides. Les seules contraintes sont des contraintes de distance : il ne contient donc aucune contrainte **S**-invariante et toutes les contraintes sont **D**-invariantes. La boucle de la ligne 2 de l'algorithme 7.4 va donc en réalité parcourir uniquement le groupe **D** des déplacements.

La figure 7.3**a** représente, avec la représentation de C , un $\mathcal{R}$-flot valide maximal pour le système engendré par les contraintes **D**-invariantes (*i.e.* tout le système). La phase de rétro-propagation est représentée à la figure 7.3**b** : les pointillés signalent les entités et les contraintes qui sont retirées. Ils signalent également les arcs ajoutés dans le DAG[17].

Il ne reste plus alors qu'une composante connexe, composée de trois points, deux contraintes de distance et quatre éléments de repère. Le nombre de degrés de repères ne correspond pas à un **D**-repère, mais on peut identifier un **D**-repère : le point central est fixé et le point de gauche est restreint. La barre de gauche, ainsi que toutes les entités géométriques qui dépendent uniquement de ces deux points sont donc identifiés, à la ligne 23, comme formant un GCS **D**-bien-contraint. Ce système est remplacé par ses entités, qui sont fixées : ceci est représenté à la figure 7.3**c**. Il ne reste donc plus qu'une barre, avec un **D**-repère : le triangle de droite est lui aussi identifié comme **D**-bien-contraint.

[16] *Cf.* p. 29
[17] *Cf.* ligne 12 de l'algorithme 7.5

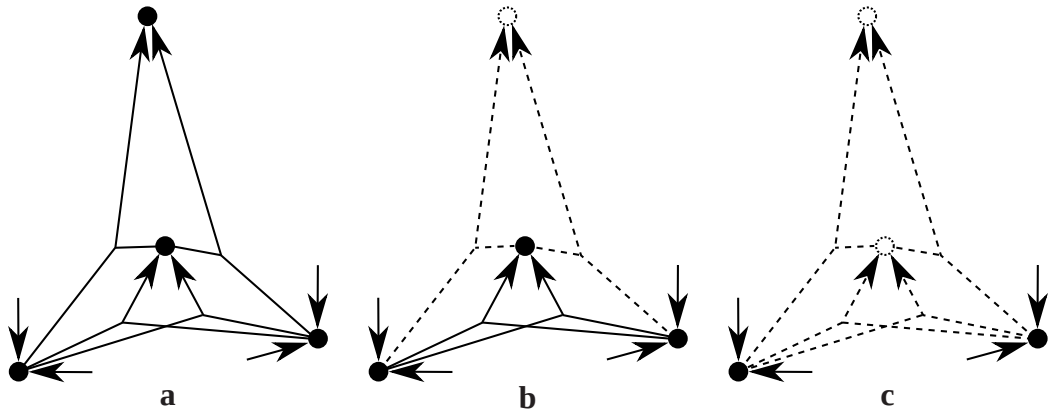


Figure 7.4 - Étapes de rétropropagation dans l'application de l'algorithme 7.4 sur le GCS de la figure 3.3.

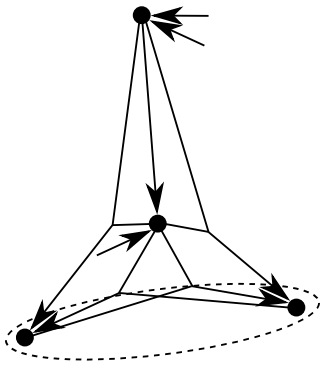


Figure 7.5 - $\mathcal{R}$-flot valide maximal du système de la figure 3.3 ; les pointillés indiquent la CFC du graphe orienté non valué.

7.2.2.2 Application sur le GCS de la figure 3.3

Considérons le GCS de la figure 3.3[18]. L'application de l'algorithme 7.4 amènera deux tours de boucle : un pour les similitudes **S**, un pour les déplacements **D**. La figure 7.4 illustre les étapes de la rétro-propagation sur le sous-système engendré par les contraintes **S**-invariantes. À l'étape **c**, il reste un **S**-repère, dont toutes les entités précédemment retirées dépendent : le sous-système engendré par les contraintes **S**-invariantes est donc **S**-bien-contraint.

La figure 7.5 montre un $\mathcal{R}$-flot valide maximal du système engendré par les contraintes **D**-invariantes, c'est-à-dire la totalité du GCS. Aucune rétro-propagation n'est possible avec notre méthode. Les pointillés indiquent la CFC contenant les entités géométriques de degré de repère non nul : il n'existe pas de $\mathcal{R}$-flot minimisant la taille de la CFC.

[18] *Cf.* p. 72

7.2.3 Limites

Nous avons donné des pistes pour une identification combinatoire des sous-systèmes G-bien-contraints dans le graphe de $\mathcal{R}$-flux, mais celles-ci ne sont encore que des pistes, car les algorithmes souffrent d'un certain nombre de limites. Certaines de ces limites sont les limites classiques des méthodes combinatoires actuelles : des redondances géométriques ne sont pas détectées, par exemple[19].

En outre, la rétro-propagation ne fonctionne que pour des graphes dont la connexité est inférieure au nombre de degrés de liberté des entités géométriques du GCS. Par exemple, on voit à la figure 5.4 que l'hexagone rigide dont le graphe de contrainte est $K_{3,3}$ ne dispose d'aucune entité que l'on puisse retirer, et ce quel que soit le $\mathcal{R}$-flot considéré.

L'identification de GCS G-bien-contraints est sensible au $\mathcal{R}$-flot utilisé au moment de la rétro-propagation : on voit à la figure 7.6 que si le $\mathcal{R}$-flot place un G-repère dans un sous-GCS G-bien-contraint, la rétropropagation peut terminer sur ce G-repère et identifier le sous-GCS (c'est le cas dans les figures 7.6**a** à 7.6**c**) ; tandis que si le $\mathcal{R}$-flot ne place pas de G-repère dans le sous-système, il se peut qu'il ne soit pas possible de l'identifier comme G-bien-contraint, comme c'est le cas dans les figures 7.6**d** à 7.6**f**).

Cet algorithme échoue également à identifier la sur-constriction[20]. La figure 7.7 montre une application de l'algorithme sur le système de la figure 6.3 : la rétro-propagation n'élimine aucune entité dans la partie sur-contrainte. La CFC contenant des entités géométriques de degré de repère nul est entourée en pointillés : le sous-graphe transmis à l'algorithme 7.6 correspond donc exactement à la partie génériquement sur-contrainte. Cela étant, si l'algorithme ne détecte pas ici la sur-constriction, il ne conclut pas non plus à la bonne constriction : il s'arrête à la ligne 15 de l'algorithme 7.6.

L'algorithme identifie comme bien-contraint *modulo* les déplacements la double-banane : la figure 7.8**a** montre un $\mathcal{R}$-flot maximal valide pour la double-banane et on y voit que le triangle central de la banane de droite forme une CFC. Le sous-graphe transmis à l'algorithme 7.6 est donc la banane de droite : elle est identifiée comme **D**-bien-contrainte puisqu'une modification du $\mathcal{R}$-flot permet d'effectuer une rétro-propagation jusqu'à un **D**-repère. C'est ce que l'on voit aux figures 7.8**b** et 7.8**c**, dans la banane de gauche : après retrait de la banane de droite, le nœud correspon-

[19] *E.g.* considérer un système 3D où un point p est contraint à être incident à une droite d et à un plan P, la droite d étant elle-même contrainte à être incidente à P.

[20] Sans surprise, puisqu'il se base sur un graphe de $\mathcal{R}$-flux et souffre donc de tous les problèmes évoqués à la section 6.2.

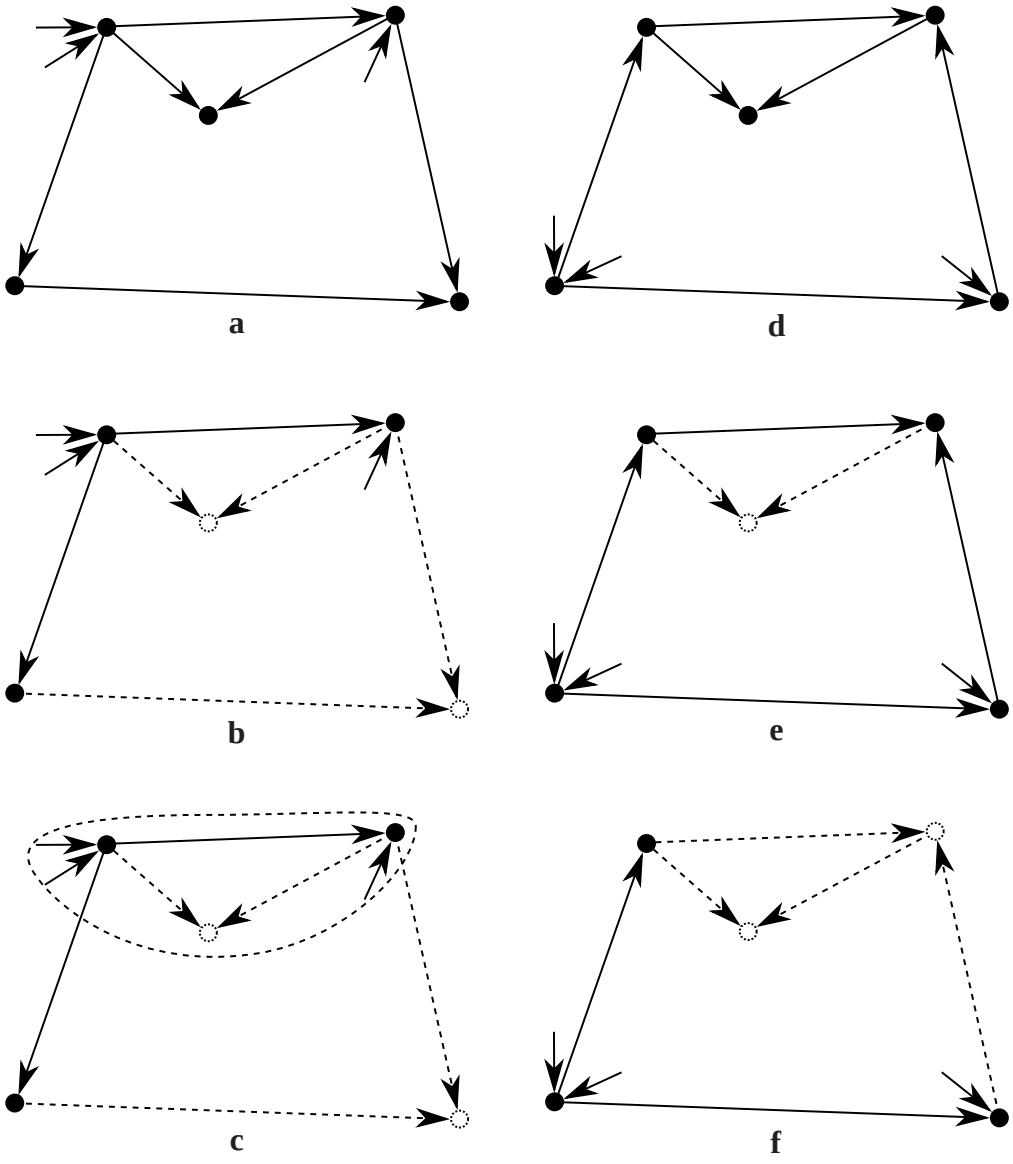


Figure 7.6 - La rétro-propagation dans un graphe de $\mathcal{R}$-flux est sensible au $\mathcal{R}$-flot : les étapes **a**, **b** et **c** montrent la rétro-propagation avec un premier $\mathcal{R}$-flot : le triangle du haut est identifié comme **D**-bien-contraint ; les étapes **d**, **e** et **f** montrent la rétro-propagation avec un second $\mathcal{R}$-flot : il n'y a aucun **D**-repère duquel dépendent des entités retirées durant la rétro-propagation.

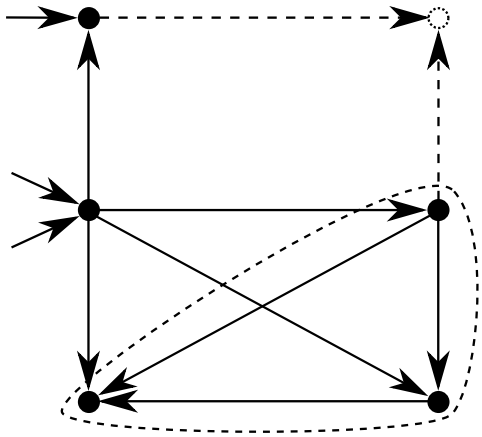


Figure 7.7 - Application de l'algorithme 7.5 pour les déplacements sur l'exemple de H : les flèches pointillées indiquent la rétro-propagation ; l'ensemble entouré en pointillés représente la CFC restante avec des entités de degré de repère nul.

dant au point inférieur peut être retiré (figure 7.8**b**), puis c'est le tour de l'un des points du triangle central (figure 7.8**c**).

7.3 Modifications de l'algorithme de paramétrisation combinatoire

Dans cette section, nous proposons des modifications de l'algorithme 4.1 de paramétrisation basé sur le graphe de $\mathcal{R}$-flux. Ces modifications visent à permettre la prise en compte de systèmes G-bien-contraints comme des entités atomiques dans le graphe de $\mathcal{R}$-flux. Ces changements nécessitent deux éléments :

– la connaissance, pour chaque système atomique, du nombre de degrés de repère nécessaire à leur fixation, et de l'ensemble des entités de ce système susceptibles d'être liées par une contrainte à une entité extérieure,
– une méthode permettant, à l'ajout d'une contrainte, d'identifier les MGS auxquels appartient cette contrainte.

Lorsqu'un sous-système $\mathcal{S}$ est identifié comme G-bien-contraint, soit *via* l'ajout d'une contrainte, soit parce que l'utilisateur a ajouté un système G-bien-contraint atomique, les contraintes internes de $\mathcal{S}$ et ses nœuds de repère peuvent être supprimés : nous ne conservons plus que les nœuds d'entité de $\mathcal{S}$, avec l'information de leur appartenance à $\mathcal{S}$. Nous ajoutons un unique nœud de repère $r_{\mathcal{S}}$, lié à l'ensemble des entités de $\mathcal{S}$. L'arc allant d'un nœud d'entité à $r_{\mathcal{S}}$ a pour capacité le nombre de degrés de liberté de l'entité. L'arc liant $r_{\mathcal{S}}$ au nœud final n_p a une capacité égale au nombre de degrés de repère d'un G-repère.

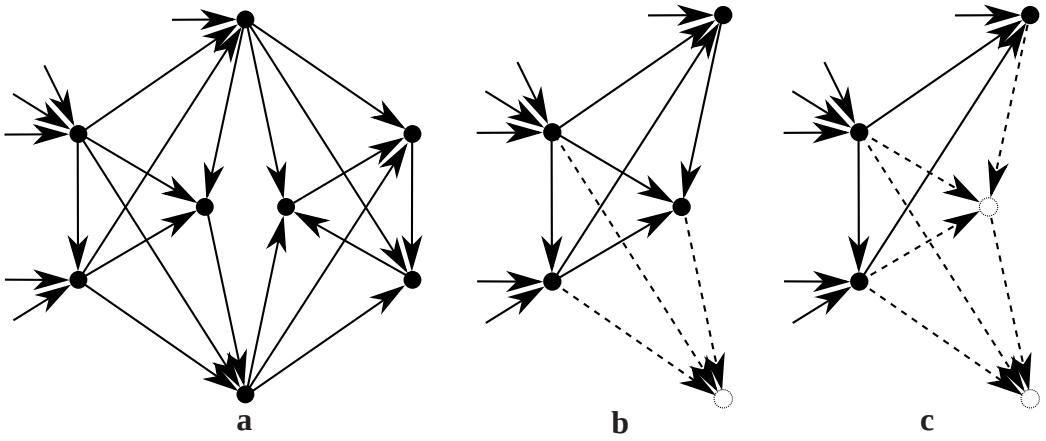


Figure 7.8 - La double-banane est **D**-bien-contrainte d'après l'algorithme 7.5

La définition d'un $\mathcal{R}$-flot maximal valide[21] doit également être modifiée : il n'est plus nécessaire, dès lors que les contraintes internes et les nœuds de repère sont supprimés, que toutes les entités géométriques de $\mathcal{S}$ soient saturées : le $\mathcal{R}$-flot est valide et maximal si :

– les nœuds d'entité et de contrainte n'appartenant pas à un système identifié comme G-bien-contraint sont saturés, comme dans la définition 48 ;
– la somme des arcs sortant d'un nœud d'entité appartenant à un système identifié comme G-bien-contraint est au maximum la capacité de son arc entrant ;
– la somme des arcs sortants des nœuds d'entité appartenant à un système identifié comme G-bien-contraint et du degré de repère de ce système est égale au nombre de degrés de repère d'un G-repère[22].

L'algorithme 4.1 peut alors être modifié pour qu'à chaque ajout de contrainte, plusieurs tests soient effectués :

– si une contrainte G-invariante est ajoutée entre des entités géométriques appartenant à un même système G-bien-contraint, elle est redondante ;
– si l'ajout de la contrainte induit la création d'un sous-système G-bien-contraint, on modifie le graphe de $\mathcal{R}$-flux en conséquence, comme expliqué ci-dessus (suppression des nœuds de contrainte et des nœuds de repère, ajout d'un nœud de repère unique).

La recherche de chemin (et leur retournement) peut se faire en considérant que n'importe quel couple d'entités géométriques d'un système G-bien-contraint est relié par une contrainte.

La figure 7.9 illustre la construction incrémentale d'un GCS avec l'algorithme 4.1 modifié pour prendre en compte la connaissance de la G-bonne-constriction de sous-systèmes. Il utilise la représentation de C , chaque arc représentant ici une

[21] *Cf.* définition 48 p. 93
[22] Avec l'exception des systèmes sous-G-définis

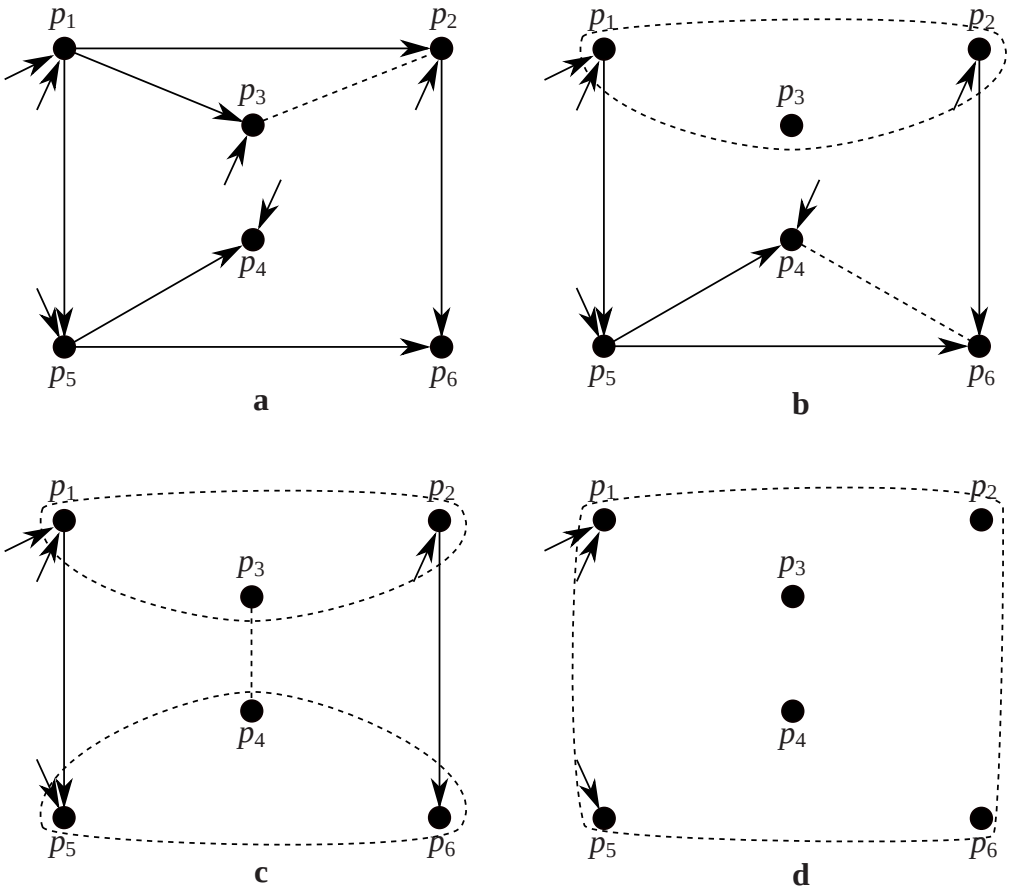


Figure 7.9 - Paramétrisation combinatoire avec connaissance de la **D**-bonne-constriction de sous-systèmes

contrainte de distance entre deux points. Les traits pointillés indiquent la prochaine contrainte de distance ajoutée. Les parties entourées en pointillés signalent un système **D**-bien-contraint. Notons que, dans la mesure où en 2D un système de deux points contraints par une distance est rigide, il ne devrait en réalité y avoir aucun arc représenté à la figure 7.9 mais, pour des raisons de clarté, nous n'affichons la connaissance de la **D**-bonne-constriction que pour des systèmes plus grands qu'un système trivial.

C 8

$\mathcal{W}$-´

Sommaire

Le meilleur moyen de résoudre un problème est d'en nier l'énoncé
– André Frossard, journaliste français

Bien souvent, les méthodes de décomposition de la littérature sont liées à l'algorithme de résolution qu'elles accompagnent. Une méthode de décomposition accompagnant un algorithme de résolution à base de règles sera elle-même à base de règles, par exemple. Dans ce cas, les méthodes de décomposition souffrent des mêmes limitations que les méthodes de résolution : lorsqu'elles sont basées sur une analyse du graphe, elles tombent dans les pièges de la combinatoire et des erreurs de la caractérisation de la rigidité structurelle ; lorsqu'elles sont basées sur un système de règles, elles ne détecteront pas la bonne constriction (*modulo* les déplacements ou un autre groupe) d'un sous-système si elle est due à un théorème géométrique qui n'a pas été pris en compte par le concepteur.

Dans ce chapitre, nous proposons un algorithme de décomposition très général, basé sur la méthode du témoin, que nous avons baptisé $\mathcal{W}$-décomposition (*angl.* : Witness-decomposition). Il est en fait basé sur l'existence d'un algorithme d'identification des sous-systèmes **D**-bien-contraints maximaux et l'algorithme que nous avons proposé au chapitre 7 est basé sur la méthode du témoin[1]. Cet algorithme pourrait toutefois être remplacé par n'importe quel autre algorithme d'identification des MRS et c'est de la puissance de cet algorithme que dépendrait directement la méthode de décomposition correspondante.

L'objectif de la $\mathcal{W}$-décomposition est de décomposer un système rigide en l'ensemble de ses sous-systèmes rigides non triviaux, c'est-à-dire différents de leur bord. Pour cela, le principe général, déjà esquissé dans deux articles traitant de la méthode du témoin [MF06, JTNM06] est de retirer une contrainte du système et de rechercher tous les MRS du système résultant. Ceux-ci sont récursivement $\mathcal{W}$-décomposés et remplacés par leur bord.

Dans la section 8.1 nous donnons les pré-requis de la $\mathcal{W}$-décomposition, c'est-à-dire les caractéristiques minimales de l'algorithme utilisé pour identifier les MRS. La section 8.2 donne alors l'algorithme précis de la $\mathcal{W}$-décomposition suivi de quelques exemples d'utilisation. La section 8.3 propose une extension multi-groupe de la $\mathcal{W}$-décomposition et enfin la section 8.4 effectue une analyse des algorithmes proposés dans ce chapitre et des extensions sont également proposées.

8.1 Pré-requis pour la $\mathcal{W}$-décomposition

L'algorithme de $\mathcal{W}$-décomposition explicité à la section 8.2 se base sur l'existence d'une méthode d'identification des MRS. Nous avons donné à la section 7.1 une telle méthode, mais elle pourrait être remplacée par une autre méthode équivalente. Dans la suite du présent chapitre, nous appelons `IdMRS` l'algorithme servant à identifier les MRS. Cet algorithme doit être capable, à partir d'un système, rigide ou non, de donner la liste de ses Sous-systèmes Rigides Maximaux non triviaux.

Définition 56. Sous-système trivial : *Un GCS $\mathcal{S}' \subset \mathcal{S}$ est dit* trivial *dans $\mathcal{S}$ s'il est égal à une base de $\mathcal{B}_{\mathcal{S}-\mathcal{S}'}(\mathcal{S}')$.*

Exemple 24. Sous-système trivial

Dans l'esquisse de la figure 7.1[2], les systèmes limités aux segments $[p_1p_4]$, $[p_1p_2]$ et $[p_2p_3]$ (c'est-à-dire constitués chacun d'un couple de points et de la contrainte de distance entre ces deux points) sont triviaux : en effet, ils sont rigides et chacun confondus avec leur bord.

[1] *Cf.* section 7.1
[2] *Cf.* p. 157

De même que pour l'expression du bord, en l'absence de confusion sur $\mathcal{S}$, on pourra dire d'un sous-système $\mathcal{S}'$ trivial dans $\mathcal{S}$ qu'il est trivial.

Notons qu'un GCS trivial peut avoir des sous-systèmes triviaux. Un système trivial dont l'ensemble de contraintes est non vide contient au moins une entité géométrique, or le système induit par cette entité seule (c'est-à-dire ne contenant aucune contrainte) est lui aussi trivial.

Notons également que la trivialité d'un GCS dépend non seulement du GCS dans lequel il est inclus mais aussi de l'univers géométrique considéré. Ainsi, un système représentant un triangle contraint par les distances de ses trois côtés n'est pas trivial si l'univers géométrique permet d'exprimer des contraintes d'angle : les trois angles font en effet partie du bord. Si l'univers géométrique ne permet d'exprimer que les trois distances, le triangle est trivial.

L'algorithme de $\mathcal{W}$-décomposition est récursif et s'arrête lorsqu'un système est $\mathcal{W}$-indécomposable.

Définition 57. **$\mathcal{W}$-indécomposabilité :** *Soit $\mathcal{S} = (C, X, A)$ un GCS rigide. $\mathcal{S}$ est $\mathcal{W}$-indécomposable si $\forall c \in C$, les MRS de $(C \backslash \{c\}, X, A)$ sont triviaux dans $\mathcal{S}$.*

La fiabilité de l'algorithme `IdMRS` utilisé est importante : si l'algorithme `IdMRS` n'est pas capable d'identifier les MRS et produit une liste contenant un ou des sous-systèmes rigides non maximaux (c'est-à-dire qui sont sous-systèmes d'un GCS rigide), alors la puissance de la $\mathcal{W}$-décomposition sera réduite puisqu'elle ne fournira pas tous les sous-systèmes rigides. Il existe alors un risque de classer comme $\mathcal{W}$-indécomposable un système qui a des sous-systèmes rigides non triviaux.

De même, si l'algorithme `IdMRS` fournit une liste contenant des systèmes non rigides, alors la $\mathcal{W}$-décomposition correspondante sera inexacte.

8.2 $\mathcal{W}$-décomposition rigide

8.2.1 Algorithme

L'algorithme de $\mathcal{W}$-décomposition vise à décomposer un système rigide en ses sous-systèmes rigides maximaux stricts. Par une récursion sur ces GCS, nous obtenons au final l'ensemble des sous-systèmes rigides. La condition d'arrêt est l'obtention d'une liste de sous-systèmes ne contenant que des systèmes triviaux.

L'idée de la $\mathcal{W}$-décomposition est de retirer une contrainte c du système $\mathcal{S} = (C, X, A)$ à décomposer. En utilisant `IdMRS`, les MRS de $(C\backslash\{c\}, X, A)$ sont identifiés en une liste de systèmes $\mathcal{S}_i$. Si cette liste est vide, on réintroduit la contrainte c dans le système et on réessaie avec une autre contrainte. Si l'identification des MRS a échoué avec toutes les contraintes, alors $\mathcal{S}$ est $\mathcal{W}$-indécomposable. Dans ce cas, on renvoie la liste contenant uniquement $\mathcal{S}$.

Si la liste des $\mathcal{S}_i$ n'est pas vide, alors on réintroduit la contrainte c puis chacun de ces systèmes est remplacé par son bord. On obtient donc un système $\mathcal{S}' = \mathcal{S} - \Sigma_i(\mathcal{S}_i) + \Sigma_i(\mathcal{B}_{\mathcal{S}-\mathcal{S}_i}(\mathcal{S}_i))$. On $\mathcal{W}$-décompose récursivement $\mathcal{S}'$. De même, on $\mathcal{W}$-décompose récursivement chacun des $\mathcal{S}_i$. On renvoie alors la concaténation des listes obtenues dans ces diverses récursions. Ces informations peuvent bien évidemment être structurées, par exemple sous la forme d'un arbre, en renvoyant un arbre dont la racine est $\mathcal{S}$ et dont les fils sont les arbres (éventuellement limités à une feuille dans le cas d'un système $\mathcal{W}$-indécomposable) renvoyés par chacune des récursions sur $\mathcal{S}'$ et les systèmes $\mathcal{S}_i$.

Le pseudo-code de la $\mathcal{W}$-décomposition est donné à l'algorithme 8.1.

À la ligne 15 de l'algorithme, il faut ajouter le bord d'un système que l'on a retiré à la ligne d'au-dessus. Le calcul du bord peut sembler aisé, dans la mesure où l'on sait que pour un GCS rigide, la connaissance de l'ensemble des distances point à point et des angles entre deux droites et entre deux plans permet de caractériser l'ensemble des solutions. Toutefois, calculer l'ensemble de ces informations sur les entités géométriques communes au système $\mathcal{S}_i$ et au reste de $\mathcal{S}$ peut mener à un système qui, quoique bien contraint, est génériquement sur-contraint.

L'algorithme de $\mathcal{W}$-décomposition en lui-même n'est pas sensible à la sur-contriction générique. Toutefois, si l'algorithme `IdMRS` utilisé est sensible à cela, il est nécessaire de s'assurer que le bord calculé n'est pas génériquement sur-contraint. Rappelons que dans le chapitre 6[3] nous donnons une méthode de calcul incrémental du bord, basée sur le témoin elle aussi, qui assure que le système ainsi calculé n'est pas génériquement sur-contraint.

La $\mathcal{W}$-décomposabilité d'un système ne présume pas de sa résolubilité : encore faut-il être capable de résoudre les feuilles, c'est-à-dire les systèmes $\mathcal{W}$-indécomposables. Le premier exemple de la section 8.2.2 termine par exemple sur des feuilles considérées comme délicates à résoudre, puisqu'il s'agit de systèmes non constructibles à la règle et au compas, en l'occurence des systèmes $K_{3,3}$ que nous avons déjà vu en figure 1.12.

En revanche, si les systèmes $\mathcal{W}$-indécomposables sont résolubles par les solveurs

[3] *Cf.* section 6.3.1 p. 139

Entrées :
$\mathcal{S} = (C, X, A)$: Un GCS rigide
Résultat :
Arbre des MRS de $\mathcal{S}$

---

répéter
2 Sélectionner une contrainte $c \in C$
3 $L \leftarrow$ liste des MRS de $(C \backslash c, X, A)$ identifiés par `IdMRS`
tant que *L contient uniquement des MRS triviaux* **faire**
Sélectionner une contrainte c qui n'a pas encore été sélectionnée
$L \leftarrow$ liste des MRS de $(C \backslash c, X, A)$ identifiés par `IdMRS`
jusqu'à *ce que toutes les contraintes aient été testées ou qu'il y ait un MRS non trivial*
si *L contient uniquement des MRS triviaux* **alors**
retourner *une feuille étiquetée par* $\mathcal{S}$
sinon
$\mathcal{A} \leftarrow$ arbre sans fils étiqueté par $\mathcal{S}$
pour chaque $\mathcal{S}_i \in L$ **faire**
13 Enraciner la $\mathcal{W}$-décomposition de $\mathcal{S}_i$ comme fils de $\mathcal{A}$
$\mathcal{S} \leftarrow \mathcal{S} - \mathcal{S}_i$
15 $\mathcal{S} \leftarrow \mathcal{S} + \mathcal{B}_{\mathcal{S}}(\mathcal{S}_i)$
16 Enraciner la $\mathcal{W}$-décomposition de $\mathcal{S}$ comme fils de $\mathcal{A}$
retourner $\mathcal{A}$

Algorithme 8.1: $\mathcal{W}$-décomposition

dont on dispose, alors on a l'assurance de pouvoir assembler leurs solutions par des transformations géométriques. En effet, lorsqu'un sous-système $\mathcal{S}'$ rigide est identifié dans un système rigide $\mathcal{S}$, son bord ne peut pas être sous-**D**-défini[4] (sinon, cela signifierait que $\mathcal{S}'$ est en libre rotation autour de son unique point commun avec le reste du système, ou en libre translation le long de l'unique droite ou de l'unique plan commun). Par définition du bord, si f_1 est une solution de $\mathcal{S}'$ et f_2 est une solution de $\mathcal{S} - \mathcal{S}' + \mathcal{B}_{\mathcal{S}-\mathcal{S}'}(\mathcal{S}')$, il existe $\varphi \in \mathbf{D}, \varphi(f_1) \equiv_{X_e} f_2$, c'est-à-dire que l'on peut donc déplacer une solution de $\mathcal{S}'$ pour qu'elle soit compatible avec une solution du système où $\mathcal{S}'$ a été remplacé par son bord.

Une autre possibilité[5] serait de ne pas remplacer les MRS identifiés par leur bord, mais de les conserver, et de sélectionner tour à tour toutes les contraintes pour s'assurer d'avoir identifié toutes les MRS stricts. Dans ce cas, toutefois, on obtient une liste de sous-systèmes rigides sans assurance d'être capable de les assembler. Rien n'assure, par exemple, que l'on se retrouve dans un cas qu'O [Owe91], S - [Sit06] ou S [Sun87] sachent assembler.

8.2.2 Exemples d'application

8.2.2.1 Premier exemple 2D : double $K_{3,3}$

Illustrons l'exécution de l'algorithme sur l'exemple de la figure 8.1**a**, qui représente un GCS $\mathcal{S}$ **D**-bien-contraint en 2D. Son graphe de contraintes est 3-connexe et possède deux sous-graphes $K_{3,3}$ rigides : $p_1p_3p_5p_9p_8p_7$ et $p_2p_{10}p_{11}p_{12}p_6p_4$ (il s'agit de systèmes comme celui représenté à la figure 1.12), connectés par trois contraintes de distance. L'ensemble étant rigide, l'algorithme `IdMRS` doit renvoyer un unique MRS. Considérons la sélection de deux contraintes à la ligne 2 de l'algorithme 8.1 : les contraintes c_1 et c_2, représentées en pointillés.

En sélectionnant la contrainte c_1, l'appel à `IdMRS` à la ligne 3 identifie quatre MRS : les deux sous-systèmes $K_{3,3}$, et chacune des deux barres rigides entre eux. Ces deux derniers sont triviaux. En remplaçant les deux hexagones rigides par leur bord, on obtient le système de la figure 8.1**b**. Les récursions montrent que le système résultant est $\mathcal{W}$-indécomposable (ligne 16), comme le sont chacun des deux hexagones rigides (ligne 13).

Si l'on ne sélectionne pas la contrainte c_1 mais plutôt la contrainte c_2, l'hexagone de

[4] *Cf.* théorème 3 p. 71

[5] Par exemple dans le cas où l'on ne dispose pas d'une méthode de calcul d'un bord qui ne soit pas génériquement sur-contraint et où `IdMRS` est sensible à la sur-constriction générique

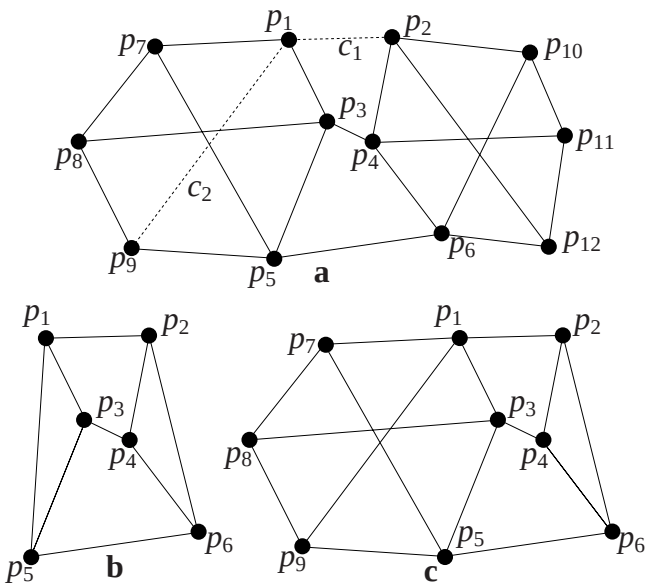


Figure 8.1 - Exemple d'utilisation de la $\mathcal{W}$-décomposition rigide sur un GCS 2D 3-connexe (les traits représentent des distances point-point) ; **a** : GCS dont le graphe de contraintes est 3-connexe fait de deux systèmes $K_{3,3}$ connectés par 3 contraintes ; **b** (resp. **c**) : systèmes obtenus en remplaçant les MRS identifiés par l'algorithme 8.1 par leur bord quand la contrainte c_1 (resp. c_2) est sélectionnée.

gauche de la figure 8.1**a** ($p_1p_3p_5p_9p_8p_7$) n'est plus rigide. L'appel à `IdMRS` n'identifie donc, comme MRS non trivial, que l'hexagone de droite de la figure 8.1**a** ($p_2p_{10}p_{11}p_{12}p_6p_4$). Une fois qu'il est remplacé par son bord, on obtient le système représenté à la figure 8.1**c**. Ce système est $\mathcal{W}$-décomposable : dès que la contrainte sélectionnée sera en dehors de l'hexagone de gauche, celui-ci sera identifié comme rigide et remplacé par son bord, ce qui ramènera au système de la figure 8.1**b**.

8.2.2.2 Second exemple 2D : hexagone « de V »

Étudions maintenant un exemple construit sur un univers géométrique plus vaste que celui de la figure 8.1, constitué uniquement de distances point-point. La figure 8.2 représente un hexagone rigide qui, comme l'hexagone de la figure 1.12, n'est pas constructible à la règle et au compas. Il est, en revanche, décomposable (il est notamment décomposable par la méthode d'O étendue à un univers géométrique incluant des angles) et nous allons montrer comment s'effectue sa $\mathcal{W}$-décomposition. Les différentes étapes sont représentées à la figure 8.3.

Supposons que la première contrainte sélectionnée soit l'angle $\widehat{p_1p_2p_3}$. Comme

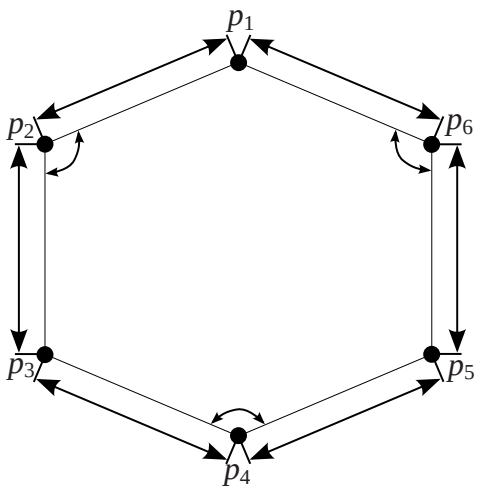


Figure 8.2 - Hexagone « de
V » [VSR92]

le montre la figure 8.3**a**, les MRS du système résultant de la suppression de la contrainte d'angle sont le triangle $p_3p_4p_5$, le triangle $p_1p_5p_6$, la barre p_1p_2 et la barre p_2p_3. Ces deux derniers sont des systèmes triviaux. Les deux triangles, en revanche, ne sont pas égaux à leur bord, puisque le bord de $p_3p_4p_5$ est le système engendré par la distance entre p_3 et p_5, par exemple.

Le remplacement des deux MRS non triviaux par leur bord mène au système représenté à la figure 8.3**b**. La récursion sur chacun de ces MRS mènera à la conclusion qu'ils sont $\mathcal{W}$-indécomposables. Concernant la récursion sur le système résultant de leur remplacement, la sélection de la contrainte de distance entre p_1 et p_2 ou entre p_2 et p_3 ne mènera pas à la détection d'un autre MRS non trivial, comme le montre la figure 8.3**c**. En revanche, si l'on sélectionne la contrainte de distance entre p_3 et p_5 (il en serait de même avec celle entre p_1 et p_5), alors le MRS $p_1p_2p_3$ est identifié et remplacé par son bord. Cela mène au système représenté à la figure 8.3**d**, qui est $\mathcal{W}$-indécomposable.

8.2.2.3 Exemple 3D

Considérons maintenant un exemple en dimension 3, avec le système de la figure 8.4**a**. Les segments (pleins ou pointillés) représentent des contraintes de distance. L'œil humain y voit un système composé d'une pyramide (*acedg*) dont la base est un quadrilatère (*aced*), qu'elle partage avec un polyèdre P non régulier (autrement, il s'agit d'un cas dégénéré) ayant deux faces triangulaires et trois faces quadrangulaires (non contraintes à être planaires).

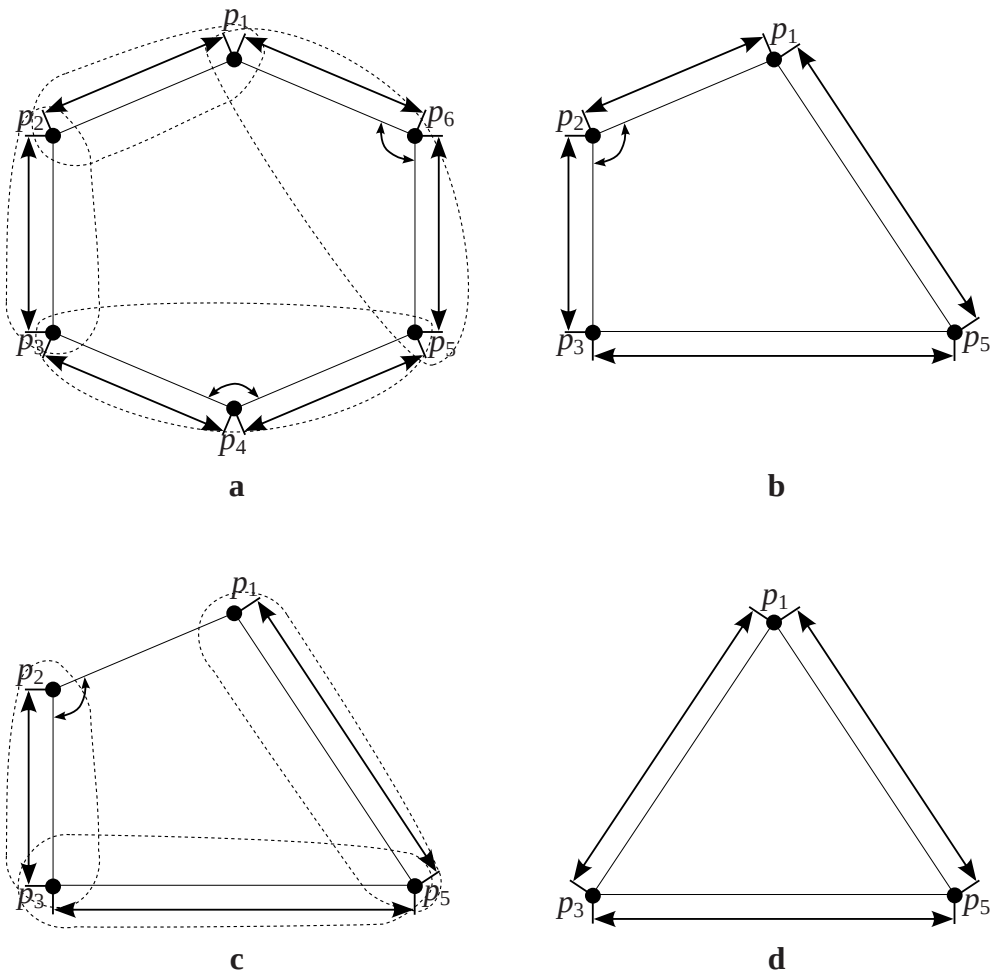
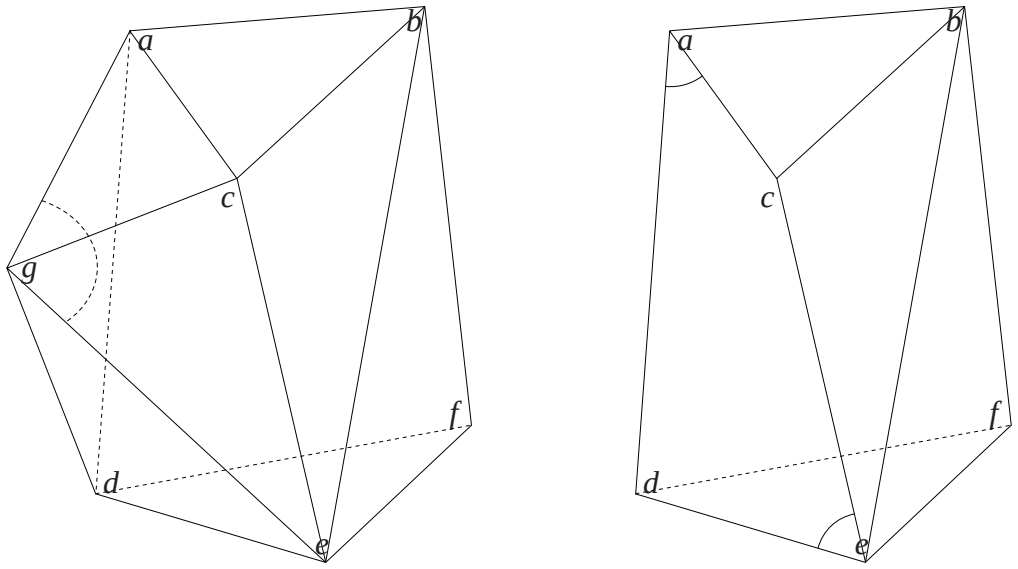


Figure 8.3 - $\mathcal{W}$-décomposition rigide de l'hexagone « de V » (*cf.* figure 8.2). Voir texte.



a **b**

Figure 8.4 - $\mathcal{W}$-décomposition rigide d'un système 3D. **a** : esquisse d'un système rigide ; **b** : remplacement de la pyramide par son bord (le système résultant est $\mathcal{W}$-décomposable).

L'ensemble est rigide, aussi l'utilisation d'`IdMRS` mène à l'identification de l'ensemble du système. Supposons que l'on sélectionne la distance qui est en diagonale d'un des quadrilatères (*b*-*e*). La pyramide est identifiée comme un MRS et est remplacée par une base de son bord, c'est-à-dire les quatre contraintes de distance qu'elle partage avec le reste du système et deux autres contraintes, ici des contraintes d'angle. Le système résultant est représenté à la figure 8.4**b**.

Le MRS de la pyramide est encore $\mathcal{W}$-décomposable : si l'on retire l'une des quatre distances qu'elle partage avec *P*, par exemple la distance *c*-*e*, on identifie comme rigide une moitié de la pyramide, *adeg*.

De même, le GCS de la figure 8.4**b** est encore $\mathcal{W}$-décomposable : si l'on retire la contrainte entre *e* et f, on identifie comme rigide le polyèdre *abcde*.

8.3 Extension multi-groupe

8.3.1 Algorithme

L'algorithme de $\mathcal{W}$-décomposition que nous avons présenté a pour objectif de décomposer un GCS rigide en ses sous-systèmes rigides. Il se base sur un algorithme

d'identification des sous-systèmes rigides maximaux. Dans cette section, nous discutons une extension multi-groupe de la $\mathcal{W}$-décomposition, c'est-à-dire une extension décomposant un GCS non forcément rigide en ses sous-systèmes G-bien-contraints avec G un groupe de transformations dans l'ensemble $\mathcal{G}$ des groupes que l'on sait gérer.

Pour cela, il est nécessaire de disposer d'un algorithme d'identification des sous-systèmes maximaux bien-contraints *modulo* n'importe lequel des groupes de $\mathcal{G}$.

Identifier des systèmes bien contraints *modulo* les translations ou les rotations est aisé si l'on dispose d'un algorithme d'identification des MRS, puisqu'il suffit de rechercher les MRS dont une direction est fixée ou dont un point est fixé. La recherche des systèmes **S**-bien-contraints maximaux en revanche doit être un algorithme à part. Là encore, l'extension de la méthode du témoin proposée à la section 7.1.2 réalise cela pour n'importe quel groupe de transformations dont on connaît les différents types de repère.

L'algorithme de $\mathcal{W}$-décomposition en lui-même nécessite peu de modifications : il convient, plutôt que d'identifier les MRS, de chercher à identifier les sous-systèmes G-bien-contraints pour chaque $G \in \mathcal{G}$. Comme l'a montré [SM06], il est préférable alors de commencer par le plus petit groupe de transformations puis de remonter dans les groupes de transformations : on ne teste un groupe G que si tous les groupes $G' \subset G$ ont déjà été testés. Dans la suite, nous appelons cet ordre partiel l'ordre d'inclusion croissant. Dans notre implantation, les groupes considérés sont les compositions des rotations, des translations et des homothéties : nous commençons donc par l'identité, puis passons aux rotations et translations, puis aux déplacements et enfin aux similitudes.

Lors de la récursion sur un sous-système G-bien-contraint, il est inutile de chercher à identifier des sous-systèmes G'-bien-contraints si $G' \subset G$ puisqu'il ne peut en exister que dans un cas. Il s'agit du cas spécifique d'un système $\mathcal{S}$ qui est à la fois G-bien-contraint et G'-bien-contraint[6]. Dans ce cas, le sous-système G'-bien-contraint maximal de $\mathcal{S}$ est $\mathcal{S}$ lui-même, ce qui signifie que la $\mathcal{W}$-décomposition de $\mathcal{S}$ pour le groupe G' aura déjà été effectuée avant l'appel récursif. Ainsi, on peut avoir l'assurance que, dans l'arbre rendu par l'algorithme de $\mathcal{W}$-décomposition, les fils d'un nœud donné sont tous bien-contraints *modulo* un groupe plus grand que le groupe de bonne constriction de leur parent.

Par conséquent, l'assemblage des fils d'un nœud dont le groupe de bonne constriction est dans $\mathcal{G}$ pourra s'effectuer sans difficultés. En effet, le théorème 3 nous assure que le bord d'un fils $\mathcal{S}_i$ par rapport au reste du système est un repère pour G_i, le

[6] Par exemple avec G les déplacements et G' le groupe engendré par les déplacements et les symétries

groupe de bonne constriction de $\mathcal{S}_i$. Les systèmes sont donc compatibles pour une jointure par transformation.

Lors du premier appel à l'algorithme de $\mathcal{W}$-décomposition multi-groupe, le groupe de bonne constriction du système est inconnu et il est même possible qu'il soit sous-contraint *modulo* tous les groupes de $\mathcal{G}$. Il faut donc commencer par une recherche, sans sélection de contraintes, des sous-systèmes G-bien-contraints pour tous les $G \in \mathcal{G}$ dans l'ordre d'inclusion croissant explicité plus haut. L'algorithme de $\mathcal{W}$-décomposition multi-groupe à proprement parler est alors utilisé sur ces systèmes. Le pseudo-code est donné à l'algorithme 8.2.

Entrées :
$\mathcal{S} = (C, X, A)$: Un GCS
G : Le groupe de bonne constriction de $\mathcal{S}$
$\mathcal{G}$: Ensemble partiellement ordonné des groupes de transformations considérés
Résultat :
Arbre des sous-systèmes de $\mathcal{S}$ et de leur groupe de bonne constriction

---

pour chaque $G' \in \mathcal{G}$ *tel que* $G' \subseteq G$, *dans l'ordre d'inclusion croissant* **faire**
  $L \leftarrow$ liste vide
  répéter
    Sélectionner une contrainte $c \in C$
    $L \leftarrow$ liste des GCS G'-bien-contraints de $(C \backslash c, X, A)$
    tant que L *contient uniquement des GCS* G'*-bien-contraints triviaux* **faire**
      Sélectionner une contrainte c qui n'a pas encore été sélectionnée
      $L \leftarrow$ liste des GCS G'-bien-contraints de $(C \backslash c, X, A)$
  jusqu'à *ce que toutes les contraintes aient été testées ou qu'il y ait un GCS* G'*-bien-contraint non trivial*
  si L *contient des GCS* G'*-bien-contraints non triviaux* **alors**
    $\mathcal{A} \leftarrow$ arbre sans fils étiqueté par $(\mathcal{S}, G)$
    pour chaque $\mathcal{S}_i \in L$ **faire**
      Enraciner la $\mathcal{W}$-décomposition de $\mathcal{S}_i$ comme fils de $\mathcal{A}$
      $\mathcal{S} \leftarrow \mathcal{S} - \mathcal{S}_i$
      $\mathcal{S} \leftarrow \mathcal{S} + \mathcal{B}_{\mathcal{S}}(\mathcal{S}_i)$
    Enraciner la $\mathcal{W}$-décomposition de $\mathcal{S}$ comme fils de $\mathcal{A}$
    retourner $\mathcal{A}$
`// On n'a trouvé aucun sous-système non trivial`
retourner *une feuille étiquetée par* $(\mathcal{S}, G)$

Algorithme 8.2: $\mathcal{W}$-décomposition multi-groupe

8.3.2 Exemple

Considérons à nouveau le système de la figure 3.3 (p. 72) et ajoutons une contrainte fixant la direction de la base du triangle. Le système est donc bien contraint *modulo* les translations.

La première phase de l'algorithme (qui n'apparaît pas dans le pseudo-code donné à l'algorithme 8.2) consiste à rechercher les MGS pour les différents groupes considérés, en commençant par le plus petit groupe. La recherche de systèmes bien contraints *modulo* l'identité, les rotations ou les homothéties sera un échec, mais la fixation d'un repère pour les translations permet d'identifier l'ensemble du système. L'algorithme 8.2 est alors exécuté sur ce système.

La boucle principale de l'algorithme va donc commencer avec les translations. Quelle que soit la contrainte sélectionnée, aucun système non trivial ne sera identifié, le seul système bien contraint *modulo* les translations étant la droite dont la direction est fournie comme paramètre.

La recherche de systèmes **D**-bien-contraints maximaux va aboutir à l'identification du système engendré par la contrainte de distance (sauf lorsque celle-ci a été sélectionnée), qui est un système trivial.

La recherche de systèmes bien contraints *modulo* la composition des homothéties et des translations va échouer à trouver des systèmes non triviaux jusqu'à ce que la contrainte sélectionnée soit la contrainte de distance. Le système identifié sera alors celui du milieu de la figure 3.3 (plus la contrainte fixant la direction de la base commune aux deux triangles).

Ce système est $\mathcal{W}$-indécomposable, le seul système maximal que l'on puisse identifier étant le triangle inférieur, qui est trivial. Le remplacement de ce système par son bord aboutit au système engendré par la contrainte de distance, auquel est ajouté la fixation de la direction de la droite passant par les deux points. Ce système est $\mathcal{W}$-indécomposable.

La $\mathcal{W}$-décomposition aboutit donc à la décomposition en deux systèmes : un système bien contraint *modulo* la composition des homothéties et des translations et un système bien-contraint *modulo* les translations.

8.4 Analyse

8.4.1 Complexité

Le temps d'exécution de l'algorithme de $\mathcal{W}$-décomposition dépend en partie de l'ordre dans lequel les contraintes sont sélectionnées : en effet, si l'on commence par parcourir des contraintes telles qu'aucun système non trivial n'est trouvé, on exécute inutilement plusieurs fois l'algorithme d'identification des MGS.

Bien plus encore, le temps d'exécution de la $\mathcal{W}$-décomposition dépend du temps d'exécution de l'algorithme d'identification des MGS. S'il a une complexité en $\mathcal{O}(y)$, alors la complexité de l'algorithme 8.1 est dans le pire des cas (une seule contrainte permet d'identifier des MRS non triviaux et elle est à chaque fois la dernière sélectionnée) de $\mathcal{O}(myl)$, où l est la profondeur de l'arbre et m le nombre de contraintes. L'algorithme 8.2 a une complexité plus élevée puisque les différents groupes doivent être testés. Avec $p = |\mathcal{G}|$, sa complexité dans le pire des cas est donc de $\mathcal{O}(lymp)$.

L'algorithme que nous utilisons, basé sur la méthode du témoin, a une complexité en $\mathcal{O}(m^2n)$, où n désigne classiquement le nombre de colonnes de la matrice – le nombre de degrés de liberté du système – et m le nombre de lignes – le nombre de contraintes. La complexité de l'algorithme 8.1 est donc dans le pire des cas de $\mathcal{O}(m^3nl)$. L'algorithme 8.2 a quant à lui une complexité de $\mathcal{O}(m^3npl)$.

8.4.2 Pouvoir de résolution

La puissance de résolution de notre algorithme dépend énormément de la fiabilité de l'algorithme d'identification des sous-systèmes bien contraints : si cet algorithme échoue à identifier tous les MGS, la $\mathcal{W}$-décomposition ne sera pas complète, de même que si des sous-systèmes sont classés comme bien contraints alors qu'ils ne le sont pas, la $\mathcal{W}$-décomposition ne sera pas correcte.

Cela étant, la puissance de la $\mathcal{W}$-décomposition réside précisément dans le fait qu'elle ne dépend pas d'une seule méthode d'identification des MGS. La $\mathcal{W}$-décomposition peut ainsi facilement être utilisée dans une architecture multi-agent, afin de profiter des puissances d'identification de différentes méthodes et ce quel que soit le paradigme de ces méthodes : à base de règles, combinatoire, symbolique, numérique voire hybride. Il est néanmoins important de garder en tête que la complexité de la $\mathcal{W}$-décomposition est alors en $\mathcal{O}(mlY)$ ($\mathcal{O}(lYmp)$ dans le cas

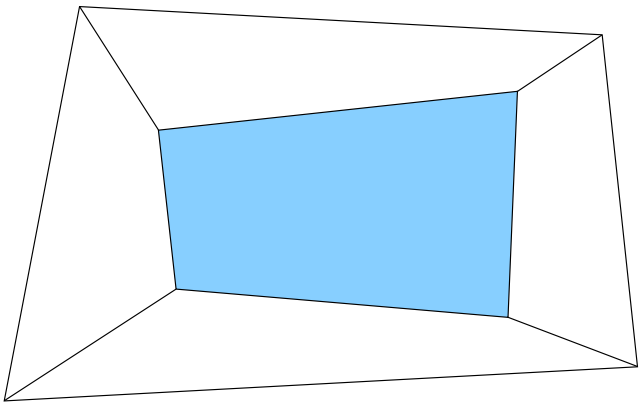


Figure 8.5 - Système $\mathcal{W}$-décomposable 2D (le système coloré en bleu est rigide)

multi-groupe) si la méthode d'identification des MGS la plus coûteuse est en $O(Y)$.

En utilisant, pour l'identification des MGS, notre méthode basée sur le témoin détaillée au chapitre 7, nous avons la certitude de disposer d'un algorithme de décomposition plus puissant que les algorithmes de la littérature, et ce pour plusieurs raisons :

– tout d'abord, il est indépendant de la connexité du graphe de contrainte. Par exemple, la figure 8.5 donne un exemple d'un GCS dont le graphe de contrainte est 4-connexe et qui est $\mathcal{W}$-décomposable, et ce quel que soit le système coloré en bleu, du moment qu'il est rigide. On peut de même générer un exemple de système n-connexe $\mathcal{W}$-décomposable pour n'importe quelle valeur de n ;
– de plus, il n'est pas basé sur une formation de clusters. Si l'on remplace le système coloré en bleu de la figure 8.5 par le système de la figure 8.6, les méthodes actuelles échouent à décomposer, alors que la $\mathcal{W}$-décomposition détecte la rigidité du système en bleu et le remplace par son bord.

Il est facile de voir par exemple que

– tous les GCS décomposables par la méthode d'O [Owe91] sont $\mathcal{W}$-décomposables, puisque les paires d'articulation sont détectées par suppression d'une contrainte ;
– tous les GCS qui sont décomposables par une méthode de formation de cluster ou sur la recherche de parties rigides minimales sont également $\mathcal{W}$-décomposables.

La décomposition ultime consiste à fournir un système d'équations triangulaire. Pour tous les systèmes algébriques, la décomposition de R -W [CG90] ou les bases de G ¨ [Buc85] permettent de telles décompositions mais elles sont inutilisables en pratique dans le domaine de la CAO, de par leur complexité algorithmique.

La $\mathcal{W}$-décomposition utilisant l'identification basée sur le témoin n'est pas aussi puissante que ces méthodes algébriques puisqu'il est possible de construire une

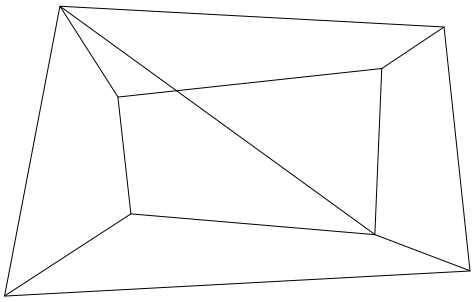


Figure 8.6 - Système $\mathcal{W}$-indécomposable 2D

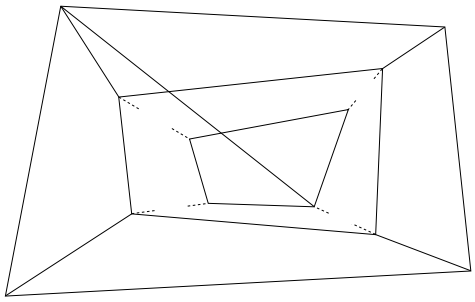


Figure 8.7 - Système $\mathcal{W}$-indécomposable 2D avec un nombre d'entités géométriques arbitraire

famille infinie de systèmes de contraintes $\mathcal{W}$-indécomposables comme celui représenté à la figure 8.7 : il n'y a aucune contrainte dans ce système dont le retrait produit un MGS plus grand qu'un système induit par une distance point-point. Toutefois, ce type de GCS est plutôt rare dans la pratique.

8.5 Correction et complétude

L'algorithme de $\mathcal{W}$-décomposition remplace explicitement un sous-système identifié comme G-bien-contraint par son bord. Le théorème 3 nous permet en outre de nous assurer que les sous-systèmes G-bien-contraints identifiés partagent un G-repère avec le reste du système et que l'on sera donc toujours capable d'assembler les solutions des sous-systèmes ensemble.

La $\mathcal{W}$-décomposition est donc correcte et complète à condition que les solveurs utilisés pour résoudre les systèmes $\mathcal{W}$-indécomposables soient corrects et complets.

B III

Nous avons proposé une extension de la méthode du témoin permettant l'identification des sous-systèmes G-bien-contraints (MGS) d'un système quel que soit son niveau de constriction. Nous avons fourni des pistes pour effectuer cette identification dans les graphes de $\mathcal{R}$-flux sans avoir recours au témoin et montré les limites de ces pistes. Nous avons adapté les algorithmes de paramétrisation de la partie II pour prendre en compte l'information de la G-bonne-constriction de sous-systèmes. Nous avons également déduit de la méthode d'identification des MGS l'algorithme de $\mathcal{W}$-décomposition qui décompose un système G-bien-contraint en ses sous-systèmes stricts G-bien-contraints.

Parmi les perspectives de ces travaux se trouve naturellement la recherche de contournement des limites que nous donnons pour l'identification des sous-systèmes G-bien-contraints sans avoir recours au témoin. Mettre au point une telle méthode permettrait de sensiblement améliorer l'efficacité de nos algorithmes : leur complexité est actuellement dominée par celle de la méthode du témoin, nécessaire pour assurer que l'ajout d'une contrainte n'amène pas une sur-constriction générique. En connaissant le groupe de bonne constriction des différents sous-systèmes, nous pourrions vérifier la non redondance d'une nouvelle contrainte en vérifiant qu'il ne s'agit pas d'une contrainte G-invariante concernant des entités géométriques appartenant à un même système G-bien-contraint.

L'extension des algorithmes de paramétrisation tirant parti de la connaissance du

groupe de bonne constriction de sous-systèmes n'est elle-même pas fiable en cela qu'elle tombe dans le piège de la caractérisation de L et échoue donc à reconnaître la sur-constriction générique dans un système comme celui de la « double-banane ». Un ajout de connaissances géométriques pourrait être envisagé en utilisant un décompte des degrés de liberté en séparant les degrés de liberté en rotation et les degrés de liberté en translation [Kra92]. Pour être intéressante, cette approche doit rester dans la lignée de nos travaux et ne pas favoriser les déplacements dans le raisonnement géométrique. Pour cela, il est nécessaire de mener une réflexion sur les degrés de liberté en homothétie et la caractérisation du nombre de ces degrés.

En outre, les algorithmes de paramétrisation, même améliorés pour tenir compte des informations de G-bonne-constriction de sous-systèmes, ne profitent pas de la puissance de résolution des méthodes de décomposition récentes et échouent par exemple à mener à un plan de construction strict pour des systèmes décomposables par la méthode d'O mais dont le bord des sous-systèmes n'apparaît pas explicitement dans le GCS. Pour corriger cela, il est nécessaire d'ajouter une connaissance géométrique en remplaçant un sous-système dont on connaît le groupe de bonne constriction par son bord. Une telle opération est aisée dans le graphe de $\mathcal{R}$-flux : il suffit de

– retirer les nœuds de contrainte du sous-système ainsi que les nœuds d'entité géométrique n'appartenant pas au bord ;
– amener les nœuds d'entité géométrique du bord à saturation en augmentant si besoin leur degré de repère ;
– utiliser l'algorithme d'ajout de contrainte pour rajouter une à une les contraintes du bord.

Il faut bien entendu faire attention, dans la dernière étape, à ne pas créer de sur-constriction générique lors de l'ajout du bord. Or ceci nécessite pour le moment l'interrogation du témoin. Il faut donc également être capable, en restant dans un processus de calcul incrémental, de mettre à jour la matrice Jacobienne pour tenir compte de la suppression des contraintes du sous-système identifié comme G-bien-contraint. Une telle extension de la méthode du témoin reste à mettre au point.

C

Ce n'est pas parce qu'un problème n'a pas été résolu qu'il est impossible à résoudre
– Agatha Christie, romancière britannique

Dans ce manuscrit, nous nous sommes intéressé à la décomposition et à la paramétrisation de systèmes de contraintes géométriques sous-contraints, avec le point de vue multi-groupe amenant à considérer la bonne constriction relativement à des groupes de transformation. Nous avons rappelé, dans la partie I, comment se formalisaient les systèmes de contraintes géométriques et leurs opérations et avons donné un aperçu des méthodes de résolution existantes en mettant un accent particulier sur les articles traitant de systèmes sous-contraints. Nous avons ensuite proposé, dans la partie II, des algorithmes incrémentaux de paramétrisation combinatoire d'un système de contraintes géométriques et une méthode pour en déduire un plan de construction par blocs. Nous avons enfin abordé, dans la partie III, la décomposition de systèmes de contraintes géométriques et l'identification des sous-systèmes bien contraints *modulo* un groupe connu.

Ces travaux proposent une nouvelle approche des systèmes de contraintes géométriques sous-contraints, généralement considérés comme des systèmes à corriger : nous les considérons comme des systèmes susceptibles de décrire un objet final dont le niveau de constriction correspond aux attentes du concepteur. Nous reprenons l'approche multi-groupe et proposons d'une part de nous ramener à un nombre

fini de solutions en fixant un repère et d'autre part en identifiant les sous-systèmes bien contraints maximaux.

1 Contributions

Nos travaux ont mené à la mise au point d'algorithmes incrémentaux, de complexité quadratique, permettant de calculer une paramétrisation combinatoire d'un GCS, sous la forme du nombre de degrés de liberté à fixer pour chaque entité géométrique. Ces algorithmes sont fondés sur les graphes de $\mathcal{R}$-flux, qui sont des graphes bipartis entités-contraintes dans lesquels nous ajoutons, pour chaque nœud correspondant à une entité géométrique, un nœud de repère dont la saturation n'est pas impérative. Ceci permet de simuler la fixation d'un nombre quelconque de degrés de liberté. Nous adaptons des algorithmes de calcul d'un couplage parfait aux graphes de $\mathcal{R}$-flux pour trouver une configuration du flux saturant tous les nœuds de contrainte et d'entité. Les degrés de repère des nœuds d'entités nous fournissent une paramétrisation combinatoire du GCS. Nous proposons un algorithme de modification de la configuration du graphe de $\mathcal{R}$-flux permettant une modification à la volée de la paramétrisation. Nous montrons également comment prendre en compte la connaissance du groupe de bonne constriction de sous-systèmes dans le calcul de la paramétrisation.

Nos algorithmes de paramétrisation sont sensibles à la sur-constriction générique. Nous proposons donc un algorithme incrémental de détection de contraintes redondantes, utilisant l'analyse de la matrice Jacobienne du système de contraintes géométriques évaluée en un témoin, ayant la même complexité que l'interrogation classique du témoin. Nous en déduisons une méthode de calcul d'une base du bord d'un système par rapport à un autre système, permettant le remplacement d'un sous-système par son bord en assurant que le résultat n'est pas génériquement sur-contraint. Nous montrons que ce remplacement permet d'assurer la correction et la complétude des méthodes de décomposition.

Nous proposons un algorithme d'identification des sous-systèmes rigides maximaux, utilisant lui aussi une analyse de la matrice Jacobienne évaluée en un témoin. Nous l'étendons en un algorithme d'identification des sous-systèmes G-bien-contraints maximaux pour tous les groupes G dont on connaît les types de repère. Nous proposons une version incrémentale de cet algorithme. Nous proposons des pistes pour identifier les sous-systèmes G-bien-contraints dans le graphe de $\mathcal{R}$-flux sans faire appel au témoin et montrons les limites de cette approche. Nous proposons une méthode de décomposition générale d'un GCS G-bien-contraint en ses sous-systèmes stricts G-bien-contraints maximaux, appelée $\mathcal{W}$-décomposition, fondée sur l'identification des MGS après retrait d'une contrainte.

2 Bilan

Les algorithmes de paramétrisation fournissent une liste du nombre de degrés de liberté à retirer à chaque entité géométrique pour revenir à un nombre fini de solutions. L'interprétation géométrique de cette paramétrisation combinatoire permet de donner un retour visuel à l'utilisateur concernant la flexibilité de l'objet décrit par le système de contraintes géométriques : en sachant quels points doivent être fixés dans l'espace, quelles directions doivent être explicitées, l'utilisateur sait aussi quels éléments géométriques sont mobiles.

L'algorithme d'identification des sous-systèmes G-bien-contraints maximaux permet d'indiquer à l'utilisateur quelles sont les parties de l'objet qui sont solidaires les unes des autres : quels sous-systèmes sont rigides, quels sous-systèmes peuvent être mis à l'échelle, *etc.*.

La méthode de vérification incrémentale de la redondance d'une contrainte nous permet d'assurer que l'objet conçu n'est pas génériquement sur-contraint et d'avertir l'utilisateur lorsqu'il ajoute une contrainte susceptible d'empêcher l'existence de solutions dans l'espace euclidien. Ces contraintes redondantes peuvent être conservées pour servir durant le parcours de l'espace des solutions, en sélectionnant uniquement les figures qui respectent ces contraintes.

Si le plan de construction (que l'on déduit de la paramétrisation) contient des sous-systèmes à plusieurs entités géométriques qui doivent être construits atomiquement, l'algorithme de $\mathcal{W}$-décomposition nous permet de décomposer ces sous-systèmes jusqu'à obtenir des systèmes suffisamment simples pour être construits directement ou pour être résolus numériquement avec une erreur faible. Les propriétés de la $\mathcal{W}$-décomposition, associées à la démonstration des conditions de correction et de complétude des méthodes de $\mathcal{W}$-décomposition, nous autorisent à assurer que nous trouvons toutes les solutions et uniquement des solutions, à condition que les méthodes de résolution des systèmes $\mathcal{W}$-indécomposables soient elles aussi correctes et complètes.

L'incrémentalité de tous nos algorithmes, qui ne s'accompagne d'aucun sur-coût de complexité ni en temps ni en espace, favorise une modélisation par essai/erreur : l'utilisateur réalise une esquisse initiale et obtient un retour visuel sur le niveau de constriction du système de contraintes géométriques. Il peut ensuite, si le résultat ne lui convient pas, modifier l'esquisse, la paramétrisation se mettant à jour automatiquement.

Dans cette démarche, tous les systèmes de contraintes géométriques sont manipulés de manière homogène, qu'ils soient rigides, bien-contraints *modulo* un autre groupe

que les déplacements ou sous-contraints *modulo* les groupes de transformation obtenus par composition des rotations, des translations et des homothéties.

3 Interfaces intuitives de modélisation par contraintes

Nos travaux permettent d'envisager la mise au point de logiciels de modélisation par contraintes accessibles à des utilisateurs non experts, à partir des éléments visuels sur le niveau de constriction, de la décomposition en sous-systèmes G-bien-contraints maximaux et de la donnée d'un plan de construction paramétré.

Sur l'illustration de la figure 8.8, nous avons représenté ce que pourrait être le retour visuel offert par notre paramétrisation dans un modeleur géométrique à base de contraintes[7]. Nous imaginons ici que l'utilisateur a spécifié les caractéristiques techniques d'une voiture au moyen de contraintes géométriques puis a fourni un plongement courbe précis aux différentes parties du véhicule.

Le logiciel de CAO fournit un dessin à l'échelle respectant les contraintes imposées par l'utilisateur. Il ajoute, sur cette figure, une représentation graphique (en rouge sur la figure 8.8) de la paramétrisation calculée. Ici, il indique qu'il faut fixer :
– la position dans l'espace du pare-chocs (**a**) ;
– une direction sur chaque roue (**b** et **c**) ;
– une direction sur la portière (**d**) ;
– une direction sur la poignée (**e**) ;
– une direction sur le cache du réservoir (**f**) ;
– une direction sur le capot (**g**) ;
– la hauteur de la vitre (**h**).
Avec ces indications, l'utilisateur a une première intuition des parties du véhicule qui sont en mouvement. Pour compléter cette intuition, nous lui indiquons quels sont les éléments dont la position dépend d'un élément de repère. L'identification des sous-systèmes rigides maximaux nous permet en outre de lui indiquer quelles sont les différentes parties rigides. La figure 8.9 montre par exemple ce que pourrait indiquer le modeleur lorsque l'utilisateur sélectionne la direction posée sur le capot. L'utilisateur voit ainsi que le capot est un élément rigide et que changer la valeur donnée à la direction déplace l'intégralité du capot.

La donnée d'un plan de construction paramétré permet de calculer les solutions à partir d'une valuation des paramètres. Elle peut donc également permettre de faire

[7] Nos algorithmes ne suffiraient en réalité pas à obtenir cette paramétrisation, dans la mesure où un grand nombre des contraintes imposées dans la conception d'un véhicule sont des contraintes de non intersection de volumes, que nous ne prenons pas en compte car elles relèvent plus de la simulation mécanique que des constructions géométriques.

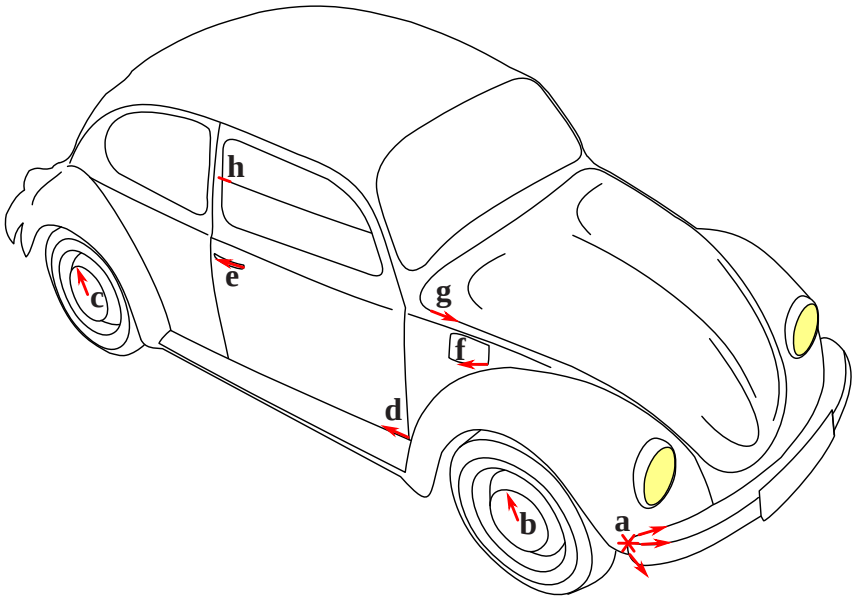


Figure 8.8 - Exemple de retour visuel possible avec notre paramétrisation

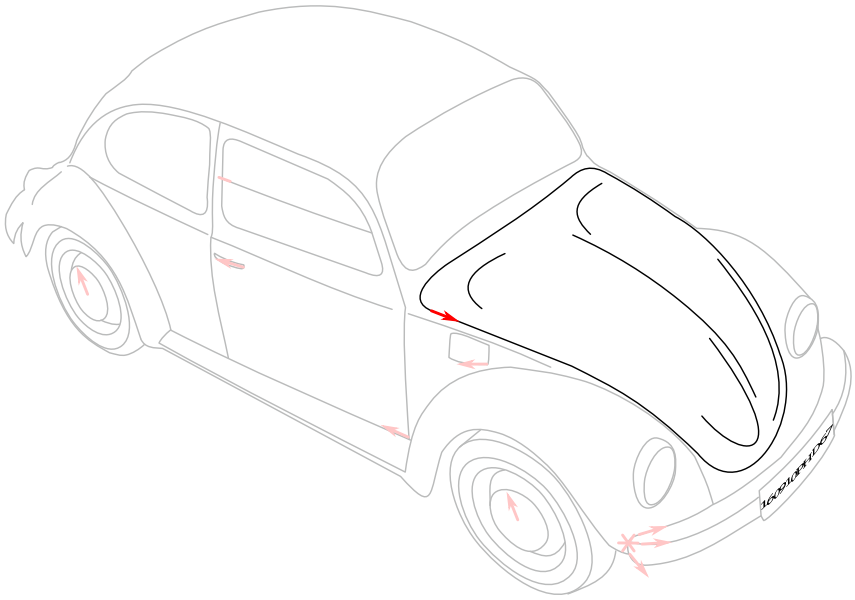


Figure 8.9 - Exemple de retour visuel possible avec l'identification des MRS

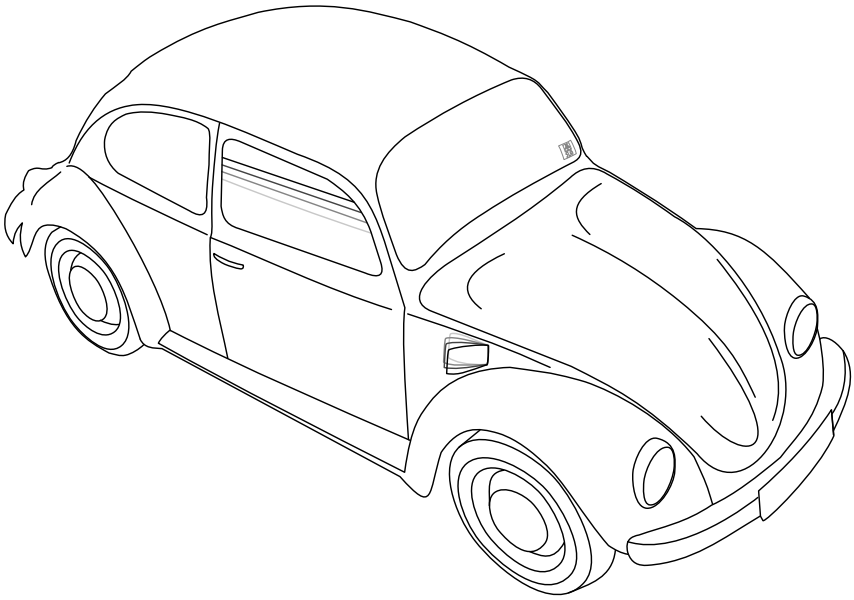


Figure 8.10 - Exemple de retour visuel possible avec un plan de construction paramétré

varier les valeurs des paramètres et de mettre à jour interactivement la/les solution(s). L'utilisateur peut alors vérifier l'intuition que lui offrent la paramétrisation et l'identification des sous-systèmes bien contraints en sélectionnant un élément de repère et en en modifiant la position. Cette approche est désormais classique dans les logiciels de géométrie dynamique tels que GeoGebra [Geo], Cabri Géomètre [Cab] ou Cinderella [Cin].

Il est même possible d'envisager de faire varier automatiquement les valeurs des paramètres pour générer une animation qui permet à l'utilisateur de visualiser un sous-espace des solutions et ainsi d'avoir une excellente intuition des libertés de mouvement des différentes parties de l'objet décrit par le système de contraintes géométriques. La figure 8.10 donne un exemple d'une telle animation, où le modeleur fait varier d'une part la direction positionnant le cache du réservoir par rapport au reste du système et d'autre part la hauteur de la vitre.

Un tel modeleur permettrait à l'utilisateur un parcours intuitif de l'espace des solutions en lui proposant plusieurs modalités de parcours :
– en faisant varier les valeurs des paramètres, l'utilisateur visualise les mouvements autorisés ;
– en ajoutant une contrainte, l'utilisateur sélectionne un sous-ensemble des solutions qui satisfait une condition particulière.

4 Perspectives

En plus des perspectives que nous avons évoquées en conclusion des trois parties[8], un certain nombre de travaux complémentaires sont à effectuer pour permettre les horizons applicatifs que nous venons d'évoquer.

Ainsi, nous avons évoqué que des mécanismes étaient connus, dans le domaine de la géométrie dynamique, pour réinterpréter interactivement un plan de construction lorsque l'utilisateur modifie les valeurs des paramètres. Toutefois, contrairement aux logiciels de géométrie dynamique, nous nous situons dans un cadre applicatif où l'utilisateur n'a pas explicitement donné un plan de construction, mais où celui-ci est calculé par le modeleur. En fonction de la paramétrisation calculée, les entités géométriques dont la position dépend d'un élément de repère donné ne sont pas les mêmes. Il est donc possible, lorsque l'utilisateur demande à modifier la valeur d'un élément de repère, que la liste des entités géométriques dont la position est recalculée ne corresponde pas à ce qu'il veut. Sur l'exemple donné plus haut de la paramétrisation d'une voiture, si l'utilisateur translate vers le bas le point fixant le pare-chocs (**a** sur la figure 8.8), faut-il opérer une translation sur l'ensemble de la voiture ou doit-on chercher, par exemple, à conserver inchangée la position du haut de la vitre ?

Lorsqu'un élément de repère change de valuation sur demande de l'utilisateur, on peut avoir à recalculer la position d'une entité géométrique qui dépend également d'autres éléments de repère. La question peut alors se poser de l'opportunité de modifier aussi les valeurs de ces éléments de repère. Sur la figure 8.8, si l'utilisateur modifie la direction de la portière (**d**), l'ensemble de la portière va subir une rotation centrée sur l'axe du mécanisme reliant la portière au reste de la voiture. Faut-il alors opérer cette même rotation sur la direction de la poignée (**e**) ? Si non, que faire lorsque le changement de la valeur d'un élément de repère empêche l'existence de solutions sans faire varier d'autres éléments de repère ?

Ces problèmes sont fortement liés à la précocité des différents éléments de repère[9] dans le DAG déduit de la paramétrisation. Les questions se posent donc d'une définition liant des entités géométriques à un élément de repère indépendamment de la précocité de ces éléments de repère ou de la mise au point d'algorithmes permettant de modifier le $\mathcal{R}$-flot sans modifier la paramétrisation afin de minimiser la précocité d'un élément de repère donné.

Une autre question liée à la liste des entités géométriques considérées comme dépendantes d'un élément de repère donné est celle des modalités d'associer visuel-

[8] *Cf.* pp. 83, 149 et 191
[9] *Cf.* définition 53 p. 124

lement l'élément de repère à cette liste. La figure 8.9 illustre l'association d'un élément de repère à un sous-système bien contraint maximal, mais dans le cas de chaînes articulées avec plusieurs éléments de repère, cette association est moins évidente. Elle pourrait passer, par exemple, par l'affichage d'un squelette du système.

Nos travaux ont montré que considérer de manière homogène les systèmes de contraintes géométriques sous-contraints et bien-contraints, avec le point de vue de l'invariance par groupe de transformations, ouvrait la porte à des logiciels de modélisation par contraintes plus intuitifs : en complément d'une approche de résolution incrémentale, notre démarche permet en effet une conception par essai/erreur grâce à des retours visuels donnant à l'utilisateur une intuition du niveau de constriction de l'objet qu'il a décrit.

Q `

A

A

Un problème créé ne peut être résolu en réfléchissant de la même manière qu'il a été créé
– Albert Einstein, physicien

Cette annexe décrit le contexte dans lequel les algorithmes décrits dans ce mémoire ont été implémentés. Nous commençons par décrire le langage de description de systèmes de contraintes géométriques développé dans notre équipe préalablement à nos travaux (section A.1) puis décrivons brièvement la structure de la plate-forme de résolution de GCS développée par Pascal M (section A.2). Enfin, nous précisons quelles ont été les contributions à ces éléments faites durant nos travaux (section A.3).

A.1 GCML

Le Geometric Constraint Markup Language (GCML) est un méta-langage basé sur XML, mis au point dans le cadre des travaux de W *et al.* [Win05, WSMF06]. Basé sur la même approche des GCS que la formalisation évoquée aux chapitres 1 et 3, ce langage permet de décrire l'univers géométrique considéré et de donner un énoncé syntaxique.

Le GCML permet de définir la syntaxe du cadre de résolution en donnant
– un ensemble de sortes ;
– un ensemble de symboles fonctionnels ;
– un ensemble de symboles prédicatifs.
Les symboles fonctionnels sont caractérisés par leur arité et leur co-arité. Les symboles prédicatifs sont caractérisés par leur arité.

Ainsi, par exemple, le code suivant décrit la syntaxe autorisée pour un univers consi-

dérant des points, des droites et des cercles et permettant l'intersection de deux cercles ou d'une droite et d'un cercle, ainsi que des contraintes de distance entre points, d'angle entre droites et d'incidence à un cercle et à une droite.

```
<syntax>
  <sorts>
    point line circle length angle
  </sorts>

  <fsymbols>
    mkpoint : -> point
    mkline : -> line
    mkcircle : point length -> circle
    intercl : circle line -> point
    intercc : circle circle -> point
  </fsymbols>

  <psymbols>
    on_pc : point circle
    on_pl : point line
    dist_pp : point point length
    angle_ll : line line angle
  </psymbols>
</syntax>
```

Nous ne détaillons pas ici la syntaxe correspondante, mais le GCML permet également de décrire un ensemble d'axiomes et de règles de construction.

Toujours dans la description de l'univers géométrique, le GCML permet d'expliciter les sémantiques associées aux GCS. Ainsi, le code suivant décrit partiellement trois sémantiques associées à la syntaxe précédente.

```
<semantics>
  <semantic type="aggregate" language="C++">
    <point>
      float x, y, z;
    </point>
  </semantic>

  <semantic type="algebraic" language="Maple">
    <distpp>
```

```
      solve({x1-x2)^2 + (y1-y2)^2 = k^2});
    </distpp>
  </semantic>

  <semantic type="visualization" language="C++/OpenGL">
    <point>
      glPushMatrix();
      glTranslatef(x,y,z);
      glutSolidSphere(0.1,32,32);
      glPopMatrix();
    </point>
  </semantic>
</semantics>
```

Enfin, le GCML permet de décrire, une fois l'univers géométrique donné, un système de contraintes géométriques. Voici, par exemple, le code décrivant le GCS de la figure 1.1, à partir de l'univers géométrique décrit ci-dessus :

```
<gcs>
  <unknowns>
    point p1 p2 p3
    line l1 l2 l3
  </unknowns>

  <parameters>
    length k1 k2
    angle a
  </parameters>

  <constraints>
    on_pl(p1, l1)
    on_pl(p2, l1)
    on_pl(p2, l2)
    on_pl(p3, l2)
    on_pl(p3, l3)
    on_pl(p1, l3)
    dist_pp(p1, p2, k1)
    dist_pp(p2, p3, k2)
    angle_ll(l1, l3, a)
  </constraints>
</gcs>
```

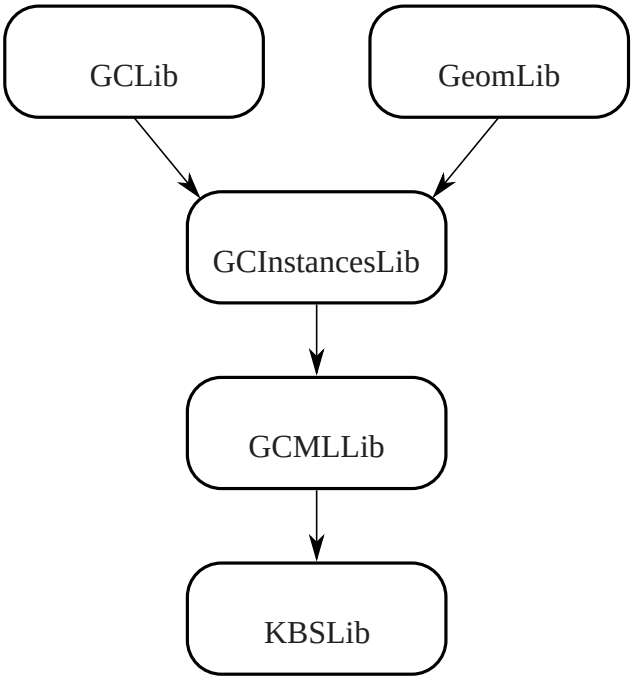


Figure A.1 - Structure de la plate-forme de modélisation pré-existante

Pour plus de détails sur le GCML, qu'il s'agisse de sa syntaxe ou de ses avantages applicatifs, le lecteur pourra se référer à [Win05].

A.2 Plate-forme de modélisation par contraintes

Pour ce qui concerne l'implémentation, nos travaux se sont faits dans le cadre d'une plate-forme de modélisation par contraintes développée en C++ préalablement à notre projet de recherche, principalement par Pascal M et Arnaud F . Cette plate-forme contient d'une part un ensemble de librairies spécifiques à la résolution de systèmes de contraintes géométriques, d'autre part un ensemble de solveurs.

Les librairies constituant la base programmatoire de la plate-forme sont :
– la GCLib ;
– la GeomLib ;
– la GCInstancesLib, qui dépend des deux librairies ci-dessus ;
– la GCMLLib, qui dépend de la GCInstancesLib ;
– et la KBSLib, qui dépend de la GCMLLib.
L'organisation des librairies est synthétisée à la figure A.1. La GCLib et la GeomLib sont les deux librairies de niveau 1, qui ne dépendent d'aucune librairie interne à la plate-forme.

La GCLib (Geometric Constraints Library) est la librairie définissant l'ensemble des classes correspondant aux notions d'univers géométrique : symbole, sorte, termes prédicatifs et fonctionnels (dont les variables), signature, valuation, groupe de transformation, plan de construction et enfin GCS.

La GeomLib (Geometry Library) définit les éléments géométriques usuels du plan et de l'espace : points, droites, cerles, plans, sphères, coniques, vecteur et mesures (représentant les métriques associées aux contraintes).

La GCInstancesLib (Geometric Constraints Instances Library) utilise les deux librairies précédentes pour créer des objets géométriques de sorte point, droite, *etc.* et instancier des groupes de transformation classiques (rotations, translations, homothéties, isométries directes, similitudes).

La GCMLLib (GCML Library) définit des outils d'analyse syntaxique de code GCML et de compilation en code C++ d'objets de classes définies dans la GCInstancesLib.

La KBSLib (Knowledge-Based Solver Library) – que nous n'avons pas utilisée durant nos travaux – fournit comme son nom l'indique un ensemble d'outils utilisés par les solveurs basés sur des systèmes experts.

Les librairies de la plate-forme ne sont dépendantes d'aucune librairie externe.

À côté des librairies, la plate-forme contient un certain nombre de solveurs. Au début de notre projet, les solveurs disponibles étaient un solveur utilisant la méthode de N -R , un solveur à base de règles utilisant les algorithmes mis au point durant le projet de recherche doctoral de Pascal M [Mat97] et un solveur par reparamétrisation conçu durant le projet de recherche doctoral d'Arnaud F [Fab06].

A.3 Contributions

Les contributions à la plate-forme développées durant le projet de recherche ont été les suivantes :

– la FlowLib, de niveau zéro, qui permet de créer et manipuler des graphes de $\mathcal{R}$-flux ;
– la GPatLib (Geometric Patterns Library), développée par une stagiaire de licence de mathématiques, étendant la GCMLLib afin de proposer un langage de description de motifs dans des systèmes de contraintes géométriques ;
– des extensions du GCML pour la GPatLib (mises au point au travers du même

stage)
- une sémantique complémentaire incluse dans la GCInstancesLib, permettant de calculer la Jacobienne évaluée en un témoin d'un GCS et de l'exprimer à l'aide de la librairie GiNaC [Gin], mise au point par Jean-David G´ à travers un stage de licence d'informatique puis lors d'un Travail d'Étude et de Recherche (TER) en première année de master d'informatique ;
- un solveur utilisant la sémantique ci-dessus pour identifier les parties rigides maximales d'un GCS, également mis au point par Jean-David G´ durant son TER.

A ´

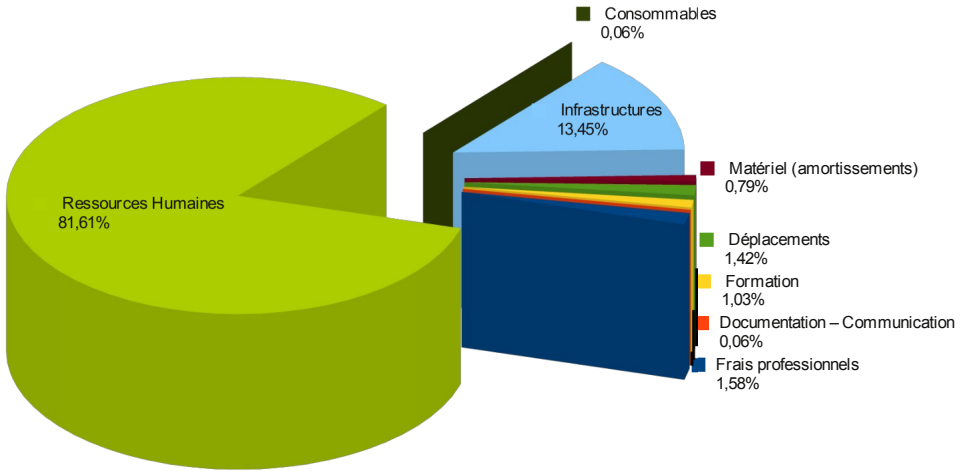
Si tu te résous toi-même, le problème du monde est résolu
– Henry de Montherlant, écrivain

Notre projet de recherche a été mené durant quatre années, au sein du Laboratoire des Sciences de l'Informatique, de l'Image et de la Télédétection (LSIIT, UMR CNRS - UdS 7005). Ses acteurs ont principalement été l'auteur du présent manuscrit et ses deux encadrants, Pascal M et Pascal S .

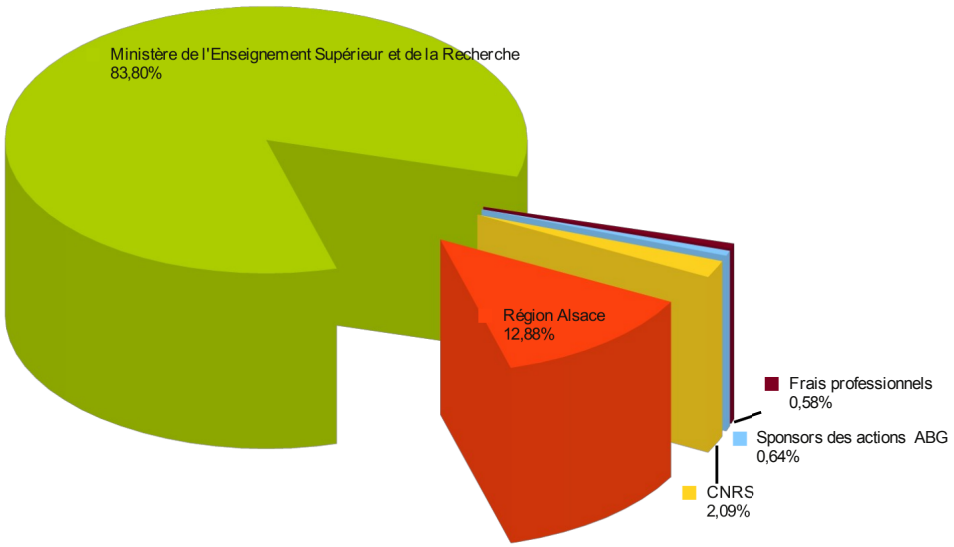
Le coût consolidé du projet est ainsi approximativement de 244 000 €, incluant
– les ressources humaines (env. 200 000 €) ;
– les infrastructures (env. 30 000 €) ;
– les déplacements (env. 3 500 €) ;
– les formations (env. 2 500 €) ;
– l'équipement scientifique (env. 2 000 €) ;
– les frais professionnels (env. 3 000 €).
Les bailleurs de fonds de ce projet sont principalement publics, puisque les recettes proviennent
– du Ministère de l'Enseignement Supérieur et de la Recherche (env. 204 000 €) ;
– de la Région Alsace (env. 32 000 €) ;
– du Centre National pour la Recherche Scientifique (env. 5 000 €).
Les recettes restantes se partagent entre quelques sponsors industriels des activités de formations de l'Université de Strasbourg (env. 1 500 €) et entre les frais professionnels que constituent les frais d'inscription universitaire annuels (env. 1 400 € au total).

Les deux graphes suivants illustrent la répartition des recettes et des dépenses.

Dépenses



Recettes



T

CAO Conception Assistée par Ordinateur

CFC Composante Fortement Connexe

DAG Graphe Orienté Acyclique – *angl. :* Directed Acyclic Graph

DOR Degré De Rigidité – *angl. :* Degree Of Rigidity

GCML Geometric Constraint Markup Language

GCS Système de Contraintes Géométriques – *angl. :* Geometric Constraint System

MRS Sous-système Rigide Maximal – *angl. :* Maximal Rigid Subsystem

MGS Sous-système *G*-bien-contraint Maximal – *angl. :* Maximal *G*-well-constrained Subsystem

T

A E

T ´

T

T

T ´ `

I

B

[AAHMS99] S. A -A , B. H , A. M et T. S : Solving geometric constraints by a graph-constructive approach. *In IV '99: Proceedings of the 3rd internation conference on Information Visualisation*, pages 250–255, London, England, Royaume Uni, 1999. IEEE Computer Society Press. – Cité en pages 42 et 45.

[ADG00] J. A. A , M. D ´ et L. G : A simple method for the synthesis of 2D and 3D mechanisms with kinematic constraints. *Mechanism and Machine Theory*, 35(5):645–674, 2000. – Cité en page 58.

[AG93] E. L. A et K. G : Continuation and path following. *Acta Numerica*, 2:1–64, 1993. – Cité en page 41.

[AKC96] R. A , G. A. K et R. H. C : Assembly modelling by geometric constraint satisfaction. *Computer-Aided Design*, 28(9):707–722, 1996. – Cité en pages 4 et 58.

[Ald88] B. A : Variations of geometries based on a geometric-reasoning method. *Computer-Aided Design*, 20(3):117–126, 1988. – Cité en page 43.

[AMRV91] B. A , H. M , H. R et K. V : Rule-based variational geometry in Computer-Aided Design. *In* D. T. P , éditeur : *Artificial Intelligence in Design*, pages 27–46. Springer-Verlag, 1991. – Cité en page 43.

[BCK88] B. B , G. E. C et B. K : Algebraic methods for geometric reasoning. *Annual Review of Computer Science*, 3:85–120, 1988. – Cité en page 41.

[BFH+95] W. B , I. F , C. M. H , J. C et R. P : A geometric constraint solver. *Computer-Aided Design*, 27(6):487–501, 1995. – Cité en pages 40 et 60.

[BKL+91] M. B , H.-J. K , P. L , F. O et D. S - : *Algebraic system specification and development: a survey and annotated bibliography*, volume 501 de *Lecture Notes in Computer Science*. Springer-Verlag, 1991. – Cité en page 14.

[Bor81] A. H. B : The programming language aspects of Thinglab, a constraint oriented simulation laboratory. *ACM Transactions on Programming Languages and Systems*, 3(4):353–387, 1981. – Cité en page 42.

[Brü87] B. B ¨ : Constructing three-dimensional geometric objects defined by constraints. *In Proceedings of the 1986 Workshop on Interactive 3D Graphics*, pages 111–129, Chapel Hill, North Carolina, USA, 1987. ACM. – Cité en page 43.

[Brü93] B. B ¨ : Using geometric rewrite rules for solving geometric problems symbolically. *Theoretical Computer Science*, 116(2):291–303, 1993. – Cité en page 43.

[BS03] B. B et J. S : Solution selectors: A user oriented answer to the multiple solution problem in constraint solving. *Journal of mechanical design*, 125(3):443–451, 2003. – Cité en page 129.

[Buc85] B. B : Gröbner bases: an algorithmic method in polynomial ideal theory. *In* N. K B , éditeur : *Multidimensional Systems Theory – Progress, Directions and Open Problems in Multidimensional Systems*, chapitre 6, pages 184–232. Reidel Publishing Company, 1985. – Cité en pages 41 et 189.

[Cab] Cabri géomètre. http://www.cabri.com/fr/. – Cité en pages 54 et 198.

[Car89] J.-C. C : *Théorie des corps – la règle et le compas*. Hermann, Paris, France, 1989. – Cité en page 30.

[Cau13] A. L. C : Sur les polygones et polyèdres, Second mémoire. *Journal de l'École Polytechnique*, 19:87–98, 1813. – Cité en page 48.

[CC02] C. A. C C : Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 19(11-12):1245–1287, 2002. – Cité en page 42.

[CG90] S.-C. C et X.-S. G : Ritt-wu's decomposition algorithm and geometry theorem proving. *In CADE '90: Proceedings of the 10th International Conference on Automated Deduction*, volume 449 de *Lecture Notes in Computer Science*, pages 207–220, Kaiserslautern, Allemagne, 1990. Springer. – Cité en pages 41 et 189.

[Cho88] S.-C. C : *Mechanical geometry theorem proving*. Mathematics and its Applications. D. Reidel, Springer, Dordrecht, Allemagne, 1988. – Cité en page 41.

[Chy85] G. W. C : Constraint management for constructive geometry. Mémoire de master, Mechanical Engineering Department, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 1985. – Cité en page 94.

[Cin] Cinderella. http://www.cinderella.de/. – Cité en page 198.

[CLC06] C. H. C , W. H. L et B. C : The geometric constraint solving based on hybrid genetic algorithm of conjugate gradient. *In* G. R. L , V. B. C. T et X. H , éditeurs : *Computational Methods*, chapitre 17, pages 1117–1121. Springer, 2006. – Cité en page 42.

[Con79] R. C : The rigidity of polyhedral surfaces. *Mathematics Magazine*, 52(5):275–283, 1979. – Cité en page 49.

[Con05] R. C : Generic global rigidity. *Discrete & Computational Geometry*, 33(4):549 – 563, 2005. – Cité en page 48.

[Cra79] H. C : Structural rigidity. *Structural topology*, 1:26–45, 1979. – Cité en page 48.

[DH00] C. D et C. M. H : A systematic framework for solving geometric constraints analytically. *Journal of Symbolic Computation*, 30(5):493–519, 2000. – Cité en page 41.

[Die05] R. D : *Graph Theory*. Springer-Verlag, New-York, New-York, USA, 2005. – Cité en page 112.

[DMS97] J.-F. D , P. M et P. S : Formal resolution of geometrical constraint systems by assembling. *In SMA '97: Pro-*

ceedings of the 4th ACM Symposium on Solid Modeling and Applications, pages 271–284, Atlanta, Georgia, USA, 1997. ACM. – Cité en page 45.

[DMS98] J.-F. D , P. M et P. S : Geometric construction by assembling solved subfigures. *Artificial Intelligence*, 99(1): 73–119, 1998. – Cité en page 45.

[Doh95] M. D : A survey of constraint satisfaction techniques for geometric modeling. *Computer & Graphics*, 19(6):831–845, 1995. – Cité en page 40.

[dRvdMB08] R. de R , H. A. van der M et W. F. B : A workbench for geometric constraint solving. *Computer-Aided Design and Applications*, 5(1–4):471–482, 2008. – Cité en page 3.

[Dur98] C. D : *Symbolic and numerical techniques for constraint solving*. Thèse de doctorat, Purdue University, West Lafayette, Indiana, USA, 1998. – Cité en page 41.

[EM85] H. E et B. M : *Fundamentals of algebraic specification*. Springer-Verlag, Secaucus, New-Jersey, USA, 1985. – Cité en page 14.

[EV01] C. E -V : *Sélection dans l'espace des solutions engendrées par un plan de construction géométrique*. Thèse de doctorat, Université Louis Pasteur, Strasbourg, France, 2001. – Cité en page 129.

[EVSD00a] C. E -V , P. S et J.-F. D : Selecting a figure using S-homotopy in a CAD system. *In 3IA '00: 4th International Conference on Computer Graphics and Articial Intelligence*, pages 153–158, Limoges, France, 2000. – Cité en page 129.

[EVSD00b] C. E -V , P. S et J.-F. D : Sketch-based pruning of a solution space within a formal geometric constraint solver. *Artificial Intelligence*, 124(1):139–159, 2000. – Cité en pages 4, 59 et 129.

[EVSMD03] C. E -V , P. S , P. M et J.-F. D : Combination of automatic and interactive tools for solution space browsing. *In GMAG '03: Proceedings of the International Conference on Geometric Modelling and Graphics*, pages 14–21, London, England, Royaume Uni, 2003. IEEE Computer Society Press. – Cité en pages 59 et 129.

[Fab06] A. F : *Contraintes géométriques en dimension 3*. Thèse de doctorat, Université Louis Pasteur, Strasbourg, France, 2006. – Cité en page 207.

[FBMB90] B. N. F -B , J. M et A. B : An incremental constraint solver. *Communications of the ACM*, 33(1):54–63, 1990. – Cité en page 42.

[Fek06] Z. F : Source location with rigidity and tree packing requirements. *Operations Research Letters*, 34(6):607–612, 2006. – Cité en page 49.

[FH96] I. F et C. M. H : Correctness proof of a geometric constraint solver. *International Journal of Computational Geometry and Applications*, 6(4):405–420, 1996. – Cité en page 58.

[FH97] I. F et C. M. H : A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics*, 16(2):179–216, 1997. – Cité en pages 43, 45 et 58.

[FJF56] L. R. F J . et D. R. F : Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 1956. – Cité en page 97.

[FMF09] C. F , D. M et S. F : Nonlinear systems solver in floating point arithmetic using LP reduction. *In SPM '09: Proceedings of the SIAM/ACM joint conference on Geometric and Physical Modeling*, pages 123–134, San Francisco, California, USA, 2009. – Cité en page 54.

[FMS04] A. F , P. M et P. S : 3D geometric constructions in virtual reality. *In IEEE-VRIC '04: 6th Virtual Reality International Conference*, pages 173–180, Laval, France, 2004. – Cité en page 3.

[FS06] A. F et P. S : Solving 3D quasi-decomposable geometric constraint systems. *In ADG '06: Proceedings of the 6th International Workshop on Automated Deduction in Geometry*, volume 4869 de *Lecture Notes in Artifical Intelligence*, pages 31–36, Pontavedra, Espagne, 2006. Springer. – Cité en pages 44 et 59.

[FS07] A. F et P. S : A formal-numerical approach to solve 3d geometric constraint systems. *In GMAI '07: Proceedings of the 2nd International Conference on Geometric Modelling and*

Imaging, pages 54–59, Zürich, Suisse, 2007. IEEE. – Cité en pages 44 et 59.

[FS08] A. F et P. S : Combining symbolic and numerical solvers to simplify indecomposable systems solving. *In SAC '08: Proceedings of the 23rd ACM Symposium on Applied Computing*, pages 1838–1842, Fortaleza, Brésil, 2008. ACM. – Cité en pages 44 et 59.

[FSSB05] A. F , L. S , P. S et D. B : Constrained gesture interaction in 3D geometric constructions. *In GW '05: 6th International Gesture Workshop*, volume 3881 de *Lecture Notes in Computer Science*, pages 324–334, Berder Island, France, 2005. Springer. – Cité en page 3.

[GC98a] X.-S. G et S.-C. C : Solving geometric constraint systems I. A global propagation approach. *Computer-Aided Design*, 30(1): 47–54, 1998. – Cité en page 41.

[GC98b] X.-S. G et S.-C. C : Solving geometric constraint systems II. A symbolic approach and decision of Rc-constructibility. *Computer-Aided Design*, 30(2):115–122, 1998. – Cité en pages 41, 55 et 56.

[GCG99] J.-X. G , S.-C. C et X.-S. G : Geometric constraint satisfaction using optimization methods. *Computer-Aided Design*, 31(14):867–879, 1999. – Cité en page 42.

[Geo] Geogebra. http://www.geogebra.org. – Cité en page 198.

[GHY02] X.-S. G , C. M. H et W.-Q. Y : Solving spatial basic geometric constraint configurations with locus intersection. *In SMA '02: Proceedings of the 7th ACM symposium on Solid Modeling and Applications*, pages 95–104, Saarbrücken, Allemagne, 2002. ACM. – Cité en pages 44 et 59.

[GHY04] X.-S. G , C. M. H et W.-Q. Y : Solving spatial basic geometric constraint configurations with locus intersection. *Computer-Aided Design*, 36(2):111–122, 2004. – Cité en pages 44 et 59.

[Gin] GiNaC is Not a CAS. http://www.ginac.de. – Cité en page 208.

[Glu74] H. G : Almost all simply connected closed surfaces are rigid. *In Proceedings of the Geometric Topology Conference*, volume

438 de *Lecture Notes in Mathematics*, pages 225–239, Park City, Utah, USA, 1974. Springer. – Cité en page 49.

[Gra02] J. G : *Counting on frameworks: mathematics to aid the design of rigid structures*. Numéro 25 de Dolciani Mathematical Expositions. The Mathematical Association of America, 2002. – Cité en pages 33 et 49.

[GSS93] J. E. G , B. S et H. S : *Combinatorial rigidity*, volume 2 de *Graduate Studies in Mathematics*. American Mathematical Society, 1993. – Cité en pages 33 et 49.

[Hav91] T. F. H : Some examples of the use of distances as coordinates for Euclidean geometry. *Journal of Symbolic Computation*, 11(5–6):579–593, 1991. – Cité en page 55.

[Hen92] B. H : Conditions for unique graph realizations. *SIAM Journal on Computing*, 21(1):65–84, 92. – Cité en pages 48, 49, 55, 69, 118, 134 et 145.

[HHM+10] T. C. H , J. H , S. M L , T. N , S. O et R. Z : A revision of the proof of the Kepler conjecture. *Discrete and Computational Geometry*, 44(1):1–34, 2010. – Cité en page 83.

[HJA97] C. M. H et R. J -A : Symbolic constraints in constructive geometric constraint solving. *Journal of Symbolic Computation*, 23(2-3):287–299, 1997. – Cité en page 44.

[HJA98] C. M. H et R. J -A : On user-defined features. *Computer-Aided Design*, 30(5):321–332, 1998. – Cité en pages 4 et 126.

[HJA05] C. M. H et R. J -A : A brief on constraint solving. *Computer-Aided Design and Applications*, 2(5):655–663, 2005. – Cité en page 40.

[HLS97] C. M. H , A. L et M. S : Finding solvable subsets of constraint graphs. *In CP 1997: Proceedings of the 3rd International Conference on Principles and Practice of Constraint Programming*, volume 1330 de *Lecture Notes in Computer Science*, pages 463–477, Hagenberg Castle, Autriche, 1997. – Cité en pages 45, 50 et 51.

[HLS98] C. M. H , A. L et M. S : Geometric constraint decomposition. *In* B. B ¨ et D. R ,

éditeurs : *Geometric Constraint Solving and Applications*, chapitre 3, pages 170–195. Springer, 1998. – Cité en pages 42, 45 et 96.

[HLS99] C. M. H , A. L et M. S : Planning geometric constraint decompositions via graph transformations. *In AGTIVE '99: Proceedings of the 1st International Workshop on Graph Transformation with Industrial Relevance*, volume 1779 de *Lecture Notes in Computer Science*, pages 309–324, Kerkrade, Pays-Bas, 1999. Springer. – Cité en page 45.

[HLS01a] C. M. H , A. L et M. S : Decomposition plans for geometric constraint systems, part I : Performance measures for CAD. *Journal of Symbolic Computation*, 31(4):367–408, 2001. – Cité en page 56.

[HLS01b] C. M. H , A. L et M. S : Decomposition plans for geometric constraint systems, part II : New algorithms. *Journal of Symbolic Computation*, 31(4):409–427, 2001. – Cité en page 56.

[HLSJS+09] K. H , A. L -S . J , M. S , I. S et N. W : Body-and-cad geometric constraint systems. *In SAC '09: Proceedings of the 24th ACM Symposium on Applied Computing*, pages 1127–1131, Honolulu, Hawaii, 2009. ACM. – Cité en page 51.

[Hof01] C. M. H : Robustness in geometric computations. *Journal of Computing and Information Science in Engineering*, 1(2): 143–155, 2001. – Cité en page 62.

[Hof06] C. M. H : Summary of basic 2D constraint solving. *International Journal of Product Lifecycle Management*, 1(2):143–149, 2006. – Cité en page 40.

[HSY04] C. M. H , M. S et B. Y : Making constraint solvers more usable: overconstraint problems. *Computer-Aided Design*, 36(4):377–399, 2004. – Cité en pages 56 et 134.

[ISO04] ISO : *ISO/TR 16570:2004: Geometrical Product Specifications (GPS) – Linear and angular dimensioning and tolerancing: +/- limit specifications – Step dimensions, distances, angular sizes and radii*. International Organization for Standardization, Genève, Suisse, 2004. – Cité en page 14.

[JA09] R. J -A : Basics on geometric constraint solving. *In EGC '09: XIII Encuentros de Geometria Computacional*, Zaragoza, Espagne, 2009. Oral presentation. Paper available at http://metodosestadisticos.unizar.es/~egc09/index_archivos/Trabajos/robert.pdf. – Cité en page 40.

[JALSR02] R. J -A , M. V. L ´ et A. S -R : Constructive geometric constraint solving : A new application of genetic algorithms. *In PPSN '02: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, volume 2439 de *Lecture Notes in Computer Science*, pages 759–768, Granada, Espagne, 2002. Springer. – Cité en pages 60 et 130.

[JALSR03] R. J -A , M. V. L ´ et A. S -R : Genetic algorithms for root multiselection in constructive geometric constraint solving. *Computer & Graphics*, 27(1):51–60, 2003. – Cité en pages 60 et 130.

[JASR97] R. J -A et A. S -R : A correct rule-based geometric constraint solver. *Computer and Graphics*, 5(21):599–609, 1997. – Cité en pages 44 et 62.

[JASR99] R. J -A et A. S -R : Combining constructive and equational geometric constraint-solving techniques. *ACM Transactions on Graphics*, 18(1):35–55, 1999. – Cité en page 44.

[JASRVM03] R. J -A , A. S -R et S. V -M : Tools to deal with under-constrained geometric constraint graphs. *In ASCM '03: Proceedings of the 6th Asian Symposium on Computer Mathematics*, 2003. – Cité en page 57.

[JASRVMVP03] R. J -A , A. S -R , S. V -M et J. V -P : Transforming an under-constrained geometric constraint problem into a well-constrained one. *In SMA '03: Proceedings of the 8th ACM symposium on Solid modeling and applications*, pages 33–44, New York, New-York, USA, 2003. ACM Press. – Cité en page 57.

[JASRVMVP04] R. J -A , A. S -R , S. V -M et J. V -P : Revisiting decomposition analysis of geometric constraint graphs. *Computer-Aided Design*, 36(2):123–140, 2004. – Cité en pages 43 et 45.

[Jer02] C. J : *Résolution de contraintes géométriques par rigidification récursive et propagation d'intervalles*. Thèse de doctorat,

Université de Nice Sophia-Antipolis, Nice, France, 2002. – Cité en page 28.

[JJ05] B. J et T. J ´ : Connected rigidity matroids and unique realization of graphs. *Journal of Combinatorial Theory*, 94(1):1–29, 2005. – Cité en page 48.

[JMS07] C. J , D. M et P. S : Modélisation géométrique par contraintes. *In* D. B et B. P´ , éditeurs : *Informatique graphique, modélisation géométrique et animation*, Traitement du Signal et de l'Image, chapitre 6, pages 185–210. Hermès-Lavoisier, 2007. – Cité en page 40.

[JNT02] C. J , B. N et G. T : A new structural rigidity for geometric constraints systems. *In ADG '02: Proceedings of the 4th International Workshop on Automated Deduction in Geometry*, volume 2930 de *Lecture Notes in Artificial Intelligence*, pages 87–106, Hagenberg Castle, Autriche, 2002. Springer. – Cité en pages 50 et 134.

[JNT03] C. J , B. N et G. T : Algorithms for identifying rigid subsystems in geometric constraint systems. *In IJCAI '03: Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 233–238, Acapulco, Mexique, 2003. Morgan Kaufmann. – Cité en pages 42, 50, 96 et 134.

[JTNM06] C. J , G. T , B. N et P. M : Decomposition of geometric constraint systems: a survey. *International Journal on Computer Graphics and Application*, 16(5,6):379–414, 2006. – Cité en pages 45, 156 et 176.

[Knu98] D. E. K : *Sorting and searching*, volume 3 de *The art of computer programming*. Addison Wesley, 1998. – Cité en page 112.

[Kon90] K. K : PIGMOD: parametric and interactive geometric modeller for mechanical design. *Computer-Aided Design*, 22(10): 633–644, 1990. – Cité en page 41.

[Kon92] K. K : Algebraic method for manipulation of dimensional relationships in geometric models. *Computer-Aided Design*, 24(3):141–147, 1992. – Cité en page 41.

[Kra90] G. A. K : *Geometric reasoning in the kinematic analysis of mechanisms*. Thèse de doctorat, University of Sussex, Brighton, England, Royaume Uni, 1990. – Cité en page 58.

[Kra91] G. A. K : Using degrees of freedom analysis to solve geometric constraint systems. *In SMA '91: Proceedings of the 1st ACM symposium on Solid modeling foundations and CAD/CAM applications*, pages 371–378, Austin, Texas, USA, 1991. ACM. – Cité en page 58.

[Kra92] G. A. K : A geometric constraint engine. *Artificial Intelligence*, 58(1-3):327–360, 1992. – Cité en pages 4, 58 et 192.

[Lam70] G. L : On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4(4):331–340, 1970. – Cité en pages 30, 32 et 47.

[LBYJA04] M. V. L ´ , E. B , E. Y et R. J -A : GA and CHC. two evolutionary algorithms to solve the root identification problem in geometric constraint solving. *In ICCS '04: Proceedings of the 4th International Conference on Computational Science*, volume 3039 de *Lecture Notes in Computer Science*, pages 139–146, Krakòw, Pologne, 2004. Springer. – Cité en pages 60 et 130.

[LK96] J.-Y. L et K. K : Geometric reasoning for knowledge-based parametric design using graph representation. *Computer-Aided Design*, 28(10):831–841, 1996. – Cité en page 44.

[LK98] J.-Y. L et K. K : A 2-D geometric constraint solver using DOF-based graph reduction. *Computer-Aided Design*, 30(11): 883–896, 1998. – Cité en pages 42 et 44.

[LKLK03] K.-Y. L , O.-H. K , J.-Y. L et T.-W. K : A hybrid approach to geometric constraint solving with graph analysis and reduction. *Advances in Engineering Software*, 34(2):103–113, 2003. – Cité en page 44.

[LLG81] R. L , V. L et D. C. G : Variational Geometry in CAD. *Computer Graphics*, 15(3):171–175, 1981. – Cité en page 42.

[LLS00] D. L , J.-C. L´ et P. S ´ : A declarative approach to a 2D variational modeler. *In IDMME '2000: Proceedings of the 3rd International Conference on Integrated Design and Manufacturing in Mechanical Engineering*, pages 105–112, Montréal, Québec, Canada, 2000. – Cité en page 54.

[LM95] H. L et D. M : Solving geometric constraints by homotopy. *In SMA '95: Proceedings of the 3rd ACM symposium*

on Solid Modeling and Applications, pages 263–269, Salt Lake City, Utah, USA, 1995. ACM Press. – Cité en page 41.

[LM96a] H. L et D. M : Solving geometric constraints by homotopy. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):28–34, 1996. – Cité en page 41.

[LM96b] R. S. L et A. E. M : Connectivity analysis: a tool for processing geometric constraints. *Computer-Aided Design*, 28(11):917–928, 1996. – Cité en pages 5, 42, 45, 49, 50, 55, 91, 96 et 118.

[LM98] H. L et D. M : Qualitative study of geometric constraints. *In* B. B ¨ et D. R , éditeurs : *Geometric Constraint Solving and Applications*, chapitre 2, pages 234–258. Springer, 1998. – Cité en page 50.

[LSRGJA05] M. V. L ´ , A. S -R , J.-F. G´ et R. J -A : Searching the solution space in constructive geometric constraint solving with genetic algorithms. *Applied Intelligence*, 22(2):109–124, 2005. – Cité en pages 4, 60 et 130.

[LY82] L. L ´ et Y. Y : On generic rigidity in the plane. *SIAM Journal on Algebraic and Discrete Methods*, 3(1):91–98, 1982. – Cité en page 48.

[Mat97] P. M : *Constructions géométriques sous contraintes en modélisation à base topologique*. Thèse de doctorat, Université Louis Pasteur, Strasbourg, France, 1997. – Cité en pages 55, 62, 74 et 207.

[MF03] L. I. M et J. D. F : Formalizing Hilbert's Grundlagen in Isabelle/Isar. *In TPHOL '03: Proceedings of the 16th International Conference on Theorem Proving in Higher Order Logics*, volume 2758 de *Lecture Notes in Computer Science*, pages 319–334. Springer, 2003. – Cité en page 83.

[MF06] M. M et S. F : Geometric constraint solving: The witness configuration method. *Computer-Aided Design*, 38(4): 284–299, 2006. – Cité en pages 34, 52, 89, 154, 156 et 176.

[MF07] M. M et S. F : Detecting all dependences in systems of geometric constraints using the witness method. *In ADG '06: Proceedings of the 6th international workshop on Automated Deduction in Geometry*, volume 4869 de *Lecture Notes*

in Artificial Intelligence, pages 98–112, Pontavedra, Espagne, 2007. Springer. – Cité en pages 52, 89 et 154.

[MF09] D. M et S. F : Interrogating witnesses for geometric constraint solving. *In SPM '09: Proceedings of the SIAM/ACM joint conference on Geometric and Physical Modeling*, pages 343–348, San Francisco, California, USA, 2009. ACM. – Cité en pages 6, 52, 53, 89 et 154.

[MFLM06] D. M , S. F , L. L et D. M´ : Another paradigm for geometric constraints solving. *In CCCG '06: Proceedings of the 18th Annual Canadian Conference on Computational Geometry*, pages 169–172, Queen's University, Ontario, Canada, 2006. – Cité en pages 52, 89 et 154.

[Moo87] R. E. M : *Methods and applications of Interval Analysis*. Studies in Applied and Numerical Mathematics. SIAM, 1987. – Cité en page 42.

[MS04] A. M et J. S : Banana spiders: a study of connectivity in 3D combinatorial rigidity. *In CCCG '04: Proceedings of the 16th Canadian Conference on Computational Geometry*, pages 44–47, Montréal, Québec, Canada, 2004. – Cité en pages 34, 48 et 143.

[MST+10] D. M , P. S , S. E. B. T , C. F¨ et J.-D. G´ : Using the witness method to detect rigid subsystems of geometric constraints in CAD. *In SPM '10: Proceedings of the 15th ACM Conference on Solid and Physical Modeling*, Haïfa, Israël, 2010. ACM.

[NDB98] A. N , M. D et W. F. B : Solving over- and underconstrained geometric models. *In* B. B ¨ et D. R - , éditeurs : *Geometric Constraint Solving and Applications*, chapitre 2, pages 107–127. Springer, 1998. – Cité en pages 56, 58 et 134.

[Nel85] G. N : Juno, a constraint-based graphic system. *ACM SIGGRAPH Computer Graphics*, 19(3):235–243, 1985. – Cité en page 42.

[NTC10] B. N , G. T et G. C : Improving inter-block backtracking with interval Newton. *Constraints*, 15(1):93–116, 2010. – Cité en page 42.

[OSMA01] J.-J. O , M. S , B. M et A. A : FRONTIER: fully enabling geometric constraints for feature-based modeling and assembly. *In SMA '01: Proceedings of the 6th ACM symposium on Solid Modeling and Applications*, pages 307–308, Ann Arbor, Michigan, USA, 2001. ACM. – Cité en pages 4 et 126.

[Owe91] J. C. O : Algebraic solution for geometry from dimensional constraints. *In SMA '91: Proceedings of the 1st ACM symposium on Solid modeling foundations and CAD/CAM applications*, pages 397–407, Austin, Texas, USA, 1991. ACM. – Cité en pages 43, 45, 62, 80, 180 et 189.

[Par09] V. P : *Manuel d'Économie politique*. Giard et Brière, Paris, France, 1909. – Cité en page 125.

[PCRT07] J.-M. P , J. C ´ , L. R et F. T : A space decomposition method for path planning of loop linkages. *In IROS '07: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1882–1888, San Diego, California, USA, 2007. IEEE. – Cité en page 58.

[Pod02] D. P : A new constructive approach to constraint-based geometric design. *Computer-Aided Design*, 34(11):769–785, 2002. – Cité en page 55.

[PŽD08] D. P , B. Ž et V D : Dealing with redundancy and inconsistency in constructive geometric constraint solving. *Advances in Engineering Software*, 39(9):770–786, 2008. – Cité en page 55.

[Sab04] K. S : Around the proof of the legendre-cauchy lemma on convex polygons. *Siberian Mathematical Journal*, 45(4):740–762, 2004. – Cité en page 48.

[SAZK06] M. S , A. A , Y. Z et N. K : Solution space navigation for geometric constraint systems. *ACM Transactions on Graphics*, 25(2):194–213, 2006. – Cité en pages 4, 60 et 129.

[Sch01] P. S : Robustness in CAD geometric constructions. *In IV '01: Proceedings of the 5th International Conference on Information Visualisation*, pages 111–116, London, England, Royaume Uni, 2001. IEEE Computer Society Press. – Cité en page 62.

[Sch02] P. S : *Résolution symbolique de contraintes géométriques*. Thèse de doctorat, Université Louis Pasteur, Strasbourg, France, 2002. – Cité en page 13.

[Ser00] P. S ´ : *Cohérence de la spécification d'un objet de l'esplace euclidien à n dimensions*. Thèse de doctorat, École Centrale Paris, Paris, France, 2000. – Cité en page 54.

[Sit00] M. S : Personal communication on the minimal dense algorithm. University of Florida, Gainesville, Florida, USA, 2000. – Cité en page 50.

[Sit06] M. S : Well-formed systems of point incidences for resolving collections of rigid bodies. *International Journal of Computational Geometry and Application*, 16(5–6):591–615, 2006. – Cité en pages 45 et 180.

[SM06] P. S et P. M : Geometrical constraint system decomposition: a multi-group approach. *International Journal of Computational Geometry and Applications*, 16(5–6):431–442, 2006. – Cité en pages 3, 7, 12, 45, 55, 58, 62, 72 et 185.

[SOZA06] M. S , J.-J. O , Y. Z et A. A : Geometric constraints within feature hierarchies. *Computer-Aided Design*, 38(1):22–38, 2006. – Cité en pages 4 et 126.

[SS02] E. S et P. S : A case study in geometric constructions. *In ICCSA '02: Proceedings of the 2002 International Conference on Computational Science and its Applications*, volume 2330 de *Lecture Notes in Computer Science*, pages 201–210, Amsterdam, Pays-Bas, 2002. Springer-Verlag. – Cité en page 55.

[SS03] E. S et P. S : Solving geometric constraints invariant modulo the similarity group. *In ICCSA '03: Proceedings of the 2003 International Conference on Computational Science and its Applications*, volume 2669 de *Lecture Notes in Computer Science*, pages 356–365, Montréal, Québec, Canada, 2003. Springer-Verlag. – Cité en page 55.

[SS06] P. S et E. S : Using the invariance under the similarity group to solve geometric constraint systems. *Computer-Aided Design*, 38(5):475–484, 2006. – Cité en pages 3, 45, 55, 58 et 62.

[ST08] I. S et L. T : Combinatorial genericity and minimal rigidity. *In SCG '08: Proceedings of the 24th annual symposium on Computational geometry*, pages 365–374, College Park, Maryland, USA, 2008. ACM. – Cité en page 48.

[Ste66] D. S : A platform with six degrees of freedom : A new form of mechanical linkage which enables a platform to move simultaneously in all six degrees of freedom developed by Elliott-automation. *Aircraft Engineering and Aerospace Technology*, 38(4):30–35, 1966. – Cité en page 102.

[Sto68] J. J. S : Geometrical problems concerning polyhedra in the large. *Communications on Pure and Applied Mathematics*, 21(2):119–168, 1968. – Cité en page 48.

[Sun87] G. S : A CAD system with declarative specification of shape. *In Proceedings of the 1st Eurographics Workshop on Intelligent CAD systems*, pages 90–104, Noorwijkerhout, Pays-Bas, 1987. Springer-Verlag. – Cité en pages 43, 45 et 180.

[Sut63] I. E. S : Sketchpad: A man-machine graphical communication system. *In Proceedings of the AFIPS Spring Joint Computer Conference*, volume 23, pages 329–346, Detroit, Michigan, USA, 1963. – Cité en pages 2 et 42.

[SZ04] M. S et Y. Z : A tractable, approximate characterization of combinatorial rigidity in 3D. *In ADG '04: Fifth International Workshop on Automated Deduction in Geometry*, Gainesville, Florida, USA, 2004. Oral presentation, paper available online at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.130.9881&rep=rep1&type=pdf. – Cité en page 51.

[Tay99] T.-S. T : On the generic rigidity of bar-frameworks. *Advances in Applied Mathematics*, 23(1):14–28, 1999. – Cité en page 48.

[Thi06] S. E. B. T : Résolution de systèmes de contraintes géométriques invariants par des groupes de transformations d'assemblage. Mémoire de master, Université Louis Pasteur, Strasbourg, France, 2006.

[TMS07] S. E. B. T , P. M et P. S : Towards an homogeneous handling of under-constrained and well-constrained systems of geometric constraints. *In SAC '07: Proceedings of the 22nd ACM Symposium on Applied Computing*, pages 773–777, Seoul, Corée du Sud, 2007. ACM.

[TSM10] S. E. B. T , P. S et P. M : Why are under-constrained systems not that bad. *In ADG '10: Proceedings of the 8th International Workshop on Automated Deduction in Geometry*, München, Allemagne, 2010.

[TSMF08] S. E. B. T , P. S , P. M et A. F : Utiliser la sous-constriction pour résoudre des systèmes de contraintes géométriques. *In MOSIM '08: Actes de la 7ème conférence internationale de Modélisation et Simulation*, pages 892–900, Paris, France, 2008.

[TW06] G. T et M. W : GPDOF: a fast algorithm to decompose under-constrained geometric constraints: Application to 3D modeling. *International Journal of Computational Geometry and Applications*, 16(5–6):479–511, 2006. – Cité en page 58.

[VA92] R. C. V et F. A : Geometric constraint propagation with quantum labels. *In* B. F , I. H et C. P , éditeurs : *Eurographics Workshop on Computer Graphics and Mathematics*, pages 211–228. Springer, 1992. – Cité en page 42.

[vdMB05] H. A. van der M et W. F. B : An efficient method to determine the intended solution for a system of geometric constraints. *International Journal of Computational Geometry and Applications*, 15(3):279–298, 2005. – Cité en pages 4, 59 et 129.

[vdMB06] H. A. van der M et W. F. B : A constructive approach to calculate parameter ranges for systems of geometric constraints. *Computer-Aided Design*, 38(4):275–283, 2006. – Cité en pages 59 et 129.

[vdMB08] H. A. van der M et W. F. B : Solving systems of 3D geometric constraints with non-rigid clusters. *In GPM '08: Advances in Geometric Modelling and Processing*, volume 4975 de *Lecture Notes in Computer Science*, pages 423–436, Hangzhou, Chine, 2008. Springer. – Cité en page 58.

[vdMB10] H. A. van der M et W. F. B : A non-rigid cluster rewriting approach to solve systems of 3D geometric constraints. *Computer-Aided Design*, 42(1):36–49, 2010. – Cité en pages 46 et 58.

[VEG06] D. V et B. R. E -G : Genetic algorithms for graph layouts with geometric constraints. *In IASTED CI '06: Procee-*

dings of the 2nd IASTED International Conference on Computational Intelligence, pages 66–71, San Francisco, California, USA, 2006. IASTED/ACTA Press. – Cité en page 42.

[VSR92] A. V , F. S et D. R : Rule-oriented method for parameterized computer-aided design. *Computer-Aided Design*, 24(10):531–540, 1992. – Cité en pages 43 et 182.

[Win05] J. W : Compilation de systèmes à base de règles pour la résolution symbolique de contraintes géométriques. Mémoire de master, Université Louis Pasteur, Strasbourg, France, 2005. – Cité en pages 203 et 206.

[Wir90] M. W : Algebraic specification. *In* J. van L , éditeur : *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 675–788. MIT Press, 1990. – Cité en page 14.

[WSMF06] J. W , P. S , P. M et A. F : A framework for geometric constraint satisfaction problem. *In SAC '06: Proceedings of the 21st ACM Symposium on Applied Computing*, pages 974–978, Dijon, France, 2006. ACM. – Cité en page 203.

[Wu86] W. T. W : Basic principles of mechanical theorem proving in elementary geometries. *Journal of Automated Reasoning*, 2(3): 221–252, 1986. – Cité en page 41.

[Yan02] L. Y : Distance coordinates used in geometric constraint solving. *In ADG '02: Proceedings of the 4th International Workshop on Automated Deduction in Geometry*, volume 2930 de *Lecture Notes in Artificial Intelligence*, pages 216–229, Hagenberg Castle, Autriche, 2002. Springer. – Cité en page 55.

[Yan03] L. Y : Solving geometric constraints with distance-based global coordinate system. *In International Workshop on Geometric Constraint Solving*, Beijing, Chine, 2003. – Cité en page 55.

[Yan06] L. Y : Solving spatial constraints with global distance coordinate system. *International Journal of Computational Geometry and Applications*, 16(5–6):533–548, 2006. – Cité en page 55.

[Ypm95] T. J. Y : Historical development of the - method. *SIAM Review*, 37(4):531–551, 1995. – Cité en page 41.

[ZG04] G.-F. Z et X.-S. G : Geometric constraint solving based on connectivity of graph. *Computer-Aided Design and Applications*,

1(1–4):469–476, 2004. – Cité en page 45.

[ZG06a] G.-F. Z et X.-S. G : Spatial geometric constraint solving based on k-connected graph decomposition. *In SAC '06: Proceedings of the 21st ACM Symposium on Applied Computing*, pages 973–977, Dijon, France, 2006. ACM. – Cité en page 45.

[ZG06b] G.-F. Z et X.-S. G : Well-constrained completion and decomposition for under-constrained geometric constraint problems. *International Journal on Computer Graphics and Application*, 16(5–6):18–35, 2006. – Cité en pages 45 et 57.

[Zho06] Y. Z : *Combinatorial decomposition, generic independence and algebraic complexity of geometric constraint systems: applications in biology and engineering*. Thèse de doctorat, University of Florida, Gainesville, Florida, USA, 2006. – Cité en page 51.

Résumé

La résolution de systèmes de contraintes géométriques (GCS) a pour objectif de produire des figures qui respectent une description technique fournie par l'utilisateur sous la forme d'une esquisse cotée. Le GCS donné par l'utilisateur peut être bien contraint (il décrit un nombre fini non nul de figures), sous-contraint (une infinité de figures) ou sur-contraint (aucune solution).

Classiquement, les systèmes sous-contraints sont considérés comme des cas d'erreur que l'utilisateur doit corriger en ajoutant des contraintes. Nos travaux proposent une autre approche, qui est celle de chercher à résoudre de manière homogène tous les systèmes de contraintes géométriques qui ne sont pas sur-contraints.

Pour cela, nous proposons des algorithmes de paramétrisation, qui indiquent quels éléments du système doivent être fixés pour qu'il y ait un nombre fini de solutions, et des algorithmes de décomposition, qui permettent d'identifier les sous-systèmes bien contraints.

Ces outils ouvrent la voie à des logiciels de modélisation par contraintes accessibles à des utilisateurs non-experts : ils permettent des retours visuels intuitifs sur le niveau de constriction du système. Comme nos algorithmes sont incrémentaux, ils permettent une approche par essai/erreur où l'utilisateur corrige l'esquisse au fur et à mesure de la résolution.

Abstract

Solving geometric constraint systems (GCS) aims at yielding figures which respect geometric requirements given by the user under the form of a technical sketch. The GCS given by the user can be well-constrained (it describes a non-zero finite number of figures), under-constrained (an infinity of figures) or over-constrained (no solutions at all).

Classically, under-constrained systems are considered as errors to be corrected by the user. Our work proposes another approach, *i.e.* try to homogeneously solve all non-over-constrained systems.

For that, we propose parameterization algorithms : they indicate which elements of the system need to be anchored for there to be a finite number of solutions ; and decomposition algorithms, which allow to identify well-constrained subsystems.

These tools open the way to constraint-based modelers which are accessible to non-expert users : they give intuitive visual feedback about the constrainedness level of the system. Since our algorithms are all incremental, they allow a trial and error approach : the user corrects the sketch as the resolution goes.