Constructions géométriques, dessin industriel et informatique

Pascal Schreck, Pascal Mathis, Arnaud Fabre

Résumé : Cet article constitue la version écrite d'un exposé présenté devant des professeurs de mathématiques de l'enseignement secondaire dans le cadre d'une journée spéciale du Plan Académique de Formation. Nous y décrivons la mise au point de quelques algorithmes de résolution de contraintes géométriques à partir d'exemples de constructions géométriques dans le domaine du dessin technique assisté par ordinateur.

Introduction

Le dessin industriel constitue une des principales applications de la géométrie dans le domaine des sciences de l'ingénieur. Son cadre théorique constitue la *géométrie descriptive* qui a été formalisée par Gaspard Monge et qui était encore enseignée il y a quelques années en mathématiques dans certaines classes de terminale. Un des buts du dessin industriel est de représenter des objets 3D sur une feuille de dessin et de spécifier leurs mesures par un *système de cotes*.

Une telle représentation utilisant plusieurs projections orthogonales (ou *vues*, voir la figure 1) permet en effet de mieux appréhender les formes et les dimensions d'une pièce ou d'un ensemble de pièces et, éventuellement, de comprendre un mécanisme. Pour réaliser cette description, il n'est pas nécessaire de dépeindre une pièce réellement existante : le dessin technique est également un outil indispensable lors de la *conception* d'objets ou d'assemblage. La cotation sert alors à définir complètement un objet en vue de sa réalisation (cotes de fabrication) ou de la vérification de sa conformité vis-à-vis de certaines fonctionnalités (cotes fonctionnelles).

À ce propos, quiconque a pratiqué le dessin technique, en particulier en reproduisant à l'échelle des *esquisses cotées* comme celle de la figure 2(a) s'est vu confronté à des problèmes de constructions géométriques un peu similaires à ceux que l'on rencontre dans les programmes de géométrie de l'enseignement secondaire. Ici, les énoncés sont donnés sous forme graphique grâce à la cotation, qui doit respecter un certain nombre de normes et conventions, et la construction est habituellement effacée pour ne retenir que le dessin « au propre » (voir la construction géométrique suggérée à la figure 2(b)).

Ce désintérêt relatif pour la manière de résoudre le problème de construction correspondant à une esquisse n'est pas le seul point de divergence entre les constructions géométriques telles qu'elles sont vues dans l'enseignement et le dessin technique, en voici quelques autres :

- en constructions géométriques, les énoncés n'utilisent habituellement pas de données numériques et on *discute* tous les cas de figure possibles, alors qu'en dessin technique, on n'envisage qu'un cas de figure, celui présenté par l'esquisse avec les données métriques imposées par la cotation ;
- les contraintes géométriques qu'on peut rencontrer dans l'enseignement sont très variées et parfois très originales tandis que la cotation en dessin technique impose un nombre assez restreint de types de contraintes ;

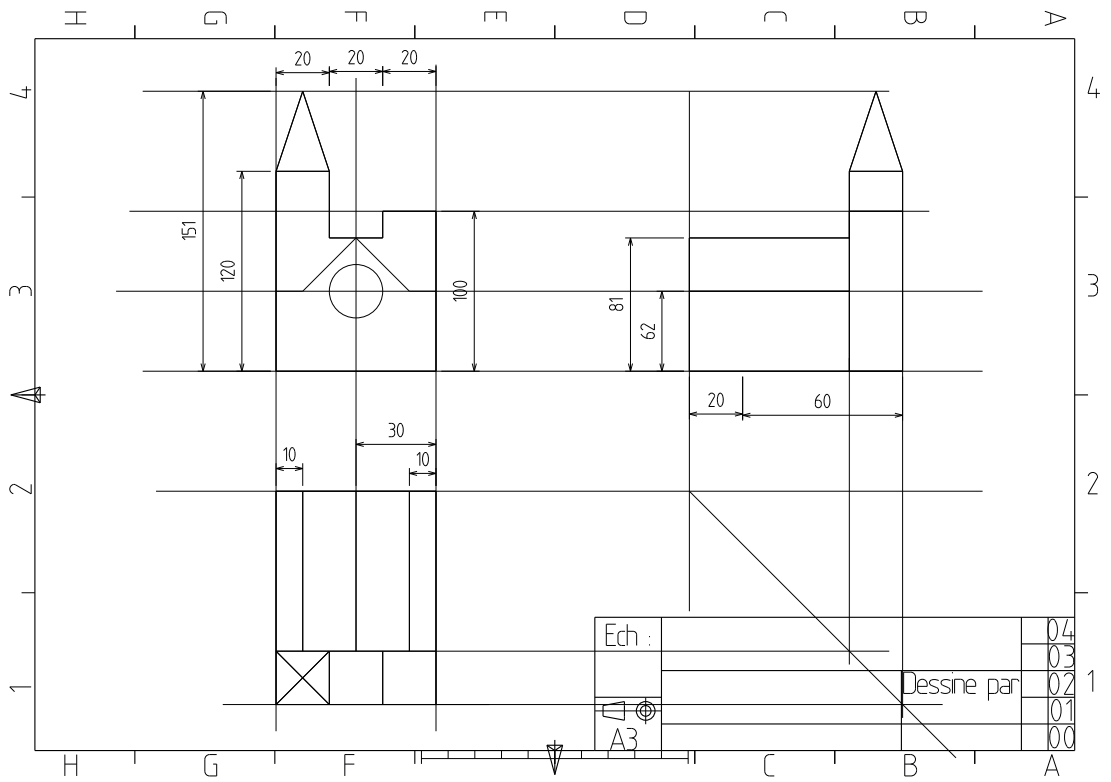


Fig. 1 – Exemple de dessin technique. La vue de face est en haut à gauche, la vue de dessus en bas à gauche et la vue de gauche en haut et à droite (un pictogramme représentant un tronc de cône dans le cartouche rappelle les conventions de vue). Ce dessin a été réalisé avec Qcad un logiciel de dessin technique gratuit et opensource (voir `http ://www.ribonsoft.com`)

– les problèmes rencontrés dans l'enseignement sont en général de petite taille et toujours astucieux, dans le sens où l'on veut faire utiliser une notion particulière aux élèves, tandis qu'en dessin technique, les dessins mettent souvent en jeu un grand nombre d'objets et de contraintes et, dans beaucoup de cas, la construction n'est pas très compliquée.

Les divergences entre les deux domaines se sont naturellement traduites par des approches très différentes lorsqu'il s'est agi d'« informatiser » les constructions géométriques. Dans le domaine de l'enseignement assisté par ordinateur, les chercheurs ont souvent suivi une approche consistant à reproduire les méthodes de construction utilisées dans l'enseignement et cela s'est traduit par des techniques essentiellement empruntées à l'intelligence artificielle. Notons ici, qu'il y a fort peu de travaux qui traitent de l'automatisation des constructions géométriques [1]. Inversement, les premières approches utilisées en Conception et Dessin Assistés par Ordinateur (CDAO ou plus simplement CAO) se sont essentiellement focalisées sur la résolution de systèmes d'équations non-linéaires en utilisant des techniques numériques basées sur la méthode de Newton-Raphson ou la méthode par continuation.

[1] Nous parlons ici des logiciels permettant de résoudre automatiquement des problèmes de construction géométriques et non des éditeurs en géométrie dynamique, comme Cabri-géomètre ou Cinderella, qui sont d'une facture plus classique.

Ces méthodes sont bien connues des mathématiciens et, malgré leur adaptation au cadre de la CAO, il nous semble qu'elles ont un rapport lointain avec la géométrie. Nous avons ainsi pris le parti de ne présenter ici que des méthodes de résolution constructive, c'est-à-dire basées sur des propriétés géométriques des figures et des objets mis en jeu. Il va par ailleurs de soi que nous n'allons pas présenter un panorama exhaustif des recherches dans ce domaine. Ainsi, et cela semble un peu paradoxal, cet article ne va pas détailler les travaux concernant l'enseignement assisté par ordinateur où, modulo une relative complexité technique, on reproduit les schémas de pensée des enseignants.

Il nous a, en effet, paru plus intéressant de montrer à un public composé de professeurs de mathématiques comment, à l'aide de quelques exemples, les constructions géométriques sont utilisées, en dehors des mathématiques, dans le contexte de l'informatisation du dessin technique. De ce point de vue, l'objectif est double, il s'agit d'une part de montrer des problèmes de constructions intéressants issus du dessin technique et d'autre part, d'illustrer, dans ce cadre particulier, des méthodes d'abstraction assez propres à l'informatique et souvent méconnues. Nous allons ainsi nous appuyer sur une série d'exemples de petite taille qui sont autant de problèmes de construction, puis dégager des méthodes permettant de traiter toute une famille de problèmes.

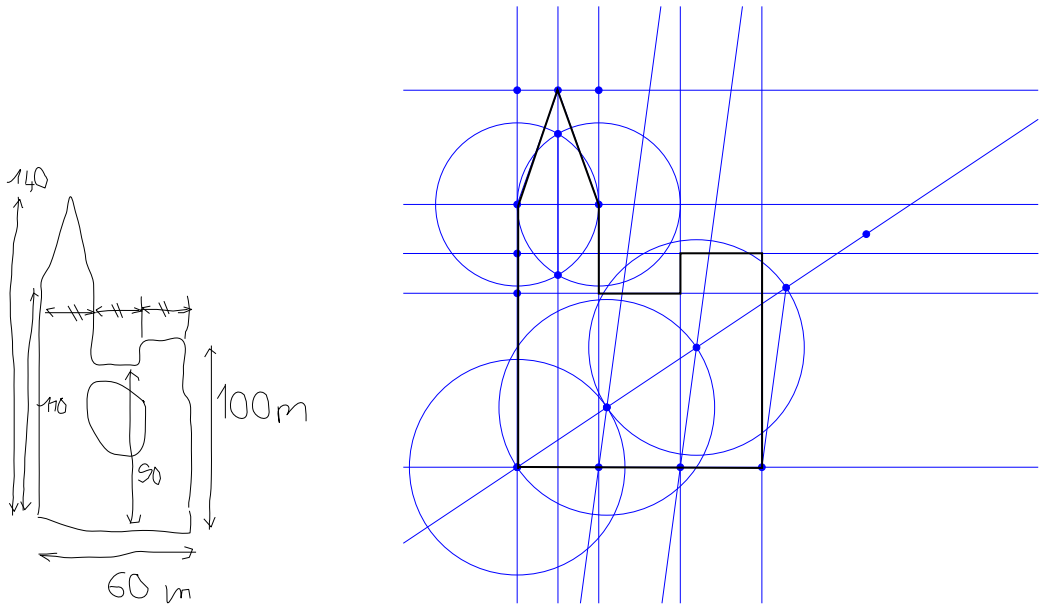
Le reste de cet article est structuré comme suit. La section 1 présente un certain nombre d'exemples introductifs que le lecteur peut essayer de résoudre avant de lire les sections suivantes. La section 2 présente un algorithme élémentaire permettant de résoudre de manière efficace beaucoup de cas simples du dessin technique. La section 3 étudie les polygones simplement contraints qui constituent un cas d'école où un petit raisonnement permet de montrer la complétude de l'algorithme de construction. La section 4 montre comment on peut exploiter systématiquement l'invariance par déplacement, qui est l'une des propriétés caractéristiques des systèmes de contraintes géométriques en dessin technique. Les techniques de décomposition des systèmes de contraintes peuvent s'étendre à d'autres groupes d'invariance, qu'on peut considérer dans un même cadre. La section 5 décrit un cadre où l'on exploite conjointement l'invariance par déplacement et l'invariance par similitude.

1. Exemples de base

Nous proposons dans cette section quelques exemples de problèmes de constructions géométriques simples dont les solutions seront développées plus loin.

Le dessin « à souris levée » donné à la figure 2(a), impose que la hauteur de la flèche soit de 140m et que les largeurs des deux flèches et de la partie entre celles-ci soient égales. Les contraintes de verticalité et d'horizontalité sont implicites. Sur ce schéma grossier, on peut aussi imaginer, parce que c'est une situation courante, que le sommet de la flèche décrit un triangle isocèle. Enfin, la rosace n'étant pas détaillée sur le dessin, nous supposerons que c'est un élément décoratif placé ici à titre d'illustration et qui ne sera pas repris dans le dessin à l'échelle. Moyennant ces compléments d'information, il y a un nombre fini de figures géométriques qui remplissent, à un déplacement près, ces contraintes. On dit parfois que la figure cotée est rigide ou isostatique ou bien contrainte modulo les déplacements puisque l'ensemble des solutions se décompose en un nombre fini d'orbites sous l'action du groupe des déplacements.

Dans cet exemple, la construction géométrique suggérée à la figure 2(b) par les lignes de construction, est simple et le dessinateur peut la réaliser facilement avec des outils de



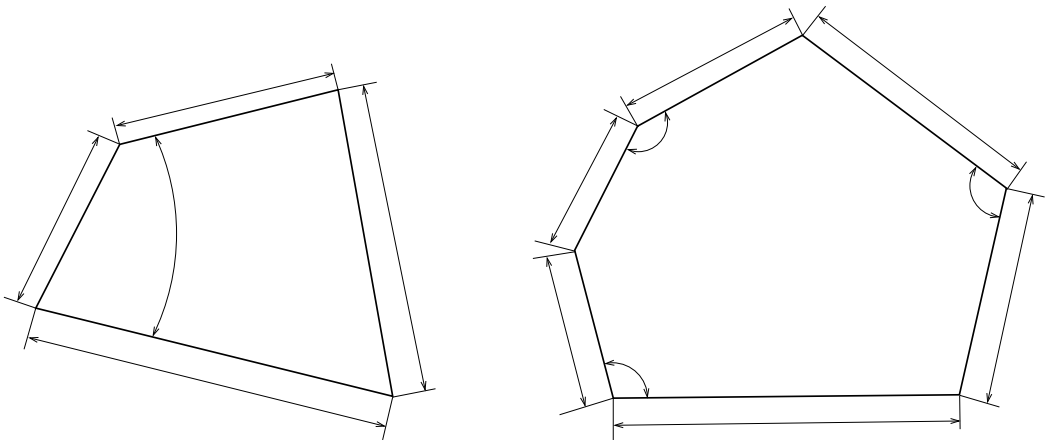
(a) esquise cotée (b) construction

Fig. 2 – Esquisse cotée (a) et construction d'une solution (b).

dessin usuels, qu'ils soient matériels ou logiciels. Remarquons qu'une approche algébrique aboutit à considérer un système global de 16 équations et 16 inconnues. Cette taille est modeste en regard des performances des outils actuels, mais on trouve facilement en dessin technique des esquisses cotées impliquant des centaines, voire des milliers, d'équations dont la résolution est coûteuse en temps. La section suivante explique comment des techniques simples de propagation de contraintes sont utilisées pour résoudre efficacement de tels problèmes.

Les deux exemples donnés à la Fig. 3, illustrent une autre facette du dessin technique. Il arrive qu'un ingénieur dessine un petit croquis avec quelques cotes fonctionnelles : ces dessins sont de petite taille, mais ils peuvent s'avérer difficiles à construire et, en tout cas, en dehors des possibilités des méthodes par propagation de contraintes. Les méthodes numériques, très générales, sont alors utilisées, mais dans le cas des polygones cotés simples, par exemple le quadrilatère et l'hexagone donnés Figs. 3(a) et 3(b), il existe des méthodes constructives permettant de trouver toutes les solutions. Remarquons que, sur ces esquisses, les valeurs des cotes ne sont pas mentionnées, cela n'est pas gênant mais impose, en toute rigueur, de trouver des solutions génériques. La section 3 décrit un algorithme général de résolution permettant, en particulier, de résoudre les problèmes de la figure 3.

Les techniques simples de propagation de contraintes ne parviennent généralement pas à résoudre des problèmes de taille importante, comme celui donné à la Fig. 4. En revanche, on peut souvent aboutir à une décomposition en sous-problèmes plus facile à résoudre. Les contraintes étant indépendantes de la position de la figure dans le plan, les méthodes utilisées se fondent habituellement sur l'invariance par déplacement de ces problèmes. Plusieurs techniques ont été développées pour découvrir et utiliser une telle décomposition, nous décrivons les plus simples à la section 4. Elles permettent, en particulier, de faire la construction correspondant à la figure 4.



(a) Quadrilatère (b) Hexagone

Fig. 3 – Deux polygones cotés simples

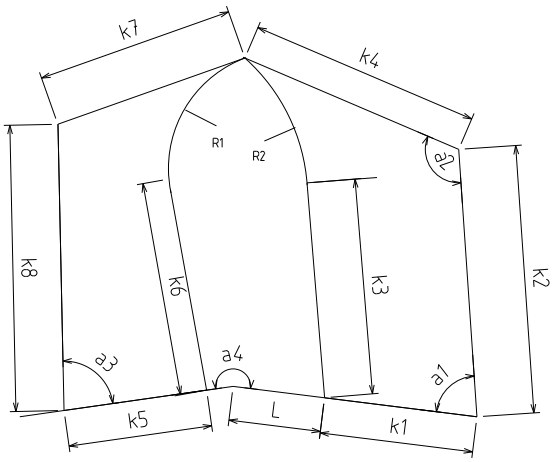
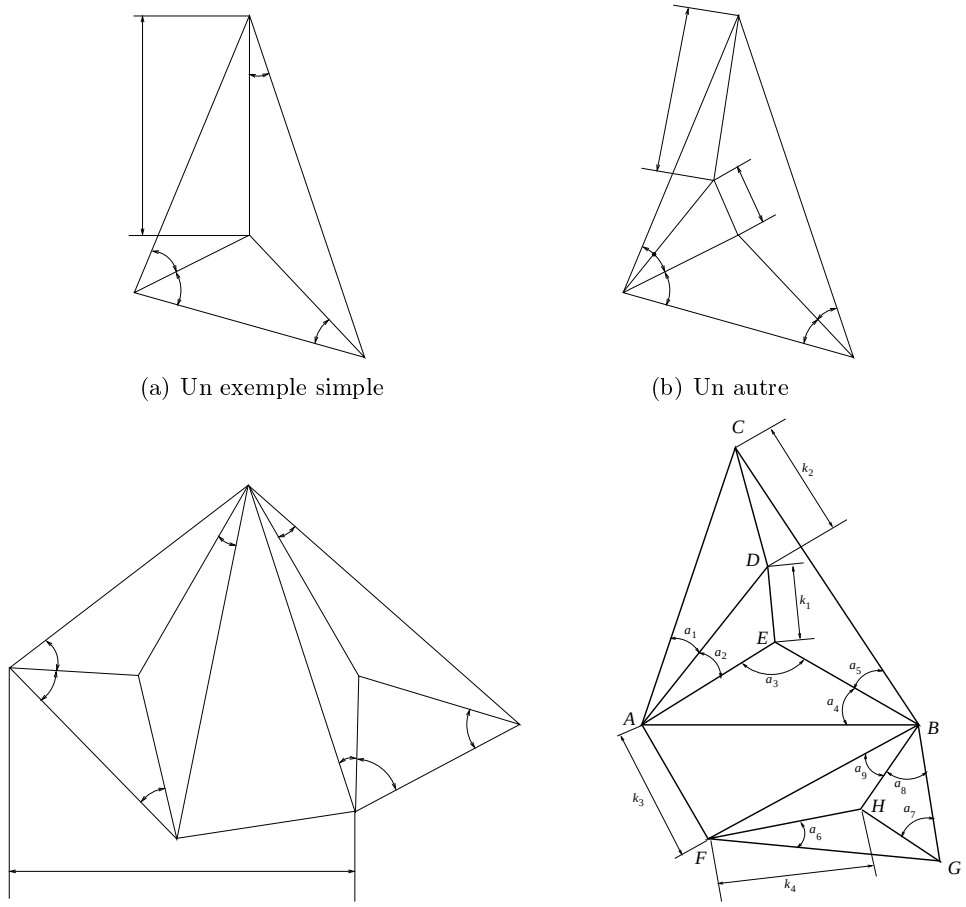


Fig. 4 – Esquisse cotée décomposable modulo les déplacements (Il y a deux contraintes de tangence qui sont implicites).

La dernière classe de problèmes, incluant les esquisses données à la Fig. 5, concerne une généralisation des techniques de décomposition au cas des systèmes de contraintes dont une partie est invariante par similitudes. La section 5 propose une méthode pour résoudre les problèmes de construction posés par les esquisses a) b) et c) de la Fig. 5 et suggère une construction pour l'esquisse d).



(a) Un exemple simple (b) Un autre

(c) Exemple avec de la décomposition (d) Un autre

Fig. 5 – Quatre exemples plus ou moins simples mettant en jeu l'invariance par similitude.

2. Propagation de contraintes

La propagation de contraintes, et son adaptation dans le cadre de la CAO, sont issues d'une idée simple : lorsqu'on a une solution concernant une inconnue dans un système de contraintes, on peut remplacer cette inconnue par cette valeur dans tout le système qui devient ainsi plus simple à résoudre dans ce cas particulier. Si on connaît toutes les solutions possibles pour l'inconnue, on a autant de systèmes plus simples à résoudre. Dans le cas particulier des constructions géométriques, on retrouve l'idée de la méthode des lieux : si on a construit deux lieux différents contenant un objet à construire, alors celui-ci appartient à leur intersection.

Cette idée s'est traduite par l'abstraction des données géométriques pour ne conserver des objets géométriques que la notion de degré de liberté, correspondant à peu près au nombre de coordonnées de l'objet, et ne garder des contraintes que la notion de degré de restriction, grosso modo le nombre d'équations réelles. Ces notions issues de la théorie de la rigidité [2,4] et appliquées, à l'origine, aux points et aux contraintes de distance, ont été implicitement étendues à divers types d'objets et de contraintes. Comme nous le verrons plus bas, cela a conduit à l'inadéquation des méthodes de propagation de contraintes aux problèmes généraux de résolution de contraintes géométriques.

Le processus d'abstraction évoqué plus haut conduit à considérer des *graphes étiquetés* où les sommets sont les inconnues du système pondérées par leur degré de liberté et les arêtes sont les contraintes pondérées par leur degré de restriction. Plus exactement, on a la définition suivante :

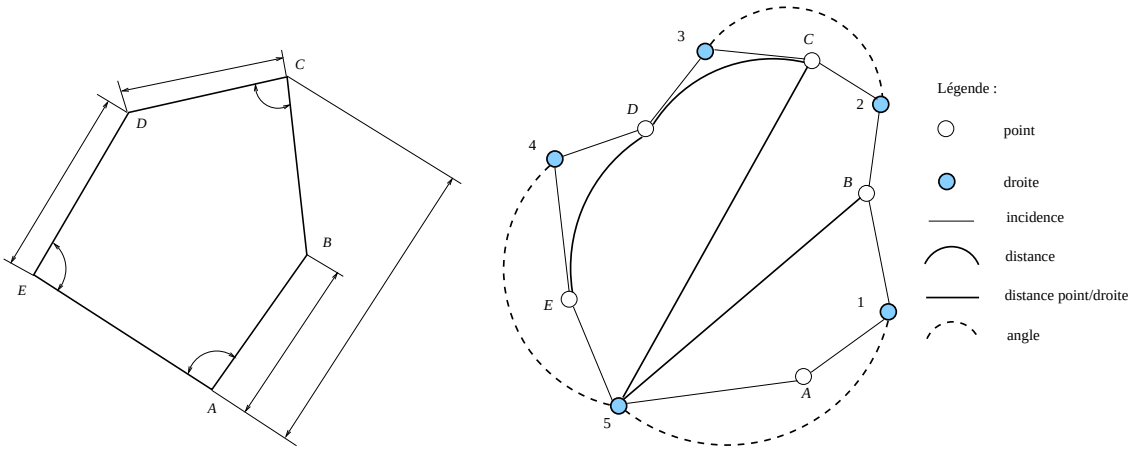
> **Définition 1** Étant donné un ensemble S, un graphe sur S est un couple $G = (S, A)$ où A est un sous-ensemble de $S \times S$. S est l'ensemble des sommets de G et A est l'ensemble des arcs de G. On peut aussi considérer des graphes non-orientés où A est un ensemble de paires (de cardinal exactement égal à 2) de sommets. La pondération des sommets et des arcs, ou des arêtes, est traduite par l'adjonction à la notion de graphe de deux fonctions de pondération, $f : S \to P_1$ et $g : A \to P_2$, où P_1 et P_2 sont des ensembles numériques. □

Nous considérons ici des *graphes de contraintes* qui sont des graphes non-orientés où S est l'ensemble des inconnues, A l'ensemble des contraintes géométriques supposées binaires, f est la fonction qui à toute inconnue associe le degré de liberté du type auquel elle correspond et g est la fonction qui à toute contrainte associe son degré de restriction. En considérant des points, des droites, des contraintes de distance, d'angle et d'incidence, nous nous plaçons dans un cadre simplifié où tous les sommets ont un degré de liberté égal à 2 et toutes les arêtes un degré de restriction égal à 1. Les graphes ont habituellement une représentation graphique où les sommets sont représentés par des pictogrammes, nous utiliserons des petits cercles, et les arêtes par des connecteurs, nous utiliserons des segments ou des arcs de cercle. Ainsi, à la Fig. 6, l'esquisse cotée de la partie gauche se traduit par le graphe représenté partie droite de la même figure : pour des raisons de lisibilité, les sommets sont dessinés différemment suivant qu'ils représentent des points ou des droites (disque grisé). Il en est de même pour les arêtes : les arêtes représentant une contrainte d'incidence sont représentées par un trait droit fin, celles représentant une distance entre deux points par un arc de cercle gras, celles représentant une distance point/droite par un trait droit gras et enfin, celles représentant des angles par un arc de cercle pointillé.

Oublions maintenant la géométrie pour ne considérer que le graphe. Deux techniques sont envisageables : l'une appelée *chaînage arrière* et l'autre *chaînage avant*. Voyons comment utiliser ces techniques sur l'exemple donné.

En adoptant le point de vue du chaînage arrière, on remarque que le sommet A n'est relié qu'à deux sommets, les sommets 1 et 5. Par conséquent, suivant la méthode des lieux, on peut dire que si on connaît 1 et 5, on connaitra A. On retire A et les arêtes incidentes pour obtenir un graphe simplifié où A n'apparaît plus. Dans ce graphe simplifié, le sommet 1 n'est relié qu'aux deux sommets B et 5. On peut donc le retirer de la même manière. On trouve ainsi une planification de la construction : A en dernier, 1 en avant dernier, *etc.*

pour arriver à un graphe extrêmement simple où l'on n'a plus que les sommets E et 5 relié par un arc. Ceci correspond au fait que les systèmes de contraintes géométriques sont, dans le cas de la CAO, invariants par déplacement. On obtient donc une solution particulière en fixant des objets géométriques particuliers, ici un point et une droite incidents. Une manière de construire la figure peut donc être : fixer le point E et la droite 5 passant par E, puis construire successivement 4, D, C, 3, 2, B, 1 et enfin A.



(a) esquisse cotée (b) graphe

Fig. 6 – Une esquisse cotée et le graphe de contraintes correspondant

Lorsqu'on agit par chaînage avant, on commence par fixer un repère de la figure, c'est-à-dire à fixer les valeurs de certains objets géométriques pour pouvoir commencer la construction. Un décompte des degrés de liberté conduit à fixer les objets correspondants à deux sommets reliés par une contrainte (on restreint ainsi le système de $2 \times 2 - 1 = 3$ degrés de liberté qui correspondent à la dimension de l'espace des déplacements)[2]. On peut, par exemple fixer les sommets E et D, ce qui correspond à tracer un segment de longueur donnée. On marque ces sommets pour indiquer ainsi qu'ils sont connus. Puisque le sommet 4 est relié par deux contraintes aux sommets E et D, on peut construire 4. On continue ainsi en marquant successivement les sommets non marqués qui sont reliés par deux contraintes à deux sommets marqués. On suit ainsi l'algorithme de la table 1.

Appliqué à notre exemple, l'algorithme parcourt successivement les sommets E, D, 4, 5, C, 3, 2, B, 1 et A, et produit en sortie la liste de ces sommets en signalant sa réussite. Remarquons qu'il y a plusieurs parcours possibles.

À l'issue de cette étape de planification, on peut finir de résoudre le système soit numériquement —on ne considère alors que des systèmes de deux équations à deux inconnues et de degré au plus deux, soit formellement en associant une méthode de construction à chaque couple de contraintes (ou un lieu à chaque contrainte).

Il est évident que cette méthode ne permet de résoudre que des problèmes simples : il s'agit en fait d'une triangulation de nature purement combinatoire de systèmes d'équations en

[2]Encore une fois, ce raisonnement abstrait sur les degrés de liberté peut facilement être mis en défaut dans des constructions géométriques simples.

```
Algorithme (planification par propagation de contraintes)
Données : S = ensembles des sommets, A = ensembles des arêtes
Sortie : liste de sommets
L = liste vide
Choisir une arête a = (p1, p2) dans A,
placer p1 à la fin de L
placer p2 à la fin de L
Tant qu'il y a des sommets de S non dans L et qu'on n'est pas en échec
faire
     chercher un sommet q de S - L relié à deux sommets de S dans L
      si cette recherche réussit  alors ajouter q à la fin de L
                                 sinon signaler un échec
                                       (et sortir de la boucle)
fait
retourner L et l'indication éventuelle d'un échec
```

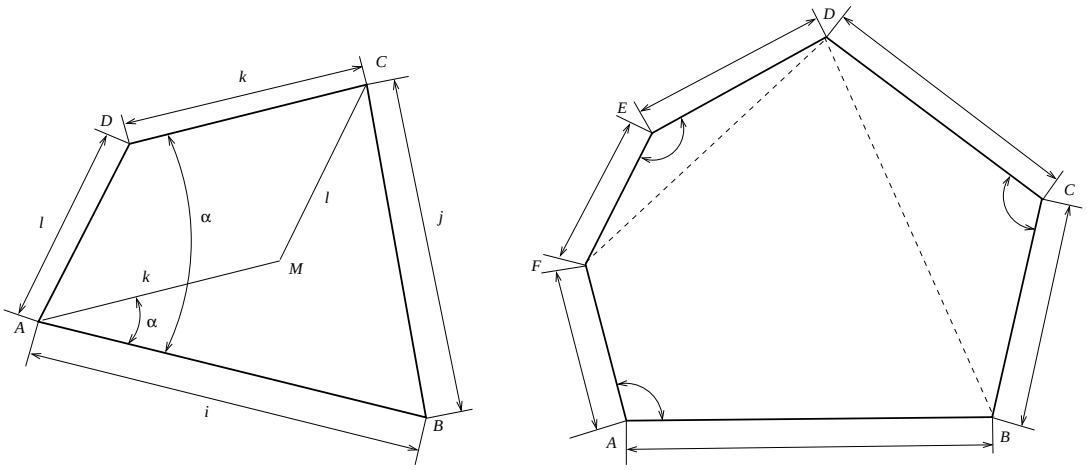
Tab. 1 – Algorithme de propagation de contraintes en chaînage avant.

blocs 2×2 tenant compte de l'invariance par déplacement. Ces méthodes sont bien sûr mises en échec par des exemples un peu plus sophistiqués, comme tous ceux des sections suivantes, et, pire, le cadre même de la propagation de contraintes est inadéquat. Il suffit, en effet, de considérer une esquisse cotée constituée d'un triangle dont on impose la valeur des trois angles : cette esquisse est sur-contrainte en général, à cause de la relation de Chasles, et il manque une information d'échelle. La méthode de propagation de contraintes ne permet de prendre en compte aucun des ces deux faits. Cependant elle est capable de résoudre de manière efficace un grand nombre de problèmes et elle est souvent employée conjointement à d'autres méthodes comme nous le verrons dans les sections 4 et 5.

3. Polygones simplement contraints

Les polygones simplement bien contraints constituent une famille d'exemples mettant en échec la propagation de contraintes. Il ne s'agit que d'exemples « jouets » de problèmes dont l'utilité essentielle ici est de montrer comment à partir d'exemples simples on essaie de généraliser des constructions particulières en termes d'algorithme et de structures de données. Les exemples typiques de problèmes que l'on veut résoudre correspondent aux problèmes classiques de construction de triangles et aux esquisses de la figure 3. Voyons comment on peut les résoudre constructivement.

Le « truc » permettant de construire le quadrilatère coté consiste à ajouter un point M matérialisant un parallélogramme comme à la Fig. 7(a)). Avec les relations d'égalité de longueurs et de parallélisme caractéristiques d'un parallélogramme, il est facile de voir que l'angle entre les droites (AB) et (AM) est connu et que la longueur AM l'est aussi. Ainsi, en commençant par fixer les points A et B, on peut poursuivre facilement en construisant le point M, puis le point C et enfin le point D symétrique de M par rapport au milieu de (A, C).



(a) Quadrilatère et parallélogramme (b) Hexagone et triangles auxiliaires

Fig. 7 – Figures auxiliaires pour résoudre des polygones simplement bien contraints

Le cas de l'hexagone est encore plus simple. Il suffit de remarquer qu'on peut déterminer les distances FD et BD, éventuellement en construisant des triangles auxiliaires, ce qui permet de construire facilement en fixant les points A et B, le quadrilatère (A, B, D, F). Le reste de la construction est immédiat.

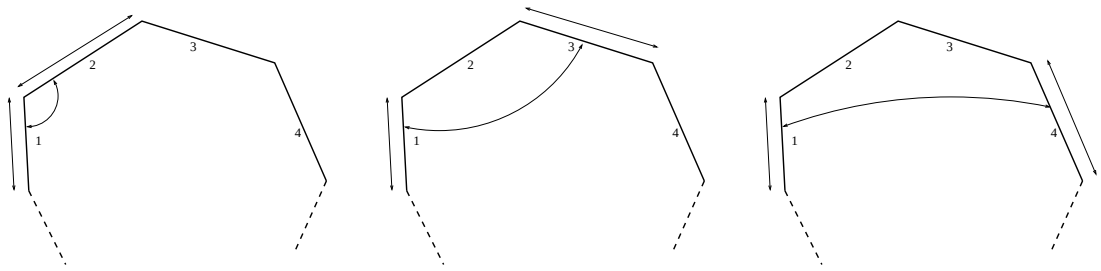
Les ingrédients utilisés pour la résolution de ces deux exemples vont nous servir pour résoudre la classe des polygones simplement contraints dont voici la définition.

> **Définition 2** Un polygone simplement contraint est un système de contraintes consistant en la donnée
> – d'une suite de n segments $E = (s_1, ..., s_n)$ formant un polygone quelconque
> – d'un ensemble de contraintes de longueur sur certains segments de E
> – d'un ensemble de contraintes d'angle sur certains couples de segments de E.
> Un polygone simplement bien contraint est un polygone bien contraint ayant un nombre fini de solutions à un déplacement près. □

En comptant les degrés de liberté et les degrés de restriction, comme on a n points, soit $2n$ inconnues réelles, il faut $2n - 3$ contraintes pour définir un tel polygone à un déplacement près (ce qui correspond au -3). Par ailleurs, ces contraintes doivent contenir au moins une contrainte de distance, elles doivent contenir au plus n contraintes de distance et au plus $n - 1$ contraintes d'angle. Nous sommes donc dans l'une des trois situations suivantes :
(i) n contraintes de distance et $n - 3$ contraintes d'angle,
(ii) $n - 1$ contraintes de distance et $n - 2$ contraintes d'angle,
(iii) $n - 2$ contraintes de distance et $n - 1$ contraintes d'angle.

Par ailleurs, en considérant la relation de Chasles, on peut regrouper les segments contraints en angle par classe : deux côtés sont dans la même classe si l'angle qu'ils font est connu ou déductible par la relation de Chasles, des contraintes d'angle données. Supposons que nous ajoutions les contraintes d'angle une à une, chaque ajout de contrainte d'angle entre deux côtés ne peut se faire qu'entre des côtés qui sont dans des classes différentes. On connecte ainsi les classes de ces deux côtés et on diminue le nombre de classes de 1. Ayant n classes avant la pose de toute contrainte d'angle, après avoir considéré toutes les

contraintes d'angle, on peut avoir respectivement 3, 2 ou 1 classes suivant que l'on soit dans les cas (i), (ii) ou (iii). Le théorème des tiroirs nous dit alors que si on considère 4 segments consécutifs, il y en a au moins deux entre lesquels il y a une contrainte d'angle et au moins deux dont la longueur est imposée. Plus exactement, en reprenant les cas précédents, on est dans l'un des trois cas :
(i) les 4 côtés sont de longueur imposée et on connaît au moins un angle,
(ii) 3 côtés sont de longueur imposée et on connaît au moins 2 angles,
(iii) 2 côtés sont de longueur imposée et on connaît tous les angles.
Un petit raisonnement montre alors qu'on est forcément dans l'une des trois situations représentées à la Fig. 8 (lorsqu'il n'y a pas de cotes sur les côtés cela signifie qu'il est indifférent qu'ils soient de longueur imposée ou non).



(a) Premier cas (b) Deuxième cas (c) Troisième cas

Fig. 8 – Les trois cas possibles pour les polygones simplement contraints

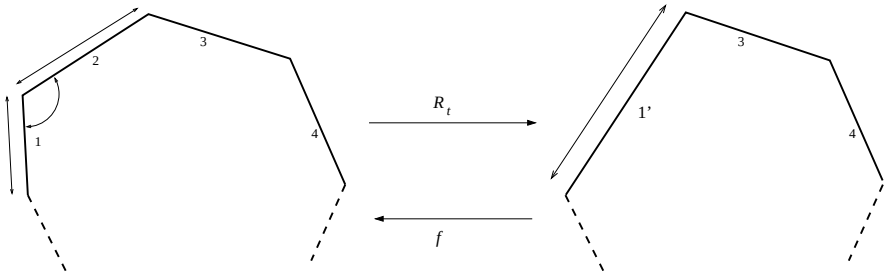
Le premier cas correspond à ce que nous avons fait lors de la résolution de l'hexagone et donne lieu à la règle que nous appellerons *règle du triangle*, représentée graphiquement à la Fig. 9 en haut. Cette règle est appliquée pour simplifier un polygone simplement contraint en remplaçant deux côtés du polygone par un seul côté dont la longueur peut être calculée. La « réciproque » de cette règle est une règle de construction permettant de construire, à partir d'une solution pour le polygone simplifié, une solution pour le polygone initial.

Le deuxième cas donne lieu de la même manière à une règle de simplification/construction, dite *règle du parallélogramme* et qui est celle que nous avons employée pour résoudre le quadrilatère coté. Elle consiste simplement à permuter les côtés numérotés 2 et 3 à la Fig. 9 au milieu, en faisant suivre les contraintes d'angle. On se retrouve alors dans un cas d'application de la règle du triangle. Comme précédemment, la réciproque de la règle permet de construire une solution du polygone simplement coté avant simplification. Finalement, le troisième cas se résout en appliquant une règle de permutation ressemblant à la précédente, suivie de la règle du parallélogramme et suivie, à nouveau, de la règle du triangle. Encore une fois, on peut retourner la règle pour faire la construction inverse de cette simplification.

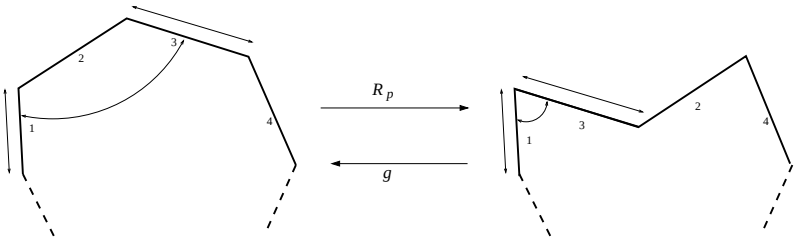
En appliquant tant que faire se peut ces règles de simplification, on aboutit au cas des triangles qui sont les cas les plus simples de polygones simplement contraints et que l'on sait parfaitement résoudre! Tout ceci conduit à l'algorithme décrit en langue naturelle à la table 2.

Par exemple, on peut construire ainsi un hexagone correspondant à la coupe d'un cube par un plan à partir des dimensions des segments tracés sur le cube (*cf.* Fig. 10) en appliquant

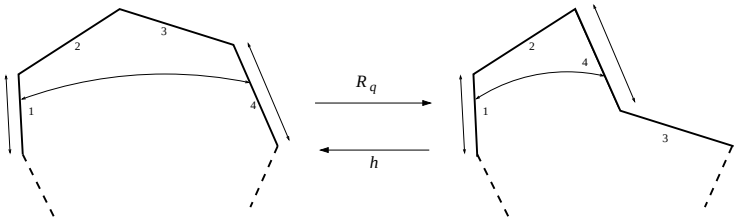
deux fois la deuxième règle du parallélogramme et en construisant le triangle dont la longueur de chaque côté est la différence des longueurs des côtés opposés de l'hexagone.



(a) Première règle : règle du triangle



(b) Deuxième règle : règle du parallélogramme



(c) Troisième règle : deuxième règle du parallélogramme

Fig. 9 – Les trois règles de simplification.

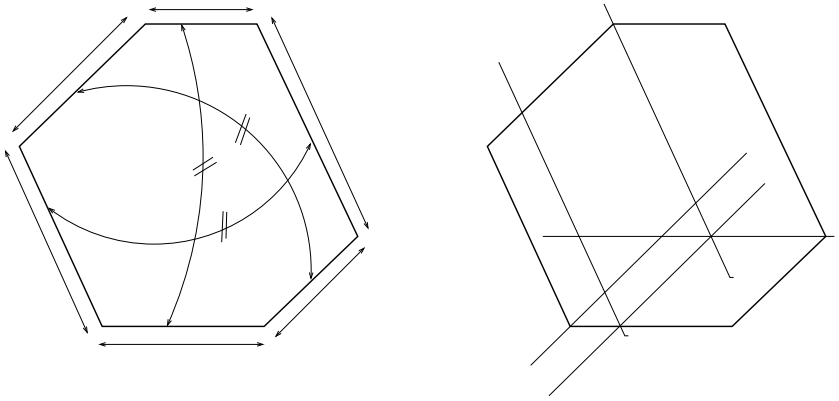


Fig. 10 – Coupe d'un cube : problème de construction (à gauche) quelques parallélogrammes utiles (à droite)

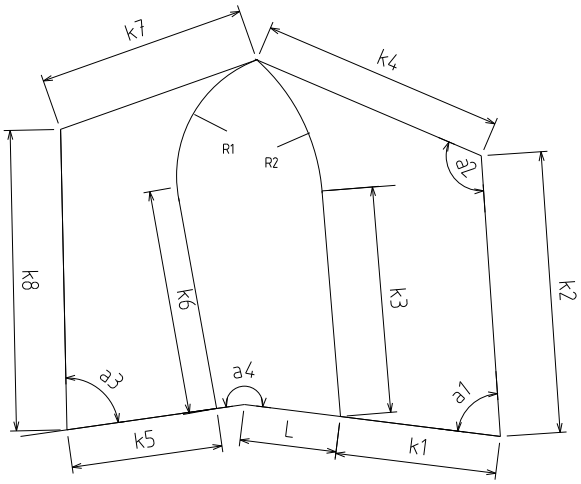
```
Algorithme(résolution des polygones simplement contraints)
Données : L = liste des côtés (avec la longueur quand elle est connue)
          A = ensemble des classes de côtés contraints en angle
Sortie : pile F des opérations à faire pour construire le polygone
F = liste vide
Tant que L est de longueur > 3 et qu'il n'y a pas d'erreur
  Si L[1] et L[2] sont de longueur connue et dans la même classe de A
     alors calculer la longueur du troisième côté 1',
           ainsi que les angles du triangle (1,2,1')
          enlever L[1] et L[2] et mettre 1' en tête de liste
          empiler sur F l'appel de fonction à f  avec les bons arguments
     sinon  si L[1] et L[3] sont de longueur connue et de même classe
            alors échanger L[2] et L[3] en mettant A à jour
                   empiler sur F l'appel de fonction à g
                     (avec les bons arguments)
            sinon si L[1] et L[4] sont de long. connue et de même classe
                   alors échanger L[3] et L[4] en mettant A à jour
                       empiler sur F l'appel de fonction à h
                         (avec les bons arguments)
                   sinon signaler une erreur et arrêter la construction
fait
Si la longueur de L n'est pas > 3 ou qu'il y a une erreur
  alors signaler une erreur
  sinon décider quel cas de triangle décrivent L et A
       empiler sur F la fonction résolvant ce cas de triangle
         (avec les bons arguments)
retourner F en signalant éventuellement une erreur
```

Tab. 2 – Algorithme pour résoudre des polygones simplement contraints

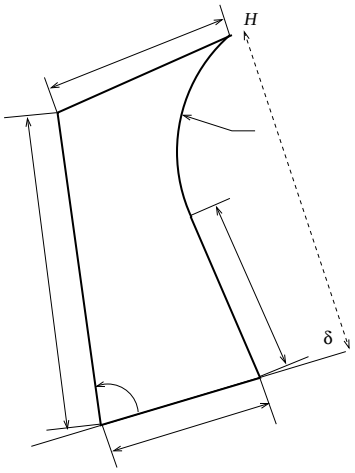
4. Décomposition tirant parti de l'invariance par déplacement

La règle du triangle de la figure 9 utilise de manière typique l'invariance par déplacement en permettant de déduire d'une figure auxiliaire une caractéristique (une distance) elle-même invariante par déplacement qui pourra être réutilisée dans une autre partie de la construction. Cela peut être généralisé à n'importe quel type de construction, et pas seulement des triangles, et à n'importe quelle caractéristique invariante par déplacement.

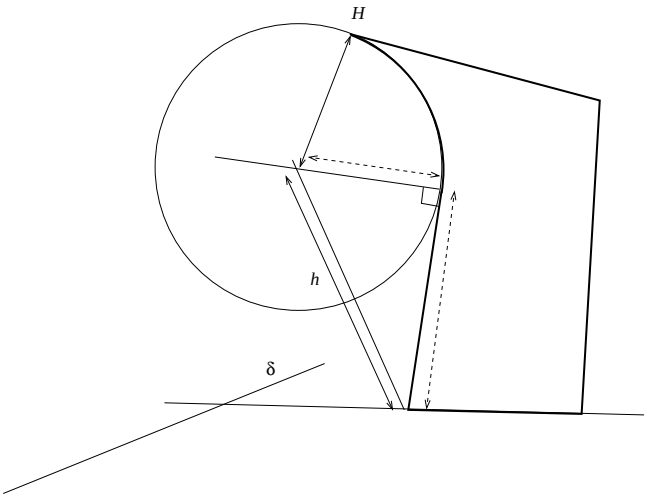
Essayons de résoudre le problème posé par l'esquisse de la Fig. 4 et reproduit à la Fig. 11(a). Comme le problème est invariant par déplacement, on commence par fixer un point et une direction en fixant, par exemple, les deux points de la partie droite tout en respectant la contrainte $k2$. On peut alors en appliquant des règles de géométrie élémentaire construire la partie droite avec, en plus, une droite attenant à la partie gauche (voir Fig. 11(c)). Puis, on ne sait pas continuer. On peut essayer de construire un autre morceau, par exemple dans l'esquisse cotée correspondant au système de contraintes restant. Cependant, celui-ci est sous-contraint modulo les déplacements.



(a) Rappel de l'esquisse cotée de la Fig. 4



(b) Sous-système de gauche (la cote en pointillés a été ajoutée).



(c) Début de la construction (notez le calcul de h)

Fig. 11 – Décomposition modulo les déplacements.

En ajoutant la contrainte de distance entre le point noté H sur la Fig. 11(b) et la droite δ que l'on peut calculer dans la partie droite (Fig. 11(c)) maintenant construite, on retrouve un système bien contraint modulo les déplacements qu'on peut résoudre simplement.

On a ainsi une construction en deux morceaux f_g et f_d qui sont deux solutions partielles et particulières : chacune a été construite en fixant un repère qui lui est propre et il n'y a aucune raison pour que la réunion de ces deux solutions partielles soit la solution globale. En revanche, puisque chacun des systèmes est invariant par déplacement, on obtient toutes les solutions d'un sous-système en faisant agir le groupe des déplacements sur les solutions particulières. Il existe ainsi un et un seul déplacement φ faisant coïncider le point H et la droite δ de la partie gauche avec le point H et la droite δ de la partie droite considérée comme fixe. Il est alors clair que la réunion de f_d et de $\varphi(f_g)$ est une solution particulière globale du système de contraintes initial. Il n'est pas difficile de montrer que ce procédé

est complet : si on a toutes les solutions particulières pour chacune des deux parties, on retrouve en faisant tous les assemblages possibles toutes les solutions particulières globales.

Nous retenons de cette construction les deux ingrédients essentiels qui pourront être utilisés de manière plus générale :

- le bord d'un sous-système est constitué des contraintes métriques que l'on peut déduire d'une solution au sous-système, par exemple la distance entre deux points.
- l'assemblage des solutions de deux sous-systèmes d'un même système de contraintes invariant par déplacement consiste à appliquer sur l'une des figures le déplacement faisant coïncider les valeurs pour les inconnues communes aux deux sous-systèmes.

Ces deux ingrédients permettent d'augmenter considérablement la puissance des méthodes par propagation de contraintes et ont donné lieu à plusieurs méthodes. Nous allons présenter deux approches différentes.

4.1 Méthode de Owen

Au début des années 90 Owen a proposé de décomposer les systèmes de contraintes géométriques en examinant, en quelque sorte, les points de faiblesse des graphes de contraintes sous-jacents. Donnons auparavant quelques définitions classiques.

Définition 3 Étant donnés deux sommets s et t d'un graphe non orienté G, un chemin de s à t est une suite finie de sommets $s_0 = s, s_1, \ldots, s_n = t$ telle que $\{s_i, s_{i+1}\} \in A$ pour tout $i \in [\![0, n-1]\!]$.
La relation d'accessibilité entre deux sommets est alors définie par :

s_2 est accessible à partir de s_1 s'il existe un chemin allant de s_1 à s_2.

Cette relation est une relation d'équivalence dont les classes sont appelées composantes connexes de G. Un graphe connexe est un graphe qui n'a qu'une composante connexe. □

En ce qui concerne les systèmes de contraintes du dessin technique, l'invariance par déplacement impose que les graphes correspondant à des systèmes bien contraints modulo les déplacements soient au moins connexes. En effet, chaque composante connexe correspond à un solide (déformable ou non) pouvant se déplacer librement par rapport aux éléments définis par les autres composantes connexes. En allant plus loin, on peut imaginer que si deux composantes connexes sont reliées par un sommet, on obtient encore deux solides pouvant avoir un mouvement relatif, par exemple une rotation si ce sommet est un point ou une translation si ce sommet est une droite. Les systèmes bien contraints modulo les déplacements doivent donc remplir un critère plus fort que la connexité.

Définition 4 Dans un graphe non orienté G, un sommet d'articulation est un sommet de G dont le retrait, avec les arêtes qui lui sont incidentes, augmente le nombre de composantes connexes (on dit qu'il déconnecte le graphe). Un graphe connexe qui ne contient pas de sommets d'articulation est dit biconnexe : dans un tel graphe, il existe au moins deux chemins distincts entre deux sommets distincts quelconques. □

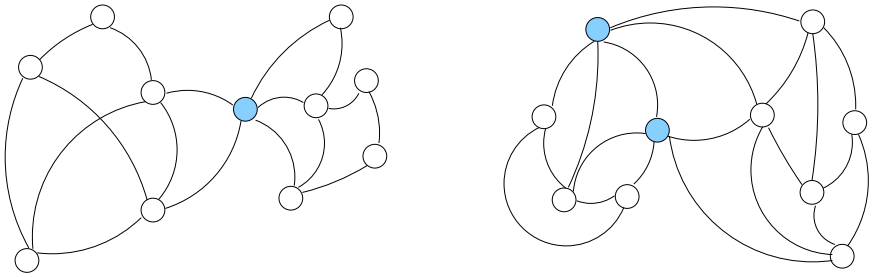
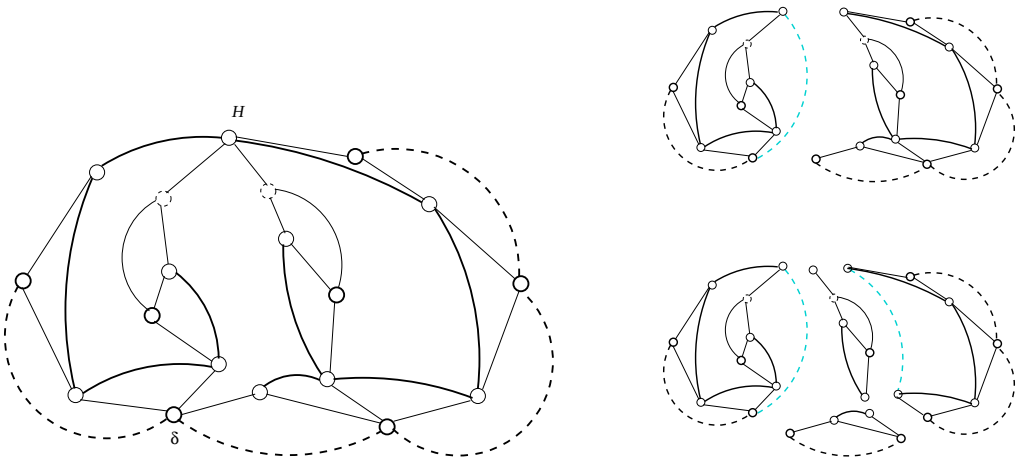


Fig. 12 – Exemples de connexité : à gauche un graphe connexe et non biconnexe, et à droite un graphe biconnexe et non triconnexe (les sommets d'articulation sont en grisé)

Il est facile de se persuader que la biconnexité est une condition nécessaire pour qu'un graphe de contraintes corresponde à un système bien contraint modulo les déplacements. Les définitions précédentes s'étendent pour des degrés de connexité supérieurs :

Définition 5 Dans un graphe non orienté biconnexe G, une paire d'articulation est une paire de sommets dont le retrait déconnecte G. Un graphe connexe qui ne contient pas de paires d'articulation est dit triconnexe : dans un tel graphe, il existe au moins trois chemins distincts entre deux sommets distincts quelconques. □

Dans le cas d'un système bien contraint, la découverte d'une paire d'articulation permet de décomposer le problème en deux parties plus petites qu'on peut résoudre, presque, indépendamment l'une de l'autre. C'est le cas de l'exemple donné à la Fig. 4 et repris en deux parties Fig. 11 : le point H et la droite δ sur cette dernière figure correspondent à la paire d'articulation du graphe de contraintes (*cf.* Fig. 13). La remarque que nous avons faite lors de la résolution de l'exemple s'applique. En effet, l'une des deux composantes (celle de gauche) est sous-contrainte : on ajoute une arête reliant les deux sommets de la paire d'articulation pour la rendre bien-contrainte. Cette arête correspond évidemment à la contrainte imposant la distance de H à δ calculée dans la partie droite du graphe. En poursuivant récursivement cette décomposition sur chacun des sous-graphes obtenus, on obtient des composantes triconnexes (ce ne sont pas tout à fait les composantes triconnexes du graphe initial à cause de l'ajout systématique d'une arête dans l'un des deux sous-graphes). Dans les cas simples, ces composantes correspondent à des cas de triangle, mais il peut arriver qu'on ait des composantes triconnexes qu'on ne sait pas résoudre constructivement et pour lesquelles il faut employer une méthode numérique itérative. Il existe des algorithmes classiques pour trouver efficacement les paires d'articulation d'un graphe (*cf.* par exemple [3]). Moyennant un tel algorithme, la méthode d'Owen donne lieu à l'algorithme décrit à la table 3. Chaque graphe de contraintes de la liste produite par cet algorithme est ensuite résolu constructivement s'il s'agit d'un cas de triangle ou numériquement sinon.



(a) graphe de contraintes.

(b) premiers découpages suivant les paires d'articulation.

Fig. 13 – Paire(s) d'articulation et découpage de graphes

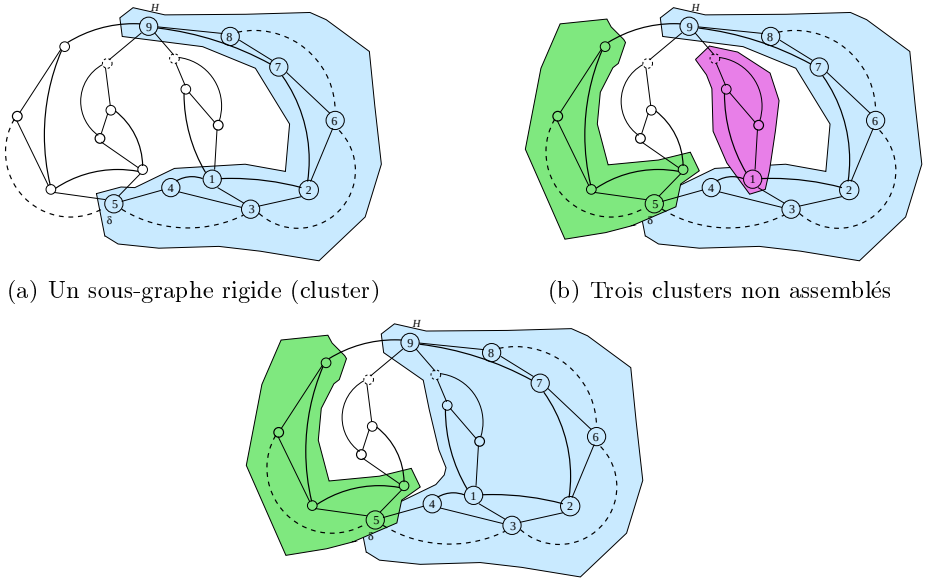
```
Algorithme(décomposition de graphes de contraintes avec la méthode d'Owen)
Données : graphe de contraintes G=(S,A)
Sortie : Liste de sous-graphes à résoudre numériquement
L = liste vide de graphes
P = pile vide de graphes
empiler G sur P
tant que P est non vide  faire
    G' = dépiler P
    chercher une paire d'articulation (s1,s2) de G'
     si cette recherche échoue (le graphe est triconnecté ou simple)
        alors placer G' à la fin de L
        sinon
           (G1, G2) = couper G suivant (s1,s2)
            si G1 est sous-contraint
               alors ajouter une arête entre s1 et s2 dans G1
                           (dont le poids est calculé dans G2)
                     empiler sur P : G2 puis G1
               sinon ajouter une arête entre s1 et s2 dans G2
                           (dont le poids est calculé dans G1)
                     empiler sur P : G1 puis G2
fait
retourner L
```

Tab. 3 – Algorithme d'Owen

4.2. Méthode de Hoffmann

Nous présentons ici une méthode mise au point par C. Hoffmann et son équipe basée sur la découverte des petites composantes rigides ensuite assemblées.

L'idée de base de cette méthode repose sur la propagation de contraintes par chaînage avant (*cf.* section 2). Initialement, une contrainte est choisie, sa résolution permet de définir une figure rigide minimale, comme par exemple deux points à une distance fixée l'un de l'autre. Puis, l'algorithme de propagation de contraintes est utilisé pour trouver le plus grand sous-graphe possible bien contraint modulo les déplacements. Ce graphe est ensuite considéré comme un objet rigide dans le plan (possédant 3 degrés de liberté). Dans la terminologie usuelle, un tel sous-graphe est appelé un *cluster*. Avec l'exemple présenté Fig. 4 et dont le graphe est donné Fig. 13(a), on peut faire de la propagation de contrainte en commençant avec la contrainte de distance en bas à droite pour obtenir un sous-graphe rigide représenté par la partie grisée Fig. 14(a) (les numéros à l'intérieur des nœuds du cluster indiquent l'ordre de parcours).



(a) Un sous-graphe rigide (cluster)

(b) Trois clusters non assemblés

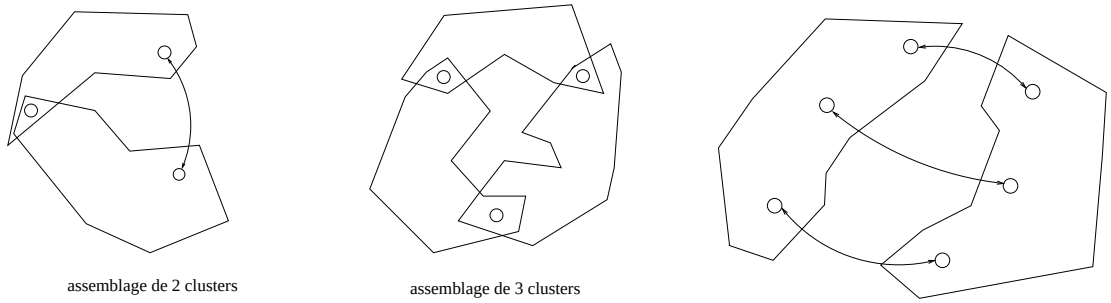
(c) Assemblage des deux premiers clusters

Fig. 14 – Exemples de formation de clusters et assemblage.

Si un cluster recouvre tout le graphe, il n'y a plus rien à faire : l'algorithme de propagation de contraintes a réussi à planifier la construction d'une solution. S'il reste des contraintes, ce processus de formation de clusters est relancé à partir d'une des contraintes non encore utilisée. Par exemple, à la Fig. 14(b), nous avons construit trois clusters dans le graphe de l'exemple 13(a).

Lorsque plusieurs clusters ont été ainsi identifiés, l'algorithme, à un deuxième niveau, procède à l'assemblage des clusters découverts suivant les deux règles illustrées à la figure 15(a) pour former ainsi des clusters plus gros qui pourront à leur tour être assemblés. En reprenant notre exemple, la Fig. 14(c) montre le résultat de l'assemblage des deux clusters

de droite de la Fig. 14(b). Le cluster de gauche peut alors être assemblé avec le cluster de droite en utilisant encore une fois la première règle d'assemblage.



(a) Les deux règles d'assemblage (b) Exemple non résolu

Fig. 15 – Assemblage de clusters

Arrêtons-nous un instant sur ces règles d'assemblage. La première, illustrée par la Fig. 15(a), dit que si C_1 et C_2 sont deux clusters possédant un sommet s en commun et une arête $c = (s_1, s_2)$ avec $s_1 \in C_1$ et $s_2 \in C_2$, alors on peut construire un cluster contenant tous les sommets de C_1 et de C_2. L'interprétation « physique » de cette règle est immédiate et intuitive. En terme de résolution de systèmes de contraintes, plusieurs points de vue peuvent être adoptés. Celui qui nous paraît le plus simple, et que nous avons mis en œuvre consiste à considérer le petit système $\mathcal{S}$ associé aux sommets s, s_1 et s_2 avec comme contraintes celle associée à c et celles qu'on peut déduire du fait que s et s_1 d'une part et s et s_2 d'autre part appartiennent au même cluster. Par exemple, si s et s_1 sont des points, la distance de s à s_1 est connue et calculable dans C_1. Lorsqu'une figure f solution de $\mathcal{S}$ est construite, on peut calculer deux déplacements, l'un, φ_1 permettant d'assembler une solution f_1 correspondant au cluster C_1 avec f, et un autre déplacement φ_2 pour assembler une solution f_2 correspondant au cluster C_2, $\varphi_1(f_1) \cup \varphi_2(f_2) \cup f$ étant alors une solution pour l'assemblage des deux clusters noté $F(C_1, C_2, s, s_1, s_2)$. Le même genre de construction est réalisé pour la deuxième règle produisant l'assemblage noté $H(C_1, C_2, C_3, s_1, s_2, s_3)$.

L'algorithme termine avec succès lorsqu'il arrive à former ainsi un seul cluster impliquant toutes les arêtes du graphe de contraintes, et il termine en échec, s'il n'arrive ni à faire de la propagation de contraintes, ni de l'assemblage de clusters alors qu'il reste des contraintes non utilisées ou plusieurs clusters. Un exemple de situation de mise en échec est présenté à la figure 15(b). Hoffmann et Fudos ont montré que la réussite ou l'échec de cette méthode ne dépend pas du choix des contraintes initiales utilisées pour fabriquer les premiers clusters, et que, par l'emploi de certaines heuristiques, le résultat numérique obtenu en cas de réussite ne dépend pas non plus de ces choix.

L'algorithme découlant de cette méthode est donné à la table 4. Nous y appellerons cluster basique la liste ordonnée des sommets obtenue par propagation de contraintes et cluster composé le résultat de l'assemblage de deux clusters (simples ou composés). Un cluster composé peut ainsi être représenté par un arbre de clusters correspondant à l'assemblage de clusters avec les fonctions F et H. Le résultat de cet algorithme est donc un cluster composé

représenté par un arbre de clusters qui décrit la manière d'assembler les clusters basiques. Cet arbre correspond ainsi à l'ordre des constructions (intra-cluster) et assemblages (inter-clusters) pour calculer une solution au système de contraintes.

```
Algorithme(méthode des clusters)
Données : graphe de contraintes G=(S,A)
Sortie : arbre de clusters basiques
Lc = liste vide de clusters (qui sont des arbres de clusters basiques)
echec = faux
tant que echec = faux et G non vide  faire
    si il existe deux clusters C1 et C2 de L assemblables avec la règle 1
    alors  fabriquer cluster C =  F(C1, C2, s, s1, s2)
           enlever C1 et C2 de L
           ajouter C dans L
    sinon
           si il existe trois clusters C1, C2 et C3 avec la règle 2
           alors fabriquer cluster C =  H(C1, C2, C3, s1, s2, s3)
                enlever C1, C2 et C3 de L
                ajouter C dans L
           sinon
              si G non vide
              alors choisir une contrainte c=(s1,s2) de G
                   L = appliquer la prop. de contraintes à partir de C
                   si L ≠ [s1, s2]  alors ajouter L dans Lc
                                          enlever les sommets de L
                                          ainsi que les arêtes incidentes
                                     sinon echec = vrai
fait
si Lc ne contient qu'un cluster C et que G est vide
   alors retourner C
   sinon signaler un échec
```

Tab. 4 – Algorithme de formation et d'assemblage des clusters basiques

5. D'autres groupes d'invariances

Il est tentant de généraliser l'utilisation de l'invariance par déplacement à d'autres groupes. Les trois premières esquisses proposées à la Fig. 5 illustrent d'une part l'intérêt de considérer l'invariance par similitude et d'autre part la pérennité des notions d'invariance de la section précédente lorsqu'on les étend à d'autres groupes. L'esquisse représentée à la figure 5(d) donne l'exemple d'un système de contraintes où produire une solution passe par la construction de clusters invariants par déplacement et de clusters invariants par similitudes.

Considérons le premier exemple (Fig. 5(a)). Une idée consiste à « oublier » la contrainte de distance : on obtient alors un système ne contenant que des contraintes d'angle qui est

sous-contraint modulo les déplacements et bien contraint modulo les similitudes. Résoudre modulo les similitudes ce nouveau système ne pose aucun problème : on fixe un repère pour les similitudes, par exemple les deux points de la base du triangle, et la construction se fait par simple propagation de contraintes (avec éventuellement utilisation de la règle de l'arc capable). L'ensemble des solutions à ce problème est invariant par similitude. Calculer la similitude qui permet de satisfaire la contrainte de distance initiale donne, à un déplacement près, une solution au problème initial. L'exemple de la Fig. 5(b) se traite de la même manière en oubliant les deux contraintes de distance et en les remplaçant par une contrainte de rapport de distances. Dans ces deux cas, l'algorithme envisageable est fort simple : identifier une contrainte d'échelle, transformer le problème invariant par déplacement en un système invariant par similitude, le résoudre, puis utiliser une similitude pour satisfaire la contrainte d'échelle.

L'exemple 5(c) se traite en étendant aux similitudes la technique de décomposition modulo les déplacements. En effet, en oubliant la contrainte de distance, on trouve un système invariant par similitude dont le graphe peut être décomposé en utilisant les techniques que nous avons vu dans la section précédente : soit en cherchant des paires d'articulation, soit en fabriquant des clusters. Le dernier exemple 5(d), plus compliqué, combine de la transformation de systèmes, de décomposition et de rattrapage de contraintes d'échelle dans plusieurs composantes. Nous en laissons la résolution à la sagacité du lecteur intéressé, l'idéal étant, bien sûr, qu'il en déduise un algorithme général pour ce genre de problèmes.

Conclusion

Ainsi se termine ce petit florilège de problèmes de constructions géométriques qu'on peut rencontrer dans le cadre de la CAO. On pourra avantageusement compléter ce point de vue en consultant des ouvrages sur la théorie de la rigidité [4,2] et des articles sur la résolution de contraintes [1,6] et sur la preuve en géométrie [5].

Nous espérons avoir montré quelques petits problèmes de construction suffisamment intéressants pour aiguiser la curiosité du lecteur en lui laissant entrevoir, au moins dans ce domaine, comment on passe d'une résolution « à la main » à un algorithme plus ou moins général.

Bibliographie

[1] A.S. (2004), Modélisation géométrique sous contraintes, *Action Spécifique du CNRS* , `http ://www.esil.univ-mrs.fr/mdaniel/gtmg/as-contraintes/index.html`

[2] Claude Berge (1958), Théorie des graphes et ses Applications, *Dunod*.

[3] Thomas H. Cormen, Charles E. Leiserson & Ronald L. Rivest (1990), Introduction à l'algorithmique, *Dunod*.

[4] Jack E. Graver, Brigitte Servatius & Herman Servatius (1993), Combinatorial Rigidity, Graduate Studies in Maths, Vol. 2, *American Mathematical Society*.

[5] Jean Mainguené & Marie-Françoise Roy (1999), *Démonstration automatique en géométrie : une approche par la géométrie analytique*, Bulletin de l'APMEP **421**, 177–188.

[6] Minime-IGG (2004), *Publications de l'équipe IGG concernant la résolution de contraintes en géométrie*, http ://axis.u-strasbg.fr/∼schreck/recherche.htm.

Pascal Schreck, Pascal Mathis, Arnaud Fabre
Équipe Informatique et Graphique du
Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection
UMR 7005, CNRS-ULP Strasbourg
schreck@dpt-info.u-strasbg.fr