Au sujet de certaines contraintes globales

Diego Olivier Fernandez Pons [1,2]

[1] Université Pierre et Marie Curie - Paris VI, 5 place Jussieu 75005 Paris
[2] ILOG S.A, 9 rue de Verdun 94253 Gentilly Cedex
diego-olivier.fernandez-pons@cicrp.jussieu.fr dofpons@ilog.fr

Résumé

Alors que les algorithmes de recherche opérationnelle permettant de résoudre certains problèmes classiques tel le chemin, l'arbre ou le couplage de coût minimal sont connus depuis plusieurs décennies, les contraintes globales correspondantes sont relativement récentes.

Ainsi Sellmann n'a-t-il montré comment réaliser une contrainte globale de plus court chemin qu'en 2003 tandis que le couplage a été résolu en 1999 par Régin après des tentatives de Focacci et al. (au moyen des coûts réduits) ou Laburthe et Caseau (au moyen de l'algorithme hongrois). Quant à l'arbre couvrant, il n'apparaît que marginalement dans un travail de Aron et Van Hentenryck en 2002.

Toutes ces contraintes globales peuvent être dérivées d'algorithmes d'énumération en ordre des coûts croissants dus à Eppstein, Murty ou Gabow. Le propos de cet article est de monter le lien entre ces algorithmes classiques de recherche opérationnelle et les contraintes globales correspondantes.

Abstract

Although algorithms to solve classical problems like minimum cost paths, spanning trees or matchings have been known for several decades, their corresponding global constraints are quite recent.

Sellmann introduced the shortest path constraint as late as 2003 while the matching case was solved by Régin in 1999 after several attempts by Focacci and al. (using reduced costs) or Laburthe and Caseau (using the hungarian algorithm). The minimum spanning tree constraint only appears marginally in a work by Aron and Van Hentenryck in 2002.

However, these global constraints can be obtained from cost-increasing enumeration algorithms of Eppstein, Murty and Gabow. This article aims to show the relation between classical operations research algorithms and global constraints.

1 Introduction

Il est désormais établi que l'une des composantes clé dans l'efficacité des méthodes de résolution de problèmes combinatoires en programmation par contraintes est l'utilisation de *contraintes globales*, et que ces contraintes sont elles mêmes basées sur des algorithmes de recherche opérationnelle.

Ainsi depuis *alldiff*, la première contrainte globale basée sur un algorithme de couplage [15], d'autres contraintes globales notamment pour les problèmes valués ont été proposées, par exemple pour les problèmes de flot à coût minimal et leur dérivés par Régin [16] ou pour des problèmes de plus court chemin par Sellmann [17]. Par ailleurs dans leur étude d'un problème d'arbres robustes dans des graphes d'intervalles [1], Aron et van Hentenryck ont montré - sans toutefois l'identifier précisément - qu'il était possible de mettre au point une contrainte globale pour l'arbre couvrant de coût minimal.

D'un autre côté, en recherche opérationnelle afin de résoudre des problèmes légèrement contraints il est d'usage de faire appel à des procédures d'énumération des solutions en ordre croissant de coût : on peut par exemple énumérer tous les chemins entre deux noeuds d'un graphe jusqu'à en obtenir un qui vérifie toutes les contraintes additionnelles voulues, ce chemin sera alors optimal. Cette méthode se retrouve communément dans des sous-problèmes de problèmes d'optimisation plus complexes par exemple pour réduire des sauts d'intégrité, générer des colonnes pour un problème maître ou étudier la sensibilité aux contraintes de la solution optimale. Eppstein [5] donne divers exemples d'applications de sa procédure d'énumération des k-plus courts chemins entre deux sommets d'un graphe.

Ces deux méthodes s'intéressent donc à des pro-

blèmes semblables mais sous des angles différents : la première se focalise sur la gestion des contraintes additionnelles la seconde mise sur l'optimalité. Il nous a dès lors semblé opportun de les rapprocher.

Nous nous efforcerons d'éclaircir les liens existants entre ces deux familles d'algorithmes et montrerons que les contraintes globales de Sellmann, Régin ou Van Hentenryck peuvent être obtenues à partir d'algorithmes d'énumération dus à Eppstein [5], Gabow [9] et Murty [14] (revu par Fussenegger et Gabow [8]). Ce faisant, nous revisiterons la notion de *coût réduit* mise en avant par Focacci et al. que nous comparerons à d'autres notions proches. Nous introduirons la notion de *coût de remplacement* et montrerons ses liens avec des problèmes classiques de la recherche opérationnelle tel le *most vital arc*, esquissant ainsi des pistes pour la recherche d'algorithmes efficaces pour d'autres contraintes globales.

2 La notion de coût de remplacement

En programmation par contraintes, la coopération des différents algorithmes de propagation est possible grâce à la mutualisation des informations dont ils disposent. Ce sont les *domaines* qui ont pour mission de de centraliser les valeurs permises et interdites pour chaque variable. Pour les problèmes de satisfaction, cette information est en quelque sorte maximale car le critère d'acceptation final est binaire (l'instantiation proposée est ou n'est pas admissible).

Pour les problèmes d'optimisation par contre, il s'agit d'une information assez pauvre. Pourtant les contraintes globales d'optimisation existantes quantifient déjà la variation du coût de la solution lors de l'ajout ou le retrait d'arcs, mais contrairement aux domaines, cette information n'est pas externalisée. Elle ne peut dès lors être utilisée par d'autres éléments (contraintes, heuristiques) lors du processus de résolution. Seuls Focacci et al. [7] ont proposé de laisser accessibles les coûts réduits de chaque affectation variable-valeur en vue d'améliorer ultérieurement leur formulation linéaire avec des coupes supplémentaires ajoutées pendant la recherche.

Notre premier objectif est d'identifier une information variationnelle relative au coût qui serait intéressante à mutualiser.

2.1 Les coûts réduits en PPC

Les coûts réduits sont une notion attachée à la programmation linéaire ; intuitivement il s'agit de la dérivée directionnelle de la fonction objectif le long des arêtes du polyèdre défini par les équations linéaires. Quand on se trouve en un point extrême du simplexe, si le coût réduit associé à une direction est négatif, on a intérêt à se diriger dans cette direction pour faire décroître le coût global de la solution (méthode de descente de gradient). Inversement, si aucune direction n'a de dérivée négative, on se trouve en un minimum local, et par convexité, en un minimum global.

Focacci et al. considèrent une formulation linéaire (d'une relaxation) du problème dans laquelle $x_{kv} \in \{0,1\}$ représente l'affectation de la valeur v à la variable x_k. A l'optimum du problème linéaire, le coût réduit $\overline{c}_{kv}$ est une borne inférieure de l'augmentation de la fonction objectif quand on affecte v à x_k ($x_{kv} = 0 \rightarrow x_{kv} = 1$) en vertu de l'inégalité :

$$\Delta f \geq \frac{\partial f}{\partial \vec{kv}} \Delta x_{kv} \qquad \Delta x_{kv} = 1$$

Le schéma général de la méthode proposée par Focacci est le suivant :

1. Considérer une relaxation du problème pour laquelle on sache calculer à la fois une solution optimale $\underline{C}$ et les coûts réduits correspondants $\overline{c}$ en un temps polynomial
2. Associer à chaque couple variable-valeur (x_k, v) son coût réduit $\overline{c}_{kv}$
3. Eliminer les affectations dont l'instanciation ferait dépasser la meilleure solution connue $\underline{C} + \overline{c}_{kv} \geq \overline{C}$
4. Réitérer jusqu'au point fixe

Cette approche requiert donc d'encapsuler un solveur linéaire dans une contrainte globale [7] mais elle peut être spécialisée à des cas où on sait obtenir les coûts réduits par des algorithmes combinatoires (couplage, arborescence) [6].

Cependant, même pour des cas particuliers dans lesquels le problème linéaire possède la propriété d'intégrité (couplage, arborescence), les bornes calculées ne sont pas exactes. Autrement dit l'estimation $\underline{C} + \overline{c}_{kv}$ peut s'avérer très inférieure à la valeur de la solution de coût minimal telle que $x_k = v$. Ainsi, seule la méthode de Régin [16] donne les bornes exactes pour le couplage.

2.2 Le coût de déviation d'Eppstein et autres variantes classiques

Les algorithmes d'énumération en ordre des coûts croissants rencontrés en recherche opérationnelle utilisent une notion proche des coûts réduits, appelée *coût de déviation* (sidetracking cost) par Eppstein.

Etant donné un graphe G, deux sommets s et t du graphe, le plus court chemin p entre s et t, et un arc

(i,k) avec $i \in p$ et $k \notin p$, on appelle coût de déviation par l'arc (i,k) le surcoût engendré par le fait d'empruter l'arc $(i,k) \notin p$ plutôt que l'arc "théoriquement optimal" $(i,j) \in p$

$$\overline{c}_{ij} = c_{ij} + d(j,t) - d(i,t)$$

où $d(i,t)$ est le plus court chemin entre i et t et c_{ij} le coût de l'arc (i,j)

On rencontre une notion sembable mais symétrique dans l'algorithme d'énumération des arbres couvrants de Gabow (T-exchange cost) : si on désire ôter l'arc $e \in T$ de l'arbre couvrant de coût minimal T, pour maintenir la propriété de connexité il va falloir le remplacer par un arc f formant un cycle dans T, contenant e. Alors $T \backslash e \cup f$ est un arbre de coût $c(T)+c(f)-c(e)$. L'algorithme associe donc à chaque arc de T son coût d'échange minimal :

$$\overline{c}_e = \min_f [c(f) - c(e)]$$

Les mêmes idées sont exploitées par Fussenegger et Gabow dans leur algorithme d'énumération des couplages : étant donné un couplage de coût minimal $\mathcal{M}$, e une arête de $\mathcal{M}$ et $\mathcal{N}$ le couplage de coût minimal ne contenant pas e, alors $\delta_e = c(\mathcal{N}) - c(\mathcal{M})$ est le surcoût engendré par le "refus" d'utiliser l'arête e.

Même si ces notions coïncident numériquement avec les coûts réduits pour certains problèmes (chemins, arbres), elles ont un fondement différent. Les coûts de déviation sont des notions discrètes, obtenues par différence de coût de solutions. Le coût réduit est une notion continue liée à la dérivation.

2.3 Les coûts de remplacement

Les remarques précédentes conduisent à définir ce que nous estimons être des information intéressantes dans le cadre d'une contrainte globale d'optimisation.

On considère un problème d'optimisation P dont on notera $OPT(G)$ le coût de la solution optimale sur le graphe G. Pour tout arc a de G on définit les fonctions :

$$\delta^+(a) = OPT_{+a}(G) - OPT(G)$$

$$\delta^-(a) = OPT(G \backslash a) - OPT(G)$$

Ces fonctions indiquent le surcoût encouru quand on décide d'imposer ou d'interdire un arc a des solutions du problème concerné. Les fonctions δ^+ et δ^- ont la particularité de caractériser les solutions du problème d'optimisation associé comme nous le verrons dans la section suivante.

A l'instar des coûts réduits, δ^+ est nulle pour tous les arcs appartenant à la solution optimale et les coûts réduits en sont une borne inférieure. On peut considérer δ^+ comme une extension au cas discret de la notion de coût réduit.

Inversement, δ^- est toujours nulle pour les arcs n'appartenant pas à la solution optimale. L'arc qui maximise δ^- est l'arc le plus vital (the most vital arc). La recherche des arcs les plus vitaux est une problématique issue des applications militaires de la recherche opérationnelle ; c'est le problème dans lequel un ennemi cherche à détruire des arcs pour ralentir le plus possible un convoi par exemple. La plupart des variantes de ce problème (destruction des arcs ou augmentation de leur coût, contrainte de ressources, etc.) sont NP-difficiles. Des algorithmes efficaces pour le calcul de l'arc le plus vital existent pour les problèmes usuels (chemins, arbres, flots).

Il convient enfin de noter que δ^- correspond aussi à la notion de regret en programmation par contraintes. L'heuristique consistant à imposer l'arc maximisant le regret dans le cadre du couplage a déjà été experimentée par Caseau et Laburthe [3] par exemple.

3 Caractérisation des solutions d'un problème d'optimisation

3.1 Exemple : le couplage

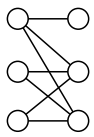


FIG. 1 – Graphe initial

Considérons le graphe biparti de la figure 1. Un algorithme de couplage appliqué à ce graphe trouverait l'une quelconque des solutions de la figure 2. Une contrainte globale au contraire ne doit pas trouver une des solutions mais caractériser l'ensemble des solutions du problème de couplage, autrement dit déterminer les arcs *nécessaires* ou *impossibles* dans toute solution. Ainsi, la contrainte globale de Régin [15] va-t-elle calculer les informations de la figure 3 de sorte à ne laisser "libres" que les arcs tels qu'il existe des solutions les contenant et des solutions ne les contenant pas.

Notons que pourvu que l'on dispose d'un algorithme polynomial résolvant le problème de décision (exite-t-il un couplage dans le graphe ?) de complexité *opt*, il est possible de caractériser l'ensemble des solutions du

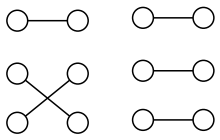


FIG. 2 – Couplages du graphe

problème en $2m \times opt$ par une procédure de double-négation : ôter un arc a du graphe, s'il n'existe plus de solution dans le graphe restant $G \backslash a$ alors qu'il y en avait dans le graphe G, on en déduit que toutes les solutions utilisent l'arc a (idem en imposant l'arc a après réduction du graphe - ce qu'il faut être également en mesure de faire).

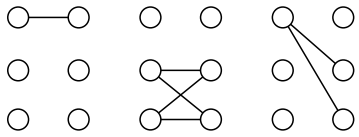


FIG. 3 – Arcs nécessaires, libres et impossibles

Enfin la contrainte globale de Régin pour le couplage atteint la complexité théorique minimale de *opt*. En effet, s'il existe une seule solution au problème, alors l'algorithme de caractérisation doit signaler que tous ses arcs sont nécessaires. On ne peut donc caractériser l'ensemble des solutions en un temps inférieur à celui qu'il faut pour trouver une seule d'entre elles.

3.2 Cas des problèmes valués

Les fonctions δ permettent de caractériser les solutions d'un problème d'optimisation lorsque l'on connaît une borne supérieure B du coût : on note $\underline{c}$ le coût d'une solution optimale du problème

1. $\delta^+(a) \leq (B - \underline{c})$ ssi il existe une solution du problème empruntant l'arc a et de coût inférieur à B
2. $\delta^-(a) \leq (B - \underline{c})$ ssi il existe une solution du problème n'empruntant pas l'arc a et de coût inférieur à B

Autrement dit, les fonctions δ^+ et δ^- permettent de partitionner l'ensemble des arcs en arcs nécessaires, libres et impossibles, tout comme le faisait la contrainte *alldiff*.

Comme précédemment la connaissance d'un algorithme d'optimisation de complexité *opt* permet le calcul de ces fonctions en un temps $2m \times opt$. Le but est de réduire autant que possible cette complexité.

4 Algorithmes d'énumération croissante

L'intérêt pour algorithmes d'énumération croissante dans la perspective des contraintes globales trouve sa source en ce que la connaissance des fonctions δ^+ ou δ^- permet d'énumérer en ordre croissant les solutions des problèmes correspondants. Cela se fait au moyen d'un schéma algorithmique classique dû à Lawler [12], lui même généralisation de l'algorithme d'énumération des couplages de Murty [14].

L'idée de l'algorithme de Lawler est de partitionner l'espace des solutions du problème P considéré en classes d'équivalence. Pour chaque classe d'équivalence seront calculés un représentant et une borne inférieure du coût de l'ensemble des solutions appartenant à cette classe. Les représentants et leurs évaluations sont introduits dans une file de priorité.

Itération centrale de l'algorithme :

1. extraire le représentant le plus prometteur (borne inférieure minimale) et sa classe d'equivalence
2. générer à partir de ce représentant une solution
3. partitionner la classe d'équivalence (privée de la solution générée) et calculer pour chaque sous-classe un représentant et une borne inférieure
4. insérer les nouvelles classes d'équivalence avec leurs représentants respectifs et leurs bornes dans la file de priorité

Le bon déroulement de la procédure de Lawler dépend de la qualité des bornes inférieures, de la possibilité de toujours générer une solution à partir d'un représentant, d'écarter cette solution de la classe extraite et de la partitionner en un nombre réduit de nouvelles classes.

Hamacher et Queyranne [10] ont poursuivi ces travaux et montré qu'il suffisait de savoir calculer la meilleure et la seconde meilleure solution de toute partition de la forme $\mathcal{P}_k = \{S \mid Req \subseteq S \subseteq E \backslash Exc\}$ où Req et Exc sont des ensembles d'arcs requis et exclus. Or pour peu que les solutions optimales du problème considéré ne puissent pas se contenir les unes les autres (par exemple les arbres ou les plus courts chemins simples mais pas les plus courts chemins quelconques) le coût c_2 de la seconde meilleure solution est $c_2 = c(opt) + \min_{e \in opt} \delta^-(e)$

De ce fait, tous les algorithmes d'énumération croissante qui vérifient la propriété de non inclusion comme les plus courts chemins sans boucles [11], les arbres couvrants [9], les couplages [8] [4], les arborescences [2] sont construits à partir de la fonction δ^-. Leur complexité est alors de $K \times opt$ où *opt* est la complexité du problème de base. Par contre, l'algorithme d'Eppstein

[5] pour les plus courts chemins avec boucles est basé sur la fonction δ^+ pour une complexité de $K + opt$.

5 Quelques contraintes globales

Nous allons à présent retrouver les contraintes globales d'arbre, de couplage et de plus court chemin à partir de la notion de coût de remplacement et de théorème de structure.

5.1 Les arbres couvrants

Seuls Aron et Van Hentenryck [1] semblent s'être intéressés au problème d'arbre couvrant. C'est pourtant l'une des contraintes globales les plus naturelles en raison de la simplicité de ses théorèmes de structure.

On appelle *théorème de structure* un théorème caractérisant l'ensemble des solutions d'un problème - en général par rapport à une solution donnée.

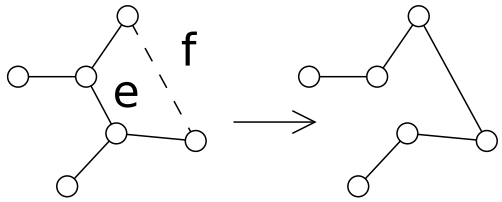


FIG. 4 – Echange d'arêtes

Théorème de structure des arbres couvrants : à partir d'un arbre couvrant T d'un graphe G, on peut atteindre tout autre arbre couvrant par une suite d'*échanges admissibles* d'arêtes de la forme (e, f) où $e \in T$ et $f \notin T$ est une arête telle que le cycle créé par f dans T contienne e (figure 4)

Théorème de structure version pondérée [Gabow 1977] : Soit T l'arbre couvrant minimal d'un graphe G.

1. On suppose donné $e \in T$, soit f la plus petite arête telle que (e, f) soit un échange admissible, alors $T \backslash e \cup f$ est l'arbre couvrant minimal de $G \backslash e$.
2. On suppose donné $f \notin T$, soit e la plus grande arête telle que (e, f) soit un échange admissible, alors $T \backslash e \cup f$ est l'arbre couvrant contenant f de coût minimal.

5.1.1 Calcul des coûts de remplacement

Les théorèmes de structure indiquent comment calculer les coûts de remplacement δ^+ et δ^-. Gabow en 1977 parvenait déjà à calculer δ^- en une complexité de $m \log m + m\alpha(m, n)$ (où α désigne l'inverse de la fonction d'Ackermann). D'autres auteurs ont réduit encore le temps et l'espace nécessaires par des techniques algorithmiques assez sophistiquées.

Etude d'un exemple

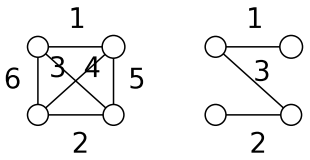


FIG. 5 – Graphe initial et arbre de coût minimal

La figure 5 montre un graphe et son arbre couvrant de coût minimal, la figure 6 les fonctions δ^+ et δ^- associées.

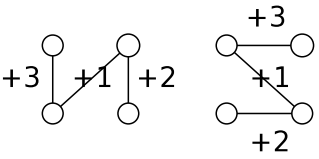


FIG. 6 – fonctions δ^+ et δ^-, les arêtes non tracées ont un δ nul

Dans la figure 6, les arêtes encadrées sont celles qui appartiennent à l'arbre couvrant minimal. Une flèche supérieure entre e et f signifie "si $e \in T$ est interdit, il faut le remplacer par $f \notin T$ pour un surcoût de $\delta^-(e) = c(f) - c(e)$". De même, une flèche inférieure entre e et f signifie "si $e \notin T$ est imposé, il faut ôter $f \in T$ pour un surcoût de $\delta^+(e) = c(e) - c(f)$". Dans les deux cas nous dirons que f est le support de e.

Le support de e est l'arête libre de coût minimal/maximal formant un cycle avec e. Comme les arêtes sont ordonnées par coûts croissants, il suffit de balayer le tableau de façon ascendante ou descendante selon le cas, jusqu'à trouver un arc formant un cycle avec e (il s'agit d'une variante de l'algorithme de Kruskal). Le calcul des coûts de remplacement se fait sur les arcs libres : l'existence d'un echange admissible (e, f) signifie que s'il l'on interdit e, on va pouvoir utiliser f à la place, ce qui suppose que e n'est pas nécessaire et que f n'est pas interdit (et symétriquement). Les figures 6 b) et c) montrent l'évolution des supports quand l'arc 3 est imposé puis interdit.

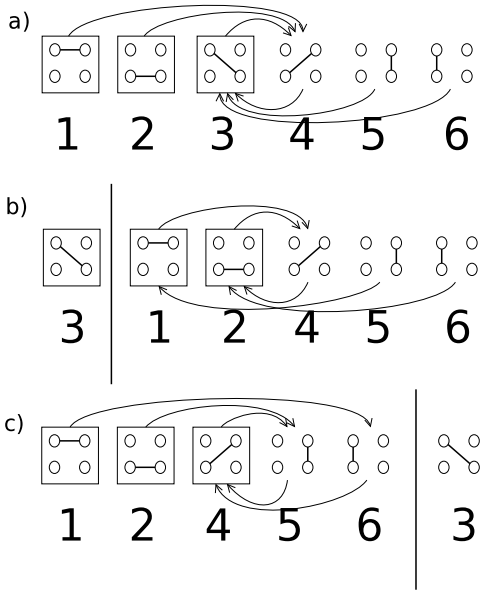


FIG. 7 – supports quand tous les arcs sont libres, quand 3 est requis et quand 3 est interdit

5.1.2 Liens entre δ^+ et δ^-

Informellement, le lien entre δ^+ et δ^- provient de la remarque que si l'on ôte un arc e d'une solution, pour retrouver une autre solution il va falloir ajouter au moins un arc dans un ensemble d'arcs de remplacement R, or δ^+ mesure le coût d'ajout des l'élements de R.

Plus précisément, si l'on a $[x_e = 0] \Rightarrow [\exists k \in R, \mid x_k = 1]$ on peut en déduire que $\delta^-(e) \geq \min_{k \in R} \delta^+(x_k)$

Les raisons pour lesquelles l'inégalité pourrait être stricte sont :

1. L'ensemble R est plus grand que nécessaire. Il convient de noter que pour la formulation PLNE du problème, R définit une inégalité de clique $\sum_{R \cup e} x_k = 1,\ x_k \in \{0,1\}$. Cela donne une indication sur la difficulté de calculer R de manière exacte dans le cas général
2. Les valeurs δ^+ sur R ne sont pas exactes (par exemple elles supposent que $x_e = 0$, ce sera le cas pour la contrainte de plus court chemin)

Dans le cas de l'arbre couvrant, si on note $m \in T$ l'arête de coût maximal de T alors toute arête $e \notin T$ telle que le cycle formé par e dans T contienne m aura m pour support (car m est l'arête de coût maximal de T donc à fortiori du cycle formé par e dans T). Ces arêtes sont exactement celles de la coupe formée par la

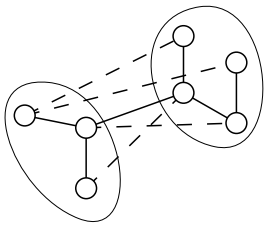


FIG. 8 – Toutes les arêtes de la coupe ont l'arête transversale pour support

suppression de m (figure 8), et pour toutes ces arêtes $\forall e \in coupe(S_m, \overline{S}_m),\ \delta^+(e) = c(m) - c(e)$.

Inversement, si on interdit l'arête m, on ne peut la remplacer que par une arête de la coupe et en particulier par celle qui minimise $c(m) - c(e)$, donc $\delta^-(e) = \min_{e \in coupe(S_m, \overline{S}_m)} \delta^+(e)$

La procédure peut être répétée récursivement sur les sous arbres créés par la coupe. C'est cette structure inductive qui permet d'aboutir à des algorithmes quasi-linéaires - de complexité $m \log m + \theta(m)$ où $m \log m$ correspond au tri des arcs et $\theta(m)$ à la gestion de structures de données de type *union-find*.

5.1.3 Voisinage de la solution optimale

Les coûts de remplacement évaluent la déviation entre l'arbre couvrant de coût minimal $T_{\min}$ et un arbre contenant une arête a donnée (en particulier cet arbre ne diffère de l'arbre minimal que par un échange d'arêtes). Il s'agit en quelque sorte d'une caractérisation du voisinage de $T_{\min}$. Pour le cas de plusieurs arêtes on a des inégalités souvent strictes :

$$c(T_{\min}) + \textstyle\sum_{a \in A} \delta^+(a) \leq \min \{\, c(T) \mid A \subseteq T \,\}$$

$$c(T_{\min}) + \textstyle\sum_{a \in A} \delta^-(a) \leq \min \{\, c(T) \mid T \subseteq E \backslash A \,\}$$

Les inégalités deviennent des égalités quand les éléments de A ont des supports disjoints.

5.2 Les couplages

Le cas des couplages a été déjà traité extensivement par Régin [16], nous ne revenons que très succintement à ce sujet.

Le *graphe résiduel* d'un couplage M sur un graphe biparti (G, I, J) est un graphe contenant tous les sommets I et J tel que :

1. A toute arête (i, j) du graphe G apparaissant dans le couplage M correspond un arc (j, i) du graphe résiduel, valué par l'inverse de son coût $-c_{ij}$

2. A toute arête (i, j) du graphe G n'apparaissant pas dans le couplage M correspond un arc (i, j) du graphe résiduel, valué par son coût c_{ij}

1	5	7
6	5	9
2	4	3

0	4	5
1	0	3
0	2	0

TAB. 1 – Matrice de coûts et matrice des coûts réduits à l'optimum

La propriété fondamentale du couplage est l'invariance des solutions par une famille de "translations" :

1. soustraction d'une même valeur à toute une ligne de la matrice des coûts
2. soustraction d'une même valeur à toute une colonne de la matrice des coûts

Ainsi la table 1 montre la matrice des coûts d'un graphe complet et sa matrice de coûts réduits qui est un extremum pour les transformations ci-dessus décrites : on ne peut faire aucune transformation supplémentaire car toutes les lignes et les colonnes contiennent au moins un zéro. Comme les solutions restent inchangées par ces transformations, on en déduit immédiatement l'existence de d'une solution optimale $(1, 1)(2, 2)(3, 3)$ car les coûts réduits correspondants sont tous nuls, donc leur somme minimale.

Ces opérations de "translation" reçoivent une justification beaucoup plus formelle dans le cadre de la théorie de la dualité linéaire qui les relie à la notion de coût réduit.

Théorème de structure des couplages [Berge 1957] : à partir d'un couplage M d'un graphe G, on peut atteindre tout autre couplage N en inversant les arcs le long d'un cycle du graphe résiduel. De plus la différence de coût entre les couplages correspond à la somme des coûts le long du cycle du graphe résiduel :

$$C(N) = C(M) + \sum_{ij \in \mathcal{C}} \overline{c}_{ij}$$

Le théorème de structure ayant une formulation relative (différence de coût de deux solutions), il est également invariant par "translation". On peut donc y remplacer les coûts par les coûts réduits. La figure 9 montre le graphe résiduel associé à la solution de la figure 10, valué par les coûts réduits. La seconde solution de la figure 10 est celle obtenue par inversion des arcs le long du cycle $x_3y_1x_1y_3x_3$ de coût 5.

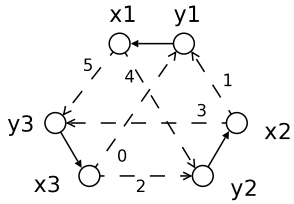


FIG. 9 – graphe résiduel associé au couplage de coût minimal

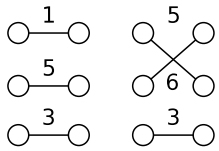


FIG. 10 – Solution optimale du problème de couplage de la table 1 et solution obtenue par inversion des arcs dans le graphe résiduel le long du cycle $x_3y_1x_1y_3x_3$ de coût 5

5.2.1 Calcul des coûts de remplacement

Le coût de remplacement d'un arc (i, j) correspond au coût du plus petit circuit dans le graphe résiduel contenant (i, j). Il peut donc se calculer par un algorithme de plus court chemin en une complexité de n^4. Régin remplace ensuite les coûts du graphe résiduel par les coûts réduits, obtenant ainsi un graphe à composantes toutes positives (les arcs inversés de coûts négatifs sont ceux appartenant à une solution or les coûts réduits des arcs d'une solution sont nuls). On peut ainsi utiliser un algorithme de plus court chemins de complexité moindre (n^3). Le calcul des coûts réduits est cubique par l'algorithme hongrois.

Fussenegger et Gabow ont réduit en 1977 la complexité de l'algorithme d'énumération des couplages de Kn^4 (Murty 1968) à Kn^3. Cette réduction était cependant uniquement due au mécanisme d'énumération utilisé par Murty lequel dans ce cas précis s'avérait inefficace. C'est le nouveau mécanisme d'énumération - également utilisé par Gabow pour l'arbre couvrant - qui a été généralisé par Hamacher et Queyranne. Dans l'optique du rapprochement avec les contraintes globales, nous nous baserons cependant sur les travaux de Fussenegger et Gabow car plus structurés que ceux de Murty.

Etant donné un couplage M, on appelle $M(x)$ le noeud image de x par M, autrement dit y tel que (x, y) appartienne à M. Nous appelons *Graphe de Gabow* le graphe contenant les seuls sommets x du graphe ré-

siduel et tel qu'à tout arc (x_i, y_j) du graphe résiduel corresponde un arc $(x_i, M(y_j))$ de même coût. Intuitivement un arc (i, j) dans le graphe de Gabow signifie "dans le couplage initial on avait $(i, M(i))$ et $(j, M(j))$, dans le nouveau couplage on aura $(i, M(j))$ et $(j, \ldots)$" autrement dit i prend l'image de j.

Lemme [Fussenegger et Gabow 1977] : Soit M un couplage d'un graphe G de coût minimal, soit (x, y) un arc de ce couplage et $\mathcal{C}$ le plus petit cycle du graphe de Gabow contenant $(x, M(x))$. Alors le couplage obtenu par inversion des arcs le long du cycle $\mathcal{C}$ est le couplage du graphe G de coût minimal ne contenant pas l'arc (x, y).

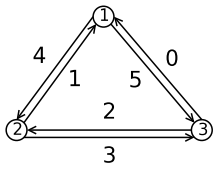


FIG. 11 – Graphe de Gabow correspondant au graphe résiduel de la figure 9

Ce lemme ramène le calcul de δ^- à un calcul de plus courts chemins en n^4 puis n^3 en utilisant les coûts réduits. Comme on obtient le coût du second meilleur couplage en calculant $\min_{e \in M} \delta^-(e)$, Fussenegger et Gabow parviennent à un algorithme d'énumération des K couplages de coût minimum en Kn^3.

5.3 Les plus courts chemins

Sellmann a introduit une contrainte globale d'optimisation pour le problème du plus court chemin entre deux noeuds s et t d'un graphe (orienté dans notre cas). Nous montrerons ses liens avec les travaux d'Eppstein sur l'énumération des chemins avec boucle.

5.3.1 Remarques préalables

Le cas des plus court chemins est un peu particulier : dans la section 3 nous signalions qu'il suffisait de savoir "réduire le graphe après avoir imposé un arc a" pour appliquer la méthode générale de double-négation et calculer $\delta^+(a)$. Pour les plus courts chemins quand l'arc a est de la forme (s, i) où s est le point de départ du chemin, il suffit de remplacer i par s. Dans le cas général, imposer des arcs à un plus court chemin rend ce problème NP-difficile. On ne peut dès lors espérer calculer δ^+ de manière exacte que pour les arcs de type (i, j) avec $i \in p$ où p est le plus court chemin entre s et t.

Plutôt que le graphe δ^+, nous utiliserons le graphe des coûts de déviation d'Eppstein (graphe réduit). Ces deux graphes coïncident pour les arcs de la forme (i, j) avec $i \in p$. Le graphe réduit est le "translaté préfixe" du graphe δ^+ en le sens que ses valeurs sont indépendantes du chemin utilisé pour arriver en un point :

propriété du graphe réduit : soit p un chemin partant de s et parvenant à un noeud i, alors tout chemin prolongeant p jusqu'à t et empruntant l'arc (i, j) a un coût supérieur à $c(p) + \overline{c}_{ij}$.

propriété du graphe δ^+ : tout chemin entre s et t empruntant l'arc (i, j) a un coût supérieur à $\delta^+(i, j)$

Ainsi, quand on impose les arcs d'un chemin entre s et i, $\overline{c}_{ij}$ ne varie pas contrairement à $\delta^+(i, j)$. La conséquence en est l'obtention de propriétés plus fortes au moyen d'une complexification significative des algorithmes. L'algorithme d'énumération d'Eppstein pour déterminer les k plus courts chemins a par exemple une complexité de $k + m \log n$ (on note que k n'est pas en facteur contrairement aux algorithmes d'énumération précédents).

Lemme [Eppstein] : Soit sp le plus court chemin entre deux points s et t d'un graphe et p un second chemin joignant ces deux noeuds.

$$\sum_{ij \in p} \delta^+_{ij} = c(p) - c(sp)$$

On pourra comparer avec l'inégalité correspondante pour les arbres couvrants

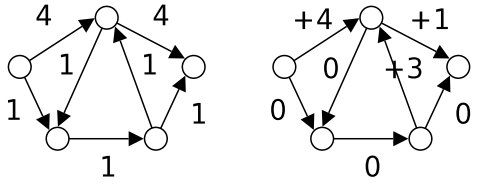


FIG. 12 – Graphe initial et graphe réduit

5.3.2 Calcul des coûts de remplacement

Le graphe réduit se calcule en un temps $m + n \log m$ au moyen de la formule d'Eppstein :

$$\overline{c}_{ij} = c_{ij} + d(j, t) - d(i, t)$$

et d'un arbre inverse des plus courts chemins (pour les distances $d(_, t)$.

La fonction δ^+ se calcule à partir du graphe réduit en $m + n \log m$ par la formule

$$\delta^+(i, j) = d(s, i) + \overline{c}_{ij}$$

au moyen d'un algorithme de plus courts chemins de source s (pour les distances $d(s, _)$)

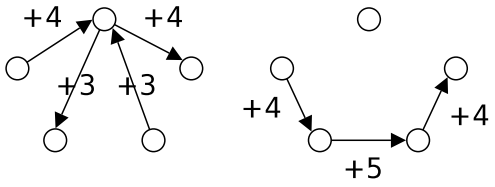


FIG. 13 – δ^+ et δ^-

Le plus court chemin est un exemple où le calcul de la fonction δ^- à partir de δ^+ donne des bornes inférieures strictes. Un argument semblable à celui utilisé pour les arbres couvrants permet de montrer que

$$\delta^-(a) \geq \min_{ij \in coupe(S_a, \overline{S}_a)} \{ d(s,i) + c_{ij} + d(j,t) \} - d(s,t)$$

Où $(S_a, \overline{S}_a)$ est la coupe créée par la suppression de a dans l'arbre des plus courts chemins d'origine s. Cependant, dans le cas des graphes orientés, l'inégalité peut être stricte si le plus court chemin entre j et t de coût $d(j,t)$ passe justement par (i,j)

La fonction $\delta^-(a)$ se calcule en $n(m + n \log m)$ par une application directe de la formule. Une borne inférieure $\min \overline{c}_{ij}$ avec i sur le chemin préfixe de a peut être calculée en temps linéaire à partir du graphe réduit. La figure 13 montre qu'il s'agit bien d'une borne inférieure car la procédure décrite aurait trouvé $+3$ ou $+4$ pour des arcs où la valeur exacte est $+4$ et $+5$.

6 Renforcement des bornes

Deux notions variationnelles qui permettent d'estimer l'incrément du coût lors de la prise de certaines décisions (branchements) pendant l'exploration de l'espace de recherche :

1. Pour les problèmes linéaires il s'agit des *coûts réduits*. Ils sont efficacement calculés par l'algorithme du simplexe
2. Pour les problèmes combinatoires polynomiaux il s'agit des *coûts de remplacement* et pour un certain nombre d'entre eux nous sommes déjà en mesure de les calculer efficacement

Il est courant en programmation linéaire en nombres entiers d'utiliser les coûts réduits pourtant définis pour le seul cas continu, comme estimation de la variation de coût de la fonction objectif. Nous nous intéressons naturellement à la possibilité de réutiliser les coûts de remplacement dans des problèmes combinatoires NP-difficiles ou sortant du cadre algorithmique de base (chemin, arbres, couplages).

Pour certains problèmes il est possible de renforcer les bornes à l'intérieur même du théorème de structure. En effet, le calcul des coûts de remplacement se ramène souvent à la résolution d'un problème classique (chemin, cycle, arbre) dans un graphe dérivé des données initiales du problème. Or certaines contraintes additionnelles peuvent s'intégrer directement dans ce problème sans augmentation significative de la complexité.

Exemple : Plus courts chemins avec un nombre borné d'arcs

La contrainte de borne sur le nombre d'arcs peut être directement introduite dans le calcul des coûts de remplacement en calculant non plus l'arbre inverse des plus courts chemins mais les arbres inverses de plus courts chemins ayant moins de k-arcs. On pose ensuite

$$\overline{c}^k_{ij} = c_{ij} + d_{k-1}(j,t) - d_k(i,t)$$

Exemple : Exploration des couplages par voisinages k-opt

Il est possible de calculer étant donné un couplage M de coût minimal, le plus petit couplage N incluant un arc (i,j) et ne différant pas de M par plus de k arcs. Il suffit de calculer dans le graphe de Gabow les cycles passant par $(i, M(i))$, de coût minimal et de longueur bornée par k, ce qui se fait par l'algorithme de Ford-Bellman.

On peut ainsi obtenir des bornes exactes pour les couplages qui diffèrent exactement par k arcs de la solution optimale afin de mettre en place une procédure de recherche explorant les voisinages k-opt.

Exemple : "Discrepancy based additive bounding for the alldifferent constraint"

L'application proposée par Lodi et al. [13] consiste à partager dans un problème de couplage les images possibles pour un noeud en "bons" et "mauvais" sommets, puis exiger qu'il y ait exactement k mauvais sommets qui soient choisis dans le couplage.

Des bornes exactes peuvent également être calculées directement dans le graphe résiduel par un problème de plus court chemin sous contraintes de ressources (les "mauvais" sommets consomment une unité de ressource). Il s'agit d'une variation du problème précédent.

Cas général ?

De manière plus générale ce sont les procédures d'additive bounding semblables à celles utilisées en

programmation linéaire en nombres entiers qui paraissent être les plus adaptées au renforcement dynamique des bornes. Elles supposent que toutes les contraintes aient accès aux coûts de remplacement.

7 Conclusion

Nous nous sommes efforcés de clarifier les liens qui existent entre les algorithmes classiques de la recherche opérationnelle tels les plus courts chemins, les arbres couvrant et les couplages, leurs équivalents énumératifs et les contraintes globales. Les algorithmes permettant un filtrage efficace de ces contraintes d'optimisation se déduisent aisément des théorèmes classiques de structure si bien que la problématique des contraintes globales apparaît comme une extension naturelle des problématiques de l'algorithmique classique.

Références

[1] Ionut D. Aron and Pascal Van Hentenryck. A constraint satisfaction approach to the robust spanning tree problem with interval data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 18–25, 2002.

[2] Paolo M. Camerini, L. Fratta, and Francesco Maffioli. The k best spanning arborescences of a network. *Networks*, 10 :91–110, 1980.

[3] Yves Caseau and Francois Laburthe. Solving various weighted matching problems with constraints. In *Proceedings of the 3rd International conference on principles and practice of constraint programming (CP)*, pages 17–31, 1997.

[4] Chandra R. Chegireddy and Horst W. Hamacher. Algorithms for findink k best perfect matchings. *Discrete Applied Mathematics*, 18 :155–165, 1987.

[5] David Eppstein. Finding the k shortest paths. *SIAM J. Computing*, 28(2) :652–673, 1998.

[6] Filippo Focacci, Andrea Lodi, and Michela Milano. Cost-based domain filtering. In *Proceedings of the 5th International conference on principles and practice of constraint programming (CP)*, pages 189–203, 1999.

[7] Filippo Focacci, Andrea Lodi, and Michela Milano. Cutting planes in constraint programming : A hybrid approach. In *Proceedings of the 6th International conference on principles and practice of constraint programming (CP)*, pages 187–201, 2000.

[8] Frank Fussenegger and Harold N. Gabow. An improved method for finding the k smallest cost assignments in order. Technical Report CU-CS-124-78, University of Colorado at Boulder, 1977.

[9] Harold N. Gabow. Two algorihtms for generating weighted spanning trees in order. *SIAM J. Computing*, 6(1) :139–150, 1977.

[10] Horst W. Hamacher and Maurice Queyranne. k best solutions to combinatorial optimization problems. *Annals of Operations Research*, 4 :123–143, 1985.

[11] John Hershberger, Matthew Maxel, and Subhash Suri. Finding the k shortest simple paths : A new algorithm and its implementation. In *ALENEX*, 2003.

[12] Eugene L. Lawler. A procedure for computing the k best solutions to discrete optimization problems. *Management Science*, 18 :401–405, 1972.

[13] Andrea Lodi, Michela Milano, and Louis-Martin Rousseau. Discrepancy-based additive bounding for the alldifferent constraint. In *Proceedings of the 9th International conference on principles and practice of constraint programming (CP)*, pages 510–524, 2003.

[14] Katta G. Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16 :628–687, 1968.

[15] Jean-Charles Régin. A filtering algorithm for constraints of difference in CSPs. In *AAAI 1994*, pages 362–367, 1994.

[16] Jean-Charles Régin. Arc consistency for global cardinality constraints with costs. In *Proceedings of the 5th International conference on principles and practice of constraint programming (CP)*, pages 390–404, 1999.

[17] Meinolf Sellmann. Cost-based filtering for shorter path constraints. In *Proceedings of the 9th International conference on principles and practice of constraint programming (CP)*, pages 694–708, 2003.