HAL
open science

Agrégations de Among à l'aide de décompositions

Jean-Philippe Metivier, Samir Loudni

To cite this version:

Jean-Philippe Metivier, Samir Loudni. Agrégations de Among à l'aide de décompositions. Journée francophones de programmation par contraintes, May 2012, Toulouse, France. hal-00810481

HAL Id: hal-00810481

https://inria.hal.science/hal-00810481v1

Submitted on 10 Apr 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Agrégations de Among à l'aide de décompositions

Jean-Philippe Métivier **Samir Loudni**
Université de Caen – GREYC – (CNRS UMR 6072)
Campus II – Côte de Nacre – Boulevard du Maréchal Juin
14000 Caen – France
prenom.nom@unicaen.fr

Résumé

La décomposition de contraintes globales en un réseau Berge-acyclique permet, et cela sans perte de filtrage, d'intégrer simplement celles-ci dans des solveurs de contraintes (et cela de manière très efficace, cf. Regular). Dans ce papier, nous proposons d'utiliser la décomposition de la contrainte Among pour modéliser des agglomérations de contraintes Among, tout en préservant l'acyclicité du réseau issu de la décomposition.

1 Introduction

Les contraintes globales jouent un rôle majeur dans la modélisation et la résolution de problèmes de grande taille [6]. Celles-ci, en exploitant la structure du problème et en proposant des algorithmes de filtrage dédiés, permettent d'établir des cohérences fortes [1].Cependant l'intégration de contraintes globales dans un solveur de contraintes est une tâche complexe qui nécessite une grande expertise et une bonne connaissance du solveur.

La décomposition de contraintes globales en contraintes plus élémentaires facilite leurs mises en œuvre. Bien que facilement intégrables, toutes les décompositions ne permettent pas d'établir le même niveau de cohérence que le propagateur dédié.

Dans le cas où la décomposition forme un réseau Berge-acyclique, alors il est possible d'établir la cohérence de domaine via l'arc-cohérence sur le nouveau réseau de contraintes. De nombreux travaux proposent d'exploiter ce type de décomposition pour diverses contraintes globales : `Among` [1, 3], `Regular` [4], ...

Dans cet article, nous nous focalisons sur la contrainte `Among` [2]. Cette contrainte globale joue un rôle important dans la modélisation et la résolution de problèmes d'allocation de ressources. Dans ce type de problèmes, plusieurs contraintes `Among` peuvent être combinées afin de modéliser l'équilibrage de la charge de travail. Nous montrons, comment en utilisant la décomposition de la contrainte `Among`, il est alors possible d'établir la cohérence de domaine sur cette conjonction.

La suite de cet article est organisée de la façon suivante. La section 2 présente la contrainte `Among` ainsi que sa décomposition en un réseau Berge-acyclique. La section 3 donne un exemple d'équilibrage modélisé par une conjonction de contraintes `Among`, et montre comment décomposer celui-ci en un réseau Berge-acyclique. La section 4 décrit une méthode pour relaxer cette conjonction de contraintes. Finalement, la section 5 montre l'efficacité de cette approche au travers de plusieurs expériementations.

2 La contrainte Among

La contrainte globale `Among` est très utile pour modéliser de nombreux problèmes d'allocation de ressources, tels que le *staff rostering* ou le *car sequencing*.

2.1 La contrainte Among

La contrainte `Among` restreint le nombre de variables pouvant être assignées à des valeurs d'un ensemble donné :

Définition 1 (La contrainte `Among` [2]). *Soit $X=\{X_1,X_2,...,X_n\}$ un ensemble de variables et S un ensemble de valeurs. Notons l et u deux bornes telles que $0 \leq l < u \leq n$. La contrainte peut être définie par :*

$$\texttt{Among}(X, S, l, u) = \{(d_1, ..., d_n) \mid l \leq |\{d_i \in S, i \in [1..n]\}| \leq u\}$$

2.2 Décomposition Berge-acyclique de `Among`

Soit la contrainte $\texttt{Among}(\{X_1,..,X_n\},S,l,u)$. Celle-ci peut être naturellement décomposée en un réseau de

†. Ce travail a été soutenu par l'Agence nationale de la Recherche, reference ANR-10-BLA-0214.

1. http ://www.emn.fr/z-info/sdemasse/gccat/

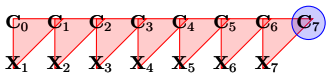

FIGURE 1 – Décomposition de la contrainte `Among` en un réseau Berge-acyclique.

contraintes ternaires et unaire Berge-acyclique [3] :

Définition 2 (Réseau de contraintes Berge-acyclique)**.** *Un réseau de contraintes est Berge-acyclique ssi son graphe d'incidence est acyclique. Le graphe d'incidence* $G = (X \cup E, e_G)$ *d'un réseau de contraintes* $R = (X, E)$ *est un graphe tel que l'ensemble des sommets contient un sommet pour chaque variable et chaque contrainte et on retrouve une arête* $\{X_i, E_j\}$ *ssi* X_i *est dans la portée de* E_j*.*

Pour décomposer `Among`, il est nécessaire d'introduire un ensemble de $n+1$ variables supplémentaires C_i. Ces variables jouent le rôle de compteur et ont pour domaine $[0..i]$. Elles comptabilisent le nombre de fois qu'une valeur de S a été affectée à la séquence de variables X_1,..,X_i.

Une contrainte unaire (cercle en bleu dans la figure 1) est ajoutée pour contraindre la dernière variable compteur à prendre une valeur comprise entre l et u. Pour chaque variable X_i ($i \in \{1, \ldots, n\}$), une contrainte ternaire (C_{i-1},X_i,C_i) (triangle en rouge dans la figure 1) est également ajoutée. Cette contrainte impose que la valeur du compteur C_i est égale à $C_{i-1} + 1$ si la variable X_i est instanciée à une valeur de S, C_{i-1} sinon.

La figure 1 présente la décomposition d'une contrainte `Among` portant sur sept variables.

Comme indiqué dans [1], pour toute contrainte globale décomposable en un réseau Berge-acyclique, établir la cohérence d'arc sur le réseau décomposé est équivalent à établir la cohérence de domaine sur la contrainte globale elle-même.

3 Among et équilibrage

Dans de nombreux problèmes de *staff rostering*, l'équilibrage de la charge de travail joue un rôle important. En plus, des contraintes `Among` portant sur l'ensemble du planning (imposant des bornes sur le nombre de jours travaillés par un employé), on retrouve également un ensemble de contraintes `Among` supplémentaires portant sur des sous-intervalles (p.e. les semaines) répartissant la charge de travail.

Dans cette section, nous montrons comment il est possible de maintenir une cohérence de domaine sur une telle conjonction de contraintes, en exploitant de manière astucieuse les variables compteurs ajoutées par la décomposition.

3.1 Exemple introductif

Considérons l'exemple suivant extrait de l'instance GPOST [2], une instance réelle de Nurse Rostering Problems (NRP).

Exemple 1. *Sur un horizon de 28 jours, une infirmière à plein temps doit travailler au plus 18 jours (en équipe de jour D ou de nuit N). Afin d'assurer un équilibrage de la charge de travail, une infirmière doit travailler entre 4 et 5 fois par semaine.*

Il est possible de modéliser ce problème comme suit.

Soit l'ensemble des jours J={1,2,...,28}. Pour chaque jour $j \in J$ est associé une variable X_j. Chaque variable prend ses valeurs sur l'ensemble des équipes possibles K={D,N,R}, avec R désignant un Repos.

Une première contrainte `Among` est utilisée pour contraindre le nombre de jours travaillés sur l'ensemble du planning. Puis pour chaque semaine, une contrainte `Among` est ajoutée pour assurer l'équilibrage. Ainsi, on obtient la modélisation suivante :

$\forall i \in [1..28],\ D_i = \{D, N, R\}$
`Among`($\{X_1, ..., X_{28}\}$,$\{D, N\}$,0,18)
`Among`($\{X_1, ..., X_7\}$,$\{D, N\}$,4,5)
`Among`($\{X_8, ..., X_{14}\}$,$\{D, N\}$,4,5)
`Among`($\{X_{15}, ..., X_{21}\}$,$\{D, N\}$,4,5)
`Among`($\{X_{22}, ..., X_{28}\}$,$\{D, N\}$,4,5)

Nous nommerons cette conjonction de contraintes `BalancedAmong`.

Définition 3 (La contrainte `BalancedAmong`)**.** *Soit* X={$X_1, ..., X_n$} *un ensemble de variables, et* S *un ensemble de valeurs. Soit* X^i *(pour* $i \in [0..\frac{n}{k}]$*) l'ensemble de variables tel que* X^i={$X_{k\cdot(i-1)+1}, ..., X_{k\cdot i}$}*. Soit* l *et* u *(resp.* l^i *et* u^i*) les bornes associées à* X *(resp. à* X^i*). La contrainte est définie par :*

$$\texttt{BalancedAmong}(X, S, k, l, u, l^i, u^i) = \texttt{Among}(X, S, l, u) \ \wedge \ \bigwedge_{i=1}^{\frac{n}{k}} \texttt{Among}(X^i, S, l^i, u^i)$$

N.B.: Les valeurs des bornes l^i et u^i peuvent être différentes pour chaque X^i.

3.2 Décomposition « naïve »

Si on applique naïvement la décomposition à `BalancedAmong` sur l'exemple 1, on obtient le réseau de contraintes de la figure 2. Le premier constat qu'il est possible de dresser est le nombre relativement élevé de contraintes ternaires engendrées par les décompositions des différentes contraintes `Among`. Par aillleurs, le nouveau réseau de contraintes obtenu n'est plus Berge-acyclique. En effet, de nombreux cycles sont présents (par exemple $\{C_0^1, C_1^1, X_2, C_1^f, C_0^f, X_1, C_0^1\}$).

2. http ://www.cs.nott.ac.uk/~tec/NRP/GPost.html

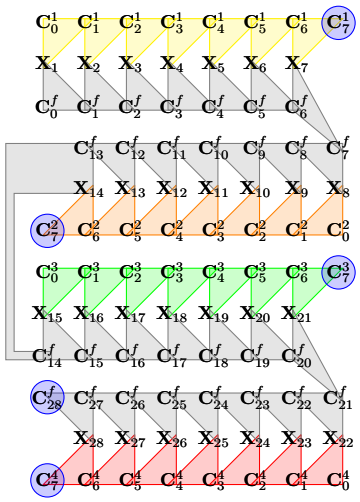

FIGURE 2 – Décomposition naïve du réseau de contraintes de l'exemple 1.

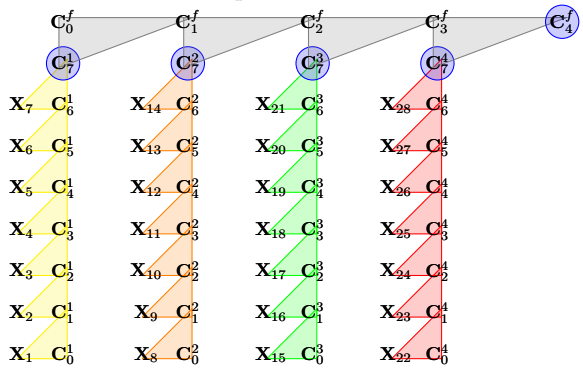

FIGURE 3 – Décomposition préservant l'acyclicité du réseau de contraintes de l'exemple 1.

3.3 Décomposition préservant l'acyclicité

Nous proposons d'exploiter les variables compteurs ajoutées par la décomposition pour d'une part rendre le réseau acyclique et d'autre part pour réduire le nombre de contraintes ternaires.

Pour chaque contrainte `Among`(X^i, S, l^i, u^i), la dernière variable compteur C_k^i correspond au nombre de fois qu'une valeur de S est affectée à une variable de X^i. Ainsi, pour connaître le nombre de variables de X prenant leur valeur dans S, il suffit de sommer les C_k^i de tous les X^i (car $\sqcup_{i\in[0,n/k]} X^i = X$). La contrainte globale somme est elle-même décomposable en un ensemble de contraintes ternaires.

La figure 3 illustre cette nouvelle décomposition utilisant une contrainte globale somme.

4 Relaxation

Pour relaxer une conjonction des contraintes `Among`, il suffit de pondérer chacune des contraintes unaires de la décomposition. Il est ainsi possible de combiner plusieurs sémantiques de violation.

La table 1 montre un exemple de fonctions de coût pénalisant tout écart δ sur un sous-ensemble (resp. l'intégralité) par un coût quadratique δ^2 (resp. linéaire $\delta \times 100$).

C_7^1	f
0	16
1	9
2	4
3	1
4	0
5	0
6	1
7	4

C_7^2	f
0	16
1	9
2	4
3	1
4	0
5	0
6	1
7	4

C_7^3	f
0	16
1	9
2	4
3	1
4	0
5	0
6	1
7	4

C_7^4	f
0	16
1	9
2	4
3	1
4	0
5	0
6	1
7	4

C_4^f	...	16	17	18	19	20	21	...
f	...	0	0	0	100	200	300	...

TABLE 1 – Sémantiques de violation associées à la décomposition du réseau de contraintes de l'exemple 1.

5 Expérimentations

Nous avons implémenté dans `ToulBar2 0.9.5`[3] les deux versions décomposées de la conjonction de contraintes `Among` ainsi que leurs versions relaxées. Les expériences ont été conduites sur un Core2Duo E8400 (2.83Ghz) avec 4GB de RAM. Les résultats reportés dans cette section (temps de calcul et nombre de backtracks) sont des sommes sur 100 instances.

5.1 Problème « pur »

Dans ce premier jeu d'expérimentations, pour chaque valeur du paramètre w, nous avons généré aléatoirement 100 instances. Chaque instance contient un ensemble de fonctions de coût unaire et une contrainte globale (`BalancedAmong`) agglomérant une conjonction de contraintes `Among`. Chaque instance contient $7 \times w$ variables ayant pour domaine l'ensemble {0,1,2}. Les contraintes unaires sont réparties sur 90% des variables, et contraignent celles-ci à prendre une valeur donnée contre un coût compris entre 1 et 10.

La contrainte `BalancedAmong` impose que les valeurs 0 et 1 soient affectées au plus $\lfloor 4.5 \times w \rfloor$ fois sur l'ensemble du planning. Pour chaque ensemble de 7 variables ($[X_1,...,X_7]$, $[X_8,...,X_{14}]$, ...) la limite est fixée entre 4 et 5 fois. Tout écart δ sur l'ensemble des variables est pénalisé par un coût linéaire de $\delta \times 100$ et chaque écart δ sur un ensemble de 7 variables est pénalisé par un coût quadratique δ^2.

Nous avons comparé trois implémentations :

1. la décomposition en un réseau Berge-acyclique de `BalancedAmong` combiné avec un ensemble de fonctions unaires (notée *Balanced*) ;
2. la décomposition de `BalancedAmong` à l'aide de contraintes `Among` décomposées et un ensemble de fonctions unaires (notée *AmongDec*) ;

3. https ://mulcyber.toulouse.inra.fr/projects/toulbar2

	Balanced		*AmongDec*		*AmongMono*	
w	#bt	time (s)	#bt	time (s)	#bt	time (s)
2	0	0.8	194	1.1	643	1.7
4	0	0.9	940	2.7	13'445	5.1
6	0	1.9	2'465	5.4	63'980	12.8
8	0	2.8	4'646	10.3	584'387	36.7
10	0	4.1	8'762	19.5	838'623	90.6

TABLE 2 – Comparaison des trois implémentations sur un problème « pur ».

	Balanced		*AmongDec*		*AmongMono*	
w	success	time (s)	success	time (s)	success	time (s)
2	100%	0.9	100%	1.3	100%	16.3
4	100%	1.9	100%	2.7	100%	77.9
6	100%	3.7	100%	5.8	100%	492.8
8	100%	6.1	100%	11.0	92%	17'937.7
10	100%	11.3	100%	29.8	28%	81'421.7

TABLE 3 – Comparaison des trois implémentations avec des contraintes `Regular` additionnelles.

3. une contrainte `BalancedAmong` exprimée à l'aide de `Among` monolithiques dans `Choco 2.1.3`[4] et des fonctions de coût unaire[5] (notée *AmongMono*).

La recherche est effectué sans limite de temps de calcul et sans utiliser d'option particulière.

Le tableau 2 montre clairement que la première implémentation permet d'obtenir des gains très substentiels à la fois en nombre de backtracks qu'en temps de résolution.

5.2 Avec des contraintes Regular additionnelles

Nous avons également généré un second jeu d'instances contenant 50% de fonctions de coût unaire (fixées de la même façon que précédemment) et une contrainte `BalancedAmong`. Pour contraindre d'avantage les instances, nous avons ajouté deux contraintes `Regular` : la première contrainte permet d'imposer un ordre sur l'enchaînement des valeurs et la seconde contrainte permet de limiter le nombre de fois successives que la valeur 1 peut être prise.

Le tableau 3 compare à nouveau les résultats des trois implémentations. La première colonne indique le pourcentage d'instances résolues en 1000 secondes, et la seconde le temps de calcul total (1000 secondes sont reportées lorsque la résolution d'une instance s'est achevés par épuisement du temps de calcul).

Une fois de plus, l'utilisation de la décomposition Berge-acyclique offre des gains substentiels.

5.3 Retour sur l'exemple : GPOST

Dans ce denier jeu d'expériences, nous avons mesuré l'apport de `BalancedAmong` sur les deux instances `GPOST`s. Nous avons utilisé la même modilisation que dans [5] et la même méthode de résolution : `VNS/LDS+CP` (et `RandomNurse` comme heuristique de choix de voisinages). Pour chaque instance, nous avons effectué 100 essais (avec un temps d'exécution maximal de 1800 secondes).

		Balanced			*AmongDec*		
Instance	Opt	succ.	avg	time (s)	succ.	avg	time (s)
`GPOSTA`	5	18	15.2	677.9	12	21.4	858.3
`GPOSTB`	3	9	44.9	732.8	5	62.1	463.8

TABLE 4 – Comparaison sur les instances `GPOST`s.

Le tableau 4 présent le nombre de succés – le nombre de fois que l'optimum est atteint –, la moyenne de la qualité des solutions retournées, et le temps moyen pour trouver une solution de coût optimal.

L'utilisation de `BalancedAmong` permet de grandement augmenter le nombre de succés (dans la cas de `GPOST-B`, celui-ci est quasiment doublé).

6 Conclusions et futurs travaux

Dans ce papier, nous avons présenté, comment en utilisant la décomposition de `Among`, il est possible d'établir la cohérence de domaine sur une conjonction particulière de `Among`. Nous avons aussi montré, expérimentalement, l'apport de cette nouvelle contrainte.

Comme futurs travaux, il serait utile de tester l'apport de `BalancedAmong` sur des problèmes de *rostering* ou de *car sequencing*. Il serait aussi intéressant d'étudier l'apport de la décomposition sur d'autres conjonctions de `Among` ou d'autres contraintes globales comme `Sum` ou `Regular`.

Références

[1] N. Beldiceanu, M. Carlsson, R. Debruyne, and T. Petit. Reformulation of global constraints based on constraints checkers. *Constraints*, 10(4) :339–362, 2005.

[2] N. Beldiceanu and E. Contejean. Introducing global constraints in chip. *Journal of Mathematical and Computer Modelling*, 20(12) :97–123, 1994.

[3] C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan, and T. Walsh. Among, common and disjoint constraints. In *CSCLP*, pages 29–43, 2005.

[4] C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan, and T. Walsh. Slide : A useful special case of the card-path constraint. In *ECAI*, pages 475–479, 2008.

[5] J-P. Métivier, P. Boizumault, and S. Loudni. Solving nurse rostering problems using soft global constraints. In *CP*, pages 73–87, 2009.

[6] H. Simonis. Models for global constraint applications. *Constraints*, 12(1) :63–92, 2007.

4. http ://www.emn.fr/z-info/choco-solver/
5. Modelisée à l'aide de contraintes réifiées.