Amélioration de la contrainte globale de cardinalité avec coûts par l'introduction de points de repère *

Margaux Schmied[1], Jean-Charles Régin[1]

[1] Université Côte d'Azur, CNRS, I3S, France

margaux.schmied@icloud.com, jcregin@gmail.com

Résumé

Le succès de la programmation par contraintes repose en partie sur l'efficacité des contraintes globales et de leurs algorithmes de filtrage. Nous étudions ici la contrainte globale de cardinalité avec coûts, qui généralise la contrainte de différence en limitant le nombre d'affectations par valeur et en imposant une contrainte sur le coût des affectations. L'algorithme classique de filtrage repose sur le calcul systématique de plus courts chemins, ce qui le rend coûteux en pratique. Nous proposons une approche plus rapide, fondée sur des bornes supérieures obtenues via des points de repère (landmarks en anglais), réduisant ainsi significativement les calculs explicites nécessaires. Les expériences montrent que, dans le meilleur des cas, elle peut être en moyenne 57 fois plus rapide.

Mots-clés

Globale contrainte, Algorithme de filtrage, Contrainte globale de cardinalité avec coûts, Arc consistance

Abstract

The success of constraint programming relies partly on the efficiency of global constraints and their filtering algorithms. We study in this article the global cardinality constraint with costs, which generalizes the difference constraint by limiting the number of assignments per value and imposing a constraint on the cost of assignments. The classical filtering algorithm is based on the systematic computation of shortest paths, which makes it costly in practice. We propose a faster approach, based on upper bounds obtained via landmarks, thus significantly reducing the explicit computations required. Experiments show that, in the best case, it is on average up to 57 times faster.

Keywords

Global constraint, Filtering algorithm, Cardinality constraints with costs, Arc consistency

1 Introduction

En Programmation par Contraintes (PPC), un problème est défini par des variables et des contraintes. Chaque variable possède un domaine qui représente l'ensemble de ses valeurs possibles, tandis qu'une contrainte exprime une relation à satisfaire entre un groupe de variables. La PPC s'appuie sur des méthodes de résolution adaptées à chaque contrainte, appelées algorithmes de filtrage.

Le succès de la PPC repose en partie sur l'efficacité de ces algorithmes. Ils éliminent des valeurs dans les domaines des variables lorsque celles-ci ne peuvent pas appartenir à une solution satisfaisant la contrainte. En d'autres termes, ils garantissent la consistance des contraintes en simplifiant le problème à résoudre.

Nous nous intéressons à la contrainte globale de cardinalité avec coûts (costGCC) [6]. Cette contrainte généralise la contrainte de différence. Elle s'applique à un ensemble de variables $X = x_1, \ldots, x_p$ avec des valeurs $V = v_1, \ldots, v_d$. Chaque valeur $v_i \in V$ doit être affectée entre $[l_i, u_i]$ fois. Un coût est associé à chaque affectation, et le coût total des affectations doit être inférieur à H. Cette contrainte est utilisée pour le routage ou l'ordonnancement, où les coûts peuvent modéliser des préférences, des poids ou des durées.

L'algorithme de filtrage associé à la contrainte costGCC [6] peut être qualifié de répétitif. Il commence par calculer un flot maximum à coût minimum (minCostMaxFlow) afin de vérifier si la contrainte est consistante, c'est-à-dire si une solution existe. Puis, il élimine les valeurs inconsistantes avec la contrainte, autrement dit qui ne peuvent pas appartenir à une solution. Plus précisément, il examine chaque valeur de chaque variable afin de déterminer s'il existe un chemin dans le graphe résiduel dont la valeur est inférieure à H.

Nous présentons une nouvelle approche visant à minimiser l'aspect répétitif de l'algorithme de filtrage. Cette approche repose sur plusieurs observations :

- **Flot suffisant** : Un flot à coût minimal n'est pas nécessaire pour chaque affectation, une simple vérification d'un coût inférieur à H suffit.
- **Approximation des coûts** : Des bornes supérieures approximatives sur les coûts des plus courts chemins sont suffisantes.
- **Points de repère** : L'utilisation de nœuds particuliers, appelés points de repère, s'avère efficace pour accélérer le calcul des plus courts chemins dans de grands graphes (contenant des millions de nœuds) [2]. Soient x et y deux nœuds d'un graphe et p un point de repère, alors $d(x,p) + d(p,y) \geq d(x,y)$ est toujours vérifiée. Cela permet d'obte-

*Cet article se base sur des résultats publiés à CP 2024[7].

nir immédiatement une borne supérieure de $d(x, y)$ entre chaque paire de nœuds x et y.

— **Impact des suppressions** : En début de recherche, le filtrage n'élimine souvent aucune valeur. Cela suggère que la marge entre H et le coût du flot minimal est généralement importante, justifiant l'usage d'approximation.

Pour amélioré l'algorithme proposé par Régin [6], nous proposons une phase de prétraitement pour limiter les calculs explicites des plus courts chemins. Notre méthode identifie des points de repère dans le graphe, permettant d'estimer des bornes supérieures sur les coûts des chemins réduisant les calculs inutiles. Ces points sont choisis selon la structure du graphe (centre (C), périphérie (O), centre et périphérie (C & O), degrée des nœuds (Deg) et aléatoire (R)) : deux calculs de plus courts chemins suffisent par repère. Nous introduisons une propriété pour détecter rapidement l'arc consistance d'une contrainte costGCC, réduisant davantage les opérations superflues.

Le Tableau 1 présente les ratios moyens d'accélération (Régin/Landmarks) obtenus avec notre méthode sur quatre ensembles de données : 1) Problème du voyageur de commerce (TSP) [1], 2) problème de stockage avec coûts (StockingCost) [3], 3) problème d'ordonnancement de tâches (FJSSP) [5, 9] et 4) problème d'assignation d'enfants à des activités (CHILD) [8]. La valeur de seuil H pour le TSP provient de l'heuristique de Lin-Kernighan [4]. Pour les autres instances, H correspond à la plus petite valeur garantissant l'existence d'une solution.

Ces résultats montrent que notre approche est toujours au moins aussi rapide que la méthode de Régin et qu'elle peut être en moyenne jusqu'à 57 fois plus rapide dans les meilleurs cas.

	C	O	C & O	Deg	R
TSP ($\leq$ 100 cities)	1.2	1.2	1.1	1.3	**1.6**
TSP ($>$ 100 & $<$ 250 cities)	**2.6**	2.5	2.5	**2.7**	2.5
TSP ($\geq$ 250 cities)	43.5	44.1	44	**56.9**	45.8
StockingCost (H)	**1.2**	1	1	1	0.9
StockingCost ($H \times 2$)	15.7	16.5	**16.9**	16	16.4
FJSSP (H)	0.8	**1.7**	0.75	1	0.8
FJSSP ($H \times 2$)	1	**1.3**	**1.3**	**1.3**	**1.3**
CHILD (H)	0.9	**1.2**	1	0.8	1
CHILD ($H \times 2$)	8.3	9	8	**9.7**	**9.7**

TABLE 1 – Ratio d'accélération (Régin/landmarks) du calcul de l'arc consistance d'une contrainte costGCC avec 4 repères.

2 Conclusion

Nous présentons une implémentation optimisée de l'algorithme d'arc consistance pour la contrainte costGCC, essentielle dans de nombreux problèmes industriels. L'algorithme classique, basé sur le calcul exact des plus courts chemins, est souvent trop lent en pratique. Nous proposons une approche reposant sur des bornes supérieures issues d'inégalités triangulaires et de points de repère, réduisant ainsi le nombre de calculs nécessaires. Les gains sont d'autant plus significatifs que plus le graphe est grand plus les coûts de calcul sont important. De plus, nous introduisons une condition suffisante, simple et rapide à évaluer, permettant de vérifier efficacement si une contrainte costGCC est arc consistante.

Remerciements

Ce travail a été soutenu par le gouvernement français, à travers le projet 3IA Côte d'Azur Investissements d'Avenir géré par l'Agence Nationale de la Recherche (ANR) avec le numéro de référence ANR-19-P3IA-0002.

Références

[1] R. Gerhard. TSPLIB–a traveling salesman problem library. *ORSA Journal on Computing*, 3(4) :376–384, 1991.

[2] A. V. Goldberg and C. Harrelson. Computing the shortest path : A search meets graph theory. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, page 156–165, USA, 2005. Society for Industrial and Applied Mathematics.

[3] V. R. Houndji, P. Schaus, and L. Wolsey. The item dependent stockingcost constraint. *Constraints*, 24(2) :183–209, April 2019.

[4] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Oper. Res.*, 21 :498–516, 1973.

[5] M. Pelleau, A. Miné, C. Truchet, and F. Benhamou. A constraint solver based on abstract domains. In *Verification, Model Checking, and Abstract Interpretation, 14th International Conference, VMCAI 2013, Rome, Italy, January 20-22, 2013. Proceedings*, pages 434–454, 2013.

[6] J.-C. Régin. Cost-based arc consistency for global cardinality constraints. *Constraints*, 7(3/4) :387–405, jul 2002.

[7] M. Schmied and J.-C. Régin. Efficient Implementation of the Global Cardinality Constraint with Costs. In *30th International Conference on Principles and Practice of Constraint Programming (CP 2024)*, volume 307, pages 27 :1–27 :18, 2024.

[8] S. Varone and C. Beffa. Dataset on a problem of assigning activities to children, with various optimization constraints. *Data in Brief*, 2019.

[9] T. Weise. jsspinstancesandresults : Results, data, and instances of the job shop scheduling problem, 2019–2020. A GitHub repository with the common benchmark instances for the Job Shop Scheduling Problem as well as results from the literature, both in form of CSV files as well as R program code to access them.