Utilisation d'algorithmes d'approximation en Programmation Par Contraintes

Arthur Godet, Xavier Lorca, Gilles Simonin

Utilisation d'algorithmes d'approximation en Programmation Par Contraintes

Arthur Godet[1*] **Xavier Lorca**[2] **Gilles Simonin**[1]

[1] Equipe TASC, IMT Atlantique, LS2N, CNRS, 44307, Nantes, France
[2] Centre Génie Industriel, IMT Mines Albi, Campus Jarlard, 81013 Albi cedex 09, France
{arthur.godet,gilles.simonin}@imt-atlantique.fr xavier.lorca@mines-albi.fr

Résumé

Dans cet article, nous présenterons les travaux préliminaires menés sur l'utilisation d'algorithmes d'approximation en Programmation Par Contraintes afin d'améliorer le calcul de bornes lors de la résolution de problèmes d'optimisation sous contraintes. L'objectif de nos travaux est d'étudier plus particulièrement quels algorithmes d'approximation présentent suffisamment de flexibilité pour être utilisés en Programmation Par Contraintes, et comment les utiliser au sein d'un propagateur qui mettra à jour les bornes de la variable-objectif à chaque noeud de l'espace de recherche. Enfin l'idée sera d'appliquer cette approche à plusieurs familles de problèmes d'optimisation afin d'en extraire une généralisation.

Abstract

In this paper we present our current work on using approximation algorithms in constraint programming to improve bounds computing when solving Constraint Optimization Problems. The objective of our work is to study more particularly which approximation algorithms have enough flexibility to be used in constraint programming, and how to use them inside a propagator which will update the bounds of the objective-variable at each node of the search space. At last, we want to apply this methodology to several families of optimisation problems in order to obtain a generalisation.

1 Introduction

La **Programmation Par Contraintes (PPC)** tire sa force de son expressivité et de sa flexibilité. L'une par sa capacité à exprimer des sous-problèmes pertinents d'un problème donné au travers de contraintes indépendantes les unes des autres. L'autre par sa capacité à décomposer un problème combinatoire en sous-problèmes pour lesquels une résolution efficace est possible. Elle se distingue ainsi des autres technologies de résolution de problèmes combinatoires, telles que la Programmation Linéaire ou la Programmation Mixte en nombre entiers, par la richesse et la variété des contraintes exprimables, qui dépasse par exemple le cadre de simples équations linéaires. Cette richesse et cette diversité s'expriment typiquement par l'utilisation de **contraintes globales**, c'est-à-dire de contraintes encapsulant des sous-problèmes combinatoires pour lesquels des traitements algorithmiques efficaces sont connus. Ceci permet généralement un meilleur *filtrage*. Toutefois, et malgré de nombreuses applications industrielles, la PPC souffre d'un certain manque d'efficacité en ce qui concerne la résolution de **Problèmes d'Optimisation sous Contraintes (COP)**.

D'un autre côté, les **Algorithmes d'Approximation** sont des algorithmes polynomiaux en temps et en espace résolvant de manière approchée des problèmes $\mathcal{NP}$-difficile avec une certaine garantie de résultat par rapport à l'optimal.

Definition 1. *Un **algorithme d'approximation** est un algorithme polynomial, en temps et en espace, résolvant un problème de minimisation (resp. maximisation) sous contraintes avec une garantie de résultat. Ainsi, si l'on note ρ le rapport d'approximation d'un algorithme d'approximation, ce dernier renverra, pour toute instance du problème, une solution dont l'évaluation ne sera pas plus grande (resp. petite) que $\rho \times opt$, où opt représente l'évaluation d'une solution optimale de l'instance du problème.*

Pour chaque problème bien connu, les recherches menées tournent autour de l'élaboration d'algorithmes présentant de meilleurs ratios d'approximation, ou de prouver qu'il n'est pas possible de faire mieux qu'un cer-

*Papier doctorant : Arthur Godet[1] est auteur principal.

tain ratio. Ainsi, pour un même problème $\mathcal{NP}$-difficile, il existe souvent différents algorithmes d'approximation le résolvant de manière non optimale. Par exemple, pour le problème $\mathcal{NP}$-difficile du *Bin Packing* [4], plusieurs algorithmes d'approximation sont connus depuis longtemps, tels que le *First-Fit* et le *Best-Fit* [5]. Toutefois, les recherches menées ont permis de mettre au point des algorithmes avec de meilleures garanties [3]. L'intérêt de s'orienter vers la théorie de l'approximation pour améliorer la PPC réside en sa forte communauté, active depuis plusieurs décennies, ayant ainsi produit de nombreux résultats sur lesquels s'appuyer.

Dans ce papier, nous présentons les travaux que nous menons pour utiliser les résultats issus de l'étude d'algorithmes d'approximation en Programmation Par Contraintes afin d'améliorer l'efficacité de résolution de COP. Dans une première section, nous discuterons des difficultés d'application des algorithmes d'approximation dans le cadre de la PPC et présenterons un algorithme générique - illustrant notre méthodologie - qui peut être embarqué au sein d'un *propagateur*. Dans une seconde section, nous présenterons les premiers résultats expérimentaux que nous avons obtenus.

2 Méthodologie

La Programmation Par Contraintes présente une forte flexibilité quant aux contraintes que l'on peut poser, ainsi qu'aux techniques de résolution déployées au sein des solveurs de contraintes. Cette flexibilité de résolution est exactement là où réside la difficulté d'utilisation d'algorithmes d'approximation en PPC. En effet, les algorithmes d'approximation ont souvent un comportement très déterministe, ne laissant pas de place à des choix arbitraires. Bien que ce cadre déterministe soit souvent à la base des démonstrations des ratios d'approximation, il s'avère très limitant en Programmation Par Contraintes. En effet, les heuristiques de sélection de la variable sur laquelle brancher et sur quelle valeur brancher peuvent amener à une situation ne correspondant pas au cadre d'application de l'algorithme d'approximation. C'est-à-dire que les instantations déjà effectuées ne correspondent pas, telles quelles, à une situation intermédiaire dans laquelle l'algorithme d'approximation aurait pu se retrouver à partir de l'instance initiale et sous certaines règles de choix arbitraires.

2.1 Exemple montrant cette limite

Pour expliquer cette limite, nous nous baserons sur le problème de minimisation du *Bin Packing*. Considérons une liste de n objets, chaque objet i possédant un poids w_i. Nous disposons également de n conteneurs homogènes de capacité B. L'objectif est de placer l'ensemble des objets dans le plus petit nombre possible de conteneurs.

Pour résoudre ce problème, comme nous l'avons dit, il existe de nombreux algorithmes d'approximation, le plus connu d'entre eux étant l'algorithme dit du *First-Fit Decreasing (FFD)*. Cet algorithme consiste à trier les objets par ordre décroissant de poids, puis de les placer chacun leur tour dans le premier conteneur pouvant le contenir.

Une modélisation possible de ce problème en Programmation Par Contraintes consiste en n variables entières x_1, ..., x_n, chacune de domaine $[1, n]$ indiquant le conteneur auquel est associé l'objet. On pose finalement la contrainte globale *bin_packing* [7, 2]. Dans le cas du problème de minimisation, on utilise également une variable z de domaine $[1, n]$ indiquant le nombre de conteneurs utilisés à l'aide de la contrainte $nvalue(z, \{x_1, ..., x_n\})$ [6] (une contrainte $max(z, \{x_1, ..., x_n\})$ peut être utilisée pour casser les symétries en l'absence de contraintes supplémentaires). On cherche donc une solution minimisant la valeur de z.

Supposons, sans perdre en généralité, que les variables x_1, ..., x_n sont déjà triées par ordre décroissant de poids. N'importe quelle situation telle qu'une variable x_j est instanciée sans qu'une variable x_i ne le soit (avec $i < j$) ne correspond à aucune étape intermédiaire dans laquelle l'algorithme du *First-Fit Decreasing* aurait pu se retrouver, cet algorithme plaçant les objets par ordre décroissant de poids dans le premier conteneur pouvant le contenir. Cette situation se produirait systématiquement si l'heuristique de sélection de variable consiste en un ordre lexicographique inversé.

Notons qu'il peut être possible dans certains cas de se ramener à un cas d'application de l'algorithme, moyennant quelques "transformations" de l'instance sur laquelle on applique l'algorithme d'approximation. Ce point sera discuté dans la prochaine sous-partie, et illustré en conservant notre exemple.

2.2 Algorithme générique

En considérant un problème de minimisation sous contraintes (un problème de maximisation sous contraintes se traiterait de manière analogue) ainsi qu'un algorithme d'approximation **Heur** de ratio d'approximation $\rho > 1$, le propagateur que l'on ajoute est décrit par l'algorithme 1 :

Tout d'abord, le propagateur teste si l'algorithme d'approximation est applicable à partir de l'état actuel des variables. Bien entendu, la fonction **detectIfAlgoApplicable()** dépend de l'algorithme d'approximation sur lequel est basé le propagateur. Cette fonction

```
Entrées : z the objective-variable, x_i variables
début
    si detectIfAlgoApplicable() alors
        inst ← buildCorrespondingInstance();
        sol ← Heur(inst);
        z.updateUpperBound(sol);
        z.updateLowerBound(sol/ρ);
    fin
fin
```

Algorithme 1 : APX-Propagator

renvoie donc s'il est possible, à partir de l'état actuel des variables, de créer une instance du problème sur laquelle appliquer l'algorithme d'approximation. Si les règles d'application de l'algorithme d'approximation ne permettent pas d'interdire des valeurs et que cela a un impact important, cette fonction devra en tenir compte.

La fonction **buildCorrespondingInstance()** construit l'instance correspondant à l'état actuel des variables et sur laquelle sera appliquée l'algorithme d'approximation. Cette fonction dépend également de l'algorithme d'approximation sur lequel est basé le propagateur.

Finalement, le propagateur applique l'algorithme d'approximation sur l'instance construite et met à jour les bornes de la variable-objectif z à partir de la valeur renvoyée par l'algorithme d'approximation. Dans le cas d'un problème de minimisation, l'algorithme d'approximation renvoie une valeur comprise entre opt et $\rho \times opt$. Ainsi on met à jour la borne inférieure de la variable-objectif par le maximum entre sa valeur actuelle et le résultat de l'algorithme d'approximation divisé par son rapport d'approximation. La borne supérieure est mise à jour par le minimum entre sa valeur actuelle et le résultat de l'algorithme d'approximation (plus grand ou égal à la valeur optimale dans le cas d'un problème de minimisation).

Il est intéressant de noter que les bornes calculées par ce propagateur sont valables dans tout l'espace de recherche.

À ce stade de notre étude, nous supposons qu'il existe un lien entre le nombre de situations applicables de l'algorithme d'approximation et l'impact, en terme de filtrage sur la variable-objectif, du propagateur dérivé de cet algorithme.

Ainsi, il pourrait être plus intéressant d'utiliser un algorithme d'approximation avec un moins bon ratio d'approximation s'il est davantage applicable qu'un algorithme d'approximation avec un meilleur ratio d'approximation mais des règles d'application plus restrictives. Dans le cas du problème de *Bin Packing*, il paraît évident que l'algorithme du *First-Fit*, ayant un rapport d'approximation de $\frac{17}{10} \times S^{OPT} + 2$ [5], est applicable dans davantage de situations que l'algorithme du *First-Fit Decreasing*, de rapport d'approximation $\frac{11}{9} \times S^{OPT} + 4$ [5]. L'algorithme du *First-Fit* a donc un ratio d'approximation de moins bonne qualité que celui du *First-Fit Decreasing*, mais le *First-Fit* étant applicable dans davantage de cas, on peut suspecter qu'il aura plus d'impact sur l'évaluation de la borne inférieure de la variable-objectif (ici le nombre de conteneurs utilisés). Le fait qu'il soit applicable dans plus de situations laisse également espérer qu'il pourra aussi mieux évaluer une borne supérieure.

3 Méthodologie expérimentale

Au moment de la rédaction de cet article, nous travaillons encore à l'élaboration et l'exécution de protocoles expérimentaux permettant d'observer les effets apportés par un tel propagateur. L'objectif est de comparer les statistiques de résolution (temps de résolution, nombre de noeuds parcourus, nombre de backtracks, nombre de fails) obtenues par un modèle et celles obtenues par ce même modèle auquel on aura ajouté notre propagateur. On observera en particulier le nombre d'appels à notre propagateur au cours de la recherche d'une solution optimale, ainsi que les effets qu'il aura eus.

L'idée est également de comparer les apports de différents algorithmes d'approximation, cherchant notamment à établir une idée du compromis qu'il peut y avoir entre rapport d'approximation et cas d'application. Pour cela, on construira deux modèles, l'un utilisant notre propagateur avec un algorithme d'approximation A, l'autre utilisant notre propagateur avec un algorithme d'approximation B. On comparera alors les statistiques de résolution des deux modèles, et l'on comparera plus particulièrement le nombre d'appels effectifs à notre propagateur dans chacun des deux modèles et les effets que chacun aura eu.

Finalement, l'objectif sera d'essayer d'appliquer cette méthodologie à différents problèmes, et différentes familles de problèmes, et d'essayer de faire émerger une généricité de l'approche.

	Temps (s)	Noeuds	Erreurs
BFD-Prop	17.36	177417	177340
Shaw	503.6	12147504	12147425
Binary model	379.5	14047374	14047295

FIGURE 1 – Résultats expérimentaux préliminaires pour le problème du Bin-Packing

Bien que la phase expérimentale soit en cours, certains résultats sont déjà très prometteurs. Dans le cadre

du problème de *Bin Packing*, nous avons généré des instances du problème en suivant le papier de Castiñeiras et al. [1]. Appliquer l'algorithme du *Best-Fit Decreasing* au noeud racine de l'arbre de recherche a permis de diviser par 20 le temps total de résolution à l'optimal de l'ensemble des 100 instances de taille 10, comme le rapporte la figure 1 (BFD-Prop étant un propagateur suivant le cadre défini et utilisant l'algorithme du *Best-Fit Decreasing*). Cela nous paraît donc très prometteur pour l'apport que peut avoir notre propagateur pour la recherche de solution optimale sur de grosses instances. On veillera toutefois à distinguer les apports de notre propagateur par rapport à un simple *presolve* (ici l'application du *Best-Fit Decreasing* au noeud racine).

4 Conclusion et Perspectives

Dans cet article, nous présentons une façon de compléter les modèles de résolution de Problèmes d'Optimisation sous Contraintes à l'aide de propagateurs basés chacun sur un algorithme d'approximation. Cette approche devrait permettre d'améliorer l'efficacité des solveurs de contraintes pour résoudre ces Problèmes d'Optimisation sous Contraintes. Les premiers résultats expérimentaux sont très prometteurs mais nécessitent d'être étendus - principalement en utilisant une implémentation réelle de notre propagateur - afin de pouvoir mieux qualifier l'apport de cette approche. Il est notamment prévu de montrer la généricité de l'approche en l'appliquant à différentes familles de problèmes.

Enfin, il faudra étudier si cette approche peut être utilisée, et le cas échéant à quel point ses effets varient, lorsque l'on ajoute des contraintes supplémentaires au problème initial.

Références

[1] I. Castiñeiras and M. De Cauwer and B. O'Sullivan : Weibull-Based Benchmarks for Bin Packing. *In Proceedings CP'12*, pages 207–222, 2012.

[2] G. Derval and P. Schaus and J-C. Régin : Improved filtering for the bin-packing with cardinality constraint. *Constraints*, 3 :251–271, 2018.

[3] W. Fernandez de la Vega and G. S. Lueker : Bin packing can be solved within $1+\epsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.

[4] M. R. Garey and D. S. Johnson : Computers and Intractability ; A Guide to the Theory of NP-Completeness. *W. H. Freeman & Co., New York, NY, USA*, 1979.

[5] D. S. Johnson and A. Demers and J. D. Ullman and M. R. Garey and R. L. Graham : Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput*, 3(4):298–325, 1974.

[6] F. Pachet and P. Roy : Automatic Generation of Music Programs. *In Proceedings CP'99*, pages 331–345, 1999.

[7] P. Shaw : A constraint for bin packing. *In Proceedings CP'04*, pages 648–662, 2004.