HAL
open science

Résolution du problème d'équilibrage des diplômes grâce à l'hybridation d'algorithmes génétiques et de la propagation de contraintes

Tony Lambert, Carlos Castro, Eric Monfroy, Maria Cristina Riff, Frédéric Saubion

To cite this version:

Tony Lambert, Carlos Castro, Eric Monfroy, Maria Cristina Riff, Frédéric Saubion. Résolution du problème d'équilibrage des diplômes grâce à l'hybridation d'algorithmes génétiques et de la propagation de contraintes. Premières Journées Francophones de Programmation par Contraintes, CRIL - CNRS FRE 2499, Jun 2005, Lens, pp.423-426. inria-00000054

HAL Id: inria-00000054

https://inria.hal.science/inria-00000054

Submitted on 25 May 2005

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.

Résolution du problème d'équilibrage des diplômes grâce à l'hybridation d'algorithmes génétiques et de la propagation de contraintes

Tony Lambert[1,2] **Carlos Castro**[3] **Eric Monfroy**[1,3]
María Cristina Riff[3] **Frédéric Saubion**[2]

[1] LINA, Université de Nantes, France (`Firstname.Name@lina.univ-nantes.fr`)
[2] LERIA, Université d'Angers, France (`Firstname.Name@univ-angers.fr`)
[3] Universidad Santa María, Valparaíso, Chile (`Firstname.Name@inf.utfsm.cl`)

1 Introduction

Les problèmes de satisfaction de contraintes (CSP) sont habituellement définis comme un ensemble de variables associées à leur domaines de valeurs possibles et par un ensemble de contraintes. Ils fournissent un cadre pour la modélisation de nombreuses applications pratiques dans le domaine de l'aide à la decision. La plupart de ces problèmes sont associés à un critère d'optimisation. Dans ce contexte, l'optimisation sous contraintes consiste à trouver une affectation de valeurs aux variables qui satisfasse les contraintes et optimise une fonction objectif donnée. Beaucoup d'algorithmes de résolution ont été proposés et peuvent être classés dans deux groupes principaux.

Les méthodes complètes explorent l'espace de recherche en totalité afin de trouver toutes les solutions ou, de détecter que le CSP n'est pas consistant. En ce qui concerne les techniques de résolution des méthodes complètes, nous sommes ici principalement concernés par la propagation de contraintes et le découpage des domaines des variables qui est une des techniques les plus usuelles de la programmation par contraintes (CP).

Les méthodes incomplètes, elles, se basent principalement sur l'utilisation de métaheuristiques fournissant une exploration plus efficace des régions intéressantes de l'espace de recherche afin de trouver des solutions. Les approches les plus communément utilisées sont basées sur des algorithmes évolutionnaires [6, 3, 7] et des algorithmes de recherche locale [1].

Dans cet article, nous nous sommes intéressés à la conception d'un cadre hybride de résolution incluant les algorithmes génétiques et la propagation de contraintes pour résoudre un problème combinatoire particulier : l'équilibrage des cursus universitaires (BACP : Balanced Academic Curriculum Problem) présenté dans [4].

Notre but est alors double : d'une part, nous visons à résoudre efficacement ce problème en combinant un algorithme génétique avec des techniques de programmation par contraintes, et d'autre part, nous proposons un cadre général pour concevoir avec précision un tel processus de résolution hybride pour identifier clairement ses caractéristiques et propriétés.

2 Modèle de calcul CP vs AG

Nous rappelons tout d'abord les notions de base liées aux CSP, à l'optimisation, et aux AG avant de présenter notre cadre général pour l'hybridation CP et AG.

Un CSP est un triplet (X, D, C) où $X = \{x_1, \cdots, x_n\}$ est un ensemble de variables prenant leurs valeurs dans leurs domaines respectifs $D = \{D_1, \cdots, D_n\}$. Une contrainte $c \in C$ est une relation $c \subseteq D_1 \times \cdots \times D_n$. Afin de simplifier les notations, D désignera également le produit cartésien des D_i et C l'union des contraintes. Un n-uplet $d \in D$ est une solution d'un CSP (X, D, C) si et seulement si $\forall c \in C, d \in c$. La propagation de contraintes, est une des techniques les plus célèbres pour résoudre les CSP. Elle se traduit par la réduction des domaines des

variables de manière itérative (suppression des valeurs qui ne satisfont pas les contraintes).

Ces réductions doivent s'alterner avec un mécanisme d'énumération afin d'obtenir un solveur complet. Les problèmes d'optimisation sous contraintes, bien que semblables à ceux de la satisfaction de contraintes, sont comparativement plus difficiles parce qu'ils n'acceptent que les solutions qui minimisent ou maximisent une fonction objectif donnée, tout en satisfaisant les contraintes.

Dans [2], K.R. Apt a défini un cadre pour la propagation de contraintes. Dans [8], le modèle est étendu avec des opérateurs de découpage et des stratégies de recherches locales afin de modéliser les différentes méthodes de résolutions hybrides par le calcul d'un point fixe d'un ensemble de fonctions sur un ensemble ordonné.

Algorithme générique itératif (GI)

Dans ce contexte, la réduction de domaines correspond au calcul d'un point fixe d'un ensemble de fonctions sur un ensemble partiellement ordonné. Ces fonctions, appelées *fonctions de réduction*, abstraient la notion de contraintes.

Le calcul est réalisé selon l'algorithm **GI** qui, supposons toutes les fonctions inflationnaires et monotones, termine alors pour chaque exécution et calcul le plus petit pointfixe commun des fonctions.

Ce cadre abstrait est étendu ici à l'hybridation de techniques de résolution.

Les Algorithmes Génétiques

Nous voulons utiliser un cadre uniforme nous permettant de combiner des techniques de propagation de contraintes et des méthodes heuristiques. Basée sur le principe de la sélection naturelle, *les algorithmes génétiques* [6] ont été appliqués avec succès aux problèmes combinatoires tels que les problèmes d'ordonnancements ou de transports.

L'application d'un algorithme génétique consiste en la génération successive de meilleurs individus en fonction du problème choisi, celui-ci est défini par les composants suivants :

- une représentation des solutions potentielles.
- une manière de créer une première population,
- une fonction d'évaluation *eval*.
- les opérateurs génétiques qui définissent la composition des enfants : deux opérateurs différents seront ici considérés : *le croisement* et *la mutation*.
- les paramètres : taille de population, les probabilités de croisement et de mutation.

Génétique : un processus d'optimisation

Dans le contexte des AG, pour la résolution d'un CSP donné (X, D, C), l'espace de recherche peut être habituellement défini par l'ensemble des n-uplets de $D = D_1 \times \cdots \times D_n$. Nous considérons des populations g de taille i, $g \subseteq D$. Un élément $s \in g$ est un individu et représente une solution potentielle du problème.

Les fonctions d'évaluation fournissent des informations quant à la qualité d'un individu et donc, d'une population. Ainsi, nous considérons la fonction d'évaluation $eval_{ind}$, telle que $eval_{ind}(s)$ représente le nombre de contraintes non satisfaites couplé avec le critère d'optimisation pour un individu s.

Nous étendons cette notion d'évaluation aux populations par $eval_{gen}$telle que $eval_{gen}(g)$ représente l'évaluation des individus de la population g. Cette fonction $eval_{gen}(g)$, peut représenter par exemple, la somme de toutes les évaluations de chacun des individus.

Les algorithmes génétiques visent à produire une nouvelle population en utilisant la selection et des opérateurs génétiques : recombinaison et mutation. Cette nouvelle population s'appelle la descendance.

Nous définissons l'ensemble des descendances possibles, c.-à-d., les séquences de populations par : $\mathcal{AG}$ Du point de vue des AG, un résultat est soit une population g qui contient une solution, soit un chemin de recherche d'une taille donnée maximum.

Structure de calcul

Afin de manipuler les différentes structures de données associées à chaque facette de la résolution, nous ajoutons un facteur génétique particulier à l'espace de recherche pour chaque CSP, sur lequel l'AG fonctionnera et où l'optimisation sera effectuée.

Définition 1 *Un CSP avec facteur génétique pour l'optimisation (ogCSP) est défini par la séquence* (D, C, p, f) *où*

- $D = D_1, ..., D_n$
- $\forall c \in C, c \subseteq D_1 \times \cdots \times D_n$
- $p \in \mathcal{AG}$
- f *: fonction objectif.*

Du point de vue de CP, une solution d'un ogCSP est un tuple qui satisfait toutes les contraintes. Du point de vue des AG, la notion de solution est liée à la fonction d'évaluation qui définit une solution comme un élément $s \in p$ tels que s est le minimum (ou le maximum) pour la fonction objectif.

Soit un ogCSP $\psi = (D, C, p, f)$, ces deux points de vue induisent deux ensembles de solutions :

- Solutions faisables : $Sol_{CP}(\psi) = \{d \in D | \forall c \in C,\, d \in c\}$
- Solutions optimales : $Sol_{\mathcal{AG}}(\psi) = \{s \mid p = (g_1, \ldots, g_m) \forall i \in [1..m], \forall s' \in g_i, eval_{ind}(s) \leq eval_{ind}(s')\}$.

Ainsi, avec les propriétés précédentes, l'AG optimise la fonction objectif, prenant ses valeurs dans un espace de recherche qui devient localement consistant avec CP. Avec des propagations de contraintes et des

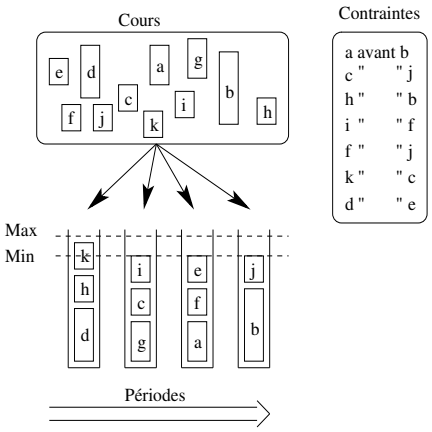

FIG. 1 – Exemple de distribution des cours.

découpage successifs, l'espace de recherche est progressivement limité au solutions faisables, parmi lesquelles l'AG doit trouver l'optimum.

3 Expérimentation

Dans cette section, nous présentons le problème BACP suivi de l'implémentation d'un prototype, développée en C++, qui nous a permis de tester nos hybridations. Le but de cette section n'est pas d'examiner un algorithme efficace sur des jeux de tests à grande échelle mais de mettre en avant l'avantage de notre cadre pour la résolution hybride de ce problème.

L'équilibrage des programmes universitaires

Le problème est de planifier les cours composant un diplôme afin d'équilibrer la charge des étudiants pour chaque période universitires. Chaque cours est doté d'un certain nombre de crédits représentant la quantité de travail nécessaire pour le suivre avec succès. La charge d'une période est la somme des crédits de chaque cours de la période. D'autres contraintes sont ajoutées : une charge maximale et minimale par période, et des rapports de précédence sont établis entre certains cours (figure 1).

Ainsi, une solution est une affectation équitable des cours aux périodes, ce que nous traduisons comme la minimisation de la période qui a la charge la plus élevée. Pour une description plus détaillée, le lecteur peut se référer à [4].

Instantiation de l'ensemble de fonctions

Nos fonctions de base sont organisées en trois ensembles : un premier ensemble des fonctions de réduction de domaines *dr*, l'ensemble des fonctions de découpage *sp*, et l'ensemble des fonctions d'AG : *ag*.

Notre algorithme génétique produit à partir d'une population k, une nouvelle population k+1 de 60 individus choisis parmi 100 issus de k.

Qualité de la solution	bacp 8	Qualité de la solution	bacp 10
24	137,08	33	9,11
23	218,23	32	25,38
21	218,43	30	25,65
20	712,84	29	1433,18
19	1441,98	27	1433,48
18	1453,73	26	1626,49
17(optimum)	1459,73	24	1626,84

FIG. 2 – Résultats en secondes avec lp_solve

Qualité de la solution	bacp 8	bacp 10	bacp 12
24	0,47	4,71	2,34
23	0,54	4,67	2,40
22	0,61	3,68	2,48
21	0,61	4,36	2,76
20	0,69	4,63	3,20
19	0,83	4,95	4,25
18	1,20	5,13	35,20
17	15,05(optimum)	5,60	
16		6,39	
15		8,53	
14		34,84(optimum)	

FIG. 3 – Résultats en secondes avec AG+CP

Les contraintes de prérequis sont facilement converties en contraintes binaires et abstraites sous forme de fonctions de réduction pour la consistance d'arc sur les domaines des cours. Les contraintes sur les périodes, la somme des charges et du nombre de cours, sont représentées en tant que contraintes globales et utilisées pour élaguer l'arbre de recherche en détectant des inconsistances.

Résultats expérimentaux

On considère les problèmes bacp8, bacp10 et bacp12 issues de la CSPlib [5]. Afin de donner un point de référence, nous présentons les résultats de [4] utilisant lp_solve pour les problèmes 8-périodes et 10-périodes (figure 2). Si lp_solve peut trouver la solution optimale pour le premier, ce n'est pas le cas pour le second.

Dans [8], nous avons présenté le système de résolution à base de contraintes établi pour l'hybridation de la propagation contraintes avec la recherche locale. Nous avons réutilisé notre système et y avons intégré le module d'AG (c.-à-d., les fonctions *ag*).

Nous avons également ajouté la notion d'optimisation. Plus précisément, nous avons adapté la fonction d'évaluation utilisée pour la recherche locale.

Le plus intéressant dans une telle hybridation est la complètude de l'association AG-CP, et les rôles

que AG et CP jouent dans le processus de recherche (voir les schémas 4, 5, et 6) : AG optimise les solutions dans l'espace de recherche progressivement rendu localement consistant en utilisant la propagation de contraintes et le découpage.

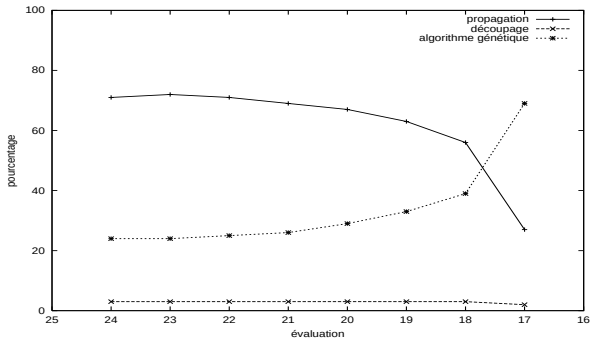

Fig. 4 – Evolution de CP vs AG durant le processus d'optimisation (problème de 8-périodes)

Afin d'évaluer les bénéfices de chacun des composants, nous avons mesuré pour CP : le nombre de réduction effectives et le nombre de découpages, et pour AG : le fait que la génération suivante est globalement meilleure que la précédente.

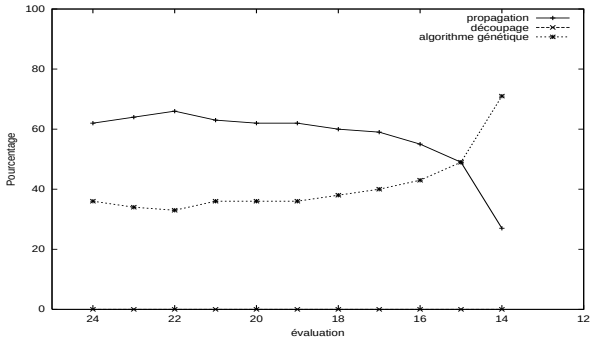

Fig. 5 – Evolution de CP vs AG durant le processus d'optimisation (problème de 10-périodes)

Au début, CP représente 70% de l'effort de recherche : la propagation de contrainte réduit l'espace de recherche. A contrario, l'AG représente environ 30%. Pendant cette période, trop peu de consistences locales sont atteintes par la propagation de contraintes, et l'AG ne trouve que des solutions avec un coût supérieur à 21. Puis, au début de la deuxième moitié du processus de recherche, en termes de coûts, CP et AG convergent l'un vers l'autre : les sous-problèmes ont en majoritairement atteint la propriété de consistance locale et les tests sur les contraintes n'améliorent plus les domaines. À la fin, AG exécute

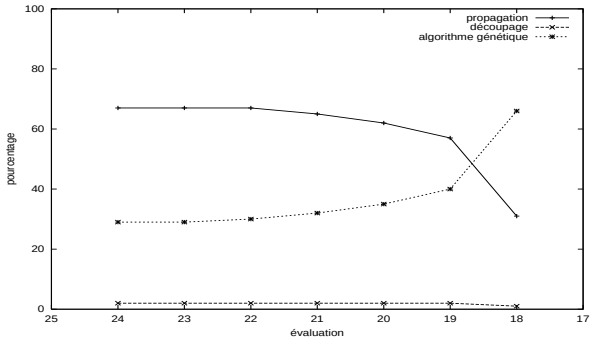

Fig. 6 – Evolution de CP vs AG durant le processus d'optimisation (problème de 12-périodes)

70% de l'effort de recherche pour trouver la solution optimale.

Dans cet article, nous avons employé un cadre général approprié pour modéliser la résolution des problèmes d'optimisation. Les résultats sur le problème permettent d'identifier les interactions entre les différents mécanismes de résolution. De telles études peuvent être employées pour la mise au point de solveurs hybrides robustes.

Références

[1] E. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization.* John Wiley & Sons, Inc., 1997.

[2] K. R. Apt. *Principles of Constraint Programming.* Cambridge Univ. Press, 2003.

[3] Nicolas Barnier and Pascal Brisset. Combine and conquer : Genetic algorithm and cp for optimization. In *CP*, page 463, 1998.

[4] C. Castro and S. Manzano. Variable and value ordering when solving balanced academic curriculum problems. In *Proc. of 6th Workshop of the ERCIM WG on Constraints*, 2001.

[5] I. Gent, T. Walsh, and B. Selman. http ://4c.ucc.ie /∼tw/csplib/, funded by the UK Network of Constraints.

[6] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co, Inc, 1989.

[7] J. H. Holland. *Adaptation in Natural and Artificial Systems.* 1975.

[8] T. Lambert, E. Monfroy, and F. Saubion. Solving strategies using a hybridization model for local search and constraint propagation. In *Proceedings of ACM SAC'2005.* ACM, 2005.