

Conception d'un algorithme de maintien de solutions dans un reseau de contraintes

Gilles Trombettoni

▶ **To cite this version:**

Gilles Trombettoni. Conception d'un algorithme de maintien de solutions dans un reseau de contraintes. [Rapport de recherche] RR-1784, INRIA. 1992. inria-00077024

HAL Id: inria-00077024
https://inria.hal.science/inria-00077024

Submitted on 29 May 2006

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.

UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

2004 route des Lucioles
B.P. 93
06902 Sophia-Antipolis
France

Rapports de Recherche

N°1784

Programme 2

Calcul symbolique, Programmation
et Génie logiciel

CONCEPTION D'UN ALGORITHME DE MAINTIEN DE SOLUTION DANS UN RÉSEAU DE CONTRAINTES

Gilles TROMBETTONI de MENTON

Novembre 92

Résumé

La plupart des problèmes de contraintes sont résolus par des méthodes de satisfaction appropriées qui garantissent la complétude mais pas nécessairement la performance.
Une autre interprétation des contraintes a comme point de départ une solution existante, perturbée par l'ajout de nouvelles contraintes et/ou l'altération de certaines variables.

Les algorithmes capables de rétablir la cohérence du système de contraintes, dits de maintien de solution, se divisent en plusieurs catégories. Le type d'algorithmes développé dans ce rapport fonctionne par propagation des valeurs, c'est à dire qu'il propage ses modifications sur le réseau de contraintes et rétablit chacune d'entre elles par ajustement local d'une variable de recalcul. L'algorithme fournit, de manière incrémentale et avec une complexité linéaire par rapport au nombre de variables, une solution proche de celle perturbée, mais ne vérifie plus la propriété de complétude.

L'algorithme conçu sur ce modèle se déroule en deux phases où la première planifie les ajustements nécessaires et la deuxième rétablit la cohérence en se basant sur ce plan. Il tire profit des contraintes numériques inégalitaires (ou de différence) et du domaine des variables impliquées. Il peut enfin intégrer des méthodes globales apparentées à des méthodes de satisfaction. La dernière partie offre des fonctionnalités supplémentaires pour mieux modéliser certains problèmes. On obtient finalement une bibliothèque de fonctions Lisp qui permet de manipuler contraintes et variables, et de déclencher le maintien de solution.

Abstract

Constraints problems are usually solved by specific satisfaction methods which insure completness but not always efficiency.
Another constraints interpretation consists in trying to recover the consistency from an existant solution, disturbed by the addition of new constraints and/or updates of variables.

Solution maintenance algorithms are based on values propagation. They propagate modifications on the constraints network and maintain the consistency of each constraint by altering the value of the variable. This type of algorithms is incremental. Its complexity is linear w.r.t. the number of variables.

The resulting algorithm works in two steps. The first one builds a plan for the required updates. The second one reinstates the consistency using this plan. It takes advantage of non-unique constraints and domains of the variables involved in. Finally, it can integrate global methods that are close to constraints satisfaction methods. The last part of this report offers additional features to enhance the expressiveness of the system.
The system is implemented as a Lisp function library.

Table des matières

1 Introduction 4

2 Le maintien de cohérence **6**
2.1 Conventions d'écriture et de représentation 6
2.2 Définition du maintien de cohérence 7
2.3 Exemple de maintien de cohérence 7
2.4 Principe de résolution 8
2.5 Remarques concernant le maintien de cohérence 8
2.6 Existant 9
2.6.1 Prose 10
2.6.2 Algorithme 10

3 Renforcement de l'algorithme **15**
3.1 Introduction 15
3.2 Phase de construction 15
3.2.1 Acceptation des cycles et évitement de certains 15
3.2.2 Principe de construction du flot 16
3.2.3 Conclusion 17
3.3 Phase d'évaluation 19
3.3.1 Introduction. 19
3.3.2 Différentes causes d'échec 19
3.3.3 Changer la mécanique de résolution 19
3.3.4 Les nœuds indéterministes 21
3.3.5 Conclusion 28
3.4 Méthodes globales de résolution 28
3.4.1 Introduction 28
3.4.2 Transformations requises 29
3.4.3 Remarques 29
3.4.4 Conclusion 30
3.5 Les cycles 31
3.5.1 Introduction 31
3.5.2 Existant. 31
3.5.3 Notre contribution 31

3.5.4 Conclusion 34
3.6 Conclusion 34

4 Capacités de modélisation et de paramétrage **35**
4.1 Introduction 35
4.2 Paramétrage 35
4.2.1 Contrôle de l'algorithme de maintien de cohérence 35
4.2.2 Gestion des solutions 36
4.3 Les liens causaux 36
4.3.1 Motivations 36
4.3.2 Intégration des liens causaux à l'algorithme existant 37
4.3.3 Conclusion 39
4.4 Les contraintes unilatérales 39
4.4.1 Motivations 39
4.4.2 Expression des relations affranchi-esclave avec des liens causaux 41
4.4.3 Le traitement des contraintes unilatérales intégré à l'algorithme de maintien 42
4.4.4 Conclusion 42
4.5 Répartition de la perturbation d'une contrainte sur plusieurs variables 43
4.5.1 Introduction 43
4.5.2 Un exemple concret 43
4.5.3 Intégration 45
4.5.4 Conclusion 46
4.6 Conclusion 46

5 Conclusion **47**

Chapitre 1

Introduction

Pour résoudre un problème, deux principaux critères sont à prendre en compte, la modélisation du problème et la facilité d'expression des connaissances qui s'y rapportent ainsi que des méthodes de résolution adéquates.
Une approche de plus en plus employée et qui semble fournir un cadre de formalisation intéressant consiste à exprimer en partie ou en totalité les informations sous forme de contraintes.
La formulation de la résolution d'un problème de contraintes est déclarative. Elle se ramène à définir ce que doit être une solution, c'est à dire définir les propriétés qu'elle doit vérifier.
Il existe des algorithmes destinés à résoudre l'ensemble des contraintes afin de déterminer une ou plusieurs solutions au problème. Certains problèmes nécessitent des algorithmes pouvant déterminer les incohérences entre les contraintes.
La notion de contraintes n'est pas nouvelle puisqu'on peut considérer les équations et inéquations sur les réels comme des contraintes algébriques et des algorithmes comme le simplexe et Gauss comme des méthodes de résolution appropriées.
Plus récemment, des travaux dans le cadre de la programmation en logique et des problèmes de satisfaction de contraintes[1], issus de l'intelligence artificielle, ont donné naissance à des systèmes ou des langages qui permettent une véritable programmation par contraintes.
En programmation en logique, l'introduction de méthodes de satisfaction de contraintes dans le mécanisme de résolution classique a donné naissance à une classe de langages pour la programmation logique avec contraintes[2]. CLP(R), Chip [DHS+88] et PrologIII [Col89] sont les CLP les plus connus.
Certains problèmes de satisfaction de contraintes constituent un ensemble de problèmes à très forte combinatoire (NP-Complets) dont les variables ont un domaine de valeurs fini. De nombreux systèmes qui utilisent des algorithmes permettant de réduire cette combinatoire ont été conçus. Citons Charme [Cha90], Pecos [Pec91] et Prose [Ber92].
Tous les systèmes précités ont un point commun : ils permettent d'obtenir une ou plusieurs solutions en affectant une valeur à chacune des variables du problème. Une autre catégorie de systèmes a le rôle complémentaire de conserver une solution après perturbation. Ces algorithmes sont appelés algorithmes de maintien de cohérence ou encore de maintien de solution.
S'ils ne sont pas très puissants (ils ne fournissent pas toujours une solution alors qu'il en existe), fait qu'ils soient incrémentaux et qu'ils fournissent des solutions proches de la solution initiale,

[1] en anglais "Constraint Satisfaction Problem" : CSP

[2] en anglais "Constraint Logical Programming" : CLP

justifie leur attrait.
Un langage nommé Constraints [SS80] est basé uniquement sur le maintien de solution et de nombreux logiciels en sont clients : dans le Cornell Synthesizer [Reps88], un générateur d'environnements de programmation, un algorithme permet de maintenir la cohérence des attributs décorant la grammaire. Des systèmes de construction interactive d'objets graphiques comme Thinglab [Bor79] ou Magritte [Gos83] ou bien les tableurs (feuilles de calcul informatisées) sont basés sur le même principe.
Ce rapport porte sur la conception et l'implantation d'un tel algorithme en s'inspirant notamment de l'algorithme de maintien de cohérence de Prose.
La première partie présente le maintien de cohérence et les algorithmes existants.
La deuxième partie porte sur le renforcement même du coeur de l'algorithme, sur une modification de la mécanique de résolution.
La troisième partie montre le paramétrage de l'algorithme qui permet de contrôler le processus de résolution ainsi que le nombre de solutions et introduit de nouvelles fonctionnalités qui offrent à l'utilisateur une capacité de modélisation des problèmes plus grande.