Octogones entiers pour le problème RCPSP

JFPC 2019

Pierre Talbot, David Cachera, Éric Monfroy, Charlotte Truchet
{pierre.talbot}@univ-nantes.fr

Université de Nantes
Laboratoire des Sciences du Numérique de Nantes (LS2N)
Équipe TASC

12 juin 2019

UNIVERSITÉ DE NANTES

Introduction

Projet AbSolute (Pelleau and al., 2013)

- Solveur de contraintes mixtes réelles et entières en OCaml.
- Basé sur la théorie de l'interprétation abstraite (notamment les **domaines abstraits**) et de la programmation par contraintes (PPC).

Concrètement on remplace le problème de satisfaction de contrainte (CSP) $\langle d, P \rangle$ par un élément d'un domaine abstrait.

Solveur classique VS solveur par interprétation abstraite

Un solveur classique en PPC :

$\texttt{solve}(\langle d, P\rangle)$
$\langle d', P\rangle \leftarrow \texttt{propagate}(\langle d, P\rangle)$
if $d' = \{a\}$ **then**
  return $\{a\}$
else if $d' = \{\}$ **then**
  return $\{\}$
else
  $\langle d_1, \ldots, d_n\rangle \leftarrow \texttt{branch}(d')$
  return $\bigcup_{i=0}^{n} \texttt{solve}(\langle d_i, P\rangle)$
end if

Solveur classique VS solveur par interprétation abstraite

Solveur par interprétation abstraite, soit *Abs* un domaine abstrait :

$\texttt{solve}(a \in Abs)$
$a \leftarrow \texttt{closure(a)}$
if $\texttt{state(a)} = \texttt{true}$ **then**
  return $\{a\}$
else if $\texttt{state(a)} = \texttt{false}$ **then**
  return $\{\}$
else
  $\langle a_1, \ldots, a_n \rangle \leftarrow \texttt{split(a)}$
  return $\bigcup_{i=0}^{n} \texttt{solve}(a_i)$
end if

Problématique

Les contraintes globales sont au cœur de la performance des solveurs de PPC actuels mais :

- Il en existe beaucoup (> 400).
- La plupart sont très spécialisées.
- Seulement un sous-ensemble implémenté dans les solveurs PPC.

Il faut réfléchir à des méthodes plus générales.
On propose les **domaines abstraits**.

Contraintes globales VS domaines abstraits

- Contraintes globales : capture une sous-structure + algorithme efficace pour cette structure.
- Domaine abstrait pour la PPC :
  - Représentation exacte, ou par sur-approximation d'un langage de contraintes.
  - Ensemble partiellement ordonné d'éléments équipé avec plusieurs opérations (consistence, entailment, join, ...).
- Plusieurs domaines abstraits **entiers et continus** : intervalle, octogone, polyhèdre, etc.
- Des **combinaisons/transformations** sur ces domaines : produit réduit, produit réduit par réification, partionnement, etc.

Contributions

- Adaptation du domaine abstrait des **octogones entiers** pour AbSolute.
- Création d'un **produit réduit générique** où les domaines communiquent par contraintes réifiées.
- On est guidé par une application d'ordonnancement : *problème de gestion de projet à contraintes de ressources* (RCPSP, RCPSP/max).

Décomposer les contraintes globales vers les domaines abstraits.

Plan

- Introduction
- Le problème RCPSP
- Domaines abstraits pour RCPSP
- Conclusion

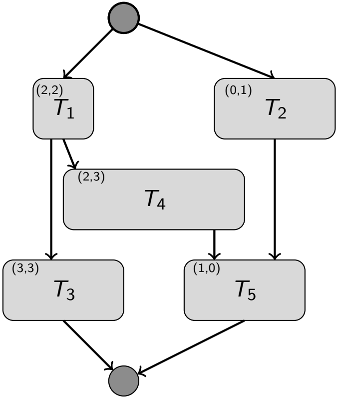
Le problème d'ordonnancement RCPSP

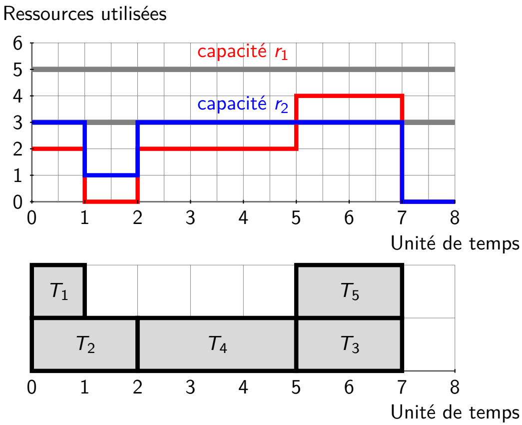
Problème d'optimisation NP-complet :

- T un ensemble de tâches, $d_i \in \mathbb{N}$ la durée de la tâche i.
- P les précédences entre les tâches : $i \ll j \in P$ si i doit terminer avant que j commence.
- R un ensemble de ressources où $k \in R$ a une capacité $c_k \in \mathbb{N}$.
- Chaque tâche i utilise une quantité $r_{k,i}$ de ressource k.

Objectif : trouver un planning (minimum) des tâches T respectant les priorités dans P sans excéder la capacité des ressources disponibles.

Exemple avec 5 tâches et deux ressources





Modèle par contraintes

- **Variables de décision** : $s_i \in \{0..h-1\}$ est la date de début de la tâche i.
- **Contraintes** :

$$\forall (i \ll j) \in P,\; s_i + d_i \leq s_j \tag{1}$$

$$\begin{aligned} &\forall j \in [1..n],\; \forall i \in [1..n] \setminus \{j\}, \\ &\quad b_{i,j} \Leftrightarrow (s_i \leq s_j \land s_j < s_i + d_i) \end{aligned} \tag{2}$$

$$\forall j \in [1..n],\; r_{k,j} + \Big(\sum_{i \in [1..n] \setminus \{j\}} r_{k,i} * b_{i,j} \Big) \leq c_k \tag{3}$$

1. Contraintes temporelles (eq. 1) : gérées par le domaine abstrait des octogones.
2. Contraintes de ressources (eq. 2 et 3) : *décomposition par tâches* de `cumulative` vers contraintes d'intervalles et octogonales.

Trois types de contraintes

- En vert : contraintes octogonales.
- En rouge : contraintes réifiées.
- En bleu : contraintes d'intervalles.

$$\forall (i \ll j) \in P,\ s_i + d_i \leq s_j$$

$$\begin{aligned} &\forall j \in [1..n],\ \forall i \in [1..n] \setminus \{j\}, \\ &\qquad b_{i,j} \Leftrightarrow (s_i \leq s_j \wedge s_j < s_i + d_i) \end{aligned}$$

$$\forall j \in [1..n],\ r_{k,j} + (\sum_{i \in [1..n] \setminus \{j\}} r_{k,i} * b_{i,j}) \leq c_k$$

Les contraintes réifiées permettent de **connecter** le domaine des intervalles et des octogones.

Plan

- Introduction
- Le problème RCPSP
- Domaines abstraits pour RCPSP
- Conclusion

Domaine abstrait pour la PPC

Treillis complet $\langle Abs, \leq \rangle$ représentables en machine avec :

- $\perp$ est le plus petit élément.
- $\sqcup$ est l'opérateur d'union (*join*) de deux éléments.
- $[\![.]\!] : C \to Abs$ est une fonction partielle transformant une contrainte en un élément du domaine abstrait.
- $closure : Abs \to Abs$ est le propagateur des contraintes du domaine abstrait.
- $entailment : Abs \times Abs \to \{T, F, U\}$: relaxation de l'ordre partielle, on peut répondre *unknown* si *true* ou *false* est trop difficile à calculer.
- ...

Octogone entier (Miné, 2004)

Un octogone entier est défini sur un ensemble de variables $(x_0, \ldots, x_{n-1})$ et contraintes

$$\pm x_i - \pm x_j \leq d$$

où $d \in \mathbb{Z}$ est une constante.

Complexité des opérations générales :

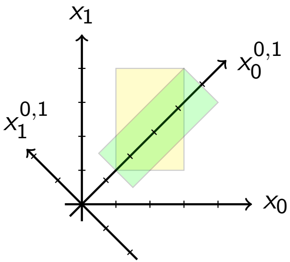
- *join* en $\mathcal{O}(n^2)$.
- *closure* : algorithme de Floyd-Warshall en $\mathcal{O}(n^3)$, version incrémentale en $\mathcal{O}(n^2)$ pour l'ajout d'une seule contrainte (Chawdhary and al., 2018). Forme normale équivalente à la **consistance de chemin** (Dechter and al., 1991).
- *entailment* en $\mathcal{O}(n^2)$, et en temps constant pour une seule contrainte.

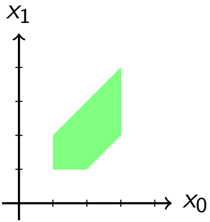
Exemple d'octogone entier

Soit les contraintes :

$$x_0 \geq 1 \wedge x_0 \leq 3 \qquad x_1 \geq 1 \wedge x_1 \leq 4$$
$$x_0 - x_1 \leq 1 \qquad -x_0 + x_1 \leq 1$$

Les contraintes de borne sur x_0 et x_1 sont représentées par la boîte en jaune, et les contraintes octogonales par la boîte en vert.





Produit direct : combinaison naïve de domaines abstraits

Nous pouvons définir un produit direct sur $Box \times Oct$ de la manière suivante :

$$(b, o) \sqcup (b', o') = (box \sqcup_{Box} b', o \sqcup_{Oct} o')$$

$$[\![c]\!] = \begin{cases} ([\![c]\!]_{Box}, [\![c]\!]_{Oct}) & \\ ([\![c]\!]_{Box}, \bot_{Oct}) & \text{si } [\![c]\!]_{Oct} \text{ n'est pas défini} \\ (\bot_{Box}, [\![c]\!]_{Oct}) & \text{si } [\![c]\!]_{Box} \text{ n'est pas défini} \end{cases}$$

$$closure((b, o)) = (closure(b), closure(o))$$

Problème : les domaines n'échangent pas d'information.

Produit réduit via contraintes réifiées

On crée un nouveau produit réduit permettant de connecter les contraintes de deux domaines abstraits via réification.

- Soit $c_1 \Leftrightarrow c_2$ une contrainte d'équivalence où $[\![c_1]\!]_{Box}$ et $[\![c_2]\!]_{Oct}$ sont définis, nous avons :

$$
\begin{aligned}
&prop_{\Leftrightarrow}(b, o, c_1 \Leftrightarrow c_2) := \\
&\quad\begin{cases}
b \models [\![c_1]\!]_{Box} \implies (b, o \sqcup [\![c_2]\!]_{Oct}) \\
b \not\models [\![c_1]\!]_{Box} \implies (b, o \sqcup [\![\neg c_2]\!]_{Oct}) \\
o \models [\![c_2]\!]_{Oct} \implies (b \sqcup [\![c_1]\!]_{Box}, o) \\
o \not\models [\![c_2]\!]_{Oct} \implies (b \sqcup [\![\neg c_1]\!]_{Box}, o)
\end{cases}
\end{aligned}
$$

Produit réduit via contraintes réifiées

On améliore l'opérateur de clôture en propageant les contraintes réifiées R.
Opérateur de clôture du produit réduit $Box \times Oct$:

$$closure_R(box, oct, R) = (\bigsqcup_{r \in R} prop_{\Leftrightarrow}(box, oct, r), R)$$

$$\begin{aligned} & closure((box, oct, R)) = \\ & \quad (closure_R(closure(box'), closure(oct'), R)) \end{aligned}$$

Soit e un élément du produit réduit, l'opérateur de clôture peut être appliqué jusqu'à réaliser le point fixe de $closure(e) = e$.

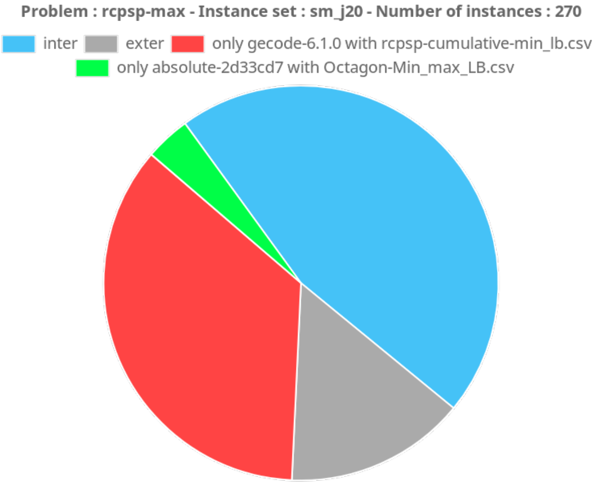
Résultat : Un produit réduit générique pour combiner des domaines abstraits.

Plan

- Introduction
- Le problème RCPSP
- Domaines abstraits pour RCPSP
- Conclusion

Évaluation

- En bref : performances bien en deça de l'état de l'art (Chuffed avec le *lazy clause*).
- En comparaison avec GeCode/cumulative :



Conclusion

- On ajoute à AbSolute le domaine abstrait des octogones entiers, et une nouvelle combinaison : le produit réduit par réification de contraintes.
- Perspectives :
  - Décomposition de la globale `sequence`, nombreuses extensions du RCPSP.
  - Clarification des liens avec les solveurs SMT (théories VS domaines abstraits).

`github.com/ptal/AbSolute`

Merci de votre attention