Optimisation lexicographique pseudo-booléenne : Une contrainte globale

Pierre-Alain Yvars[1], Mathieu Roisin [2], David Annebicque[2], Bernard Riera[2]
[1] ISAE-Supméca, Quartz, EA7393, France
pierre-alain.yvars@isae-supmeca.fr
[2] CReSTIC, Université de Reims Champagne-Ardenne, Reims, France
{mathieu.roisin, david.annebicque, bernard.riera}@univ-reims.fr

Mots-clés *: Programmation par contraintes, contrainte globale, optimisation lexicographique, synthèse de contrôleur logique.*

1 Introduction

Un problème d'optimisation lexicographique est un problème multi-objectif dans lequel chaque objectif doit-être minimisé ou maximisé dans un ordre donné. Lorsque les objectifs sont à valeurs pseudo-booléennes dans {0,1} on parle alors d'optimisation pseudo-booléenne lexicographique [1]. Nous proposons une contrainte globale permettant d'exprimer et de traiter un objectif de ce type dans un CSP à l'aide de techniques de programmation par contraintes.

2 Une contrainte globale pour l'optimisation lexicographique pseudo-booléenne

Soit le problème de satisfaction de contraintes *CSP(V, D, C)* avec :
- *V* : l'ensemble des variables du problème,
- *D* : l'ensemble des domaines des variables de *V*,
- *C* : l'ensemble des contraintes à satisfaire par les variables de *V* dans *D*.

Soit $O = \{o_1, \ldots, o_p\}$ l'ensemble des *p* variables d'optimisation totalement ordonné avec $O \subset V$ tel que $\forall i \in \{1, \ldots, p\},\ o_i \in \{0,1\}$.

Soit $\Lambda = \{\lambda_1, \ldots, \lambda_p\}$ l'ensemble des *p* opérations d'optimisation à réaliser sur chacune des variables de *O* tel que $\forall i \in \{1, \ldots, p\}, \lambda_i \in \{min, max\}$.

On souhaite représenter le problème d'optimisation lexicographique pseudo-booléen suivant : 1er objectif : $\lambda_1\ o_1$, 2ème objectif : $\lambda_2\ o_2$, ..., pème objectif *:* $\lambda_p\ o_p$.

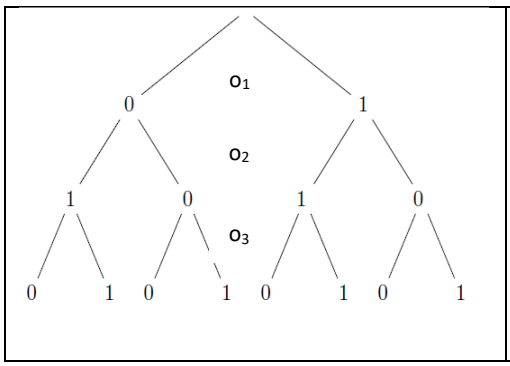
Alors, on définit la contrainte globale *Blo* (pseudo *B*oolean *l*exicographic *o*ptimisation) comme suit : *Blo*$([\lambda_1, \ldots, \lambda_p], [o_1, \ldots, o_p])$.

Le principe général de la contrainte *Blo* est de générer à la pose de la contrainte une table de données binaire correctement ordonnée et lors de la propagation de filtrer cette table de données à la manière d'une contrainte en extension bien connue dans la littérature [3]. La génération de la table s'effectue en deux temps :
- Génération d'un arbre binaire des choix de valeurs binaires dans l'ordre des *p* objectifs : Pour chaque nœud de l'arbre, ses deux successeurs seront générés avec la règle suivante : si l'on est sur un nœud *min* (resp *max*) , alors la valeur 0 (resp 1) sera celle du successeur à gauche et la valeur 1 (resp 0) sera celle du successeur à droite.
- Génération des chemins ordonnés de l'arbre précédent et création de la table de données correspondante : Cette génération est réalisée en parcourant l'arbre de manière préfixée de la

gauche vers la droite. Chaque chemin constitue alors un tuple qui sera inscrit dans la table. On est alors certain que la table est totalement ordonnée selon les combinaisons d'objectifs à atteindre et par ordre croissant d'une variable interne d'index à valeur dans le domaine fini $\{1, \ldots, 2^p\}$.

La propagation s'effectuera alors par réduction des lignes de la table à l'aide d'un algorithme connu de filtrage de type contrainte en extension appelée également contrainte table [3]. Pour assurer le respect de l'ordre il suffira de modifier la stratégie de recherche du solveur utilisé de manière à commencer par examiner les valeurs de la variable d'index de la table en commençant par la borne inférieure de son domaine.

La Figure 1 illustre ce principe sur le cas d'un problème à trois objectifs o_1, o_2, o_3 sur lesquels on souhaite faire l'optimisation lexicographique *min* o_1, *max* o_2 *et min* o_3.



index	o_1	o_2	o_3
1	0	1	0
2	0	1	1
3	0	0	0
4	0	0	1
5	1	1	0
6	1	1	1
7	1	0	0
8	1	0	1

FIG. 1 – Arbre binaire ordonné min o_1, max o_2, min o_3 et table correspondante

3 Conclusions et perspectives

La contrainte *Blo* a été implémentée dans le solveur de programmation par contraintes en domaines mixtes (domaines finis + domaines continus) de l'environnement DEPS Studio[1] ainsi que dans le langage DEPS[2] [4] et a été mise en œuvre sur plusieurs cas d'étude de synthèse de contrôleur logique pour la commande de système à évènements discrets [2].

4 Références

[1] Joao Marques-Silva, Josep Argelich, Ana Graça and Inês Lynce. Boolean lexicographic optimization: algorithms & applications. *Annals of Mathematics and Artificial Intelligence,* 62 : 317–343, 2011.

[2] Mathieu Roisin, Dimitri Renard, David Annebicque, Pierre-Alain Yvars et Bernard Riera. *De la modélisation par intension du problème de commande logique à la génération de code ST (IEC 61131-3).* 15ème conférence sur la modélisation des systèmes réactifs, MSR'25, 2025.

[3] Hélène Verhaeghe. *The extensional constraint*. Phd Thesis, UCL, Louvain, 2021.

[4] Pierre-Alain Yvars and Laurent Zimmer. DEPS: a model- and property-based language for system synthesis problems. *Software and Systems Modeling* 23 : 973–1002, 2024.

[1] https://www.depslink.com

[2] https://www.documentation.depslink.fr/