



CORNFLOWER

A Library for Global Constraints, Hybrid Search Methods and Symmetry Breaking

Meinolf Sellmann, Serdar Kadioglu and Yuri Malitsky
 Department of Computer Science, Brown University
 {sello,serdark,ynm}@cs.brown.edu

INTRODUCTION

Cornflower Project aims to solve the algorithmic problems that arise in the context of optimization driven decision support systems. Our main goal is to develop techniques that allow inexperienced users to exploit optimization power efficiently. We integrate and hybridize ideas developed in different communities. Particularly, we focus on development of high-level constraints that allow users to model the problems as conjunctions of intuitive substructures and provide hybrid methods for their efficient combination. Moreover, we develop automatization techniques for the handling of symmetries that can be the cause of severe inefficiencies when handled poorly.

GLOBAL CONSTRAINTS

We devise propagation algorithms for global constraints which are very appealing because of two different aspects:

- They facilitate problem modeling, because problem specification becomes a much more intuitive process when larger semantic building blocks offer more expressiveness, which is essential in order to facilitate access to computational optimization support.
- They allow the solver to be aware and to exploit problem structure that does not need to be crushed into pieces to adapt for a modeling language of limited expressiveness — as it is the case, for instance, in mathematical programming, where all problems must be stated as conjunction of linear inequalities.

HYBRID SEARCH METHODS

Our first goal is to devise new and more robust solution techniques that automatically adapt to the problem or problem instance that needs to be solved. Particularly, we investigate how the efficiency of exact solvers can be boosted by heuristics, randomization, and learning.

Second, by exploiting methods from mathematical programming and constraint programming we devise methods for the tight integration of constraints that can be employed by solvers automatically, for example via the exchange of dual information.

CONTEXT-FREE GRAMMAR CONSTRAINTS

Example: Language of all correctly bracketed expressions.

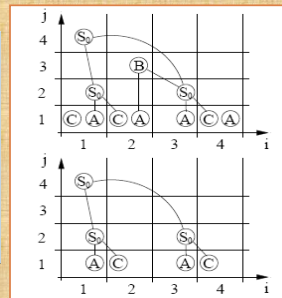
$S_0 \rightarrow S_0 S_0$ $S_0 \rightarrow AC$ $S_0 \rightarrow BC$
 $B \rightarrow AS_0$ $A \rightarrow [$ $C \rightarrow]$

[] [] [[]]

Context-Free Grammar G

Initial Domains

1) The filtering algorithm first works bottom-up by computing sets S_j for increasing j after initializing S_{1i} with all non-terminals that can produce in one step a terminal in the domain of X_i .



2) Then the algorithm works top-down removing all non-terminals from each set S_j which cannot be reached from $S_0 \in S_{1n}$ (Kadioglu, Sellmann AAAI'08).



Updated Domains

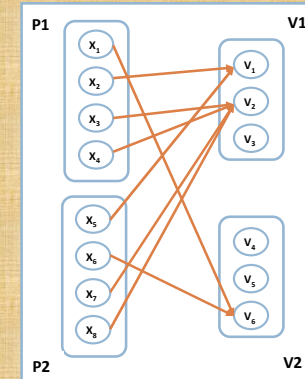
KNAPSACK CONSTRAINT

Given a knapsack problem and threshold values for the total profit of the included items and the accumulated weight of the included items; the Knapsack Constraint incrementally computes which items that must be included or excluded (Malitsky *et al* CP'08).

SYMMETRY BREAKING (SB)

Detection of symmetric parts of the search space in order to avoid redundant work is essential for the solution of symmetric combinatorial problems. However, the existing methods to handle symmetries in constraint programs require deep understanding and skill from the user. We aim to provide algorithms that can break automatically detectable symmetries, thus making symmetry breaking part of the problem solution even for users who are not aware of symmetries at all.

STATIC STRUCTURAL SB



We define the *signature* of a value v to be the tuple that counts for each variable partition, the number of variables x that are assigned to the value v [Flener *et al*, 2006].

For instance, under the assignment that is shown on the left the signatures for each value are;

- $v_2 \in V_1$ has signature (2,2)
- $v_1 \in V_1$ has signature (1,1)
- $v_6 \in V_2$ has signature (1,1)

□ $X_1 \leq X_2 \leq X_3 \leq X_4$ and $X_5 \leq X_6 \leq X_7 \leq X_8$

□ GlobalCardinality ($[X_1, \dots, X_4]$, $[v_1, \dots, v_6]$, $[\text{count}_{v_1}^1, \dots, \text{count}_{v_6}^1]$)

□ GlobalCardinality ($[X_5, \dots, X_8]$, $[v_1, \dots, v_6]$, $[\text{count}_{v_1}^2, \dots, \text{count}_{v_6}^2]$)

□ $(\text{count}_{v_1}^1, \text{count}_{v_1}^2) \geq_{\text{LEX}} (\text{count}_{v_2}^1, \text{count}_{v_2}^2) \geq_{\text{LEX}} (\text{count}_{v_3}^1, \text{count}_{v_3}^2)$

□ $(\text{count}_{v_4}^1, \text{count}_{v_4}^2) \geq_{\text{LEX}} (\text{count}_{v_5}^1, \text{count}_{v_5}^2) \geq_{\text{LEX}} (\text{count}_{v_6}^1, \text{count}_{v_6}^2)$

AVAILABLE METHODS

- **Context-Free Grammar Constraints:** Incremental and non-incremental filtering algorithms as an extension to ILOG Solver and as a standalone C++ application.
- **Knapsack Constraint:** Incremental filtering algorithm as a standalone C++ application.
- **Static Structural Symmetry Breaking:** As an extension to ILOG Solver to handle piecewise variable and value symmetry.

www.cs.brown.edu/research/cornflower