Improved Constraint Propagation via Lagrangian Decomposition

David Bergman[1], Andre A. Cire[2], and Willem-Jan van Hoeve[3]

[1] School of Business, University of Connecticut
[2] University of Toronto Scarborough
[3] Tepper School of Business, Carnegie Mellon University

david.bergman@business.uconn.edu,acire@utsc.utoronto.ca,
vanhoeve@andrew.cmu.edu

Abstract. Constraint propagation is inherently restricted to the local information that is available to each propagator. We propose to improve the communication between constraints by introducing Lagrangian penalty costs between pairs of constraints, based on the Lagrangian decomposition scheme. The role of these penalties is to force variable assignments in each of the constraints to correspond to one another. We apply this approach to constraints that can be represented by decision diagrams, and show that propagating Lagrangian cost information can help improve the overall bound computation as well as the solution time.

1 Introduction

Modern finite-domain constraint programming (CP) solvers employ a constraint propagation process in which domain changes for the variables are propagated between constraints. To allow for more communication and knowledge sharing between constraints, several techniques have been proposed. One possibility is to propagate more structural information than variable domains, such as (relaxed) decision diagrams [1, 10]. Another option, in the context of optimization problems, is to combine constraints with the objective function, and utilize mathematical programming relaxations for stronger cost-based filtering [7, 15]. These approaches, however, have in common that consistency checks are done separately and independently for each constraint. Higher-order consistencies, such as pairwise consistency [12] can consider multiple constraints simultaneously, but may suffer from a relatively high computational cost.

We propose an alternative, and generic, approach to improve the propagation between constraints based on *Lagrangian decomposition* [9]. In Lagrangian decomposition, the constraint set of a given problem is partitioned into structured subproblems, each of which is defined on a duplicate copy of the original variables. To link the subproblems, constraints are added to ensure that each of the duplicates is equal to the original variable. These latter constraints are then relaxed with an associated Lagrangian multiplier, and moved to the objective. This results in independent subproblems that can be separately optimized.

Intuitively, the idea is to force the variables to take the same value in each constraint, via the Lagrangian penalties, which are iteratively updated. This will somehow synchronize the consistency checks for each of the constraints; instead of allowing each constraint to check its consistency w.r.t. an arbitrary tuple, we iteratively arrive at tuples with minimal disagreement.

Since constraint programming has been designed to work with (global) constraints that capture a specific structure of the problem, the application of Lagrangian decomposition in this context seems natural and promising. Indeed, we show that the Lagrangian decomposition is not only useful to improve the bound on the objective, but can also be applied for cost-based domain filtering.

The structure of the paper is as follows. We first provide an overview of the most relevant related work in Section 2. In Section 3 we recall the Lagrangian decomposition scheme. We apply this to constraint programming models in Section 4. Experimental results on instances with multiple `alldiff` constraints are given in Section 5, while Section 6 provides results on set covering problems. We conclude in Section 7.

2 Related Work

Lagrangian relaxations have been widely applied in operations research as well as constraint programming. One of the first applications in CP is the work by Benoist et al. [2] on the Traveling Tournament Problem. A formal treatment was provided by Sellmann [16] who showed that optimal Lagrangian multipliers may not result in the most effective domain filtering. Recently, [8] introduced a framework for automated Lagrangian relaxation in a constraint programming context. That work explicitly generalizes Lagrangian relaxations to CP problems using measures of constraint violations, or degrees of satisfiability.

Adapting weights for improving propagation has also been applied in the context of Valued Constraint Satisfaction Problems [6]. In that work, a linear programming model is proposed for computing Optimal Soft Arc Consistency, but Lagrangian relaxations are not used. Khemmoudj et al. [13] combine arc consistency with Lagrangian relaxation for filtering constraint satisfaction problems (CSPs). They consider binary CSPs (i.e., each constraint has at most two variables in its scope) in extensional form. Lastly, Bergman et al. [3] introduce Lagrangian relaxations in the context of propagating (relaxed) decision diagrams.

3 Lagrangian Decomposition

Lagrangian decomposition has been introduced to strengthen Lagrangian bounds for integer linear optimization problems [9]. Consider an integer linear program of the form:

$$(P) \quad \max\{fx \mid Ax \leq b, Cx \leq d, x \in X\},$$

for some feasible set X, where $x \in \mathbb{R}^n$ is a vector of variables, $f \in \mathbb{R}^n$ represents a 'weight' vector, A and C represent constraint coefficient matrices, and b and

c are constant right-hand size vectors. This is equivalent to the reformulated program

$$\max\{fx \mid Ay \leq b, Cx \leq d, x = y, x \in X, y \in Y\},$$

for any set Y containing X.

The Lagrangian decomposition of P consists in dualizing the *equality* constraints $x = y$ with Lagrangian multipliers $\lambda \in \mathbb{R}^n$:

$$\begin{aligned} L_P(\lambda) &:= \max\{fx + \lambda(y - x) \mid Cx \leq d, x \in X, Ay \leq b, y \in Y\} \\ &= \max\{(f - \lambda)x \mid Cx \leq d, x \in X\} + \max\{\lambda y \mid Ay \leq by \in Y\} \end{aligned}$$

The *Lagrangian dual* is to find those Lagrangian multipliers λ that provide the best bound:

$$\min_{\lambda} L_P(\lambda).$$

Guignard and Kim [9] show that the optimal bound obtained from this Lagrangian decomposition is at least as strong as the standard Lagrangian bounds from dualizing either $Ax \leq b$ or $Cx \leq d$. Lagrangian decomposition may be particularly useful when the problem is composed of several well-structured subproblems, such as those defined by (global) constraints in CP models.

4 Application to Constraint Programming

We apply Lagrangian decomposition to constraint optimization problems (COPs), which include constraint satisfaction problems (CSPs) as special case. It is important to note that this approach will transform each of the original constraints into an 'optimization constraint'; instead of representing a witness for feasibility the constraint now has to represent a witness for *optimality*, even if the constraint is not directly linked to the objective function, or in case of feasibility problems.

When the variables have numeric domains, the method from Section 3 can be directly applied. In general, however, domains need not be numeric, and we will therefore focus our discussion on this more general case. Consider a COP with variables $x_1, \ldots, x_n$ that have given finite domains $x_i \in D_i$:

$$\begin{array}{lll} \max f(x_1, \ldots, x_n) & & \\ \text{s.t.} \quad C_j(x_1, \ldots, x_n) & j \in \{1, \ldots, m\} & \\ \quad x_i \in D_i & i \in \{1, \ldots, n\} & \end{array} \tag{1}$$

For simplicity we assume here that all variables appear in all constraints, but we can allow a different subset of variables for each constraint. Also, C_j may represent any substructure, for example a global constraint, a table constraint, or a collection of constraints. We introduce for each variable x_i and each constraint $j = 1, \ldots, m$ a duplicate variable y_i^j with domain D_i. We let the set y_i^1 represent our 'base' variables, to which we will compare the variables y_i^j for $j = 2, \ldots, m$.

The reformulated COP is as follows:

$$\begin{array}{lll} \max & f(y_1^1, \ldots, y_n^1) & \\ \text{s.t.} & C_j(y_1^j, \ldots, y_n^j) & j \in \{1, \ldots, m\} \\ & y_i^1 = y_i^j & i \in \{1, \ldots, n\}, j \in \{2, \ldots, m\} \\ & y_i^j \in D_i & i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\} \end{array}$$

To establish the Lagrangian decomposition, we relax the constraints $y_i^1 = y_i^j$ and move these into the objective as $y_i^1 \neq y_i^j$ with associated Lagrangian multipliers. To measure its violation, we propose to represent $y_i^1 \neq y_i^j$ with the set of constraints $((y_i^j = v) - (y_i^1 = v))$ for all $v \in D_i$, where $(y_i^j = v)$ is interpreted as a binary value representing the truth value of the expression. Lastly, we define a Lagrangian multiplier for each i, j $(j \geq 2)$ and each $v \in D(x_i)$ as a vector

$$\overline{\lambda}_i^j := \lambda_i^j[v].$$

The Lagrangian objective function can then be written as:

$$\begin{aligned} \max \quad & f(y_1^1, \ldots, y_n^1) + \sum_{j=2}^{m} \sum_{i=1}^{n} \sum_{v \in D(x_i)} \lambda_i^j[v]((y_i^j = v) - (y_i^1 = v)) \\ = \; & f(y_1^1, \ldots, y_n^1) + \sum_{j=2}^{m} \sum_{i=1}^{n} \left(\lambda_i^j[y_i^j] - \lambda_i^j[y_i^1] \right) \end{aligned}$$

This leads to the following decomposition (for any given set of multipliers $\overline{\lambda}_i^j$):

$$\begin{aligned} \max & \left\{ f(y_1^1, \ldots, y_n^1) - \sum_{j=2}^{m} \sum_{i=1}^{n} \lambda_i^j[y_i^1] \mid C_1(y_1^1, \ldots, y_n^1) \right\} \\ & + \sum_{j=2}^{m} \left(\max \left\{ \sum_{i=1}^{n} \lambda_i^j[y_i^j] \mid C_j(y_1^j, \ldots, y_n^j) \right\} \right) \end{aligned}$$

which are m independent subproblems. Let z_j be the optimal objective value for subproblem $j \in \{1, \ldots, m\}$. Then $\sum_{j=1}^{m} z_j$ is a valid bound on $f(x_1, \ldots, x_n)$. Note that the duplicate variables have only been introduced for the formal description of the method. In practice, all constraints C_j use the original variables.

Design choices The Lagrangian decomposition scheme can be adapted by allocating parts of original objective to different subproblems. Moreover, we can introduce equality constraints between any pair of subproblems. We will illustrate the latter in the following example.

Example 1. Consider the following CSP:

$C_1 :$ `alldiff`$(x_1, x_2, x_3) \quad C_2 :$ `alldiff`$(x_2, x_4, x_5) \quad C_3 :$ `alldiff`(x_3, x_5)
$x_1 \in \{a, b\}, x_2 \in \{b, c\}, x_3 \in \{a, c\}, x_4 \in \{a, b\}, x_5 \in \{a, b, c\}$

This CSP is domain consistent as well as pairwise consistent, and has one solution $(x_1, x_2, x_3, x_4, x_5) = (b, c, a, a, b)$.

We construct a Lagrangian decomposition based on the constraints C_1, C_2, C_3. To link these, we only need to introduce the constraints $y_2^2 = y_2^1$, $y_3^3 = y_3^1$, $y_5^3 = y_5^2$, and their associated multipliers. This yields the following three subproblems, with respective objective values z_1, z_2, z_3:

$$z_1 = \max \left\{ -\overline{\lambda}_2^2[y_2^1] - \overline{\lambda}_3^3[y_3^1] \mid \texttt{alldiff}(y_1^1, y_2^1, y_3^1) \right\}$$
$$z_2 = \max \left\{ \overline{\lambda}_2^2[y_2^2] - \overline{\lambda}_5^3[y_5^2] \mid \texttt{alldiff}(y_2^2, y_4^2, y_5^2) \right\}$$
$$z_3 = \max \left\{ \overline{\lambda}_3^3[y_3^3] + \overline{\lambda}_5^3[y_5^3] \mid \texttt{alldiff}(y_3^3, y_5^3) \right\}$$

This CSP can be considered as a COP with a zero-valued objective function so that the value $z_1 + z_2 + z_3$ is an upper bound on the satisfiability of this problem, for any Lagrangian multipliers; if the bound is below zero, the problem is unsatisfiable. And so, the optimal Lagrangian decomposition bound is 0. □

Cost-based domain filtering In addition to pruning the search based on the overall bound $L_P(\lambda)$ and a given lower bound B, we can apply cost-based domain filtering. The difference with existing cost-based filtering methods is that the bounds from the different subproblems can all be conditioned on a specific variable/value pair. To this end, let $z_j|_{x_i=v}$ be the optimal objective value for subproblem $j \in \{1, \ldots, m\}$ in which $y_i^j = v$. We have the following result:

Proposition 1. *If $\sum_j z_j|_{x_i=v} < B$ then v can be removed from D_i.*

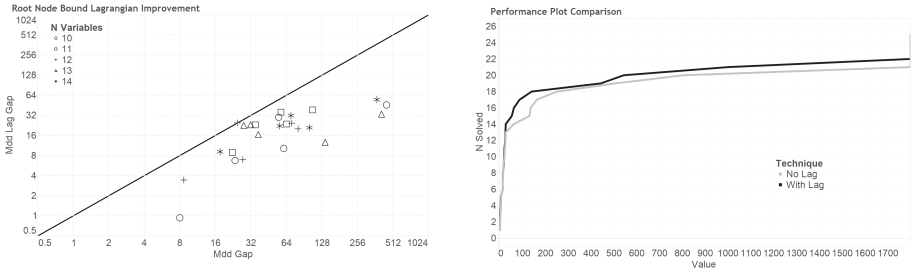
This result may be particularly effective when there is no single subproblem that collects all variables. We continue our example to give an illustration.

Example 2. Continuing Example 1, consider the following Lagrangian multipliers (all others are zero): $\lambda_2^2[b] = 0.5, \lambda_3^3[a] = 0.5, \lambda_5^3[a] = 0.5$. This yields $z_1 = -0.5$, $z_2 = 0.5$, $z_3 = 0.5$, and a total bound of 0.5. Even though this is not optimal, when we condition $x_2 = b$ or $x_5 = c$, the bound becomes -0.5 in both cases, and by Proposition 1 we can remove those values from their respective domains. We can similarly remove values a from D_1, b from D_2, and c from D_3 using the multipliers $\lambda_2^2[c] = -0.5, \lambda_3^3[c] = 0.5, \lambda_5^3[c] = 0.5$. □

Example 2 implies the following result:

Proposition 2. *Cost-based filtering based on Lagrangian decomposition can be stronger than pairwise consistency.*

Implementation issues To apply Lagrangian propagation efficiently, it is important that each constraint is optimized efficiently. For many constraints optimization versions are already available [11], to which the Lagrangian costs can be immediately added. For example, in our experiments we represent constraints by decision diagrams, which permit to find the optimal solution quickly via a shortest (or longest) path calculation. Also, cost-based propagators are available that



a. Root node gap comparison

b. Performance plot

Fig. 1: Evaluating the impact of the Lagrangian decomposition on systems of multiple `alldiff` constraints. (a) compares the root node gap obtained with Lagrangian decomposition (Mdd Lag Gap) and without (Mdd Gap) and (b) depicts a performance profile comparing the number of instances solved (N Solved) within a given time limit (horizontal axis) with Lagrangian decomposition (With Lag) and without (No Lag).

filter sub-optimal arcs from the decision diagram. Second, the search for optimal Lagrangian multipliers can be done with different methods [14]. Regardless, any set of multipliers results in a valid relaxation, and we do not necessarily need to solve the Lagrangian dual to optimality. In our implementation, we compute the multipliers once at the root node and reuse them during the CP search process.

5 Application: Multiple Alldifferent Constraints

As first application, we consider systems of multiple overlapping `alldiff` constraints, as in [1]. These are defined on a set $X = \{x_1, \ldots, x_n\}$ of variables with domain $\{1, \ldots, n\}$. Each `alldiff` constraint is defined on a subset of variables $S_j \subset X$, for $j = 1, \ldots, k$. We then consider the following COP:

$$\max \left\{ \sum_{i=1}^{n} w_i x_i \mid \texttt{alldiff}(S_j) \quad \forall j \in \{1, \ldots, k\} \right\}$$

We generated instances with $n = 10, 11, 12, 13, 14$, $k = 4$, and $|S_j| = n - 2$ for all $j = 1, \ldots, 5$. For the Lagrangian decomposition, we partition the `alldiff` constraints into two arbitrary subsets of size two, and define one multi-valued decision diagram (MDD) for each subset. In other words, we apply MDD propagation to these subsets of `alldiff` constraints. The two MDDs thus formed are the basis for the Lagrangian decomposition, which follows the description in Section 4 (where the j-th MDD represents constraint set C_j).

We implemented the MDD propagation as well as the Lagrangian decomposition as a global constraint in IBM ILOG CPO 12.6, similar to [5]. The (near-)optimal Lagrangian multipliers were computed using the Kelly-Cheney-Goldstein method [14], using IBM ILOG CPLEX 12.6 as the linear programming

solver. We fix the CP search to be lexicographic in the order of the variables, to ensure the search tree is the same across all instances. We compare the performance with and without Lagrangian multipliers.

In Figure 1.a we show the root node percent gap for the 25 instances (where the optimal value was obtained by formulating an integer linear program and solving the instances using CPLEX). The reduction in the gap can be substantial, in some case several orders of magnitude. This reduction in the optimality gap and additional cost-based filtering due to the Lagrangian multipliers enables more instances to be solved in shorter computational time, as depicted in Figure 1.b. Depending on the configuration of the `alldiff` systems the improvement can be marginal and in some cases negligible.

6 Application: Set Covering

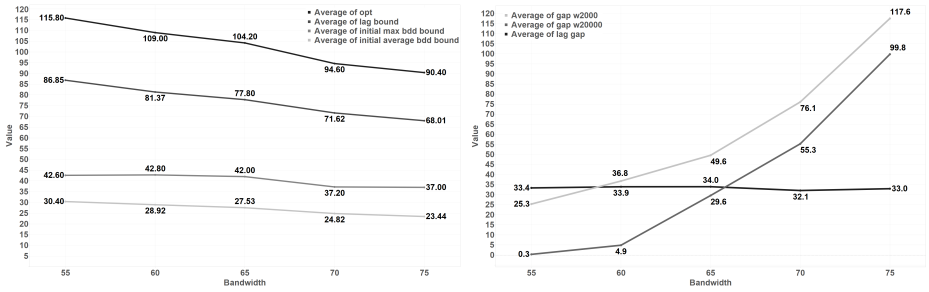
The set covering problem is defined on a universe of n elements $U = \{1, \ldots, n\}$. Given a collection of subsets $C_1, \ldots, C_m \subseteq U$ and weights w_i $(i = 1, \ldots, n)$, the problem is to find a set of elements $S \subset U$ of minimum total weight such that all $S \cap C_j$ is not empty for all $j = 1, \ldots, m$. Using a binary variable x_i to represent whether element i is in S, the problem can be formulated as the following COP:

$$\min \left\{ \sum_{i=1}^{n} w_i x_i \mid \sum_{i \in C_j} x_i \geq 1 \quad \forall j \in \{1, \ldots, m\}, x_i \in \{0, 1\} \quad \forall i \in \{1, \ldots, n\} \right\}$$

Instead of defining a subproblem for each separate constraint, we create exact binary decision diagram (BDD) representations for collections of them. That is, using the construction method described in [4], we create a BDD by adding constraints one at the time, until the exact width exceeds a given limit (in our case 100 nodes on any layer). We then create the next BDD, and so forth. This forms a partition of the constraint set, each of which is represented by an exact BDD. For the instances we considered, we construct 10 or 11 BDDs per instance.

We also slightly modify the Lagrangian decomposition method by representing the original objective function in each of the BDDs, and dualizing constraints $x_i^j = x_i^{j'}$ for every pair (j, j'). Hence, the Lagrangian bound is no longer the sum of the bounds of the respective BDDs, rather the average over the objectives.

In previous work [4], it was shown that the bounds from BDDs were most effective when the constraint matrix has a relatively small bandwidth. We therefore used the same benchmark generator to evaluate the impact of the Lagrangian decomposition for increasing bandwidths. We generated instances with $n = 150$ variables, randomly generated costs, and uniform-randomly selected subsets C_j from within a given bandwidth of size 55 to 75 (five instances for each bandwidth). To generate the costs, we let $c(i)$ represent the number of subsets C_j that contain element i. Then the cost for variable x_i is taken uniform randomly in $[0.75 * c(i), 1.25 * c(i)]$. The results are shown in Figure 2, showing the average over the five instances per bandwidth. Figure 2.a depicts four lines: the optimal



a. Impact of Lagrangian decomposition b. Comparison with single relaxed BDD

Fig. 2: Evaluating the bound from Lagrangian decomposition for set covering problems of varying bandwidth.

solution (found by CPLEX), the average bound without using Lagrangian decomposition, the maximum bound without using Lagrangian decomposition, and lastly the average bound when using the Lagrangian decomposition. Lagrangian decomposition generates bounds of much better quality than the independent BDDs. For example, for bandwidth 65 the average bound of 27.53 is improved to 77.80 using the Lagrangian decomposition, on average.

We also compare the Lagrangian decomposition to the original BDD *relaxation* from [4] that represents all constraints in a single BDD respecting a given maximum width. A larger width leads to a stronger relaxation and better bounds. Figure 2.b compares the percent gap (between the lower bound and the optimal solution) of the BDD relaxation for maximum widths 2,000 and 20,000 with that of the Lagrangian decomposition. We note that the BDD relaxation with maximum width 2,000 has about the same memory requirements as the separate BDDs for the Lagrangian decomposition. As the bandwidth increases, the quality of BDD relaxation rapidly declines, while the Lagrangian decomposition is much more stable and outperforms the BDD relaxation (decreasing the gap from 117.6% to 33% for bandwidth 75 and maximum width 2,000). This demonstrates that Lagrangian decompositions can be used to improve BDD-based optimization when a single BDD relaxation can no longer provide sufficient power to represent the entire problem. We do note, however, that the Lagrangian decomposition takes more time to compute (on average 60s) compared to the single BDD relaxation (on average 1.4s for width 2,000 and 17s for width 20,000).

7 Conclusion

We have introduced Lagrangian decomposition in the context of constraint programming as a generic approach to improve the constraint propagation process. The key idea is that we penalize variables in different constraints to take different assignments. We have shown how this approach can be utilized for stronger cost-based domain filtering, and that it leads to improved bounds for systems of `alldiff` constraints and set covering problems.

References

1. Andersen, H.R., Hadzic, T., Hooker, J.N., Tiedemann, P.: A Constraint Store Based on Multivalued Decision Diagrams. In: Proceedings of CP. LNCS, vol. 4741, pp. 118–132. Springer (2007)
2. Benoist, T., Laburthe, F., Rottembourg, B.: Lagrange relaxation and constraint programming collaborative schemes for traveling tournament problems. In: Proceedings of the International Workshop on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2001) (2001)
3. Bergman, D., Cire, A.A., van Hoeve, W.J.: Lagrangian Bounds from Decision Diagrams. Constraints 20(3), 346–361 (2015)
4. Bergman, D., van Hoeve, W.J., Hooker, J.N.: Manipulating MDD Relaxations for Combinatorial Optimization. In: Proceedings of CPAIOR. LNCS, vol. 6697, pp. 20–35. Springer (2011)
5. Cire, A.A., van Hoeve, W.J.: Multivalued Decision Diagrams for Sequencing Problems. Operations Research 61(6), 1411–1428 (2013)
6. Cooper, M.C., de Givry, S., Sanchez, M., Schiex, T., Zytnicki, M., Werner, T.: Soft arc consistency revisited. Artificial Intelligence 174(7–8), 449–478 (2010)
7. Focacci, F., Lodi, A., Milano, M.: Cost-based domain filtering. In: Proceedings of CP. LNCS, vol. 1713, pp. 189–203. Springer (1999)
8. Fontaine, D., Michel, L., Van Hentenryck, P.: Constraint-based lagrangian relaxation. In: Proceedings of CP. LNCS, vol. 8656, pp. 324–339 (2014)
9. Guignard, M., Kim, S.: Lagrangian Decomposition: A Model Yielding Stronger Lagrangian Bounds. Mathematical Programming 39, 215–228 (1987)
10. Hoda, S., van Hoeve, W.J., Hooker, J.N.: A Systematic Approach to MDD-Based Constraint Programming. In: Proceedings of CP. LNCS, vol. 6308, pp. 266–280. Springer (2010)
11. van Hoeve, W.J., Katriel, I.: Global constraints. In: Handbook of Constraint Programming, pp. 169–208. Elsevier (2006)
12. Janssen, P., Jégou, P., Nouguier, B., Vilarem, M.C.: A Filtering Process for General Constraint-Satisfaction Problems: Achieving Pairwise-Consistency Using an Associated Binary Representation. In: IEEE International Workshop on Tools for Artificial Intelligence, 1989. Architectures, Languages and Algorithms. pp. 420–427. IEEE (1989)
13. Khemmoudj, M.O., Bennaceur, H., Nagih, A.: Combining arc consistency and dual Lagrangean relaxation for filtering CSPs. In: Proceedings of CPAIOR. LNCS, vol. 3524, pp. 258–272. Springer (2005)
14. Lemaréchal, C.: Lagrangian Relaxation. In: Jünger, M., Naddef, D. (eds.) Computational Combinatorial Optimization, LNCS, vol. 2241, pp. 112–156. Springer (2001)
15. Régin, J.C.: Arc Consistency for Global Cardinality Constraints with Costs. In: Proceedings of CP. LNCS, vol. 1713, pp. 390–404. Springer (1999)
16. Sellmann, M.: Theoretical Foundations of CP-Based Lagrangian Relaxation. In: Proceedings of CP. LNCS, vol. 3258, pp. 634–647 (2004)