Linear Relaxations and Reduced-Cost Based Propagation of Continuous Variable Subscripts

Greger Ottosson[1] and Erlendur S. Thorsteinsson[2]

[1] Information Technology, Department of Computing Science; Uppsala University; PO Box 311; S-751 05 Uppsala; Sweden. greger@csd.uu.se
[2] Graduate School of Industrial Administration; Carnegie Mellon University; Schenley Park; Pittsburgh, PA 15213; U.S.A. esth@cmu.edu

Abstract In hybrid solvers for combinatorial optimization, combining Constraint (Logic) Programming (CLP) and Mixed Integer Programming (MIP), it is important to have tight connections between the two domains. We extend and generalize previous work on automatic linearizations and propagation of symbolic CLP constraints that cross the boundary between CLP and MIP. We also present how reduced costs from the linear programming relaxation can be used for domain reduction on the CLP side. Computational results comparing our hybrid approach with pure CLP and MIP on a configuration problem show significant speed-ups.

1 Introduction

The topic of this paper can be seen as the merger of three lines of research in the area of hybrid techniques of Constraint (Logic) Programming (CLP) and Mixed Integer Programming (MIP).

Firstly, while studying a configuration problem, we continue along the lines of Rodošek, Wallace and Hajian [16] in providing automatic linearizations of arithmetic and symbolic CLP constraints. The object of study here are terms with variable subscripts in continuous constraints, i.e., a structure which in part is modeled with the `element` constraint in CLP.

Secondly, we continue a line of research by Focacci, Lodi and Milano [5, 6, 7, 8], who have used the *reduced costs* of a separate assignment subproblem for propagation in a CLP framework. We generalize this by showing how reduced-cost based inference can be applied to constraints whose linearization is a part of a larger linear programming relaxation.

Finally, we show how this fits nicely into the modeling framework Mixed Logical/Linear Programming (MLLP) and a hybrid CLP–MIP solver, which is a part of our previous line of research [9, 10, 14, 15].

This paper is structured as follows. Section 2 introduces our application, a class of configuration problems, with CLP, MIP and hybrid MLLP models. Sections 3 and 4 describe how the variable subscripts are linearized in MLLP. Section 5 describes reduced costs and how they can be used for inference using those linearizations. Finally, Section 6 gives computational results.

2 A Configuration Problem

Many industrial products come in different configurations, aiming to closely satisfy the needs of individual customers. In one class of such problems there are components, each one of a set of possible types, and the aim is to find a feasible configuration with a type and quantity for each component that optimizes some criteria. Components supply or consume attributes/resources (such as weight, cost or effect), and these resources are constrained or are a part of the objective. In our problem all quantities are integral and in addition there are a set of logical side-constraints, the *configuration* constraints.

Given is a set of components C, possible component types T_i for component i, and a set of attributes R. Variable t_i is the type of component i, q_i is the quantity of component i and r_k is the quantity of attribute/resource k. The weight/cost per unit of attribute/resource k is denoted by c_k, R_k is the minimum amount of attribute/resource k used/produced and $A_{k,i,j}$ defines how many units of attribute/resource k are produced/consumed by each component i if it is of type j. Let $q = (q_i)_{i \in C}$ and $t = (t_i)_{i \in C}$. Then,

$$
\begin{aligned}
\min \quad & \sum_{k \in R} c_k r_k \\
\text{s.t.} \quad & r_k = \sum_{i \in C} \Big(q_i \times A_{k,i,t_i} \Big), && \forall k \in R, && (1) \\
& h_l(q,t), && \forall l \in L, && (2) \\
& r_k \geq R_k, && \forall k \in R, && (3) \\
& t_i \in T_i,\ q_i \in \mathbb{Z}_+, && \forall i \in C.
\end{aligned}
$$

Note that the resource consumption "lookup" is handled in (1) through a variable subscript on A, i.e., A_{k,i,t_i} where the *variable* t_i is the type of component i. This term is the core of the problem, and is, as we shall see, modeled differently in CLP and IP. Equation (2) states the set of configuration constraints, e.g., $q_1 > 0 \Rightarrow q_2 = 0$ or $\texttt{alldiff}(t_1, \ldots, t_n)$.

2.1 A CLP Model

In CLP the configuration problem is modeled with variable subscripts (`element` constraints) in the following way (assuming that $T_i = \{1, \ldots, n_i\}$):

$$
\begin{aligned}
\min \quad & cr \\
\text{s.t.} \quad & r_k = \sum_{i \in C} q_i a_{ki}, && \forall k \in R, && (4) \\
& \texttt{element}(t_i, [A_{ki1}, \ldots, A_{kin_i}], a_{ki}), && \forall i \in C, k \in R, && (5) \\
& r_k \geq R_k, && \forall k \in R, && (6) \\
& t_1 = 1 \Rightarrow t_2 \in \{1,2\}, && && (7) \\
& q_1 > 0 \Rightarrow q_2 = 0, && && (8) \\
& t_i \in T_i,\ q_i \in \mathbb{Z}_+,\ a_{ki} \in \mathbb{Q}, && \forall i \in C, k \in R.
\end{aligned}
$$

where a_{ki} is the consumption of resource k by component i, and the other variables and constants as before. Constraint (7)–(8) are two examples of a logical side-constraints. Note that (4) is non-linear and that there are $|C| \times |R|$ `element` constraints.

2.2 A MIP Model

In a MIP model, the lack of variable subscripts and the linear requirement means that the component types t_i have to be replaced with 0–1 variables t_{ij}, where t_{ij} is 1 if component i has type j, 0 otherwise. Similarly, the quantity of the i-th component, q_i, is disaggregated into q_{ij} for types $j \in T_i$ (constraints (11)–(12)):

$$
\begin{aligned}
\min \quad & cr \\
\text{s.t.} \quad & r_k = \sum_{i \in C} \sum_{j \in T_i} q_{ij} A_{kij}, && \forall k \in R, && (9) \\
& \sum_{j \in T_i} t_{ij} = 1, && \forall i \in C, && (10) \\
& q_{ij} \leq M t_{ij}, && \forall i \in C, j \in T_i, && (11) \\
& q_i = \sum_{j \in T_i} q_{ij}, && \forall i \in C, && (12) \\
& r_k \geq R_k, && \forall k \in R, && (13) \\
& t_{21} + t_{22} \geq t_{11}, && && (14) \\
& b \leq q_1 \leq M b,\ q_2 \leq M(1-b), && && (15) \\
& t_{ij} \in \{0,1\},\ q_i, q_{ij} \in \mathbb{Z}_+,\ b \in \{0,1\}, && \forall i \in C, j \in T_i. &&
\end{aligned}
$$

Equations (14)–(15) in this model formulate the configuration constraints from before.

2.3 An MLLP Model

The MLLP model is similar to the CLP model, although the underlying solution strategy employs LP relaxations in conjunction with the constraint propagation:

$$
\begin{aligned}
\min \quad & cr \\
\text{s.t.} \quad & r_k = \sum_{i \in C} q_i A_{kit_i}, && \forall k \in R, && (16) \\
& r_k \geq R_k, && \forall k \in R, && (17) \\
& t_1 \in \{1\} \Rightarrow t_2 \in \{1,2\}, && && (18) \\
& b \in \{0\} \Rightarrow q_1 = 0,\ b \in \{1\} \Rightarrow (q_1 \geq 1, q_2 = 0), && && (19) \\
& t_i \in T_i,\ q_i \in \mathbb{Z}_+,\ b \in \{0,1\}, && \forall i \in C. &&
\end{aligned}
$$

Note that the cost "lookup" is handled through a variable subscript on A, i.e., by A_{kit_i} where the variable t_i is the type of component i. The next two sections describe the linear relaxations used for these variable subscripts, and how the relaxations link the discrete and continuous parts of the MLLP model.

3 Linear Relaxation of c_y

The term $q_i \times A_{k,i,t_i}$ is the core of the configuration problem. We will, however, begin with a simpler linearization, of a single subscripted constant c_y. This would be the term which would occur in the MLLP model if it was limited to unit quantities, i.e., if $q_i \equiv 1$, and is equivalent to the traditional `element` constraint in CLP. A subscripted constant, c_y, occurring alone in a term is compiled as in the following example:

$$\begin{array}{l} 2x + c_y \leq 18, \\ c \in \{1.0, 4.5, 6.1\}, \end{array} \iff \begin{array}{l} 2x + z \leq 18, \\ \texttt{element}(y, [1.0, 4.5, 6.1], z). \end{array}$$

Bounds-consistency on the `element` constraint involves producing increasingly tighter bounds on z as the domain D_y of y is reduced. A straight-forward linearization of this constraint is identical to how this structure is modeled in MIP, i.e., we introduce decision variables $b_1, \ldots, b_k$ for $D_y = \{1, \ldots, k\}$, where b_i is 1 if $y = i$ and 0 otherwise, and add

$$\begin{array}{rll} & 0 \leq b_i \leq 1, \ \forall i, & (20) \\ \texttt{element}(y, [c_1, \ldots, c_k], z) \iff & b_1 + \cdots + b_k = 1, & (21) \\ & z = c_1 b_1 + \cdots + c_k b_k. & (22) \end{array}$$

to the LP. This linear relaxation is no stronger than bounds-consistency on the `element` constraint (actually equivalent to it). Thus there is little incentive to use this relaxation unless these linear constraints connect in a beneficial way to some other linear constraints, or useful information (dual values, reduced costs, etc.) can be derived and used by including them.

4 Linear Relaxation of $c_y x$

We now turn our attention back to the basic structure (1) of the configuration problem. The term $c_y x$ is a subscripted constant multiplied by a continuous variable. In the configuration problem it denotes the cost of buying x units of type y at cost c_y each. In MLLP this structure is replaced by a continuous variable z when the model is compiled and the constraint $\texttt{element}(y, [c_1, \ldots, c_k] \times x, z)$ is introduced into the problem. For example

$$\begin{array}{l} 2x + c_y x \leq 18, \\ c \in \{1.0, 4.5, 6.1\}, \end{array} \iff \begin{array}{l} 2x + z \leq 18, \\ \texttt{element}(y, [1.0, 4.5, 6.1] \times x, z). \end{array}$$

When propagating this variant of the `element` constraint upon a change of D_y, the domain of y, cuts of the form

$$c_{\min} x \leq z \qquad \text{and} \qquad c_{\max} x \geq z \tag{23}$$

are updated, where $c_{\min} = \min\{c_i : i \in D_y\}$ and $c_{\max} = \max\{c_i : i \in D_y\}$.

This is compact, but a stronger relaxation of $\texttt{element}(y, [c_1, \ldots, c_k] \times x, z)$ is achieved by disaggregating x into bins x_i, and linking to the corresponding decision variable z:

$$0 \leq b_i \leq 1, \qquad \forall i, \qquad (24)$$

$$b_1 + \cdots + b_k = 1, \qquad (25)$$

$$x = x_1 + \cdots + x_k, \qquad (26)$$

$$z = c_1 x_1 + \cdots + c_k x_k, \qquad (27)$$

$$x_i \geq 0, \qquad \forall i, \qquad (28)$$

$$x_i \leq M b_i, \qquad \forall i. \qquad (29)$$

Equations (24)–(25) are the same as for c_y above, but in addition, x is disaggregated to $x_1, \ldots, x_k$, and connected through the big-M constraints (29) to $b_1, \ldots, b_k$. Intuitively, this relaxation is stronger than bounds-consistency simply because all the coefficients are weighed into the LP relaxation. As we shall see, this makes a big difference in computational efficiency.

This formulation is not the smallest convex hull relaxation possible. The purpose of the variables $b_1, \ldots, b_k$ and the big-M constraints (29) is only to ensure that at most one of $x_1, \ldots, x_k$ is non-zero. This is unnecessary in our framework since this is implicitly enforced by the connection to y. As an alternative to this intuitive explanation, consider deriving the convex hull formulation of this constraint from the general disjunctive formulation [1]. The disjunction equivalent of $\texttt{element}(y, [c_1, \ldots, c_k] \times x, z)$ is

$$\bigvee_{i \in D_y} \big(-z + c_i x = 0 \big), \qquad (30)$$

and the general disjunctive formulation a) applied to (30) gives b) since $\alpha = 0$, which then simplifies to c), which are equations (26)–(27) above.

$$\begin{array}{lll}
\bigvee a_i x \leq \alpha, & & \\
\Updownarrow & (z, x) = \sum_i (z_i, x_i), & \\
x = \sum_i x_i, & -z_i + c_i x_i \leq 0, \quad \forall i, & z = \sum_i c_i x_i, \\
a_i x_i \leq \alpha b_i, \quad \forall i, & z_i - c_i x_i \leq 0, \quad \forall i, & \\
\sum_i b_i = 1. & \sum_i b_i = 1. & x = \sum_i x_i. \\
\text{a)} & \text{b)} & \text{c)}
\end{array}$$

Thus, we can dispose of (24), (25) and (29), given that upon domain reduction $i \notin D_y$ of y, we enforce $x_i = 0$. This is similar to the modeling and branching with Special Ordered Sets (SOS) of type 1 in MIP.

This tight relaxation, which computational tests (Sec. 6) show is an invaluable tool, links the discrete and continuous parts of the MLLP model (Sec. 2.3), The continuous (linear) part includes the resource constraints (16), with $q_i A_{kit_i}$ linearized as detailed above.[1] Figure 1 illustrates the effect of this. The quantities

[1] Unify q_i with x, t_i with y and A_{kit_i} with c, to map between model and linearization.

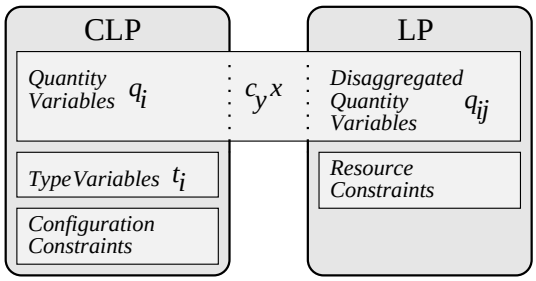

Figure 1. The build-up of the MLLP model

are present in both the CLP and LP parts, although in different forms. CLP has q_i, whereas the linearization introduces the q_{ij} variables into the LP. The component types are only present in the CLP part, because as proven above, they add nothing to the relaxation and are therefore not included.

5 Reduced costs

Using reduced costs for inference in optimization methods is standard practice in OR and is available in several MIP solvers, e.g., CPLEX [11] and XPRESS MP [4]. We will only give a brief description of reduced costs here, for more details see [3, 12]. In a basic solution to the Linear Program (LP),

$$\begin{aligned} \min \quad & z = cx \\ \text{s.t.} \quad & Ax = b,\ x \geq 0, \end{aligned}$$

the variables are partitioned into *basic* and *non-basic* variables [3], call them x_B and x_N, respectively. If we partition the constraint matrix $A = [B\ N]$ and the objective function $c = [c_B\ c_N]$ in the same manner, we can write the problem above as

$$\begin{aligned} \min \quad & z = c_B x_B + c_N x_N \\ \text{s.t.} \quad & Bx_B + Nx_N = b, \\ & x_B, x_N \geq 0. \end{aligned} \tag{31}$$

The solution to equation (31) is $x_B = B^{-1}b - B^{-1}Nx_N$. In a *basic solution*, x_N will be 0 and therefore $x_B = B^{-1}b$. We note that the value of the objective function will then be

$$\begin{aligned} z = c_B x_B + c_N x_N &= c_B(B^{-1}b - B^{-1}Nx_N) + c_N x_N \\ &= c_B B^{-1}b + \underbrace{(c_N - B^{-1}N)}_{\bar{c}} x_N = c_B B^{-1}b. \end{aligned}$$

The reduced costs, $\bar{c}$, are defined for the non-basic variables $x_N = \{x_{i_1}, \ldots, x_{i_k}\}$ (which are all zero in the basic solution). The reduced cost $\bar{c}_{i_j}$ then corresponds

to the *cost per unit increase* to the objective function value if x_{i_j} would take on a non-zero value. The basic solution is an *optimal basic solution* if all the reduced costs are non-negative, indicating that there is no benefit in having any of the non-basic variables taking on non-zero values.

A process called *reduced cost fixing* [18] uses the reduced costs of relaxed 0–1 variables in MIP problems, obtained from the optimal linear programming relaxation solution $\bar{x}$, to potentially fix variables to zero without having to branch on them. This can be done for a non-basic variable x_{i_j} if $c\bar{x} + \bar{c}_{i_j} > cx^*$, i.e., if the current objective value plus the reduced cost $\bar{c}_{i_j}$ of the variable x_{i_j} exceeds the objective value of the incumbent solution x^*, the best solution found so far in the branch-and-bound search. This generalizes to variables at their lower *or* upper bound, and to general integer variables.

5.1 Reduced-Cost Based Propagation in CLP

Recently, Focacci et. al. [8, 5, 7] have adapted variable fixing to a CLP framework and used it successfully for the domain pruning of the `alldiff` constraint and the `path` constraint in ILOG Solver. This is done by shadowing these global constraints with a linear formulation of the assignment problem [18] and solving it to optimality while computing the corresponding reduced costs. This is done incrementally using the Hungarian algorithm, but it could also be done with linear programming. In the assignment problem, the variable $x_{ij} = 1$ corresponds to $x_i = j$ in $\texttt{alldiff}(x_1, \ldots, x_k)$. As before, if $c\bar{x} + \bar{c}_{ij} > cx^*$ for a non-basic variable x_{ij} and incumbent solution x^*, then $x_i \neq j$. This rule is used within a standard fix-point propagation loop and can thus interact and communicate with other constraints. From a CLP perspective, this is a new way of effectively using the objective function for domain reduction in specific global constraints/structures. On the other hand, from an OR point-of-view, reduced-cost fixing is given added value through its integration with other inference algorithms.

Note that the assignment problem is completely separate there from the rest of the model; it is only used in the propagation loop. The assignment problem is also *totally unimodular* [12], which means that the optimal solution to the linear relaxation will always be integral. This will not be the case for general hybrid CLP–MIP models, but the reduced costs will still be well-defined for non-basic variables.

5.2 Reduced-Cost Based Propagation in MLLP

In a hybrid CLP–MIP solver, such as MLLP, where the inference of CLP is combined with a linear relaxation, we can also use reduced-cost based propagation, given that our constraints are linearized. The reduced cost propagation then becomes a part of the regular fixed-point propagation loop, which all the discrete and mixed constraints of MLLP are a part of.

Again we start with the simple subscripted constant, c_y (Sec. 3). Assume the current optimal LP solution is $\bar{x}$, with reduced costs $\bar{c}_1, \ldots, \bar{c}_k$ for the decision variables $b_1, \ldots, b_k$, and x^* is the incumbent MLLP solution. Then the following

rule defines reduced-cost based domain reduction for the `element` constraint (assume minimization), if linearized by (20)–(22):

$$\begin{array}{l}\texttt{do } \forall i \in D_y \\ \quad \texttt{if } c\bar{x} + \bar{c}_i \geq cx^* \texttt{ then } D_y = D_y - \{i\}\end{array}$$

This domain pruning is equivalent to standard constraint propagation, i.e., valid in this subproblem and all its extensions (this node and below in the search tree). As we noted earlier, remember that the linearization of c_y is no stronger than simple bounds-consistency, but that it allows us to do this reduced-cost based propagation, which may be beneficial in some cases.

Returning to our configuration problem, a similar reduced-cost based propagation rule for the term $c_y x$ (Sec. 4) can be used on the linearization of (26)–(28).

$$\begin{array}{l}\texttt{do } \forall i \in D_y \\ \quad \texttt{if } c\bar{x} + \bar{d}_i \geq cx^* \texttt{ then } D_y = D_y - \{i\} \\ \texttt{do } \forall i \in D_y \\ \quad x \leq \lfloor (cx^* - c\bar{x})/\bar{d}_i \rfloor\end{array}$$

where $\bar{d}_1, \ldots, \bar{d}_k$ are the reduced costs for $x_1, \ldots, x_k$, respectively. The last line bounds the quantity variable; the standard variable fixing in MIP, which is also applicable in MLLP.

6 Computational Testing

The first interesting object of study is to compare how CLP, MIP and MLLP perform on this problem. Figure 2 shows average number of nodes and average time required to solve to proven optimality instances of different sizes (note the logarithmic scale). The first, easily solved, class is a set of seven instances with real-world data; the following classes have ten instances each, with data generated to closely resemble the original instances.

For size '16x20' (16 components, 20 types), CPLEX solves 8 of the 10 instances to proven optimality within 100 000 nodes, at which point the search is aborted. For size '20x24' it only solves 4 of the 10 instances and for size '26x30' it only solves 3 of the 10 instances. SICStus Prolog [2] solves 6 instances of size '16x20' to proven optimality within 500s (50 000–100 000 nodes), 7 instances of size '20x24' on and no '26x30' instances. MLLP solves all instances of all sizes, a magnitude or more faster. It should be emphasized that MLLP uses the relaxation of $c_y x$ described in Sec. 4, without which MLLP performs much worse.

The second method we want to study is the reduced-cost based propagation. It has been shown to be clearly beneficial in a CLP context [5, 7, 8], where it adds optimization oriented domain reduction, but is it equally effective in a framework like MLLP that is already focusing the search around the relaxation optimum? Figure 3 shows the effect on the number of nodes and the time required to solve to proven optimality. The reduction in nodes is significant, around 10-50% is saved (higher percentage as the problems grow larger), and the overhead is not considerable so the CPU time saved is almost as much.

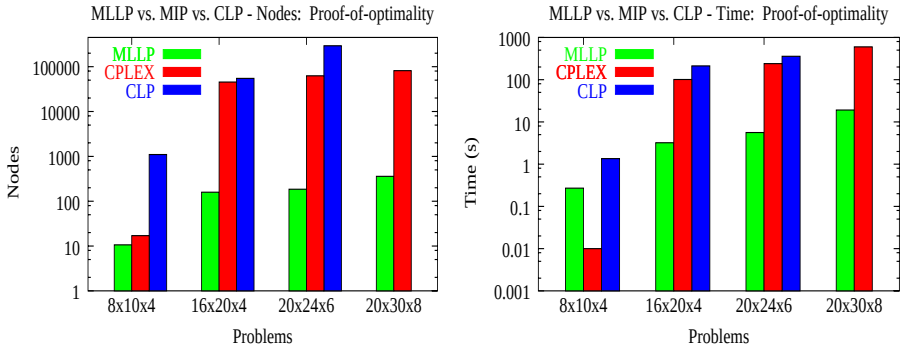

Figure 2. MLLP vs. MIP vs. CLP on a configuration problem

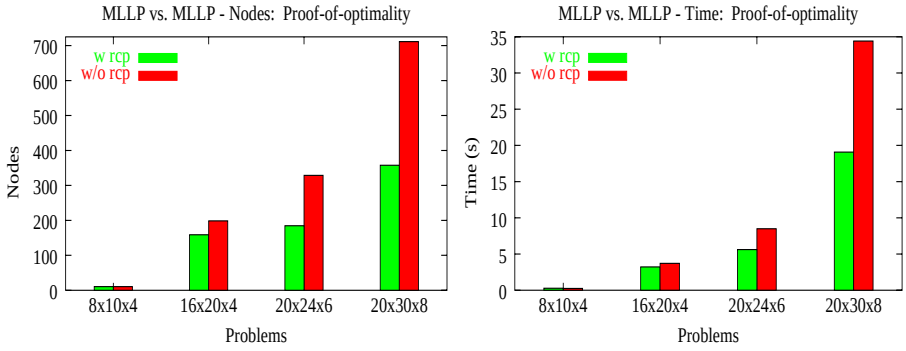

Figure 3. The impact of reduced-cost propagation, with RCP vs. without RCP

7 Conclusion

In this paper, we presented and compared CLP, MIP and hybrid MLLP models for a configuration problem. For the hybrid MLLP model, we showed how a continuous variant of the `element` constraint, which is central to the problem, can be linearized and how the reduced costs from the LP relaxation can be used for domain pruning. Computational results were included, comparing different models and techniques on a set of real-world instances. The results showed that the linear relaxation introduced is an invaluable tool and that the reduced cost propagation also contributes to an efficient solution. The hybrid MLLP approach was shown to out-perform both pure CLP and MIP, and was able to solve instances larger than the other two methods could handle.

Acknowledgements

We would like to thanks Tacton Systems [13, 17] for providing the configuration problem and initial data, Mats Carlsson, Swedish Institute of Computer Science (SICS), for reading drafts of this paper, and Prof. John Hooker, Carnegie Mellon University (CMU), for general guidance and inspiration.

References

[1] E. Balas. Disjunctive programming. In P. L. Hammer, E. L. Johnson, and B. H. Korte, editors, *Discrete Optimization II, 5*, pages 3–51, Amsterdam, 1979. Annals of Discrete Mathematics, North-Holland.

[2] Mats Carlsson et al. SICStus Prolog User's Manual. SICS research report, Swedish Institute of Computer Science, 1995. URL: http://www.sics.se/sicstus.

[3] Vašek Chvátal. *Linear Programming*. W. H. Freeman and Company, New York, 1983.

[4] Dash Optimization, Inc. *XPRESS–MP: User Manuals*, 1999. www.dashopt.com.

[5] Filippo Focacci, Andrea Lodi, and Michela Milano. Cost-based domain filtering. In Joxan Jaffar, editor, *Principles and Practice of Constraint Programming*, volume 1713 of *Lecture Notes in Computer Science*. Springer, October 1999.

[6] Filippo Focacci, Andrea Lodi, and Michela Milano. Integration of CP and OR methods for matching problems. In *CP-AI-OR'99 Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems*, Feb 1999.

[7] Filippo Focacci, Andrea Lodi, and Michela Milano. Solving TSP with time windows with constraints. In *Sixteenth International Conference on Logic Programming*, November 1999.

[8] Filippo Focacci, Andrea Lodi, Michela Milano, and Danielo Vigo. Solving TSP through the integration of OR and CP techniques. In *CP98 Workshop on Large Scale Combinatorial Optimisation and Constraints*, October 1998.

[9] John N. Hooker, Greger Ottosson, Erlendur S. Thorsteinsson, and Hak-Jin Kim. On integrating constraint propagation and linear programming for combinatorial optimization. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 136–141. AAAI, The AAAI Press/The MIT Press, July 1999.

[10] John N. Hooker, Greger Ottosson, Erlendur S. Thorsteinsson, and Hak-Jin Kim. A scheme for unifying optimization and constraint satisfaction methods. *Knowledge Engineering Review, special issue on AI/OR*, 15(1):11–30, 2000.

[11] ILOG. *Using the CPLEX Callable Library*, 1997. www.cplex.com.

[12] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, 1988.

[13] Klas Orsvärn and Tomas Axling. The Tacton view of configuration tasks and engines. In *Workshop on Configuration, Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, 1999. URL http://www.tacton.com, Technical Report WS-99-05.

[14] Greger Ottosson, John N. Hooker, Hak-Jin Kim, and Erlendur S. Thorsteinsson. On integrating constraint propagation and linear programming for combinatorial optimization. In *Proceedings of the First International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR-99)*, February 1999.

[15] Greger Ottosson, Erlendur S. Thorsteinsson, and John N. Hooker. Mixed global constraints and inference in hybrid CLP–IP solvers. In Susanne Heipcke and Mark Wallace, editors, *Proceedings of the Fifth International Conference on Principles and Practice of Constraint Programming (CP-99)'s Post-Conference Workshop on Large Scale Combinatorial Optimisation and Constraints (LSCO&C)*, volume 4 of *Electronic Notes in Discrete Mathematics, www.elsevier.nl/locate/endm*. Elsevier Science, October 1999.

[16] Robert Rodošek, Mark Wallace, and Mozafar Hajian. A new approach to integrating mixed integer programming and constraint logic programming. *Baltzer Journals*, 1997.

[17] Tacton Systems. *Tacton Configurator User's Manual*, 1999. URL http://www.tacton.com.

[18] L. A. Wolsey. *Integer Programming.* John Wiley, New York, 1998.