The Benefits of Global Constraints for the Integration of Constraint Programming and Integer Programming

Michela Milano[*] mmilano@deis.unibo.it
Greger Ottosson[†] greger@dashopt.com
Philippe Refalo[‡] refalo@ilog.fr
Erlendur S. Thorsteinsson[§] esth@cmu.edu

∗ Dipartimento di Ingegneria; Università di Ferrara; Via Saragat, 1; 41100 Ferrara; Italy;
† Dash Optimization, Inc.; 115 River Road, Suite 1203; Edgewater, NJ 07020; U.S.A.
‡ ILOG S.A.; Les Taissounieres; 1681, route des Dolines; 06560 Sophia Antipolis; France
§ GSIA; Carnegie Mellon University; Schenley Park; Pittsburgh, PA 15213; U.S.A.

Abstract

Efforts aimed at combining Operations Research and Constraint Programming have become increasingly prominent and successful in the last few years. It is now widely recognized that integrating inference in the form of constraint propagation and relaxation in the form of linear programming can yield substantial results. In this paper we argue the benefits of global constraints as a basis for such an integration. We demonstrate the advantages of modeling with global constraints, explain their operational benefits and illustrate this with a series of case studies.

Introduction

The last decade has seen an increasing success in the combination of the models and methods of Operations Research (OR) with those of Constraint Programming (CP). The initial cultural barriers have largely disappeared and the gains from taking advantage of both inference in the form of constraint propagation, and, e.g., continuous relaxations through Linear Programming (LP) are now widely acknowledged. Increasingly, techniques from OR are being applied within CP frameworks, and vice versa. Examples of the former include several Integer Programming (IP) techniques, such as *linear relaxations* (Ottosson & Thorsteinsson 2000; Ottosson, Thorsteinsson, & Hooker 1999; Refalo 1999; 2000; Rodošek, Wallace, & Hajian 1997), *Lagrangian relaxations* (Caseau & Laburthe 1997b; Focacci, Lodi, & Milano 2000), *variable fixing* through *reduced costs* (Focacci, Lodi, & Milano 1999a; Ottosson & Thorsteinsson 2000), and specialized graph algorithms, such as *matching* (Focacci, Lodi, & Milano 1999a) and *network flow* algorithms (Régin 1999).

An important modeling technique from CP used in these approaches are *symbolic* or *global constraints*. Decompositional modeling using these high-level abstractions is largely unrecognized in the OR community where most models are traditionally only formalized in linear form. One noteworthy exception is Special Ordered Sets (SOS) (Beale & Tomlin 1970; de Farias, Johnson, & Nemhauser 1999), present in most commercial IP solvers, which are essentially specially treated cardinality constraints.

In CP, global constraints serve both declaratively as building blocks of the problem statement and operationally as *software components* encapsulating specialized pruning techniques. In many cases, these techniques or *filtering algorithms* have their origin in results from OR and discrete mathematics. A typical example is the *edge finding* technique for scheduling that originated in the work of (Carlier & Pinson 1990; 1994) and subsequently has been embedded in global constraints (Caseau & Laburthe 1994; Nuijten & Aarts 1995; Baptiste, Pape, & Nuijten 1995) for modelling limited resource availability. Another interesting example is the `alldifferent` constraint (Régin 1994), which constrains a set of variables to be assigned different values. The problem of pruning inconsistent values from the domains of the variables of this constraint can be recast as the problem of finding all maximal bipartite matchings in the corresponding value graph. A graph theory result of (Berge 1970) allows this to be done in a less naïve and more efficient manner using a flow algorithm and calculating strongly connected components. Other global constraints in CP, such as the global cardinality constraint (Régin 1996), similarly derive their properties and base their propagation on graph theory and graph-based algorithms.

Historically, global constraints enable effective pruning on the basis of *feasibility* reasoning; values are removed from domains if proven infeasible. In optimization problems, global constraints can also be used for *optimality* reasoning; values are removed from domains if proven sub-optimal (Focacci, Lodi, & Milano 1999a; Ottosson & Thorsteinsson 2000). This second kind of filtering can be achieved by embedding relaxations or special purpose (optimization) algorithms in global constraints.

The purpose of this paper is to summarize and illustrate the benefits of global constraints as a basis also for a closer integration of CP and IP. To facilitate this discussion we begin with a short overview of modeling and solving in CP and IP. We then argue the benefits of hybrid modeling with global constraints and illustrate this with a series of case studies.

Solution Techniques

In CP, search is combined with inference aimed at reducing the amount of choices needed to explore. This is done through constraint propagation rules encapsulated within global constraints. In this way a CP model decomposes the problem into sub-structures, each of which is handled separately with specialized algorithms. The effect of the propagation of individual constraints is communicated through

the domains of shared variables.

In IP, on the other hand, branching is most commonly interleaved with solving a global relaxation of the problem, which eliminates the exploration of nodes for which the relaxation is infeasible or provenly sub-optimal. Besides giving a lower bound on the solution, the relaxation also provides a point in space around which the search can be centered. In the case of a good relaxation this point should be close to the optimal solution of the problem. This fact is exploited in the traditional IP branch-and-bound search. A common strategy is to branch on the fractional values that most violate the integrality requirements, thus effectively exploring regions close to the relaxation optimum. Branching on inconsistencies is preferable when the relaxation is tight and conforms closely to the original problem, since then the inconsistencies are few and easy to resolve with enumeration.

Modeling Techniques

In IP, the underlying linear form is historically a natural part of the solution process. This very restricted form of problem representation has also permeated the higher levels of problem formalization and modeling. IP uses, e.g., 0–1 variables extensively. The 0–1 variables specify the available decision choices (often exhaustively) and the associated constraints encode, in linear form, the decisions that have to be made. The decisions are then implicitly enforced through the branching on the values of the 0–1 variables.

The lack of high-level modeling constructs in IP for disjunctions and combinatorial constraints makes the process of modeling harder and more error-prone. In addition, and more importantly, much of the problem structure is lost before the problem reaches the solver algorithms. To do something more intelligent than basic branch-and-bound the solver has to *recover* the structure of the problem, despite the fact that the structure was already known to the modeler, or *assume* what the structure of the model was. Some recent work by Bockmayr and Kasper (Bockmayr & Kasper 1998) has addressed this issue by showing how the idea of global constraints can be carried over from CP to IP. As an example they introduce global `tsp` and `assign` constraints that exploit various results from polyhedral theory to generate cutting planes for these structures.

Furthermore, the black-box structure of IP solvers that is common in OR does not permit flexible, problem-specific search strategies to be defined. This is most probably due to the fact that OR grew out of the applied mathematics community rather than computer science.

In comparison, CP has had more influence from computer science and programming language design than from mathematics. This has affected not only solution strategies, but also the interface and modeling practises. A wider range of global constraints are available and the user can define new ones in various ways. Also, search strategies are both more supported and exploited in CP.

At the same time as solution technologies of CP and IP merge, so will modeling practises. Algebraic modeling will become more expressive and flexible, allowing global constraints and problem-specific pre-processing, inference and search.

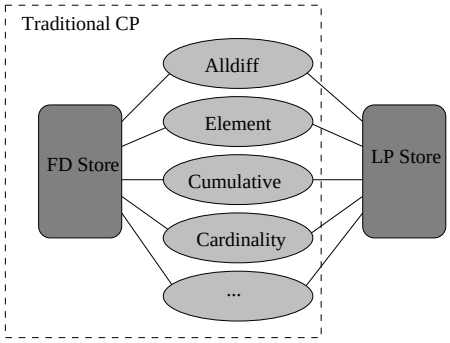

Figure 1: The interactions of global constraints

Global Constraints

Modeling using high-level, global constraints can have several advantages:

Models More expressive constraints leads to smaller models whose variables and constraints have a closer mapping to the problem at hand. This simplifies the task of modeling and maintenance.

Inference Global constraints capture structure and deliver that structure intact to the solver. This allows the solver to apply effective specialized inference algorithms.

Relaxations Relaxations can be dynamically updated, allowing a tighter formulation to be maintained.

Search Information to guide the search can be collected by these constraints, allowing more precise guidance.

Visualization Global constraints can have a visualization, i.e., the state of the constraint and the involved variables can be displayed (during search) in a constraint-specific way.

From a solving perspective, the most important feature is that relaxations and inference can be combined and applied to the same structures. In particular, this allows constraint propagation and linear programming to meet, despite the difference in the models they traditionally are applied to.

Furthermore, relaxations of overlapping global constraints gives a second channel for communication, as depicted in Fig. 1. In CP, constraints communicate only through the *constraint store*, i.e., the domains of the shared variables (a.k.a. the FD Store). A global relaxation (like LP) adds to this by providing a separate means of interaction. Inference on one constraint will here strengthen the LP relaxation, which therefore also strengthens the relaxations of other global constraints since their relaxations are also a part of the same LP.

Object Oriented Modeling and Solving

It is interesting to examine the parallels between modeling and solving using global constraints, and Object Oriented Software Design and Development (OOSDD). As

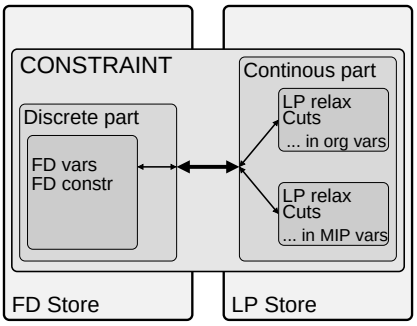

Figure 2: The structure of global constraints

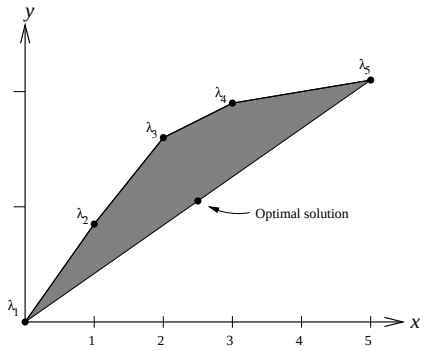

Figure 3: Piecewise-linear function illustrating the SOS–2 formulation.

mentioned in the previous sections, a global constraint captures a sub-structure in the problem and allows that substructure to be passed around and operated on as a whole. OOSDD builds on a similar principle, where each class captures specific functionality in the program and has built-in methods to operate on its data.

The methodology of OOSDD can thus be applied to a large degree to the modeling and solving of combinatorial optimization problems, as it relates to breaking the problem structurally into coherent pieces and providing filtering algorithms for each one, hence the term *Object Oriented Modeling and Solving (OOMS)*. The correspondence between the two also provides a roadmap to how a modeler/solver system can be implemented in software, both in terms of the interface between global constraints and the rest of the system (Fig. 1) and how a mixed global constraint breaks down into individual parts and how they communicate (Fig. 2).

In what follows, we study a series of cases where integration efforts based on global constraints have shown the benefits of combining relaxations and constraint propagation. They vary in application areas from production planning, scheduling, traveling salesman with time-windows, to configuration problems.

Piecewise-Linear Optimization

Piecewise-linear functions are well-studied, both in the literature and in industrial applications. A function is piecewise-linear when its values lie on linear segments as illustrated in Fig. 3. Optimizing a problem having constraints $y = f(x)$ where f is piecewise-linear is called *piecewise-linear optimization*. A problem having arbitrary piecewise-linear functions is generaly equivalent to an IP. An equivalent IP formulation can be obtained by reformulating each constraint $y = f(x)$ into

$$x = \lambda_1 b_1 + \cdots + \lambda_k b_k, \quad (1)$$

$$y = \lambda_1 f(b_1) + \cdots + \lambda_k f(b_k), \quad (2)$$

$$\lambda_1 + \cdots + \lambda_k = 1, \quad (3)$$

$$\lambda_i \geq 0, \quad i = 1, \ldots, k, \quad (4)$$

$$\text{SOS–2}(\lambda_1, \ldots, \lambda_k), \quad (5)$$

where b_i are the x-coordinates where the slope of the function changes. The linear constraints in this formulation express that the solution set contains solutions that are convex combinations of the extreme points of the functions, that is of points $(b_i, f(b_i))$ by way of positive λ_i variables. Thus the linear constraints (1)–(4) represent the convex hull of the solution set. The SOS–2 requirement (5) forces the number of λ_i variables that are different from zero to be at most two and also adjacent in the set. This constraint is usually handled with specialized branching in IP solvers, not with constraint propagation as is the case with cardinality constraints in CP.

Althought the goal of the SOS–2 requirement is to provide structural information to the IP model, traditional MIP solvers consider all these constraints independently. As a consequence, they loose the global structure of piecewise-linear functions and they do not update the linear formulation correctly during the search.

Recently, Ottosson, Thorsteinsson and Hooker (Ottosson, Thorsteinsson, & Hooker 1999) and Refalo (Refalo 1999) have shown how to really preserve the piecewise structure as a global constraint that permits maintaining a tight formulation during the search.

In (Ottosson, Thorsteinsson, & Hooker 1999) the constraint is given a linear convex hull formulation in the original space, i.e., the two dimensional space (x, y) given by the two axes, and thus no new variables need to be introduced (if one discrete indicator variable is also introduced, then the segments can be semi-continuous). Besides an improved relaxation, which is dynamically updated and tightened during the search as a part of the constraint propagation, the global piecewise constraint can also provide information for better search strategies. The black dot on the cut in Fig. 3 denotes the LP-values of x, y in a typical situation where the LP relaxation has been solved and the minimization has identified the optimal solution. A natural branching strategy is now to focus the search around the indicated segment, e.g, by splitting the domain of x into two parts, or into three parts, forcing the solution up onto the piecewise function. This is obvious and easy to calculate for a solver that treats this structure as a global constraint

In (Refalo 1999) the interaction between constraint propagation and linear formulation is investigated. A tight cooperation scheme is introduced that for each constraint $y = f(x)$ maintains the convex hull of the formulation, taking into ac-

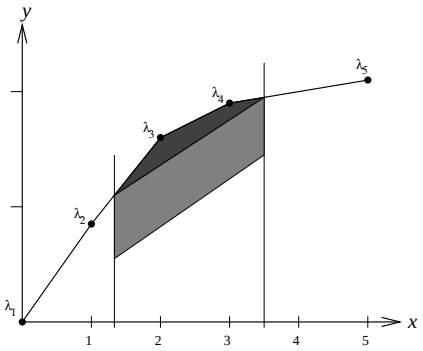

Figure 4: Piecewise-linear functions illustrating how constraint propagation on bounds can give a tighter relaxation (dark shaded area).

count the domain reduction perfomed during the search. To maintain this convex hull, cutting planes are added in the two dimensional space (x, y). The benefits of this approach are illustrated in Fig. 4: Assume the bounds of x have been tightened. Then a tighter formulation (shown in dark) can be constructed by intersecting these new bounds with the solution set of $y = f(x)$. On some problems, cutting plane generation can outperform standard IP approaches. Note that these cuts are only locally valid since they depend on domain reductions performed on x and y at a particular node of the search tree. Consequently, they must be removed when backtracking, similarly to standard constraint propagation.

The Matching Structure

Matching is a very common structure that occurs as a basic building block of many problems. Caseau and Laburthe (Caseau & Laburthe 1997a) first exploited this structure when embedding lower bounds in global constraints. These lower bounds were derived from solving the Assignment Problem (AP) to optimality by special purpose algorithms. The AP has a linear formulation that is totally unimodular, thus its optimal (continuous) solution is integer. The AP can therefore be solved by linear programming or by specialized incremental algorithms, e.g., the Busaker and Gowen flow algorithm or the Hungarian algorithm, in polynomial time. They also defined an approximative regret function which was used as information for search heuristics.

Focacci, Lodi and Milano (Focacci, Lodi, & Milano 1999a) have extended the notion of regret with that of reduced costs by incorporating the AP structure within the `alldiff` and `path/cycle` constraints. A linear formulation of the assignment problem (Wolsey 1998):

$$\min \quad cx = \sum_i \sum_j c_{ij} x_{ij} \tag{6}$$

$$\text{s.t.} \quad \sum_j x_{ij} = 1, \qquad \forall i, \tag{7}$$

$$\sum_i x_{ij} \leq 1, \qquad \forall j, \tag{8}$$

$$0 \leq x_{ij} \leq 1, \qquad \forall i, j, \tag{9}$$

is embedded in the global constraint and solved to optimality while computing the corresponding reduced costs. The assignment $x_{ij} = 1$ corresponds to the CP variable instantiation $x_i = j$ in `alldiff`$(x_1, \ldots, x_k)$. The reduced-cost propagation rule is that if $c\bar{x} + \bar{c}_{ij} > cx^*$ for a non-basic variable x_{ij}, incumbent solution x^*, current solution $\bar{x}$ and its reduced costs $\bar{c}$, 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 CP 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. Here the inference algorithms for the `alldiff` constraint (Régin 1994) work in combination with a relaxation to reduce the search space.

The Cycle Structure

The assignment structure is also a basic block in the Traveling Salesman Problem (TSP) and its variants. In graph terminology, the assignment corresponds to finding a set of disjoint sub-tours that minimizes the sum of costs of selected arcs. The cycle constraint, or the TSP structure, includes this and the additional requirement of a *single* cycle.

The CP model for the TSP and its variants involves a `cycle` constraint where the same technique as used for the `alldiff` constraint can be exploited. A first improvement to pure CP techniques was given by Caseau and Laburthe in (Caseau & Laburthe 1997b), where a propagation scheme for the `cycle` constraint was proposed together with branching strategies and bounding methods that further improved the performances of CP systems. In (Focacci, Lodi, & Milano 1999a) the use of reduced costs has been proven to be even more effective. In fact, experimental results show that the information on the lower bound derived from solving an AP to optimality permits solving small TSPs that are not solvable without this information. In addition, the use of reduced-cost based propagation leads to a 10–20-fold average reduction in the number of failures, with respect to using the lower bounds only. For pure TSPs, despite the performance improvement achieved with reduced-cost based propagation, the state of the art OR branch-and-cut algorithms are still superior. When the TSP variants are considered, however, the techniques described above obtain results which are comparable with state of the art OR approaches (Ascheuer, Fischetti, & Grötschel).

An improvement to this work has been obtained in (Focacci, Lodi, & Milano 2000; Refalo 2000) by adding a set of subtour elimination cuts (Padberg & Rinaldi 1990) to the AP formulation, aimed at strengthening the linear formulation of the AP. The resulting problem is again solved to optimality and used for domain pruning based on reduced costs and improved bounding. As before, the relaxation combines with the propagation of the `cycle` constraint (Caseau & Laburthe 1997b) within the global constraint.

Variable Subscripts

In CP, the $\texttt{element}(y, [c_1, \ldots, c_k], z)$ constraint is ubiquitous in many optimization problems, commonly used to look up a cost associated to a discrete decision variable. Bounds-consistency on the `element` constraint involves producing increasingly smaller domains or 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, i.e, a disjunction, is modeled in IP. 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

$$\texttt{element}(y, [c_1, \ldots, c_k], z)$$

$$\Updownarrow$$

$$0 \leq b_i \leq 1, \qquad \forall i, \tag{10}$$

$$b_1 + \cdots + b_k = 1, \tag{11}$$

$$z = c_1 b_1 + \cdots + c_k b_k, \tag{12}$$

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 higher-dimensional relaxation unless these new variables $(b_1, \ldots, b_k)$ 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.

We believe there are many problems where bounds-consistency techniques can eliminate the introduction of 0–1 variables, and in effect reduce the LP size. Furthermore, in this case branching on y implicitly performs a search which is similar to SOS-style branching in IP, without the need for the modeler to specify this explicitly.

Quantified Variable Subscripts

A variant of the variable subscripted constants discussed in the previous section was investigated by Ottosson and Thorsteinsson (Ottosson & Thorsteinsson 2000). Occurring in a configuration problem, a type and quantity are to be decided for each component, which results in a term of the form $c_y x = z$. Here, y indicates the type of the component, x its quantity and z the resulting cost.

A linear 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:

$$0 \leq b_i \leq 1, \qquad \forall i, \tag{13}$$

$$b_1 + \cdots + b_k = 1, \tag{14}$$

$$x = x_1 + \cdots + x_k, \tag{15}$$

$$z = c_1 x_1 + \cdots + c_k x_k, \tag{16}$$

$$x_i \geq 0, \qquad \forall i, \tag{17}$$

$$x_i \leq M b_i, \qquad \forall i. \tag{18}$$

Equations (13)–(14) are the same as for c_y above (the plain `element` constraint), but in addition, x is disaggregated to $x_1, \ldots, x_k$, and connected through the big-M constraints (18) to $b_1, \ldots, b_k$.

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 (18) is only to ensure that at most one of $x_1, \ldots, x_k$ is non-zero. This is unnecessary since this is implicitly enforced by the connection to y. Thus, we can dispose of (13), (14) and (18), 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 SOS–1 variables in IP, but again, it is implicit within this global constraint. Note that the resulting relaxation (15)–(17) is the convex hull formulation of the disjunction (Balas 1979) underlying the `element` constraint.

Computational tests (Ottosson & Thorsteinsson 2000) show that this tight relaxation is an invaluable tool if combined with constraint propagation. Furthermore, reduced-cost based propagation (Ottosson & Thorsteinsson 2000) can be applied to the variables $x_1, \ldots, x_k$, achieving domain reduction on y. A hybrid approach using this relaxation and reduced-cost based propagation along with regular CP propagation is shown to out-perform both pure CP and IP, and is able to solve instances larger than the other two methods alone can handle.

The two variants of the `element` constraint described in this and the previous section illustrate three of the benefits of global constraints; *smaller relaxations*, *improved search* and *improved propagation*, e.g., by using the reduced costs.

Conclusion

Modeling and solving are tightly integrated activities, and it is no surprise that the different underlying solution technologies of CP and IP have shaped the modeling traditions into different forms. When the two techniques now are being merged on the technology level, we also have to re-evaluate our modeling practises. So far, most efforts of integration have been done within the CP community, which also traditionally has had the more flexible framework.

As part of this, global constraints play a very import role. They preserve structure, allowing improved inference and search to be performed more effectively. We believe this also holds true when adding relaxations, and this is illustrated on several examples. Not only can OR methods be incorporated within these constraint abstractions, but in several cases the relaxations can be made sharper and more effective than in IP solvers operating only on linear formulations. Add to this the combination of constraint propagation and relaxations that elevate each other, and we have a clear indication that global constraints are worth further study and application in hybrid CP–IP contexts.

References

Ascheuer, N.; Fischetti, M.; and Grötschel, M. A polyhedral study of the asymmetric travelling salesman problem with time windows. *Networks*. to appear.

Balas, E. 1979. Disjunctive programming. In Hammer, P. L.; Johnson, E. L.; and Korte, B. H., eds., *Discrete Optimization II, 5*, 3–51. Amsterdam: Annals of Discrete Mathematics.

Baptiste, P.; Pape, C. L.; and Nuijten, W. 1995. Incorporating efficient operations research algorithms in constraint-based scheduling. In *1st Joint Workshop on Artificial Intelligence and Operational Research*.

Beale, E., and Tomlin, J. 1970. Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. In Lawrence, J., ed., *Proceeding of the Fifth International Conference on Operation Research*. Tavistock Publications.

Berge, C. 1970. *Graphes et hypergraphes*. Paris: Dunod.

Bockmayr, A., and Kasper, T. 1998. Branch-and-infer: A unifying framework for integer and finite domain constraint programming. *INFORMS J. Computing* 10(3):287 – 300.

Carlier, J., and Pinson, E. 1990. A practical use of Jackson's preemptive schedule for solving the job-shop problem. *Annals of Operations Research* 269–287.

Carlier, J., and Pinson, E. 1994. Adjustment of heads and tails for the job-shop problem. *European Journal of Operational Research* 146–161.

Caseau, Y., and Laburthe, F. 1994. Improved CLP Scheduling with Task Intervals. In *Proceedings of the Eleventh International Conference on Logic Programming*. MIT Press.

Caseau, Y., and Laburthe, F. 1997a. Solving various weighted matching problems with constraints. In *Principles and Practice of Constraint Programming*, volume 1330 of *Lecture Notes in Computer Science*, 17–31.

Caseau, Y., and Laburthe, F. 1997b. Solving small TSPs with constraints. In Naish, L., ed., *Proceedings of the 14th International Conference on Logic Programming*, 316–330. Cambridge: MIT Press.

de Farias, I. R.; Johnson, E. L.; and Nemhauser, G. L. 1999. A branch-and-cut approach without binary variables to combinatorial optimization problems with continuous variables and combinatorial constraints. *Knowledge Engineering Review, special issue on AI/OR, submitted*.

Focacci, F.; Lodi, A.; and Milano, M. 1999a. Cost-based domain filtering. In Jaffar, J., ed., *Principles and Practice of Constraint Programming*, volume 1713 of *Lecture Notes in Computer Science*. Springer.

Focacci, F.; Lodi, A.; and Milano, M. 1999b. Solving TSP with time windows with constraints. In *Sixteenth International Conference on Logic Programming*.

Focacci, F.; Lodi, A.; and Milano, M. 2000. Cutting planes in constraint programming: An hybrid approach. In *CP-AI-OR'00 Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems*.

Hooker, J. N., and Osorio, M. A. 1999. Mixed logical/linear programming. *Discrete Applied Mathematics* 96–97(1–3):395–442.

Hooker, J. N.; Ottosson, G.; Thorsteinsson, E. S.; and Kim, H.-J. 1999. On integrating constraint propagation and linear programming for combinatorial optimization. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, 136–141. AAAI.

Hooker, J. N.; Ottosson, G.; Thorsteinsson, E. S.; and Kim, H.-J. 2000. A scheme for unifying optimization and constraint satisfaction methods. *Knowledge Engineering Review, special issue on AI/OR*.

Nuijten, W. P. M., and Aarts, E. H. L. 1995. A computational study of constraint satisfaction for multiple capacitated job shop scheduling. *European Journal of Operational Research*. Accepted for publication.

Ottosson, G., and Thorsteinsson, E. S. 2000. Linear relaxations and reduced-cost based propagation of continuous variable subscripts. In *Proceedings of the Second International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR-00)*.

Ottosson, G.; Thorsteinsson, E. S.; and Hooker, J. N. 1999. Mixed global constraints and inference in hybrid CLP–IP solvers. In Heipcke, S., and Wallace, M., eds., *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.

Padberg, M., and Rinaldi, G. 1990. An efficient algorithm for the minimum capacity cut problem. *Mathematical Programming* 47:19–36.

Refalo, P. 1999. Tight cooperation and its application in piecewise linear optimization. In Jaffar, J., ed., *Principles and Practice of Constraint Programming*, volume 1713 of *Lecture Notes in Computer Science*. Springer.

Refalo, P. 2000. Linear formulation of constraint programming models. In *CP-AI-OR'00 Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems*.

Régin, J.-C. 1994. A filtering algorithm for constraints of difference in CSPs. In *Proc. of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, 362–367.

Régin, J.-C. 1999. Arc consistency for global cardinality constraints with costs. In Jaffar, J., ed., *Principles and Practice of Constraint Programming*, volume 1713 of *Lecture Notes in Computer Science*. Springer.

Régin, J.-C. 1996. Generalized arc consistency for global cardinality constraint. In *AAAI-96: Thirteenth National Conference on Artificial Intelligence*.

Rodošek, R.; Wallace, M.; and Hajian, M. 1997. A new approach to integrating mixed integer programming and constraint logic programming. *Baltzer Journals*.

Wolsey, L. A. 1998. *Integer Programming*. New York: John Wiley.