On Integrating Constraint Propagation and Linear Programming for Combinatorial Optimization

Greger Ottosson‡ greger@csd.uu.se
John N. Hooker† jh38+@andrew.cmu.edu
Hak-Jin Kim† hk2z+@andrew.cmu.edu
Erlendur S. Thorsteinsson† esth@cmu.edu

† Graduate School of Industrial Administration
Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A.
‡ Computing Science Dept., Uppsala University
PO Box 311, S-751 05 Uppsala, Sweden

March 16, 2000

Abstract

Integer programming and constraint (logic) programming are two traditional techniques for solving combinatorial optimization problems; the former based on linear programming relaxations and the latter on constraint propagation. Attempts to combine them have mainly focused on incorporating either technique into the framework of the other — traditional models have been left intact. We argue that a rethinking of our modeling traditions is necessary to achieve the greatest benefit of such an integration. We propose a declarative modeling framework in which the structure of the constraints indicates how LP and CP can interact to solve the problem.

1 Introduction

Linear programming (LP) and constraint propagation (CP) are two complementary techniques with potential for integration to benefit the solution of combinatorial optimization problems. Integer programming (IP) has been successfully applied to a range of problems, such as capital budgeting, bin packing and traveling salesman problems. Constraint (logic) programming (CLP) has in the last decade been shown to be a flexible, efficient and commercially successful technique for scheduling, planning and allocation. LP and CP tend to be used separately in IP and CLP, respectively, and only recently have attempts been made at combining them.

Several articles compare CLP and IP [7, 2]. They report experimental results that illustrate some key properties of the techniques. IP is very efficient for problems with good relaxations, but it suffers when the relaxation is weak or when its restricted modeling framework results in big models. CLP, with its more expressive constraints, has smaller models that are closer to the problem and behaves well for highly constrained problems, but it lacks the "global perspective" of relaxations.

Some attempts have been made at integration. In [6], CP and LP relaxations are simultaneously used to prune domains and establish bounds. A systematic procedure is used to create a "shadow" MIP model, from a CLP model with logical and symbolic constraints. The modeler may annotate constraints to indicate which solver should handle them – CP, LP or both.

Bockmayr and Kasper propose an interesting framework in [1] for combining CLP and IP, in which several approaches of integration or synergy are possible. They investigate how symbolic constraints can be incorporated into IP much as cutting planes are. They also show how a linear system of inequalities can be used in CLP by incorporating them as symbolic constraints. They also discuss a closer integration where both linear inequalities and domains appear in the same constraint store.

2 Characterization

Both CLP and IP rely on branching to enumerate regions of the search space. But within this framework they use dual approaches to problem solving: inference and search. CLP emphasizes inference in the form of constraint propagation, which removes infeasible values from the variable domains. It is not a search method, i.e., an algorithm that examines a series of complete labellings until it finds a solution. IP, by contrast, does exactly this. It obtains complete labellings by solving linear programming relaxations of the problem in the branching tree.

IP has the advantage that it can generate cutting planes (inequalities implied by the constraint set) that strengthen the linear relaxation. This can be a powerful technique when the problem is amenable to polyhedral analysis. But IP has the disadvantage that its constraints must be expressed as inequalities (or equations) involving integer-valued variables. Otherwise the linear programming relaxation is not available. This places a severe restriction on IP's modeling language.

CP can accelerate the search for a solution by

- reducing variable domains (and in particular by proving infeasibility),
- tightening the linear relaxation by adding bounds and cuts in addition to classical cutting planes, and
- eliminating search of symmetric solutions, which are often more easily excluded by using symbolic constraints.

LP can enhance the solver by

- finding feasible solutions early in the search by "global reasoning," i.e., solution of an LP relaxation,
- similarly proving infeasibility earlier in the search,
- similarly providing stronger bounds that accelerate the proof of optimality, and by
- providing reasons for failure or a poor solution, so as to produce nogoods.

How can we model our problems so that propagation techniques can be applied in IP? How can we make best use of LP in a CLP framework? Even more importantly, what framework is needed to truly unify the techniques?

The approaches taken so far for integration of CP and LP are (a) to use both models in parallel, and (b) to try to incorporate one within the other. The more fundamental question of whether a *new* modeling framework should be used has not yet been explored in any depth. The success of (a) depends on the strength of the links between the models and to what degree the overhead of having two models can be avoided. Option (b) is limited in what it can achieve. The high-level symbolic constraints of CLP cannot directly be applied in an LP model, and, conversely, the vast research in the mathematical programming community of cutting planes relies in most instances on 0–1 decision variables and a linear relaxation.

We propose taking a step back to investigate how one can design a model to suit the solvers rather than adjust the solvers to suit the traditional models.

3 Modeling for Hybrid Solvers

We begin with the framework of mixed logical/linear programming or MLLP [4, 5], in which conditional constraints link the discrete and continuous elements of the problem. The antecedents of the conditionals contain discrete variables, and the consequents contain continuous variables. A model has the form,

$$\begin{array}{ll} \min & cx \\ \text{s.t.} & h_i(y) \rightarrow A^i x \geq b^i, \quad i \in I, \\ & x \in R^n, y \in D, \end{array}$$

where y is a vector of discrete variables and x a vector of continuous variables. $h_i(y)$ is a constraint that contains only variables in y. A problem is solved by a search that branches on the discrete variables. CP is applied to the discrete part to reduce the search and help determine when partial assignments satisfy the antecedents. An LP solver is applied to the continuous side to detect infeasibility and guide the search. A feasible solution is obtained when the truth value of every antecedent is determined, and the LP solver finds an optimal solution subject to the enforced inequalities. Computational tests reported in [4] suggest that an MLLP framework not only has modeling advantages but can often permit more rapid solution of the problem than traditional mixed integer programming solvers.

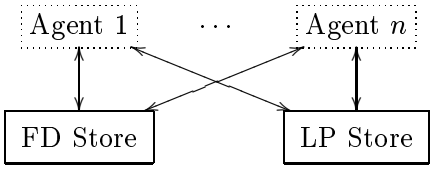
For example, consider the *Network Design Problem*. The objective is to decide which arcs of a network to use for flow so as to minimize cost. There is a fixed cost F_{ij} for using

an arc (i, j), and a unit cost c_{ij} for flow on arc (i, j).

$$\begin{array}{lll} \min & \sum_{j,k} z_{ij} & \\ \text{s.t.} & \neg y_{ij} \rightarrow z_{ij} = x_{ij} = 0 & \forall i, j \\ & y_{ij} \rightarrow z_{ij} = F_{ij} + c_{ij} x_{ij} & \forall i, j \\ & \sum_i x_{ij} - \sum_i x_{ji} = S_j & \forall j \\ & 0 \leq x_{ij} \leq M_{ij} & \forall i, j \end{array}$$

Here, x_{ij} is the flow placed on the arc (i, j) and M_{ij} is the arc capacity. S_j is the net supply available at node j. A nice property of this model is that it can be given a relaxation that not only is as strong as the traditional LP relaxation but is smaller and, because of its network flow structure, can be solved much more rapidly [3].

Using the framework of [1], MLLP has two constraint stores.



A classical finite domain (FD) constraint store contains domains, and the LP constraint store contains linear inequalities. The non-primitive constraints can access and add to both constraints stores. Since only domain constraints $x_i \in D_i$ exist in the FD store, integrality constraints can remain therein as primitive constraints, and the continuous arithmetic constraints are accessible from the LP constraint store. There are no continuous variables in the FD store and no discrete variables in the LP store. The conditional constraints act as the prime inference agents connecting the two stores, reading domains of the FD store and adding inequalities to the LP store.

4 Conclusion

LP and CP have long been used separately, but they have the potential to be integrated as complementary techniques in future optimization frameworks. But to do this fully and in general, the modeling traditions of mathematical programming and constraint programming also must be integrated. We propose a unifying modeling and solution framework that aims to do so; continuous and discrete constraints are naturally combined using conditional constraints, allowing a clean separation and a natural link between constraints amenable to CP and continuous inequalities efficiently handled by LP.

References

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

[2] Ken Darby-Dowman and James Little. Properties of some combinatorial optimization problems and their effect on the performance of integer programming and constraint logic programming. *INFORMS Journal on Computing*, 10(3):276–286, Summer 1998.

[3] J. N. Hooker and Hak-Jin Kim. Succinct relaxations for some discrete problems. Technical report, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213 USA, 1998.

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

[5] John N. Hooker, Hak-Jin Kim, and Greger Ottosson. A declarative modeling framework that integrates solution methods. *Annals of Operations Research, Special Issue on Modeling Languages and Approaches, to appear*, 1998.

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

[7] Barbara Smith, Sally Brailsford, Peter Hubbard, and H. Paul Williams. The Progressive Party Problem: Integer Linear Programming and Constraint Programming Compared. In *CP95: Proceedings 1st International Conference on Principles and Practice of Constraint Programming)*, Marseilles, September 1995.