Representations of the *all_different* Predicate of Constraint Satisfaction in Integer Programming

H.P. Williams • Hong Yan
Faculty of Mathematical Studies, University of Southampton, Southampton, UK
Department of Management, The Hong Kong Polytechnic University, Hong Kong
hpw@maths.soton.ac.hk • mshyan@polyu.edu.hk

The use of predicates to state constraints in Constraint Satisfaction is explained. If, as an alternative, the traditional approach of Integer Programming (IP) is used, it is desirable to model these constraints so as to give as tight a linear programming relaxation as possible. One of the most commonly used predicates is the "*all_different*" predicate. Different applications of this are described together with different IP formulations. The facet defining constraints for the convex hull are then described and proved to be facet defining.
(*Constraint Satisfaction; Constraint Logic Programming; all_different Predicate; Facets; Convex Hull*)

1. Introduction

Constraint Satisfaction (CS) (Finite Domain Programming, Constraint Logic Programming, Logic Programming etc.) is proving to be a powerful tool for modeling and solving many problems in operations research. For most of these problems it is an alternative to Integer Programming (IP) (Discrete Optimization). While IP methods offer much more mathematical sophistication, being usually based on solving the linear programming (LP) relaxation, CS often has representational advantages. Firstly it is often a far more compact representation, and secondly to some users it is a more natural representation. A discussion of some case studies comparing CS and IP is given below.

The representational advantage of CS lies in the use of *predicates* to represent constraints. These predicates are far richer than the restricted linear equalities and inequalities of LP and IP. CS methods treat these predicates through the processing of the resultant model. Such discussion is beyond the scope of this paper but standard references are Tsang (1994) and Van Hentenryck (1989) as well as others. This can, in some circumstance, result in quicker processing than the corresponding IP model. Examples of comparisons are described by, for example, Brailsford et al. (1995), Darby-Dowman and Little (1998), Hooker (2000), and Proll and Smith (1998). Quicker processing normally results from the smaller data structures modeled by CS. The underlying methods are, however, less sophisticated than IP from a mathematical (but perhaps not a computer-science) point of view. The great advantage provided by IP is that it normally makes use of the LP relaxation. This is usually very quick to solve and provides a *bound* on the optimal objective value. It also additionally provides a *proof of optimality* for optimization models (something that CS can do only by complete enumeration). It is well known, however, that there are often alternative IP models for a problem, which may differ markedly in their computational tractability. See for example Williams (1974, 1978) and Crowder et al. (1983). This arises from the possibility of giving IP formulations whose set of feasible integer solutions is the same but whose set of solutions to the corresponding LP relax-

INFORMS Journal on Computing © 2001 INFORMS
Vol. 13, No. 2, Spring 2001 pp. 96–103
0899-1499/01/1302/0096$5.00
1526-5528 electronic ISSN

ations are not. Reducing the size of the polytope of solutions to the LP relaxation normally has a dramatic effect in reducing solution time. In theory one might like to represent the IP by constraints representing the *facets of the convex hull* of feasible integer solutions. Normally, however, there will be an astronomic number of these and one has to content oneself with a more modest aim, possibly adding constraints on an "*as-needed basis*."

One possible compromise is to represent the facet defining linear representation for individual constraints, where known. The intersection of the convex hulls of these individual constraints will not, of course, in general produce a convex-hull representation for the whole model (see for example, Jeroslow 1989 and Balas 1974) but will be more constrained than if the original "looser" constraints are used.

Facets of the convex hull for some individual linear constraints have already been analyzed. In particular the Knapsack polytope has been investigated by Balas (1974), Wolsey (1975) and Hammer et al. (1975). Much less attention has been paid to more complex constraints which often arise in CS in the form of predicates. There is now great interest in using such predicates in designing modeling languages for Mathematical Programming (MP) or CS systems. See for example McKinnon and Williams (1989), Hooker (1998) and Fourer (1998). It would be of great advantage to know "good" representations of these predicates as systems of linear equations or inequalities if MP is to be used. By "good", we mean "having tight LP relaxations" but also being modest in size. It was to this end that McKinnon and Williams (1989) provided a modeling system based on the "greater-than-or-equal" constraint. Yan and Hooker (1999) have produced a facet description of the "If cardinality of $S \geq s$ then cardinality of $T \geq t$" constraint. In this paper we produce a facet description of one of the most commonly used CS predicates. This is the "*all_different*" predicate (variously known in different systems as "*all_distinct*," "*not_equal*," etc.).

$$x_i \in \{a_1, a_2, \cdots, a_k\}, \quad i \in \{1, 2, \cdots, n\}$$
$$x_i \neq x_j, \quad \text{for all } i, j.$$

The representation of this constraint by a predicate is clearly compact. The formulation using IP constraints can, however, be cumbersome. We give some possible representations in Section 3. Before this, however, in Section 2, we report applications of it in order to motivate subsequent discussion. In Section 4, we give, and prove, the facet description.

2. Examples of Use of the "*all_different*" Predicate

2.1. The Assignment Problem

Suppose we wish to assign members of a set I uniquely to members of J. The conventional IP model defines variables $x_{ij} \in \{0, 1\}$ with constraints

$$\sum_{j \in J} x_{ij} = 1, \text{ all } i \quad \text{(all members of } I \text{ must be assigned somewhere.)}$$
$$\sum_{i \in I} x_{ij} \leq 1, \text{ all } j \quad \text{(no member of } J \text{ can have more than one member of } I \text{ assigned to it)}$$

This IP formulation has $|I||J|$ variables and $|I|+|J|$ constraints.

A CS formulation replaces the index j by a set of variables y_i defined by

$$y_i = \text{member of } J \text{ to which } i \text{ is assigned.}$$

We than replace the IP constraints by the CS constraint:

$$\textit{all\_different}_i\ (y_i).$$

This model has $|I|$ (CS) variables and 1 (CS) constraint.

2.2. The Progressive Party Problem

This is described by Brailsford et al. In the IP model variables are used of the form:

$$x_{ijt} = \begin{cases} 1 & \text{iff person } j \text{ is assigned to party } i \text{ at time } t, \\ 0 & \text{otherwise,} \end{cases}$$

for $i \in I$, $j \in J$, and $t \in T$.

No person can return to the same host at another party at another time. The IP constraint is:

$$\sum_{i,t} x_{ijt} \leq 1, \text{ for all } j.$$

This part of the model has $|I||J||T|$ variables and $|J|$ constraints.

The CS formulation replaced the index i by a set of variables:

$$y_{jt} = \text{Host to which } j \text{ is assigned at time } t,$$

and the IP constraint by the CS constraint

$$all\_different_{j,t}\ (y_{jt}).$$

2.3. A Manufacturing Problem

This is described by Darby-Dowman and Little (1998). In the IP model,

$$x_{ij} = 1 \text{ if task } j \text{ is assigned to machine } i.$$

No two tasks can be assigned to the same machine. The IP constraint is

$$\sum_j x_{ij} \le 1, \text{ all } i$$

where $i \in I$, $j \in J$.

As in the assignment problem variables y_j are introduced (instead of indices i) with the interpretation

$$y_j = \text{Machine to which task } j \text{ is assigned},$$

and the CS constraint

$$all\_different_j\ (y_j).$$

2.4. The School Timetabling Problem

We wish to assign Teachers to different classes at different periods. The IP formulation is

$$x_{ijk} = \begin{cases} 1, & \text{if Teacher } i \text{ is assigned to Class } j \\ & \quad \text{in Period } k \\ 0, & \text{otherwise.} \end{cases}$$

The IP constraints are

$\sum_i x_{ijk} \le 1$, all j, k, (Class j in Period k cannot have more than one Teacher)

$\sum_j x_{ijk} \le 1$, all i, k, (Teacher i cannot teach more than one Class in Period k)

$\sum_k x_{ijk} \ge D_{ij}$, all i, j, (Teacher i must teach Class j for at least D_{ij} Periods per week)

where $i \in I$, $j \in J$ and $k \in K$. This model has $|I||J||K|$ variables and $|J||K| + |I||K| + |I||J|$ constraints.

One possible CS formulation is to replace indices i by variables:

$y_{jk} = 0$ if no teacher is assigned to class j in period k

$y_{jk} =$ teacher assigned to class j in period k.

We introduce the CS predicates:

$$y_{jk} = i \rightarrow x_{ijk} = 1, \text{ all } i, j, k$$

$$all\_different_{j>0}\ (y_{jk}), \text{ all } k.$$

This model has $|I||J||K| + |J||K|$ variables and $|I||J||K| + |K|$ constraints.

2.5. Cryptograms

These puzzles are found in newspapers and puzzle books. Their solution by IP and CS methods is mentioned by Wilson (1990), Barth (1996) and recently Hajian et al. (1999). An example is to "solve" the addition sum

$$\begin{array}{cccc} S & E & N & D \\ M & O & R & E \\ \hline M & O & N & E & Y \end{array}$$

by assigning the letters to digits $\in \{0, 1, \cdots, 9\}$ so that different letters are assigned to different digits. One possible IP formulation involves variables:

$$x_i = \text{ value assigned to letter } i$$

$$\delta_{ij} = \begin{cases} 0 & \text{if } x_i < x_j \\ 1 & \text{if } x_j < x_i \end{cases}$$

where $i, j \in \{S, E, N, D, M, O, R, Y\}$, and the "addition" constraint,

$$\begin{aligned} &1000S + 100E + 10N + D \\ &+ 1000M + 100O + 10R + E \\ &= 10000M + 1000O + 100N + 10E + Y \end{aligned}$$

holds.

$$x_i - x_j - 10\delta_{ij} \le -1, \text{ all } i, j, \ i < j$$

$$x_j - x_i + 10\delta_{ij} \le 9, \text{ all } i, j, \ i < j.$$

This model has 64 variables and 57 constraints.

One CS formulation retains only the "addition" constraint and introduces the CS constraint

$$all\_different_i\ (x_i).$$

This CS model has eight variables and two constraints.

3. Different Integer Programming Formulations of the "*all_different*" Predicate

The purpose of this section is to show how it is possible to formulate the "*all_different*" predicate in different ways using systems of linear constraints. However, the quality of these formulations depends on both the tightness of the LP relaxation and the number of constraints and variables (the "compactness").

We assume that

$$x_i \in \{a_1, a_2, \cdots, a_k\}, \quad i \in \{1, 2, \cdots, n\},$$

and that $\{a_1, a_2, \cdots, a_k\}$ is an ordered list. There is no loss of generality in taking $a_j = j-1$ for all j.

Call the formulation used in Section 2.5 "Formulation 1:"

$$\delta_{ij} = \begin{cases} 0 & \text{if } i < j \\ 1 & \text{if } i > j \end{cases}$$

$$x_i - x_j - k\delta_{ij} \le -1, \quad \text{all } i, j, \ i < j$$
$$x_j - x_i + k\delta_{ij} \le k-1, \quad \text{all } i, j, \ i < j$$

This formulation has $n(n-1)$ constraints and $n(n-1)$ extra variables. This formulation is very "bad" in the sense that the LP relaxation is very weak. In order to compare formulations, it is convenient to project out all the extra variables in the LP relaxation so that the polytope for the LP relaxation is in the space of the x variables. This may be done by Fourier-Motzkin Elimination (see, for example Williams (1974)). Doing this results in the polytope defined by

$$-1 \le x_i - x_j \le k-1, \text{ all } i, j.$$

Clearly, this is much larger than the convex hull of feasible integer solutions to

$$\textit{all\_different}_i\ (x_i)$$

As an alternative to Formulation 1, we now describe "Formulation 2". We introduce variables

$$\lambda_{ij} = \begin{cases} 1 & \text{if } x_i \text{ takes value } j \\ 0 & \text{otherwise} \end{cases}$$

and the constraints,

$$x_i - \sum_{j=0}^{k-1} j\lambda_{ij} = 0, \quad \text{all } i \tag{1}$$

$$\sum_{j=0}^{k-1} \lambda_{ij} = 1, \quad \text{all } i \tag{2}$$

$$\sum_{i=1}^{n} \lambda_{ij} \le 1, \quad \text{all } i \tag{3}$$

This formulation has kn extra variables and $2n+k$ constraints.

Hajian et al. (1999) apply both Formulation 1 and Formulation 2 to the Cryptogram given in Section 2. They apply Formulation 1 by making use of "nonzero" variables, i.e. variables that can be either negative or positive only. These variables can then be adjudicated in the branching process.

Both Formulation 1 and Formulation 2 are "*polynomial*" in the sense that the number of constraints and variables are polynomial functions of n and k. It can be shown that projecting out the λ_{ij} variables in Formulation 2 in the LP relaxation by Fourier-Motzkin elimination produces the facet constraints described in Section 4. Therefore Formulation 2 is "tight" in the sense that all the vertices give integer solutions.

It is straightforward to see that the facet constraints in Section 4 are all derivable as non-negative combinations of these above, although at this stage we do not know that there are not other facet constraints.

In order to obtain the constraints described in Section 4 we perform the following operations.

Adding constraints (1) for $i = i_1, i_2, \cdots, i_k$ to the last h of constraints (3) taken in multiples of $1, 2, \cdots, h$ respectively gives

$$x_{i_1} + x_{i_2} + \cdots + x_{i_h} - (k-h-1) \sum_{j=0}^{k-h-1} \sum_{i=1}^{n} \lambda_{ij} \le \frac{h(h-1)}{2} \tag{4}$$

Then adding constraints (2) for $i = 1, 2, \cdots, h$ in multiples of $(k-h-1)$ gives

$$\begin{aligned} x_{i_1} + x_{i_2} + \cdots + x_{i_h} + (k-h-1) \sum_{j=k-h}^{k-1} \sum_{i=1}^{n} \lambda_{ij} &\le \frac{h(h+1)}{2} + h(k-h-1) \\ &= \frac{h(2k-h-1)}{2} \end{aligned} \tag{5}$$

Adding in the appropriate non-negativity constraints "$-\lambda_{ij} \le 0$" gives

$$x_{i_1} + x_{i_2} + \cdots + x_{i_h} \le \frac{h(2k-h-1)}{2}.$$

The constraints

$$x_{i_1} + x_{i_2} + \cdots + x_{i_h} \ge \frac{h(h-1)}{2}$$

can be obtained in a similar manner.

This formulation has $2^{n+1}-2$ constraints and n variables. Note that the number of constraints is an *exponential* function of n making impractical for large n. It does, however, provide a way of demonstrating that it (and Formulation 2) represents the convex hull of feasible solutions. This is done in the next section. When n is small, or $n << k$, it may be more efficient to deal with this projected model than the $n(k+1)$ variables of the lifted model.

4. Convex Hull Representation of the "*all_different*" Predicate

Given a set of integers $A = \{a_1, \cdots, a_k\}$, for all i, and $a_i \ne a_j$ for all i, j. We consider

$$\textit{all\_different}\ (x_i),\ x_i \in A,\ \text{for } i = 1, \cdots, n. \qquad (6)$$

The predicate (6) means that $x_i \ne x_j$, for all $i \ne j$. Denoted by S the set of feasible points satisfying (6), and by $conv(S)$ the convex hull of S. We will identify all facets of $conv(S)$.

Since $a_i \ne a_j$ for all i and j, and a_i is integer for all i, it is sufficient to consider

$$A = \{0, 1, \cdots, k\}. \qquad (7)$$

It is clear that if $k < n-1$, rule (6) is infeasible. So, we only consider $k \ge n-1$.

THEOREM 1. *A set of integers A is given as* (7). *If $k = n-1$, and $n > 2$, then*

$$\sum_{i=1}^{n} x_i = \frac{n(n-1)}{2} \qquad (8)$$

is the only facet of $conv(S)$.

PROOF. Since A is given by

$$A = \{0, 1, \cdots, n-1\},$$

and all x_i are different, each x_i takes a distinct value from A. Therefore we always have

$$\sum_{i=1}^{n} x_i = \sum_{t=0}^{n-1} t = \frac{n(n-1)}{2}.$$

Furthermore, it is easy to see that there are n affinely independent points on it (*i.e.* permutations of A).

EXAMPLE 1. Consider *all_different*(x_1, x_2, x_3) defined on $A = \{0, 1, 2\}$. All six feasible points are marked on Figure 1. It is clear that they are all on the plane

$$x_1 + x_2 + x_3 = 3.$$

Another special case is that $n = 2$. We then have the following result.

THEOREM 2. *For $n = 2$, and $k \ge 1$, all the facets of $conv(S)$ are given as follows.*

$$x_1 + x_2 \ge 1$$
$$x_1 + x_2 \le 2k - 1$$
$$0 \le x_1, x_2 \le k.$$

Figure 2 shows the result.

We then consider the general situation. The following lemma is needed in the proof of the main result given later.

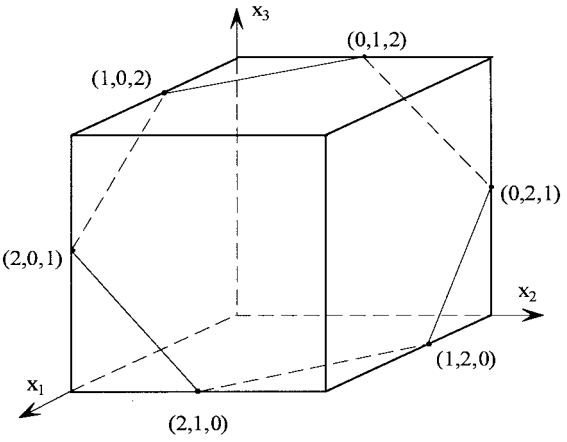

Figure 1 Hyperplane in Example 1

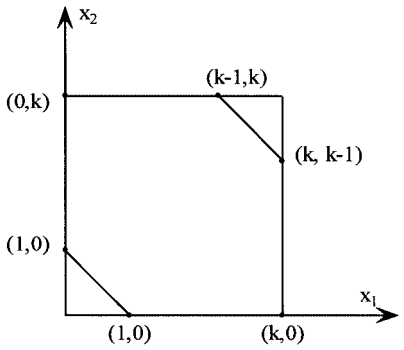

Figure 2 A Special Case of $conv(S)$

LEMMA 1. *Given a vector*

$$(a_1, a_2, \cdots, a_n),$$

where not all components a_i's are the same and they are arranged in increasing order $a_1 \le \cdots \le a_n$, and a vector

$$(b_1, b_2, \cdots, b_n), \quad b_i \ne b_j, \text{ for all } i, j.$$

Rearrange components b_i's in decreasing order,

$$b'_1 > b'_2 > \cdots > b'_n. \tag{9}$$

Then

$$\sum_{i=1}^{n} a_i b'_i < \sum_{i=1}^{n} a_i b_i. \tag{10}$$

PROOF. We show this by induction. For $n = 2$, if

$$a_1 < a_2 \text{ and } b_2 > b_1,$$

Then

$$(a_1 b_2 + a_2 b_1) - (a_1 b_1 + a_2 b_2) = (a_1 - a_2)(b_2 - b_1) < 0.$$

Assume that (10) holds for $n-1$. Rearrange the b_i's such that (9) holds. If $b'_n = b_n$, then (10) holds for n. Assume that

$$b'_n = b_t, \quad t \le n-1.$$

Then

$$\begin{aligned}\sum_{i=1}^{n} a_i b'_i &= \sum_{i=1}^{n-1} a_i b'_i + a_n b'_n = \sum_{i=1}^{n-1} a_i b'_i + a_n b_t \\ &< \sum_{i=1, i\ne t}^{n-1} a_i b_i + a_t b_n + a_n b_t \\ &< \sum_{i=1, i\ne t}^{n-1} a_i b_i + a_t b_t + a_n b_n \\ &= \sum_{i=1}^{n} a_i b_i.\end{aligned}$$

Thus, (10) holds for all n.

LEMMA 2. *The inequalities*

$$\sum_{t=1}^{h} x_{i_t} \ge \frac{h(h-1)}{2}, \quad h = 1, \cdots, n, \tag{11}$$

$$\sum_{t=1}^{h} x_{i_t} \le \frac{h(2k-h+1)}{2}, \quad h = 1, \cdots, n. \tag{12}$$

are facets of $conv(S)$.

PROOF. Take any feasible point $(x_1^\circ, \cdots, x_n^\circ)$ from *all_different*$(x_1, \cdots, x_n)$. There is at most one variable that could be equal to 0. Without loss of the generality, let $x_1^\circ = 0$. Then for the variables left, there is at most one that could be less than 1, but not 0. Let $x_2^\circ = 1$. Repeating these, we thus have

$$\sum_{t=1}^{h} x_{i_t} \ge \sum_{i=1}^{h} x_i^\circ = \sum_{i=0}^{h} i = \frac{h(h-1)}{2}.$$

Thus, (11) is feasible. Similarly, (12) is feasible.

Next, we show that they are facets. For $(i_1, \cdots, i_h) \subseteq (1, \cdots, n)$, let $(x_{i_1}, \cdots, x_{i_h})$ take values

$$\begin{array}{l}(0,\ 1,\ \cdots,\ h-1,\quad h)\\ (1,\ 2,\ \cdots,\quad h,\quad 0)\\ (2,\ 3,\ \cdots,\quad 0,\quad 1)\\ \quad\cdots\\ (h,\ 0,\ \cdots,\ h-2,\ h-1)\end{array}$$

And let other variables take values from $\{h+1, \cdots, n\}$ alternatively. It is not difficult to see that these are n affinely independent points. They are all on (11). Thus, (11) defines a facet. Similarly, (12) defines a facet.

LEMMA 3. *Given $b \ge 0$, $h \le n$. If*

$$a_{i_1} x_{i_1} + \cdots + a_{i_h} x_{i_h} \ge b, \tag{13}$$

defines a facet of all_different$(x_1, \cdots, x_n)$ defined on A, then it is a positive scalar multiple of (11). *Similarly, if*

$$a_{i_1} x_{i_1} + \cdots + a_{i_h} x_{i_h} \le b. \tag{14}$$

defines a facet of all_different$(x_1, \cdots, x_n)$ defined on A, then it is a positive scalar multiple of (12).

PROOF. Assume that (13) defines a facet of *all_different*$(x_1, \cdots, x_n)$ defined on A. If a_{i_j}'s are different, then without loss of the generality, assume that

$$a_{i_1} \le a_{i_2} \le \cdots \le a_{i_h}.$$

Consider a point $p^\circ = (x_1^\circ, \cdots, x_n^\circ)$ on the facet,

$$a_{i_1}x_{i_1}^\circ + \cdots + a_{i_h}x_{i_h}^\circ = b.$$

Rearrange the components of p° in decreasing order, we have a new point p'

$$p' = (x_1', \cdots, x_n').$$

Since p' is also a feasible point,

$$a_{i_1}x_{i_1}' + \cdots + a_{i_h}x_{i_h}' \geq b. \tag{15}$$

On the other hand, by Lemma 1, we have

$$a_{i_1}x_{i_1}' + \cdots + a_{i_h}x_{i_h}' < b. \tag{16}$$

This contradicts (15). Thus we must have $a_{i_j} = \alpha$, for $j = 1, \cdots, h$ and α is a constant. Thus (13) is rewritten as

$$\alpha(x_{i_1} + \cdots + x_{i_h}) \geq b. \tag{17}$$

It is clear that $\alpha > 0$, since otherwise (17) cannot be valid. Therefore, (17) can be written as

$$x_{i_1} + \cdots + x_{i_h} \geq \bar{b}, \tag{18}$$

where $\bar{b} = b/\alpha \geq 0$. Since (13) is a facet, $\bar{b}$ takes the smallest value, which is $\frac{h(h-1)}{2}$. The case for (14) can be shown similarly. This completes the proof.

To summarize Lemma 2 and Lemma 3, we have the following result.

THEOREM 3. *A is a set of integers given as* (7). *For* $k \geq n$, *all facets of* $conv(S)$, *are given by*

$$\sum_{t=1}^{h} x_{i_t} \geq \frac{h(h-1)}{2}, \quad h = 1, \cdots, n, \tag{19}$$

$$\sum_{t=1}^{h} x_{i_t} \leq \frac{h(2k-h+1)}{2}, \quad h = 1, \cdots, n. \tag{20}$$

5. Conclusion and Further Work

This result, together with the lifted formulation of which it is the projection, shows that the *all_different* predicate can be expressed reasonably concisely in integer programming in its tightest possible form. Further work will consist of (i) computational experiments with this formulation on practical problems and (ii) the representation of other commonly used constraint-satisfaction constraints in terms of facets of their convex hull.

Acknowledgments

This work was supported by The Hong Kong Polytechnic University Research Grant A-PA97.

References

Balas, E. 1974. Disjunctive programming: facets of the convex hull of feasible points. Working Paper 348, GSIA, Carnegie Mellon University, Pittsburgh, PA.

Barth, P. 1996. *Logic-based 0-1 Constraint Programming*. Kluwer, Boston, MA.

Brailsford, S.C., P.M. Hubbard, B.M. Smith, H.P. Williams. 1995. The progressive party problem: integer linear programming and constraint programming compared. U. Montanar, F. Rosi, eds. *Principles and Practice of Constraint Programming*. Springer-Verlag, Berlin, Germany.

Darby-Dowman, K., J. Little. 1998. Properties of some combinatorial optimization problems and their effect in the performance of integer programming and constraint logic programming. *INFORMS J. Computing* **10** 276–286.

Crowder, H., F.L. Johnson, M.W. Padberg. 1983. Solving large-scale zero-one linear programming problems. *Operations Research* **31** 803–834.

Fourer, R. 1998. Extending a general-purpose algebra modeling language to combinatorial optimization: a logic programming approach. D.L. Woordruff, ed. *Advances in Computational and Stochastic Optimization, Logic Programming and Heuristic Search*. Kluwer, Boston, MA.

Hajian, M.T., R. Rodosek, B. Richards. 1999. Introduction of a new class of variables to discrete and integer programming problems. *Annals of Operations Research* **86** 39–51.

Hammer, P.L., E.L. Johnson, U.N. Poland. 1975. Facets of regular 0-1 polytopes. *Mathematical Programming* **8** 178–206.

Hooker, J.N. 1998. Constraint satisfaction methods for generating valid cuts. D.L. Woodruff, ed. *Advances in Computational and Stochastic Optimization, Logic Programming and Heuristic Search*. Kluwer, Boston, MA.

Hooker, J.N. 2000. *Logic Based Methods for Optimization*. Wiley-InterScience, New York.

Jeroslow, R. 1989. Logic-based decision support: mixed integer model formulation. *Annals of Discrete Mathematics* **40** North-Holland, Amsterdam, The Netherlands.

McKinnon, K.I., H.P. Williams. 1989. Constructing integer programming models by the predicate calculus. *Annals of Operations Research* **21** 227–246.

Proll, L., B. Smith. 1998. Integer linear programming and constraint programming approaches to a transportation design problem. *INFORMS J. Computing* **10** 265–275.

Tsang, E. 1994. *Foundations of Constraint Satisfaction*. Wiley, Chichester, UK.

Van Hentenryck, P. 1989. *Constraint Satisfaction in Logic Programming*. MIT Press, Boston, MA.

Williams, H.P. 1974. Experiments in the formulations of integer programming problems. *Mathematical Programming Study* **2** 180–197.

Williams, H.P. 1978. The formulation of two mixed integer programming problems. *Mathematical Programming* **14** 325–331.

Williams, H.P. 1986. Fourier's method of linear programming and its dual. *The American Mathematical Monthly* **93** 681–334.

Wilson, S.M. 1990. The formulation and solution of logical puzzles and some implication for artificial intelligence. *Bulletin of the Institute of Mathematics and Its Applications* **26** 86–90.

Wolsey, L.A. 1975. Facets for a linear inequality in 0-1 variables. *Mathematical Programming* **8** 165–178.

Yan, H., J.N. Hooker. 1999. Tight representation of logic constraints as cardinality rules. *Mathematical Programming* **85** 363–377.

Accepted by John N. Hooker, Jr.; received March 1999; revised June 2000, July 2000; accepted August 2000.