NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

TIGHT REPRESENTATION OF LOGICAL CONSTRAINTS AS CARDINALITY RULES

John N. Hooker, Hong Yan

EDRC 70-06-92

Tight Representation of Logical Constraints as Cardinality Rules *

J. N. HOOKER
H. YAN
Graduate School of Industrial Administration
Carnegie Mellon University, Pittsburgh, PA 15213 USA

June 1992

Abstract

We address the problem of finding a "tight" representation of complex logical constraints in a mixed integer programming model by describing a convex hull representation of cardinality rules. A cardinality rule asserts that if at least k of the propositions $A_1, \ldots, A_m$ are true, then at least l of the propositions $B_1, \ldots, B_n$ are true.

As rule-based systems and other types of logic modeling grow in popularity, logical rules and propositions can play an increasingly important role in mathematical programming models. Such simple logical constraints as "if A is produced, then either B or C must be produced" have long been a part of mathematical programming. But much more complex logic models are now being formulated, and they can also be embedded in mathematical programming models.

A logic model is a description of a problem expressed in some logical formalism, usually propositional or predicate logic, from which facts about its solution can be deduced. Expert systems and other rule-based systems are logic models, as are programs written in the logical programming language PROLOG. It is useful to incorporate a logic model into a mathematical programming model when neither model alone adequately describes reality, or when the logic model contains constraints or heuristic rules that ease the

*The first author is partially supported by AFOSR grant 91-0287 and ONR grant N00014-92-J-1028. Both authors are partially supported by NSF grant 1-55093.

solution of the mathematical model by reducing the number of alternatives that must be examined.

When logical constraints become a significant component of mathematical programming models, the quality of their representation becomes an important issue. We wish to address this problem by showing how best to represent a particular type of logical formula, namely a *cardinality rule*, in a mixed integer programming (MIP) model. We focus on cardinality rules because we have observed that logical constraints take this form in a large variety of applications. In fact McKinnon and Williams [11] use cardinality conditions as a basic form in which to express logical constraints.

A cardinality rule has the form,

> If at least (or at most) k of the propositions $A_1, \ldots, A_m$ are true, then at least (or at most) l of the propositions $B_1, \ldots, B_n$ are true.

An *elementary* cardinality rule is one in which each A_i and each B_j is a *literal* (an atomic proposition or its negation) and no atomic proposition occurs more than once. The rules one typically finds in expert systems,

> if $A_1, \ldots, A_m$ are all true, then B is true,

are special cases of elementary cardinality rules, as are *cardinality clauses* of the form,

> at least (or at most) k of $A_1, \ldots, A_m$ are true.

The usual method for representing logical conditions in an MIP model has two stages. They are first rewritten as a conjunction of logical "clauses," which is to say in conjunctive normal form (CNF). A *clause* is a disjunction of literals, such as,

$$x_1 \vee \neg x_2 \vee x_3,$$

where $\neg$ means "not," and $\vee$ means "or." Each clause is then written as an inequality in 0-1 variables, which for this example is,

$$x_1 + (1 - x_2) + x_3 \geq 1,$$

where x_j is interpreted as true when $x_j = 1$ and false when $x_j = 0$.

Unfortunately this is typically a very "loose" representation of the logical conditions, in the sense that its linear relaxation (which replaces $x_j \in \{0, 1\}$ with $0 \leq x_j \leq 1$) describes a polytope that has many fractional extreme

points. This complicates the solution of the model, since most solution techniques make essential use of the linear relaxation.

The difficulty with CNF representation is not that the individual clauses are poorly represented. In fact, each clause receives the tightest possible representation, namely a *convex hull representation.* This is a set of 0-1 inequalities whose linear relaxation describes the convex hull of the 0-1 points satisfying them. Rather, the difficulty is that there are typically a large number of clauses in CNF. This results in a loose representation of the formula as a whole, even though each individual clause is tightly represented. (R. Jeroslow discusses this principle in [9].) This problem is particularly acute for a cardinality rule, since the number of clauses in its CNF equivalent grows exponentially with the rule's length if no new variables are added (and is apparently quite large even if new variables are added).

Thus when logical constraints occur or can be expressed in the form of elementary cardinality rules, it is far better to give each a convex hull representation directly than to convert it to CNF first. The resulting description is no shorter than the CNF equivalent, because it contains all the inequalities that appear in CNF. But it avoids the loose representation that results from using *only* the inequalities in the CNF representation.

We will therefore state an algorithm that generates for any elementary cardinality rule a set of 0-1 inequalities that provide a convex hull representation of it. Our main result is that these inequalities in fact describe the facets of the convex hull.

Several authors have contributed to the inequality representation of logical formulas in CNF. Tseitin [12] showed how to convert any formula of propositional logic to CNF in linear time by adding new variables. Cook [3, 4], Dantzig [5], Blair *et al.* [2], and Williams [13, 14, 15] discussed the use of 0-1 inequalities to represent logical constraints in CNF, and Karp [10] used them to show that integer programming is NP-hard. The shortest such representation, however, was recently proposed by Wilson [16].

Some attention has also been paid to the representation of cardinality clauses (recall that these are a special case of elementary cardinality rules). Hadjiconstantinou and Mitra [6] described an algorithm for the automatic conversion of logical formulas, including cardinality clauses, into inequality form. But they do not consider the tightness of the representation. Hooker [8] described a generalized resolution procedure that generates all the undominated implications of a set of cardinality clauses. This can lead to a tighter representation because it in effect generates valid cuts. But it can be

computationally expensive if carried to completion on a large set of clauses. In research that proceeded concurrently with ours, Araque and Chandru [1] found a convex hull representation of cardinality clauses. But their work has a different focus, because it does not consider cardinality rules in general, and it describes the polyhedron associated with a set covering formulation of the problem, which has twice the dimension of the polyhedron we study.

We begin in Section 1 by augmenting propositional logic with cardinality rules to obtain "cardinality logic." Since our convex hull representation is designed for elementary cardinality rules, we show how to convert any formula of cardinality logic to a conjunction of elementary cardinality rules (*cardinality normal form*) in linear time. We also show that the CNF of an elementary cardinality rule grows exponentially with the length of the rule, when no variables are added. When additional variables are used, CNF is polynomial but still quite large using what is apparently the best known conversion.

Section 2 states our algorithm for generating the convex hull description of a cardinality rule and proves its correctness. In Section 3 we use an example to illustrate its application.

1 Cardinality Rules

We begin by augmenting propositional logic with cardinality rules. We then show how to write any formula of cardinality logic in cardinality normal form. Finally we discuss what is involved in the reduction of elementary cardinality rules to CNF.

1.1 Cardinality Logic

A cardinality rule is written,

$$(A_1, \ldots, A_m)_k \Rightarrow (B_1, \ldots, B_n)_l, \tag{1}$$

and is read, "if at least k of $A_1, \ldots, A_m$ are true, then at least l of $B_1, \ldots, B_n$ are true." We assume $m \geq k \geq 0$, $n \geq 0$, and $l \geq 1$. The phrase "at least" can be replaced with "at most" by writing,

$$(\neg A_1, \ldots, \neg A_m)_{m-k} \Rightarrow (\neg B_1, \ldots, \neg B_n)_{n-l}.$$

Ordinary rules have $m = k$ and $l = 1$ in (1). When $m = 1$ or $n = 1$ we will omit the parentheses, so that $(A)_1 \Rightarrow (B)_1$ is written $A \Rightarrow B$.

When the antecedent is a tautology, which is to say $k = 0$, (1) asserts that at least / of $B\backslash,\ldots, B_n$ are true. In such cases we can assume without loss of generality that $m = 0$. When the consequent is a contradiction, which is to say / > n, (1) asserts that fewer than k of $A_i,\ldots, A_m$ are true. In such cases we can assume that $n = 0$ and / = 1. We therefore suppose that every cardinality rule satisfies a) either $m > k > 1$ or m = = 0, and b) either $n > l >$ lor(n,/) = (0,l).

Propositional logic is recursively defined to consist of *atomic propositions* $x\backslash, X2,\ldots,$ which are regarded as formulas, plus all formulas of the forms -«A, A **A** 5, A **V** 5, A *D B,* and A = fi, where A and 5 are formulas. Here **A** means "and," **V** means "or," **D** means "implies," and = means "is equivalent to." Implication and equivalence are defined by,

$$(A\ D\ B) \quad =_{def} \quad (-A\ V\ 5) \tag{2}$$

$$(A = B) \quad =_{def} \quad ((A\ D\ B)\ /\backslash (B \quad D\ A)). \tag{3}$$

Cardinality logic is recursively defined to consist of all the above formulas and all formulas of the form (1), where $A_i,\ldots, A_m$ and $B_i,\ldots, B_n$ are formulas of cardinality logic.

1.2 CARDINALITY NORMAL FORM

We now show how to convert any formula of cardinality logic to cardinality normal form in linear time. If the formulas to be represented are already elementary cardinality rules, this conversion is not necessary, and one can proceeed directly to the linear inequality representation (Section 2).

We first review conversion to conjunctive normal form (CNF) in ordinary propositional logic. This can be done by a) using (2) and (3) to remove all occurrences of **D** and =; b) using De Morgan's laws,

$$-(A\ A\ S) \quad = \quad (-LAVNFI)$$

$$->(A\ V\ B) \quad = \quad (-«A\ A\ -«5),$$

and the equivalence ->-»A = A to absorb all negations into literals; and c) using the distribution law,

$$A\ V\ (F\ L\ A\ C) \quad = \quad (A\ V\ B)\ A\ (A\ V\ C),$$

to transform the result to a conjunction of clauses.