MATHEMATICAL PROGRAMMING STUDIES

MATHEMATICAL PROGRAMMING STUDY 2

Approaches to Integer Programming

Edited by M.L. BALINSKI

December (1974)

NORTH-HOLLAND PUBLISHING COMPANY – AMSTERDAM

This STUDY is also available to non-subscribers in a book edition.

Printed in The Netherlands

PREFACE

Integer programming, in its often disparate guises, has been advancing with vigor yet by fits and starts in many different directions. The group theoretical or algebraic approach, first investigated by Gomory in 1958, is now well established and being pushed forward computationally and theoretically. In particular, a spate of recent activity concerns polytopal descriptions of regions having integer valued extreme points. Enumeration, or "branch-and-bound," spurred by the successes of Little *et al.* in solving traveling salesman problems in 1963, and perhaps the most effective approach for practical computation, is being pursued. Heuristic devices for finding "good solutions" quickly, such as that used with such surprising results for the traveling salesman problem by Lin in 1965, continue to be proposed. And, since the enormously successful marriage of "relaxation" and "branch-and-bound" affected by Held and Karp (again for the traveling salesman) in 1971, the use of relaxation in integer (and, incidentally, otherwise modified) programming problems is being explored.

Along with these various and other tendencies has come the need for generalized settings or theories explaining the "place" or categorizing the type of this or that approach, the need for sets of reasonable problems with which to experiment and by which to compare approaches, and, finally, and most important, the expanding need of the practitioner to solve real problems.

All of these directions, tendencies, and needs are displayed in this STUDY: as such it is a fairly reliable witness to the state of the art and major preoccupations and efforts in integer programming today.

Paper 1, by Breu and Burdet, focusses on enumeration but is motivated by pragmatic computational experimentation with "reasonably" difficult or large problems. Guided by the results of computer tests, it passes in review various ideas, tactical choices and overall strategies which have been or can be used in defining a branch and bound method. On this level it takes on the role of a survey, categorizing and comparing. At the same time it reports extensively on the results of tests, and gives careful descriptions for the generation of test problems.

Paper 2, by Burdet and Johnson, is concerned with the group problem

(itself a relaxation of the) underlying integer programs, but attacks it by means other than the analysis of the group structure. This is at least partly motivated by the potentially large order of the groups encountered. Using results about the generation of valid cuts for the group problem with subadditive functions, and more particularly, "diamond gauge functions," the paper develops a new cutting algorithm for solving the problem in which the computational effort is not directly related to the order of the group.

Paper 3, by Fulkerson, Nemhauser and Trotter, identifies the particularly interesting, because difficult, minimim cardinality set-covering problems which arise in seeking the 1-width of incidence matrices of Steiner triple systems. There is pure mathematical interest in seeing new information concerning old speculations from the solution of one of these problems (having 117 constraints and 27 variables). There is both theoretical and computational interest in the challenge of solving the other problem (having 330 constraints and 45 variables), which has so far resisted the efforts of theoretician and practitioner alike.

Paper 4, by Geoffrion, notes the success of the relaxation idea in solving various special problems (notably the traveling salesman) and develops the idea in a general framework for use in enumerative approaches to integer programs. Certain already known approaches emerge as special cases, such as the "penalty" approach, "surrogate" constraints, and some cuts. Application to three particular types of integer programs—0,1 problems, 0,1 problems with "multiple choice," and 0,1 problems with location type inequalities—is carefully investigated.

Paper 5, by Ibaraki, Ohashi and Mine, proposes a heuristic approach and program for the solution of mixed integer programming problems which extend earlier ideas of Hillier. Extensive computational evidence is presented to argue the validity and excellence of the approach.

Paper 6, by Johnson, analyses the group problem which results from a relaxation of the mixed integer programming problem. Although considerably more difficult than the group problem for the pure integer problem, subadditive functions are again used in providing cuts and faces of the polyhedron of feasible solutions to the group problem.

Finally, Paper 7, by Williams, investigates the effect on computation of formulating each of five problems arising in practice as two different integer programs. In each case a branch-and-bound program is used. Definite differences are found and some morals are drawn from these.

M.L. Balinski

CONTENTS

Mathematical Programming Study 2 (1974) 1–50. North-Holland Publishing Company

BRANCH AND BOUND EXPERIMENTS IN ZERO–ONE PROGRAMMING*

Raymond BREU** and Claude-Alain BURDET***

University of Ottawa, Ottawa, Ontario, Canada

Received 28 May 1974
Revised manuscript received 23 August 1974

This paper investigates, both from a theoretical and practical point of view, the ability of a general branch and bound approach to solve pure zero-one programming problems. Large scale experiments show that our algorithm is able to compete successfully with special purpose implicit enumeration codes.

An effort has been made in the presentation to put various concepts into the proper perspective from the point of view of an efficient computer implementation. We give a brief general description of those "classical" aspects which were found useful in our framework. Most of the paper, however, deals with new ideas emerging from pragmatic analysis.

Introduction

The efficiency of mathematical programming codes and their ability to solve large scale problems in the area of linear and mixed-integer programming has been greatly increased over the past few years. In fact many practical problems can now be solved [27, 30, 35].

While the branch and bound approach proposed by Beale and Small [7], Land and Doig [22], Little et al. [26] and others has proved quite successful [8, 28] for mixed integer problems with *many* linear (continuous) and *few* integer-constrained variables, it has not received much attention in the all-integer context.

The area of pure integer (also zero–one) programming has evidently not enjoyed the same development trend; it seems to remain limited to an order of magnitude (50 to 100 zero–one variables, depending on the degree of difficulty of the problem) reached by the better *enumerative* codes [2, 16, 18]. In order to break this barrier, several new approaches have recently been proposed to construct cutting planes [3, 4, 11, 12, 19] and new algorithms [5, 10, 13, 21]. The practical value of such "sophisticated" techniques has yet to be tested in a realistic implementation, however.

* This project was partially supported by the National Science Foundation (grant GP 37510X) and the U.S. Office of Naval Research (contract N0014 67 A 0314 00u NR 047 048)

** Visiting Fellow 1972 1973.

*** Associate Professor of Industrial Administration, on leave from the Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pa.

In spite, or perhaps because of their conceptual simplicity, the so-called rudimentary branch and bound algorithms have generated little enthusiasm in pure zero–one programming. Such an LP based approach is expected to introduce a mass of arithmetic operations, problems of numerical accuracy and various additional difficulties. Furthermore, as pertinently observed by Balas in [1] all that is required to solve pure integer programs are boolean and/or integer additions. In reality, however, things are somewhat different: in order to become efficient, Balas' algorithm must be reinforced by a multitude of special purpose devices; LP solutions, in particular, are extremely useful [16]. We investigate here an alternate methodology based on the *flexibility* of a general Branch and Bound approach and we show that it is competitive with other currently developed and implemented algorithms. This opinion is supported by the following comments:

(i) If CPU time is taken as a measure, our code's performance on "hard" and/or "large" problems is satisfactory (see Tables X.11 and X.12).

(ii) The approach proves *reliable* in the sense that it solves *all* test problems from the literature even those previously unsolved.

(iii) In spite of the fact that our code is set up to handle mixed problems, its performance on highly structured pure zero one problems is good (see Table X.13).

(iv) The code can also contend with subsidiary issues of a practical nature such as *sensitivity* and *multiple* (sub-)optimal *solutions* within a given percentage of the optimum.

(v) Through its ability to solve both pure and mixed formulations, our approach allows the investigation of integrality requirements and their influence on the solutions.

Success in solving difficult zero-one (pure or mixed) problems depends upon a few essential elements. First, one has to adequately control the effects of numerical errors and the number of arithmetic operations, in order to generate updated subproblems and to perform a great many pivot steps without distortion of the "exact problem structure". A careful organization of the computer operations is also required during the calculation of bounds and branching choices (tactics). Finally, the natural flexibility of the method ought to be exploited: during the execution, one should have ready access to the mass of information accumulated during previous node computations. This can be accomplished by the adaptive strategies suggested in Section III.2.

The paper is self-contained and surveys those classical aspects which

were found useful in the development of new techniques. We frequently attempt to test the relative merit of elementary yet *powerful* devices and of more elaborate results. Section I outlines the general framework and terminology used throughout the paper. It exposes the interested reader to some practical details of the implementation. Section II contains a description of methods to compute bounds and to choose a branching variable. In Section III we discuss strategic principles to determine the sequence of tactical operations.

Throughout the study we try to remain faithful to a *pragmatic* philosophy, testing a wide variety of computationally simple devices in our experimental code (all-FORTRAN, all-in-core). Initially all ideas which were implemented stemmed from classical results and common sense. But later, we gradually obtained significant improvements of the overall performance by observing the solution process and making ad hoc modifications. The practical value of a given property critically depends upon *the form of its implementation*; this is not always apparent from the literature where such aspects are rarely developed in sufficient detail. The pattern of our research is different in that it *originates* at the level of a coded implementation. In this precise context, our investigations have led us to exploit elementary properties of integer programs, some of which have not been previously explored. The ultimate criterion to validate algorithmic proposals in this manner is based on computational performance and our judgment in that respect is subjective. It is colored by our general approach and the particular programming style of our code. The selection of test problems is a further element of subjectivity. We do not feel that performance of a code on problems which can be solved in less than a couple of seconds is meaningful. In view of the exponential growth underlying integer programs, the extrapolation of any measure of efficiency becomes extremely ambiguous. We thus base our judgment on difficult and/or large problems.

I. Branch and bound

I.1. *General framework and terminology*

Consider the linear program

$$\begin{aligned} &\text{minimize } x_0 = c\,x, \\ &\text{subject to } A\,x = b \end{aligned} \tag{1}$$

with vectors x and c (n components), b (m components) and a m by n matrix A; we assume that the linear program (1) is in Dantzig format and that the first n_{zo} components of the vector x are *structural binary* (zero–one) variables, i.e.,

$$x_i = 0,1 \quad \text{for all } i \in N_{zo} = \{1, 2, \ldots, n_{zo}\}. \tag{2}$$

Depending on the nature of the rows of A, some (or all) slack variables may also be integer constrained.

The following terms are used throughout the paper:

Node. Any partition of the index set N_{zo} into three mutually exclusive sets:

$$N_{zero} \cup N_{one} \cup N_{free} = N_{zo}$$

and the *corresponding subprogram*

$$\begin{aligned} &\text{minimize } x_o = c\,x, \\ &\text{subject to } A\,x = b, \\ &\qquad x_i = 0, \qquad i \in N_{zero}, \\ &\qquad x_i = 1, \qquad i \in N_{one}, \\ &\qquad 0 \le x_i \le 1, \qquad i \in N_{free}. \end{aligned}$$

(There are $2^{(n_{zo}+1)} - 1$ possible nodes.)

If the subprogram has an optimal solution, then the corresponding tableau in *explicit* format is denoted as follows:

$$\begin{aligned} x_o &= \bar{x}_o + \sum_{j \in \bar{N}} \bar{c}_j x_j, \\ &\vdots \\ x_i &= \bar{x}_i - \sum_{j \in \bar{N}} \bar{a}_{ij} x_j \end{aligned}$$

where $\bar{N}$ is the (optimal) non-basic index set for the subprogram and one has

$$\begin{aligned} &\bar{c}_j \ge 0 && \text{for all } j \in \bar{N}, \\ &0 \le \bar{x}_i \le 1 && \text{for } i \in N_{free}, \\ &0 \le \bar{x}_i && \text{otherwise.} \end{aligned}$$

Furthermore, if $0 < \bar{x}_i < 1$ for some $i \in N_{free}$, then the basic variable x_i is said *fractional.*

Successor node. A node k is called *successor* of a node h if one has

$$N^{(k)}_{\text{zero}} \supseteq N^{(h)}_{\text{zero}},$$

$$N^{(k)}_{\text{one}} \supseteq N^{(h)}_{\text{one}},$$

$$N^{(k)}_{\text{free}} \subseteq N^{(h)}_{\text{free}}.$$

Bound (of a given node). Any real number β such that $\hat{x}_0 \geq \beta$ for all (integer) feasible solutions $\hat{x}$ of the subprogram corresponding to that node; in particular, one has

$$\hat{x}_i = 0 \text{ or } 1 \quad \text{for all } i \in N_{\text{free}}.$$

Penalty (at a given node). Any quantity $p \geq 0$ which can be used to produce a valid bound $\beta = \bar{x}_0 + p$ from the LP optimal value $\bar{x}_0$ of the corresponding subprogram.

Cut-off value. The quantity γ is called a cut-off value when the solutions to (1, 2) which satisfy $x_0 > \gamma$ are of no interest.

Level (of a node). Number of fixed variables at that node.

Pending node. A node which has been explicitely identified but is not yet either fathomed or branched (see below).

Fathomed node. A node which need not be further considered because
– either its subprogram is infeasible
– or its bound is above the cut-off value ($\beta > \gamma$), implying that no solution of the corresponding subprogram is of interest.

Branched node. A node whose two direct successors are either pending or fathomed.

Feasible, optimal, suboptimal solutions. As customary in mathematical programming, taking also the integrality requirements (2) into account.

Breadth (of a level). Number of branched or pending nodes at that level.

Depth (of a branch). Number of branched nodes on that branch.

Front. Set of pending nodes (at a given time) during the execution; we shall also use the *maximal* front as a cardinal measure of performance.

Tactics. Considerations made for a single node and with the corresponding local information; for instance penalty and bound computations, determination of a branching variable at that node.

Strategy. Rules and guidelines governing the sequence of tactical operations; for instance, the choice of the next node, and the control of breadth and front. In contrast to tactics, strategies invoke informations gathered from several nodes (subtree).

Tactical improvement. A modification of a procedure which reduces the amount of computations or generally increase the efficiency *at a single node.*

Strategic improvement. A modification which brings about a reduction of the number of nodes to be considered for further tactical operations such as bound computations, branching, storage, etc.

One may remark that neither tactical nor strategic improvements need bring about an overall improvement; a tactical improvement may be strategically catastrophical because it triggers an avalanche of additional nodes. Similarly a strategic improvement may reduce the number of nodes but require many expensive computations for each node.

The basic framework in which we propose to operate is best referred to as Branch and Bound [7, 22, 26, 29]. The algorithm may be sketched as follows:

(1) Initialize the algorithm by defining a cut-off value (e.g., $\gamma = +\infty$) and put the zero level node ($N_{\text{zero}} = N_{\text{one}} = \emptyset$, $N_{\text{free}} = N_{\text{zo}}$) into the front.

(2) Select a node of the front for further branching. Determine its two successors (subprograms) and compute the corresponding bounds. Update γ whenever new integer solutions are found with a better (i.e., smaller) objective function value than the current cut-off γ. Update the front by removing the branched node and adding its two successors (if they are not immediately fathomed). Fathom the nodes of the front which are dominated by the new cut-off value γ. Iterate (2).

(3) The algorithm terminates with an empty front. The k best solutions are collected or one shows that there are only $h < k$ feasible (integer) solutions.

The algorithm contains three crucial procedures:
- *bound computation* (tactical),
  branching variable selection (tactical),
- *node selection* (strategic).

And the efficiency of these procedures, measured by the overall performance of the algorithm, depends entirely on the following:

(i) Detective ability to divine a path (branch) to the optimal solution or, at least, to a very good suboptimal solution; this provides a *tight cut-off value early* in the execution.

(ii) Power of discernment to eliminate fruitless branches as close as possible to their root; this is the job of the bound machinery (optimality) and/or the arsenal of logical implications (feasibility) described in Section II.

(iii) Computational effectiveness to derive with sufficient accuracy the information required by (i) and (ii), *at a low cost*.

Theoretically, observations of this type offer little more than the frustration of a trivial philosophical ratiocination; but one must constantly keep them in mind if an implementation is to be more than a vaguely coherent conglomerate of subroutines which conscienciously hammers through numerical arabesques with the elegance of a pachyderm.

I.2. *Some linear programming aspects*

The computational results presented here are obtained with the help of an all FORTRAN LP code using the product form of the revised Simplex method. In order to handle large and structured problems, array storage and all arithmetic manipulations handle only non-zero entries explicity. All LP's are processed in their primal formulation. In the present zero one context, it is necessary to treat all upper bounding constraints implicity.

It is well known that linear programs occurring in the integer programming context (particularly all integer) often possess peculiar characteristics. Thus special care must be given to the various "exceptional" cases of LP theory. The specific configuration of our implementation also puts a heavier emphasis on some LP subroutines which are otherwise expected to play an accessory role as explained below.

I.2.1. *Reinversion and Eta-file*

Branch and Bound implementations require very frequent (re)construction of a basis; this is due to a multitude of numerical and conceptual reasons. One must frequently control numerical accuracy with great care to avoid eliminating integral solutions. The branching process also necessitates a Phase I or dual optimization for *every* node and thereby creates unusually voluminous "Eta files" (i.e., unnecessarily long sequences of pivots). Most powerful bounds are based on additional *backward transformations* which generate updated rows; these backward transformations involve a number of arithmetic operations proportional to the length of the Eta file and it is therefore indispensable to keep this number of pivots to a minimum by frequent reinversion. There are also several other minor organizational details which have the tendency to unduly lengthen the Eta file.

I.2.2. *Primal versus dual optimization*

As mentioned in the preliminary comments of this section, we use primal LP formulations. But in an all-integer context with all structural variables zero or one, it is always possible to start with a dual *feasible* solution (replacing x by $1-x$ whenever the cost is negative).

Thus one may either apply a dual pivot selection rule or use a primal approach (with Phase I) for the LP post-optimization in any subprogram (node). Table X.2 contains the corresponding comparison. One would expect dual pivoting to be more efficient for the node-reoptimization occuring after branching. However, experiments show the opposite, as primal re-optimization seems more efficient on subprograms with few variables and many constraints at the lower levels.

I.2.3. *Arborescence*

The tree structure is constructed node by node and stored sequentially in an array called *front*. Variables are flagged and nodes represented by bookkeeping arrays; each bit is activated individually in order to limit storage requirements. The array storage structure in a computer always gives a natural lexicographic order to the tree; but further preference rules can be superimposed at virtually no additional cost. It is possible, for example, to reinforce the lexicographic order (which favors the so-called Last-In-First-Out node selection rule) with a second order relation derived from the bounds; as a result, nodes are activated according to their bounds *in combination with LIFO* (*wishful branching*, see Section II).

The all-in-core storage of the tree and these multiple ordering techniques allow strategic flexibility in the node selection; this would be lost if the system were to rely heavily on intermediary storage media. In our code a minimal amount of information is stored for each node, and in-core memory space was never found to be a scarce resource; there are also strategic and tactical advantages in limiting the front to a modest size (see Section III).

I.2.4. *Control parameters*

We have introduced various *sets of parameters* to control numerical accuracy and storage needs; rather than describing everyone of the more than 40 parameters in detail, we list below the role of each parameter set:

- control the accuracy of *individual operations* (round off errors, zero tests, integrality tests),

- error propagation control (solution checks with reference to the initial system, before and after reinversion tests),
- reinversion control,
- capacity control (number of entries in the Eta file, tree size),
- choice of *pivot selection* rules (for primal or dual feasible tableaux),
- organizational parameters to regulate tree storage.

In comparison to linear programming, the role and importance of control parameters is amplified in integer programming. Their effect is sometimes surprising, not easily understood and lacks theoretical foundation. One easily convinces oneself that dichotomous branching, for instance, represents a foreign element in the flow of LP computations. Due to the influence of round-off errors, it creates infeasibility and inconsistency of the linear subprograms. The high degree of degeneracy in most "exact subprograms" contributes to further complicate the matter; particularly during the solution of some set covering problems, numerically absurd situations may arise.

I.2.5. *The influence of finite arithmetics on the algorithm*

This brief description of numerical aspects would be incomplete without the following remarks; in order to understand the real meaning of the present numerical experiments, it is imperative to realize how the concepts of performance, accuracy and reliability interact:

Measures of performance (e.g. *nodes* or *isec* in Appendix X) are easily influenced by accuracy control parameters. Consider bound calculations, for instance, where the real influence of numerical errors is difficult to assess: one has to introduce control parameters so that, in spite of its inaccuracy, the bound activated by the algorithm remains *valid in an exact sense*; otherwise optimal solutions may be lost. Practically the choice of a value for such control parameters is often a subjective matter: inaccurate (i.e., not rigorously valid) bounds may seem satisfactory, because their potentially disastrous effect in skipping solutions is unlikely to occur. Control parameters also effect performance dramatically: the problem L–S D2 was solved with identical strategies and different sets of control parameters (delivering always the true optimum) and the corresponding solution times ranged from less than 70 seconds to more than 150 seconds. Similar ranges were also found for other measures like *nodes*. Which performance should then be reported?

In all tables of the appendix, we adopt a *conservative attitude*, sacrificing

excellence to *reliability*, in order to ascertain that the optimization is indeed rigorous.

II. Tactical computations in the Branch and Bound algorithm

II.1. *Bounds*

The following developments are presented under the assumption that all structural variables are *bivalent*. This is the focal point of our experiments; the analysis however is easily extended to the mixed case and all subroutines are implemented in this general form.

II.1.1. *Simple bounds*

The primary role of a bound is to fathom nodes; thus good bounds may have a drastic effect on the final phase of the algorithm where optimality must be proved. It should be pointed out however that a good bound is not necessarily one with a good numerical value; the price which must be paid to compute this value is an equally significant factor. A good comparative measure for the quality of a bound can be obtained by comparing it to LP bounds (see BD 1, below).

There is an abundant list of bound proposals in the literature. We have experimented with most of those which seemed promising in the present context; a few of them are mentioned in this subsection, not necessarily bacause they were found very efficient, but rather because they represent the starting point for the improved bounds of the subsections II.1.2 and II.5.

BD 1. *Optimal LP value.* The quantity $\beta = \bar{x}_0$ is easily seen to yield a valid bound; it is generally not very effective but provides a good basis for comparisons.

BD 2. *Optimal LP value with cut(s).* If a valid cut is appended to the subprogram then the bound $\beta = \bar{x}_0$ may (sometimes) be improved. Usually the integrality requirements of the basic variables x_i, $i \in N_{\text{free}}$ are used to calculate valid cuts and this may require a costly LP updating procedure. After inserting the cut into the subprogram, the LP becomes primal infeasible but remains dual feasible; thus every dualpivot step performed to restore primal feasibility furnishes a valid bound. But explicit pivoting is usually too expensive : instead a so-called *penalty* is computed to indicate the improvement after one (single) dual pivot step. Pivoting need not be explicity carried out to determine this penalty value.

BD 3. *Dantzig cut* [14]. The cut $\sum_{j\in\bar{N}} x_j \geq 1$ is valid and yields the penalty

$$p = \min_{j\in\bar{N}_{free}} \bar{c}_j,$$

where $\bar{N}$ denotesthe index set of non-basic (free) variables (integer constrained).

BD 4. *Gomory's mixed-integer cut* [20]. Consider the inequality

$$\sum_{j\in\bar{N}} (-\pi_j)\, x_j \leq -\pi_0 \qquad (\pi_0 > 0).$$

Choose a fractional basic 0-1 structural variable x_i to generate the following coefficients:

$$\pi_0 = 1$$

and, for $j \in \bar{N}$,

$$\begin{aligned} \pi_j &= f_{ij}/\bar{x}_i && \text{if } f_{ij} \leq \bar{x}_i, \\ &= (1 - f_{ij})/(1 - \bar{x}_i) && \text{otherwise,} \end{aligned}$$

where f_{ij} is the (positive) fractional part of the updated coefficient $\bar{a}_{ij}$ when x_j is an integer constrained variable. Note that $\pi_j \leq 1$ holds true. (For continuous variables one has

$$\begin{aligned} \pi_j &= \bar{a}_{ij}/\bar{x}_i && \bar{a}_{ij} \geq 0, \\ &= -\bar{a}_{ij}/(1 - \bar{x}_i) && \text{otherwise.)} \end{aligned}$$

The corresponding penalty is

$$pg = \min_{j\in\bar{N}} \{\bar{c}_j/\pi_j : \pi_j \neq 0\}.$$

These penalties have been successfully used by Tomlin [34] in mixed integer programming.

BD 5. *Dichotomous penalties.* Consider branching on a fractional (basic) 0-1 variable x_i; one introduces the cuts

$$x_i \leq 0, \qquad x_i \geq 1$$

which are valid for the corresponding nodes, thus obtaining the penalties

$$pd = \min_{j\in\bar{N}} \{\bar{x}_i \bar{c}_j/\bar{a}_{ij} : \bar{a}_{ij} > 0\} \qquad \textit{down} \text{ penalty}$$

$$pu = \min_{j\in\bar{N}} \{(\bar{x}_i - 1)\, \bar{c}_j/\bar{a}_{ij} < 0\} \qquad \textit{up} \text{ penalty.}$$

For the initial node one has the penalty

$$p = \min \{pd, pu\}.$$

Note that dichotomous penalties can also be derived for non-basic 0–1 variables; one then has $pd = 0$, $pu = \bar{c}_j$. This information will be useful for gap branching (see subsection II.2.3).

II.1.2. *Improved bounds*

The most effective bounds require a certain amount of preliminary computations; once this tactical set-up cost is incurred, it becomes advantageous to engage in the computation of further *marginal improvements* of the bound and to exploit all *readily available* information. The resulting increase in tactical efficiency depends upon the form of this information retrieval. The following remarks and example illustrate a generally effective improvement policy.

(1) Bounds generated from penalties which in turn are computed from a valid cut should benefit from *all* integrality requirements; not only during the construction of the cut, but also afterwards, during the penalty computations where *integrality of the non-basic variables* can be used.

(2) When the dual pivot selection rule indicates that an *integer* constrained variable should enter the basic at a *fractional* value, then the *same updated cut row* can be *used again* to form a new cut. Thus a *chain of cuts* can be constructed with few arithmetic operations.

As an illustration of the above principles we now describe penalties (derived from Gomory's mixed cut) which were found more efficient than those proposed by Tomlin [34].

Example 1 (BD 7). If x_i is a fractional basic variable, then the corresponding mixed integer Gomory cut reads

$$\sum_{j \in \bar{N}} (-\pi_j) x_j \leq -1,$$

where π_j is defined in BD 4.

If $j_0 \in N_{zo}$, then x_{j0} has an upper bound of 1; furthermore one has $\pi_{j_0} \leq 1$; hence after one dual step the variable x_{j_0} obtains the value $\tilde{x}_{j_0} = \pi_{j_0}^{-1} \geq 1$; thus if $\pi_{j_0} < 1$, $\tilde{x}_{j_0}$ is > 1 and one may automatically perform another dual step which brings $\tilde{x}_{j_0}$ back to 1.

The following penalty computation reflects these two dual steps.

$$\text{ipg} = \bar{c}_{j_0} + \bar{c}_{j_1}(1 - \pi_{j_0})/\pi_{j_1},$$

where

$$\bar{c}_{j_0}/\pi_{j_0} \leq \bar{c}_{j_1}/\pi_{j_1} \leq \bar{c}_j/\pi_j \quad \text{for all } j \neq j_0, j_1, \pi_j > 0.$$

Example 2 (BD 8). There is yet another way to use the mixed integer cut. Compute the (down) penalty pd_i, i.e., introduce the cut $x_i \leq 0$ which also reads

$$\sum_{j \in \bar{N}} (-\bar{a}_{ij})\, x_j \leq -\bar{x}_i$$

and perform a dual step (with corresponding slack s, x_{j_0} variable entering the basis)

$$\sum_{\substack{j \in \bar{N} \\ j \neq j_0}} \bar{a}_{ij}/\bar{a}_{ij_0}\, x_j - 1/\bar{a}_{ij_0}\, s + x_{j_0} = \bar{x}_i/\bar{a}_{ij_0}, \qquad (*)$$

$$\sum_{\substack{j \in \bar{N} \\ j \neq j_0}} (c_j - \bar{a}_{ij}/\bar{a}_{ij_0}\, \bar{c}_{j_0})\, x_j + \bar{c}_{j_0}/\bar{a}_{j_0} s - x_0 = -\bar{x}_0 - \bar{c}_{j_0}\bar{x}_i/\bar{a}_{ij_0}. \qquad (**)$$

Now, if x_{j_0} is an integer variable, and $\bar{x}_i/\bar{a}_{ij_0}$ is *not* integer, then (*) can be used to compute a mixed integer Gomory cut (note that $s = 0 - x_i$ is an integer variable). The corresponding improved penalty (Example 1) using the updated objective function (**) yields the improved down penalty $\text{ipd} = \text{pd} + \text{ipg}$. A similar treatment holds for the up penalty ipu.

Example 3 (chains; **BD** 6). The idea of *chained cuts* circumvents some of the obstacles encountered by cutting plane algorithms. Most of the cost of creating a cutting plane from different fractional variable lies in the cost of up-dating the corresponding rows; for a *chain* of cuts we simply observe that once a row has been updated it is easily kept in updated form during subsequent pivot steps.

Construct for instance the cut described in BD 4 for a given fractional variable x_{i_0}. Selection of a dual pivot brings another integer variable x_{j_0} into the basis and the updated form of the corresponding row reads

$$\sum_{\substack{j \in \bar{N} \\ j \neq j_0}} (\pi_j/\pi_{j_0})\, x_j - \pi_{j_0}^{-1} s + x_{j_0} = \pi_{j_0}^{-1}.$$

Clearly this row (i.e., the fractional variable x_{j_0}) can be used to generate a new Gomory cut; and so on.

Such chain reactions may be triggered at virtually every tactical bound

or penalty computation with the advantage of avoiding explicit pivots as for penalties.

Further improvements can be deduced from *logical implications* (see subsection II.5). Finally, note that improvement procedures (particularly chains) automatically generate special updated rows (see subsection II.4).

A final comment. As remarked in the introductory part of subsection II.1.1, the purpose of the bound mechanism is to fathom nodes; not arbitrary nodes, but those explicitly identified and inserted into the *front*. There is a trade-off between bound improvement at a given node and branching to its successors; the latter can also be viewed as an improvement with the difference that it increase the size of the arborescence and the front.

In any case, when a node is selected for bound computation, one should keep in mind that the following may occur :

(a) the node is pruned due to its "good" bound,

(b) the algorithm backtracks and *branches.*

A posteriori it is obvious that, no matter how powerful and efficient the bound computation is, branching has to occur at certain nodes. The basis for such strategic considerations concerning bounds is straight forward : compute only those bounds which cause pruning. One should apply *flexible* bound tactics and engage in expensive bound computations only when there is a reasonably good chance that pruning will occur; the difference between the cut-off value γ and the bound β may be used here as a basis for this choice.

II.2. *Branching*

The tactical selection of a branching variable resembles compass navigation in a labyrinth. Confronted with an abominable exponential mess, one has little hope of gaining theoretical insight. This is probably the reason why "branching" has received almost no attention in the literature in comparison with the opulance of the "bound industry". But it is equally important to obtain good values for both γ and β; and γ is a direct product of the branching strategies (see Section III). The next subsections explore tactical results in this area.

II.2.1. *Ordinal branching*

BR 1. *Lexicographic priority.* The implementation of a computer program entails by fiat a natural ordering of its array operations. If the variables

are initially ordered according to some measure of "importance", systematic branching on the one free variable x_j ($j \in N_{\text{free}}$) with smallest index j will automatically take this property into account.

BR 2. *Cost ranking.* This is a variant of BR 1, where the absolute value $|c_j|$ is adopted as measure of importance.

II.2.2. *Wishful branching.*

BR 1 and BR 2 fail to distinguish between basic and nonbasic variables at the LP optimum as well as between *fractional, integer valued* or *fixed* variables. The following rules aim at restoring integral feasibility under the bias of the objective function.

BR 3. *Maximal fractional part.* Select the branching variable x_{i_0} according to

$$|\bar{x}_{i_0} - 0.5| = \min_i |\bar{x}_i - 0.5|,$$

among all fractional variables x_i.

BR 4. *Maximal penalty* [26]. If BD 4 and BD 5 are used to compute penalties then one may select x_{i_0} according to $p_{i_0} \geq p_i$ for all i.

BR 5. *Pseudo-costs* [8]. This branching rule seems to enjoy a successful carreer in large scale mixed-integer programming with a vast number of continuous variables. In the present all integer context, however, it has not quite lived up to our expectations (see Table X7).

Consider a node k with LP-value $x_0^{(k)}$, its branching variable x_i (with a fractional value $\bar{x}_i^{(k)}$), and its successor nodes with corresponding LP-values $\bar{x}_0^{(k+1)}$ and $\bar{x}_0^{(k+2)}$. Define

$$\text{pcd}_i = \frac{x_0^{(k+1)} - x_0^{(k)}}{\bar{x}_i^{(k)}},$$

$$\text{pcu}_i = \frac{\bar{x}_0^{(k+2)} - x_0^{(k)}}{1 - \bar{x}_i^{(k)}}$$

(assuming that node $k+2$ corresponds to $x_i = 1$). pcd and pcu give the *deterioration* of the objective function value subsequent to fixing the variable. It can be computed statistically during the branch and bound algorithm [8].

To select a branching variable at node k, one now chooses

$$\text{pc}_i = \min \{\text{pcu}_i (1 - \bar{x}_i^{(k)}), \text{pcd}_i \bar{x}_i^{(k)}\}$$

and x_{i_0} is selected such that $\text{pc}_{i_0} \geq \text{pc}_i$, for all i.

Note that pseudo-costs provide no bounds.

BR 6. *Feasalties.* Pseudo-costs and penalties work reasonably well. But one feels intuitively that they fail to capture the nature of the problem with respect to *feasibility*, in both the LP and integral sense. They are based on the objective function only. We therefore propose to introduce a measure of feasibility degradation into the choice of a branching variable and this type of tactical rule was found to exhibit surprising clairvoyance. It is also efficient because few arithmetic operations are involved.

Take any zero–one variable x_i and impose tentatively $x_i = 0$; if the current LP becomes infeasible, then *one* dual pivot step is performed on the "most infeasible" row whereby only the new right-hand side $\tilde{x}$ need be computed. Then define the following (down) feasalty

$$\mathrm{fd}_i = \sum_{k \in N_{\mathrm{free}}} (\min \sigma_k \{0, \tilde{x}_k\} + \min \sigma_k \{0, 1 - \tilde{x}_k\}),$$

where the σ_k are given parameters (for instance $\sigma_k = 1$, all k or $\sigma_k = \lceil c_k \rceil$).

The (up) feasalty fu_i is obtained in a similar way upon imposing temporarily $x_i = 1$. Finally, compute the feasalty

$$\mathrm{f}_i = \min \{\mathrm{fd}_i, \mathrm{fu}_i\}$$

and choose the branching variable x_{i_0} according to $\mathrm{f}_{i_0} \geq \mathrm{f}_i$, for all i.

One may feel that the influence of the objective function is unnecessarily eliminated from the above development. This can be corrected by adding to f_i any one of the penalties described earlier. The single dual step dichotomous penalty is often sufficient. More generally we can combine feasalties and penalties in the following convex manner:

$$\lambda\, \mathrm{fd}_i + (1 - \lambda)\, \mathrm{pd}_i, \qquad \lambda \in [0, 1]$$

(and similarly for fu and pu).

The parameter λ may be chosen (at any node) to reflect one or the other tendency in the branching rule; in particular, the node may be dual degenerate and all penalties vanish. Thus the branching choice will be based on feasalties alone. On the other hand, a single dual step may restore primal feasibility and feasalties will vanish. The choice of a branching variable is then left to the penalties. These two extreme cases correspond to $\lambda = 1$ resp. 0. One easily imagines criteria based on measures for dual degeneracy and primal infeasibility to select intermediate values for λ. Note also that feasalties can be derived for both basic and non-basic variables.

Of course, the concept of feasalty may also be used in a statistical manner as in the case of pseudo-cost branching.

Thus pseudo-feasalties are defined as :

$$\mathrm{pfd}_i = \mathrm{fd}_i/\bar{x}_i, \qquad \mathrm{pfu}_i = \mathrm{fu}_i/(1 - \bar{x}_i),$$

and one will choose the branching variable according to the convex combination

$$\lambda\, \mathrm{pfd}_i + (1 - \lambda)\, \mathrm{pcd}_i$$

(similarly for pfu and pcu).

Convex combinations of the type presented in this section should be kept in proper perspective. One should not expect minor modifications of the λ coefficients to have a dramatic effect on any given problem. The primary motivation for such devices is their low cost. They involve very little additional overhead. It furthermore gives one the opportunity to let experimentation choose heuristically its own approach since theory fails to provide adequate support.

II.2.3. *Gap branching*

The branching rules of the previous subsection are all based on some quantified measure: penalty, feasalty, etc. The general philosophy for branching tactics is simply to hope that the optimal solution lies on the branch which currently seems most "promising". This optimism is not always rewarded: backtracking could often be avoided if an element of "safety" were introduced into tactical branch selections. *Gap branching* is an attempt in this direction. The absolute magnitude of the penalty and a *relative* measure are both taken into account by the difference

$$\Delta p_i = |\mathrm{pu}_i - \mathrm{pd}_i|.$$

This also requires but few computational modifications to the standard choice of the branching variable.

BR 7. *Gap penalties.* Denote by pd_i and pu_i any of the (down, resp. up) penalties, pseudo-costs or (pseudo-) feasalties described in II.2.2.; and set $p_i = \min\{\mathrm{pd}_i, \mathrm{pu}_i\}$. Define

$$g_i = \lambda\, p_i + (1 - \lambda)\, \Delta p_i, \qquad \lambda \in [0, 1].$$

The branching variable x_{i_0} is then chosen according to $g_{i_0} \geq g_i$ for all i.

One has the following special cases :

$\lambda = 1$, a pure BR 4-type rule.

$\lambda = 0$, a pure gap selection rule.

$\lambda = \frac{1}{2}$, where g_i reduces to

$$g_i = \tfrac{1}{2}\min\{pu_i, pd_i\} + \tfrac{1}{2}|pu_i - pd_i| = \tfrac{1}{2}\max\{pu_i, pd_i\}.$$

The purpose of gap penalties is to branch on variables which are "safe", one of the two branches being *clearly "better" than the other*; thus there is little chance that one will have to backtrack to that node.

Finally, a general expression can be formed to include penalties, feasalties, pseudo-quantities and the various related gaps; this simply requires several λ control parameters.

II.3. *Miscellaneous devices*

II.3.1. *Cut-off values*

The concept of *cut-off value* is very useful to accelerate the algorithm and to control storage requirements. The following list is self-explanatory:

CV 1. *Current best value.* As integer feasible solutions are discovered, γ is systematically updated to represent the best objective function value (current optimum). If p solutions to the integer program are desired, γ is is simply chosen to be the p-best value among all feasible solutions at any given stage (or $\gamma = +\infty$ if the algorithm has not yet identified p feasible solutions).

CV 2. *All-integer cut-offs.* In the all-integer case, the above γ value can be improved as follows: New solutions must satisfy

$$x_0 = \sum_{j \in N_{zo}} c_j x_j < \gamma;$$

hence, upon division by $\delta > 0$, one obtains

$$\sum_{j \in N_{zo}} (c_j/\delta)\, x_j < \gamma/\delta;$$

now if δ is such that (c_j/δ) is integral for all $j \in N_{zo}$
the left-hand side must be integer, i.e.,

$$\sum_{j \in N_{zo}} (c_j/\delta)\, x_j \leq \gamma/\delta - 1$$

(the quantity $\gamma/\delta = \sum_{j \in N_{zo}} (c_j/\delta)\, x'_j$ is integral because $\gamma = \sum_{j \in N_{zo}} c_j x'_j$ for some integer solution x'). Finally, multiplying by δ one has

$$x_0 = \sum_{j \in N_{zo}} c_j x_j \leq \gamma - \delta = \tilde{\gamma} \qquad (\delta > 0)$$

i.e., $\tilde{\gamma}$ is an improved cut-off value.

CV 3. *Near optimality.* In some cases it is not necessary to determine the optimum of the given integer program but merely to detect a good integer feasible solution, garanteed to lie within a certain range $\rho > 0$ of the optimum. The value of ρ may be defined in absolute or percentage terms to yield the following cut-off values:

$$\hat{\gamma} = \gamma - \rho \qquad (\text{or } \hat{\gamma} = \tilde{\gamma} - \rho).$$

CV 4. *All (sub)optimal solutions in a given range* ρ. This case is the opposite of CV 3, and the cut-off value is *increased.* Suboptimal solutions within the given range $\rho > 0$ are stored:

$$\hat{\gamma} = \gamma + \rho \qquad (\text{or } \hat{\gamma} = \tilde{\gamma} + \rho).$$

II.3.2. *Rounding*

As the algorithm proceeds, there are many opportunities to "hit" a feasible solution accidentally. For instance when the LP optimal solution of a node can be rounded to an integer feasible solution. This involves only few additions and may sometimes curtail the search by providing an unexpectedly good cut-off value.

Rounding also interacts advantageously with the logical implications of subsection II.5.

(a) The list of implications will guide rounding and speed up the local search process.

(b) Rounding is a type of partial enumeration at the node, i.e., within a facet of the unit hypercube defined by the fractional variables. If that facet has no feasible solution, a new logical implication has been found.

(c) Rounding and logicals operate in close contact with one another: both procedures are frequently used and involve a great many *simple* operations. For both, the philosophy is to engage in such operations to a limited degree only; just enough to take advantage of a small probability of success. Any massive rounding search could be catastrophical because it neglects other sources of information.

II.3.3. *"Trouble-shooting"*

The bulk of computations occurs during LP operations. Branching and LP characteristics of integer programs cause variables to repeatedly jump in and out of the basis; and this disagreable phenomenon creates superfluously long Eta files, frequent reinversions, inaccuracies... This can be detected by performance statistics and identified during the execution. The remedy is to eliminate the *trouble-maker* (variable) with an ad hoc

dichotomy: if it is a structural zero–one variable, simply branch on it; and otherwise keep it *in* the basis on one branch, and *out* on the other.

There are further occasions where trouble-shooting is recommended: dual degeneracy and small updated costs can be eliminated by simultaneous branching on the degenerate variables; a variable which frequently appears in logicals is a prime candidate for branching; fixing it will reduce the degree and strengthen the influence of all implications containing that variable.

In fact, the idea of *trouble-shooting is quite* general and need not be based on rational arguments. A set of *control statistics* will guard against any abnormal behavior by triggering appropriate counter-measures. Consider the order, for instance, in which variables are processed by branching and logical subroutines. Because it is column-oriented, the LP set-up allows some flexibility in this respect; it is advantageous to take those variables first which are most "relevant". This is true of all operations where chronology has an influence, as in the construction of logical implications.

Many such decisions are based on "hunches" and empirical evidence. Their track record can be evaluated statistically and *correction factors* determined for each problem.

II.4. *Special updated rows*

II.4.1.

Most of the implications used in zero–one programming are derived from *linear* expression with *integrality* conditions. The continuous LP structure and the discrete nature of the problem have little in common; and *equivalent* representations of the linear expression may lead to genuinely different implications.

It is therefore advantageous to take every opportunity of performing this type of computation (see subsection II.5). Of course there is a limit to such recommendations due to the high price of the updating process. The information which could be extracted from an updated row does not seem to compensate for the prohibitive cost of updating. Thus we propose to use special updated rows to amplify the potential advantages of this idea without expensive updating computations. There are two kinds of special updated rows.

(i) Rows which must be updated anyway during the flow of computations: *objective function*, *pivot rows*, and eventually also *fractional rows* and *chains* (subsection II.1.2).

(ii) Linear expressions which are constructed ad hoc and updated for this purpose.

Special updated rows of type (i) are of course chosen for the simple reason that they involve no *set-up cost*.

II.4.2. *Explicit row combinations*

In some occasions the variables of an integer (sub)program are found to interact in a particular manner which can be expressed by a linear expression

$$\sigma(x) = \sum_i \alpha_i x_i.$$

Since a linear combination of several rows can be updated in one single backward transformation, relatively few arithmetic operations are involved.

(1) *Canonical expressions* ($\alpha_i = 0, \pm 1$). linear relations of this form arise from compounded logical implications and may be used for simultaneous branching, for instance.

(2) *Structural expressions.* If the initial LP has some structure (blocks), one can be use partial rows: $\alpha_i = a_{ki}$ for some or all i in the row k). This structure may have a strong influence on solutions.

(3) *Fractional expressions* ($\alpha_i = \bar{x}_i^{-1}$). It may be desirable to isolate "almost integer" fractional variables for trouble-shooting purposes; one may then "lump" them together in a special updated row.

II.4.3. *Branching on a special updated row*

Special updated rows introduce new branching variants. Consider two values σ^+ and σ^- satisfying

(a) $\sigma^- < \sigma^+$,

(b) $\sum_{i \in I^-} \alpha_i \leq \sigma^-$; $\sum_{i \in I^+} \sigma_i \geq \sigma^+$,

where

$$I^- = \{i : \alpha_i < 0\}, \qquad I^+ = \{i : \alpha_i > 0\}.$$

One then defines the dichotomy:

$$\sigma(x) \leq \sigma^-, \qquad \sigma(x) \geq \sigma^+.$$

If one chooses $\sigma^- = \sum_{i \in I^-} \alpha_i$, for example, and $\sigma^+ = \sigma^- + \min_i |\alpha_i|$, the dichotomy corresponds to a simultaneous branching step because the first inequality ($\sigma(x) \leq \sigma^-$) becomes an equality which is only satisfied by $x_i = 1$ for all $i \in I^-$, $\bar{x}_i = 0$ for all $i \in I^+$; the second inequality ($\sigma(x) \geq \sigma^+$) in turn is valid for all other solutions.

Apart from branching, special updated rows are extremely useful for the derivation of logical implications because they furnish many new implications (without set-up cost).

Finally, if the coefficients α_i are integral, one may always generate a Gomory cut (and chains) from the special updated row.

II.5. *Logical implications*

A rudimentary Branch and Bound scheme may require an astronomical number of nodes. *Fathoming* tests are therefore necessary. A large number of infeasible or suboptimal zero–one solutions can be eliminated by checking the rows of the *initial (sub) tableaux* and/or *special updated rows*. This results in a list of logical implications which are stored to be available at no significant retrieval cost.

Logical implications have been studied, in a rather general context, by several authors under the name fathomings tests, filters, preferred inequalities, etc. [2, 16, 21, 32]. Our experience, however, has been that only trivial implications increase efficiency. The derivation of "sophisticated" logical relationships tends to require a prohibitive number of operations. We propose instead a *cascading* approach to generate only those relations (of higher complexity) which can be derived economically.

In every tactical branching or bounding operation the logical machinery proves reliable, and consistently increases the overall efficiency of the algorithm (see Table X.3).

II.5.1. *Elementary implications*

The complexity of a logical implication can be measured by the number of variables it actively involves; it is called *degree* [32].

Degree zero (LP feasibility). Group the coefficients of each linear relation $\sum_{j\in N} a_{ij} x_j = b_i$ for all i, according to their sign, and define:

$$J_i^+ = \{j \in N\colon a_{ij} > 0\}, \qquad J_i^- = \{j \in N\colon a_{ij} < 0\}.$$

with

$$u_j = \begin{cases} 1 & \text{if } j \in N_{zo}, \\ \text{an upper bound for the variable } x_j & \text{if } j \notin N_{zo}, \\ +\infty & \text{if } x_j \text{ has no known upper bound.} \end{cases}$$

The following expressions characterize the range of the artificial variable

$$y_i = b_i - \sum_{j \in N} a_{ij} x_j,$$

$$b_i - \sum_{j \in J_i^+} a_{ij} u_j = r_i^- \leq y_i \leq r_i^+ = b_i - \sum_{j \in J_i^-} a_{ij} u_j. \qquad (*)$$

For every solution of the IP the artificial variables y_i must all vanish. Thus the subproblem has no (integer) feasible solution if $r_i^+ < 0$ or $r_i^- > 0$ for some i.

Degree one. Choose a variable $x_j, j \in N_{zo}$; fix x_j to 0 (or 1) and investigate the remaining terms of the above relation (*).

One sees that the subproblem has no (integer) feasible solution with

$$x_j = \begin{cases} 0 & \text{if } r_{i0}^+ < 0 \text{ or } r_{i0}^- > 0, \\ 1 & \text{if } r_{i1}^+ < 0 \text{ or } r_{i1}^- > 0. \end{cases}$$

Such conditional statements are derived when the variable x_j is proposed for branching.

Higher degrees. In general, any combination of variables $x_j, j \in N_{zo}$ can be fixed. Assume

$$J_0 \subseteq N_{zo}, \qquad J_1 \subseteq N_{zo}, \qquad \text{with } J_0 \cap J_1 = \emptyset,$$

and define

$$r_i^+ = b_i - \sum_{J_1} a_{ij} - \sum_{J_i^- - J_1 - J_0} a_{ij} u_j,$$

$$r_i^- = b_i - \sum_{J_1} a_{ij} - \sum_{J_i^+ - J_1 - J_0} a_{ij} u_j.$$

If $r_i^+ < 0$ or $r_i^- > 0$, one obtains

$$\sum_{J_0} x_j + \sum_{J_1} x_{-j} \geq 1,$$

where x_{-j} denotes $1 - x_j$.

A systematic analysis of all such higher order relations is impractical because of the large number of cases $(\binom{n}{2}, \binom{n}{3}, \ldots, \binom{n}{k})$ which must be explicitly considered.[1]

II.5.2. *Compounding.*

Elementary implications are stored along with the tree structure. One has the following observations.

[1] In a private communication, K. Spielberg pointed out that this situation is not so disadvantageous in mixed linear programming with many continuous variables and that higher degree relations may be expected to perform better for such problems.

- Any valid implication at a given node is also valid for *all* the *successors* of that node.
- Any valid implication at a given node (N_{zero}, N_{one}, N_{free}) yields an expansion which is valid throughout:

$$\sum_{N_{zero}} x_j + \sum_{N_{one}} x_{-j} + \sum_{J_0} x_j + \sum_{J_1} x_{-j} \geq 1.$$

Such relations are used to identify infeasible nodes without LP computations. Known implications will speed up the search for new implications; thus chronology is not immaterial and generates a *compounding* effect. This particular manner of automatically selecting implications whose higher degree is determined by the flow of operations is very efficacious.

II.5.3. *Reduction and cascades*

Large sets of logical implications can be reduced to equivalent and simpler systems [21, 32]. The aim is to derive implications of degree zero or one (eventually two). However, our experience shows that the amount of work required by a reduction destroys the resulting pay-off. Only two special cases (degree two and *cascades*) seem profitable in 0–1 programming.

Lemma (reduction). *If* $\sum_J x_j \geq 1$ *and* $\sum_K x_j \geq 1$ (*where* $J = J_0 \cup \{-j: j \in J_1\}$, *and similarly for* K) *are both valid* (*at a given node*) *and if there exists exactly one index* $|l|$ *such that* $l \in J$ *and* $-l \in K$, *then*

$$\sum_{(J-\{l\})\cup(K-\{-l\})} x_j \geq 1$$

is also valid.

Proof. The proof is immediate by contradiction.

The lemma may be used as follows to produce simpler implications.

Example 1 (degree 2). If $x_j + x_i \geq 1$ and $x_j + x_{-i} \geq 1$ are both valid, then $x_j \geq 1$ must hold.

Example 2 (Cascades). If $\sum_P x_j + \sum_L x_j \geq 1$ ($\pm j_0 \notin P$, $j_0 \in L$) and $x_{j_0} + x_{-j} \geq 1$ for all $j \in P$ are valid implications, then $x_{j_0} + \sum_L x_j \geq 1$ is also valid.

This can be established by repeated application of the reduction lemma. Note that if the rest term $\sum_L x_j$ is of degree zero or one, then any implication of degree $|P| + |L|$ can be reduced to a cascade of degree one or two. A similar statement holds when $j_0 \in L$.

In these two examples reductions are obtained at virtually no additional cost. The necessary sweep through existing logical implications of degree ≤ 2 involves only few boolean operations.

II.5.4. *Computational aspects*

The first logical implications are generated from the rows of the *initial tableau.* Further implications are derived from the subtableau at each node, and from special updated rows as the algorithm proceeds.

The *objective function* (cut-off inequalities), *pivot rows* and *chains* are also used consistently for this purpose. However the net savings that result from logical implications depend less on their intrinsic logical strength than on the efficiency of the implemented computational procedure.

For most problems, known logical implications are used as often as possible in fathoming tests, penalty and bound calculations. They are stored in compact form (bit-wise) to minimize retrieval costs.

II.5.5. *Penalty improvement*

For penalty or bound computations, logicals tend to involve too many operations to yield a significant pay-off. In special situations, however, a *cumulative* effect may act favorably. Consider the following examples to illustrate this idea.

The logical implication

$$\sum_{j \in J_0} x_j + \sum_{j \in J_1} x_{-j} \geq 1 \qquad (*)$$

implies that at least one x_j, $j \in J_0$ or x_{-j}, $j \in J_1$, must take value 1. The penalties pd_j, pu_j (for all $j \in J_0 \cup J_1$) combine with (*) to yield the following valid penalty

$$\mathrm{p} = \min \{ \min_{j \in J_0} \mathrm{pu}_j, \min_{j \in J_1} \mathrm{pd}_j \}. \qquad (**)$$

Furthermore, when several logical implications $(J_0^k, J_1^k : k = 1, 2, \ldots)$ are available such that

(i) the index sets J_0^k, J_1^k, are all disjoint (an index belongs at most to one set),

(ii) J_0^k and J_1^k (for all k) contain no current basic index,

penalties can be *cumulated*:

$$\mathrm{p} = \sum_k p^k,$$

where p^k is defined by (**). This cumulative effect increases the efficiency of penalty computations.

Example 1 (See BD 5 in Section II). Improved bounding rule BD 9. Consider an implication with $J_1^k = \emptyset$ and $J_0^k \subset \bar{N}$ = current non-basic set. Then one has the dichotomous penalties

$$pd_j = 0, \qquad pu_j = \bar{c}_j \qquad p^k = \min_{j \in J_0^k} \bar{c}_j.$$

Example 2 Improved bounding rule BD 10. Usually penalties are computed for basic variables from a cut (BD 5 or BD 6), written in the form

$$\sum_{j \in \bar{N}} (-\pi_j)\, x_j \leq -1.$$

Let the variable x_{j_0} enter the basis at the first dual step, and assume $j_0 \notin J_0^k$ for all k. Compute the penalty $\tilde{p} = \bar{c}_{j_0} / \pi_{j_0}$ from the cut.

As in example 1, one now obtains

$$p = \tilde{p} + \sum_k p^k,$$

with

$$p^k = \min_{j \in J_0^k} \{\bar{c}_j - (\pi_j / \pi_{j_0})\, \bar{c}_{j_0}\}.$$

With dichotomous cuts, one will improve the up (resp. down) penalties in this manner.

Finally, logical implications can also influence LP optimization. They represent GUB rows which need not be explicitly implemented in the tableaux; the results presented in the Appendix, however, make no use of this property.

II.6 *Surrogate constraints (theory and practice)*

Several years ago, Glover [18] and Balas [2] proposed the use of linear row combinations called surrogate constraints. In his implicit enumeration algorithm, Geoffrion [16] found surrogates to be efficient. The basic idea is to use a linear expression

$$c\,x + y\,(b - A\,x) + s = \gamma \qquad (\gamma = \text{cut off}).$$

In Geoffrion's case, the y_i coefficients are LP-optimal *dual* variables and the resulting expression becomes the *updated objective function* as shown below.

$$y = c_B B^{-1} \qquad (B = \text{Basis of the LP optimal tableau})$$

yielding

$$c x + c_B B^{-1}(b - A x) + s = \gamma, \qquad s \geq 0$$

which can be written as

$$c_B x_B - c_B B^{-1} B x_B + c_N x_N - c_B B^{-1} N x_N + c_B B^{-1} b + s = \gamma$$

or

$$\bar{c} x + s = \gamma - c_B B^{-1} b = \gamma - \bar{x}_0.$$

But $s \geq 0$ implies

$$\sum_{j \in \bar{N}} \bar{c}_j x_j \leq \gamma - \bar{x}_0$$

or

$$x_0 = \sum_{j \in \bar{N}} \bar{c}_j x_j + \bar{x}_0 \leq \gamma.$$

Logical implications can now immediately be derived from this inequality:

Degree 0. If $\gamma < \bar{x}_0$, then the node cannot generate desirable feasible solutions.

Degree 1. If $\bar{c}_j > \gamma - \bar{x}_0$, then $x_j = 0$ must hold for all successors.

Degree 2. If $c_j + \bar{c}_k > \gamma - \bar{x}_0$, then $(1 - x_j) + (1 - x_k) \geq 1$ holds for all successors.

These tests for logical implications can be performed at *every* step of a dual LP optimization since the objective function is always dual feasible. The same holds for penalty calculations (improved or *chains* of section II.1).

Example. Construct a (mixed-integer) cut

$$\sum_{j \in \bar{N}} (-\pi_j) x_j \leq -1$$

to yield, after a dual step, the following objective function cut

$$\sum_{\substack{j \in \bar{N} \\ j \neq j0}} (\bar{c}_j - \bar{c}_{j0}) \frac{\pi_j}{\pi_{j0}} x_j + \frac{\bar{c}_{j0}}{\pi_{j0}} s \leq -\bar{x}_0 + \gamma - \frac{\bar{c}_{i0}}{\pi_{j0}}.$$

The logical tests now read :

Degree 0. If $\gamma < \bar{x}_0 + \bar{c}_{j_0}/\pi_{j_0}$, then there is no desirable feasible solution

Degree 1. If $\bar{c}_j > \gamma - \bar{x}_0 - (1 - \pi_j)\,\bar{c}_{j_0}/\pi_{j_0}$, then $x_j = 0$ must hold.

These tests are now stronger because

$$\bar{c}_{j_0} > 0, \qquad \pi_{j_0} > 0, \qquad \pi_j \leq 1.$$

III. Strategies

Strategies are general guidelines which govern the overall flow of operations. They influence the node selection and other *tactical computations.* The efficiency of a strategy cannot be captured by a mathematical argument; the influence of a strategic choice depends heavily upon implementation details and/or the structure of the tree.

Naturally, one may study *in abstracto* the properties of a given strategy, but such efforts cannot characterize the behaviour of a practical implementation without taking technical details into account : relative speed of computers' elementary operations, storage capacity limits, information transfer speed and/or *tactical* aspects (such as relative efficiency of a branch forward versus backtracking step) of different branching and bounding computations, etc.

III.1. *Basic node selection rules*

At each branching step, the algorithm has to choose one among a set of pending nodes, called the front.

NS 1. *Depth first.* Priority is given to the immediate successor nodes, provided one, at least, is pending. The aim is obviously to identify quickly an integer feasible solution. The real pay-off depends upon the efficacy of the resulting cut-off tests. This strategy is compatible with LP operations because subprograms are always readily available in updated form from the preceding node.

NS 2. *Breadth first.* The node of the front with the *best bound* is automatically chosen for branching. If bounds are reliable predictors, this rule will find good solutions rapidly.

NS 3. *Alternate depth-breadth.* The strategies NS 1 and NS 2 may be engaged alternately according to a predetermined pattern, for instance

"depth first", initially, but "breadth first" as soon as backtracking becomes necessary.

NS 4. *Last-in-first-out* (LIFO) [34]. This is essentially NS 1; but backtracking is made to the node most recently introduced into the front.

NS 5. *Mixed depth-breadth* [8]. Let β^h denote the better bound of the two immediate successors of the node h, β_b the best bound in the front, and q a given positive real. Then

$$\text{if } \beta^h > \beta_b + q, \text{ select the } \beta_b \text{ node,}$$

$$\text{if } \beta^h \leq \beta_b + q, \text{ select the } \beta^h \text{ node.}$$

This rule imposes a depth tendency at the beginning of the algorithm; then it turns into a breadt strategy. When the value β_b is good, almost all new nodes are dominated by an amount $\geq q$; and the node b is preferred.

NS 6. *Front control.* This flexible rule will influence the size of the front in function of the parameter $\alpha \in [0, 1]$.

Let β_w = worst bound (of the front) below cut-off value. Then

$$\text{if } \beta^h > \alpha\,\beta_w + (1 - \alpha)\,\beta_b, \text{ select the } \beta_b \text{ node,}$$

$$\text{if } \beta^h \leq \alpha\,\beta_w + (1 - \alpha)\,\beta_b, \text{ select the } \beta^h \text{ node}$$

if no successor is pending, take the β_b node.

This strategy corresponds to NS 3 for $\alpha = 1$ and to NS 2 for $\alpha = 0$. For $\alpha \cong 0.8$, one has a breadth strategy as long as the front remains small (at the beginning, for instance). It becomes a depth strategy as the front becomes larger. This type of strategy controls storage requirements in an adequate self-regulating manner.

NS 7. *Single and simultaneous branching.* The flow of LP iterations suggests that significant savings can be achieved if *only one of the successor nodes* is formed. This is called *single branching* and the algorithm follows a straight downward path. The aim is to find a good solution; a good cut-off value γ may then cause most "open" branches to be pruned. Single branching requires gap branching tactics to reduce the frequency of backtracking.

Simultaneous branching is a tactical variation of this approach; several successive variables are fixed to a given value *at the same node*. A penalty for the other branch can be computed from special updated rows of type (ii).

There is little theoretical difference between simultaneous and repeated single branching. The difference lies in the organization of computer operations. The amount of work performed on "open" branches is another

factor influencing the efficacy of the algorithm. A single branching step differs from double branching in that all LP operations are omitted for expediency (see Fig. 1a,b). Simultaneous branching groups several single branching steps into one and combines several "open" branches into one. In fact, at the practical level of a coded implementation, there exists an entire spectrum of options : simultaneous, single and double branching are but three particular strategic choices in this manifold. Our experiments favored the stepwise approach of single branching; probably because single branching determines a set of bounds and penalties which guides effectively the choice of branches, while simultaneous branching has to rely more heavily on an initial hunch.

But single branching is not able to efficiently cope with massive backtracking; conservative versions of double branching are better equipped for this situation. They store much information as it becomes available.

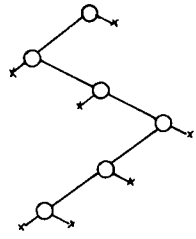

Fig. 1a. Single branching.

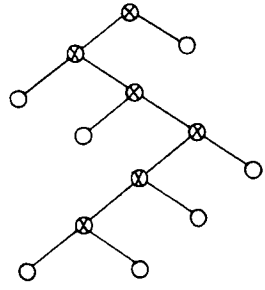

Fig. 1b. Double branching.

○ pending nodes which are computed (LP opt.) and stored in the front
× "open" branches to nodes which are identified but not computed
⊗ branched nodes which are computed but not stored

III.2. *Adaptive and learning strategies*

The behaviour and effectiveness of a strategy can be analysed by *a posteriori* observation of the tree structures; it reveals the actual flow of operations generated by the algorithm. The arborescence exhibits the ability of a particular combination of tactical and strategic rules to extract the relevant information from the problem.

Assume that the nodes of the tree in a given problem possess common characteristics. Then the experience gathered by the algorithm as it goes through earlier branches may indicate how an initially chosen strategy

should be modified. For such *adaptive* strategies one has to carry a number of statistical appraisals of the flow of operations.

Similarly, operation control parameters will record the life time experience of the computer code. These control parameters are constantly tuned for each class of problems by appropriate statistics. Thus the computer code *learns* as it solves problems.

Statistical devices of this type are used to choose a numerical value of several "free" parameters which are best determined heuristically: the α parameter of NS 6, λ in BR 7, special updated row parameters, reinversion, and other numerical and operational parameters, etc.

There are three distinct levels of decisions:

(1) *Tactical* decisions (node dependent).

(2) *Strategic* decisions and adaptive measures (branch and subtree or front dependent).

(3) Structural decisions (problem class dependent).

In our general branch and bound approach the *same* code is capable of handling problems of fundamentally different structure: mixed or pure zero–one, dense or sparse, etc.

This flexibility is maintained at all levels of our implementation and regulated by strategic parameters.

IV. Numerical experiments

As observed at various points in this report, numerical results in integer programming should be examined with caution because of many elements of "subjectivity"; test problem set, computer implementation, accuracy and reliability, computer characteristics (machine, compiler) etc.; all are vital factors which make intrinsic evaluation and comparison difficult.

Nevertheless, by running a code through a multitude of problems, one ultimately gains a feeling for its performance characteristics. In this paper, we try to convey such messages by emphasizing those aspects which highlight our experience in the general Branch and Bound framework.

Naturally most readers will also want to see "hard facts" and for this purpose we present a series of tables in Appendix X which are derived from a series of numerical experiments. The primary purpose of these tables is to illustrate *tendencies, not polished performance.*

But one still wants a quantitative measure which allows in some sense a comparison with other methods (particularly implicit enumeration, with imbedded LP); this type of experiment would be very interesting but it

cannot be based simply on comparing independent experiments (different machines). To be serious, such a study must make certain that comparisons are based on similar quantities (particularly with respect to accuracy and reliability as mentionned in subsection 1.3).

The tables in Appendix X are chosen to illustrate the highlights of our study; due to the complex nature of our code where dichotomies and LP operations interact in many different ways, it is difficult to quantify the individual effect of a particular choice.

Our intent here is to present *numerical observations* rather than results. We identify a few technically relevant options; this provides a basis for the difficult study of problem characteristics and the corresponding (adaptive) strategic decisions. Throughout this report and in the comments of Appendix X, we indicate how a given device affects the algorithm. Our current research with adaptive strategies seeks to determine "when this effect seems desirable"; for it is only the combination of the two which can produce good *results*.

IV.1. *Improvements*

A first set of tables (X.1, X.3, X.4, X.6, X.9) shows whether marginal improvements are worthwhile or not.

Penalty improvement based on readily available information is advantageous. The same holds for logical implications.

The tables indicate the individual improvement trends. A larger increase in efficiency is possible with composite improvement procedures.

IV.2. *Conceptual differences*

Tables X.2, X.5, X.7 and X.8 illustrate basically different approaches within the context of our Branch and Bound code.

IV.3. *Test problems* (see also Appendix Y)

Finally, in the Tables X.11, X.12 and X.13, one finds typical run statistics. They illustrate the power of our approach.

In Table X.11 particularly, one observes that our performance on small and "easy" problems (under 1 minute CPU) is relatively poor; this is not surprising since we deploy an entire arsenal to kill a fly. On the other hand, our code is the first to solve L–S D 2 and E.

The strategy in Table X.11 is a good and reliable one with adequate

overall performance; it does not yield the best time for every problem but rather shows how well a fixed strategy works. It also provides us with a good basis of reference in our research with adaptive and learning strategies.

IV.4. *Epilogue*

The practitioner should not be discouraged by the fact that 50 by 50 problems (P 22) can be difficult; this is a mathematical reality which has in fact little to do with the practicability of a method.

Appendix X. Numerical results

X.0. *Notations*

M	Number of rows (constraints).
N	Number of variables (slacks and artificials not included).
Z_{opt}	Optimal value of the objective function.
IMP	Number of improvements (incumbent suboptimal solutions); number of solutions which were found to deliver the best objective value at some point during the execution.
OPT AT NODE	Node at which optimal solution is found (value of the *nodes* counter at that time which indicates the number of nodes which had to be created before the optimum was discovered).
NODES	Total number of branched nodes during the course of calculations. More precisely, when a node has to be chosen in the front according to a node selection rule, this counter is increased by 1.

Note that this accounting method differs significantly from the following less stringent one used by some authors:

Number of nodes which were branched to produce *both* successor nodes as pending nodes in the front.

This differs from *nodes* in the following cases:

- If one or two of the successor nodes are fathomed immediately (without ever belonging to the front), then the branched node is counted as *node* but *not* in the latter accounting.
- During a single branching phase, backtracking to the second branch of a node in the front causes the counter *nodes* to be augmented while this would not be registered in the other method.
- Explicitely identified feasible solutions are also counted as *nodes* but not in the other case. Our choice of the *nodes* counter is motivated by a desire to obtain representative measurement for the bulk of tactical operations.

ISEC	CPU solution time in seconds (rounded to nearest seconds; Univac 1108, Exec 2). *Isec* does not include

	input/output, but includes *all* preprocessing, such as rearrangement, scaling, initialization.
MAXFR	Maximal number of nodes in the front. This measures the maximum amount of storage required by the tree organization.
DEGREE 0	Number of logical implications of degree 0 (logical infeasibility) encountered by the algorithm.
DEGREE 1	Number of logical implications of degree 1 (preferred variable).
DEGREE 2	Number of logical implications of degree 2.
CASC	Number of logical cascades (logical implications of degree > 2 which are reducible to a degree ≤ 2).
LPITER	Number of LP iterations (pivot step).
PU	Up dichotomous penalty.
PD	Down dichotomous penalty.
PG	Gomory–Tomlin penalty, improved by using integrality requirements of nonbasics (see BD 7 in subsection II.1.2).
IPU, IPD	Improved up or down penalty (elementary chain, see BD 8 in subsection II.1.2).

(For a definition of the terms *node*, *pending*, *fathomed*, *branched* or *front* see subsection I.1).

X.1. *Further conventions*

D1	Use of logical tests of degree ≤ 1.
D2C	Similarly for degree ≤ 2, including cascades.
RD	Rounding.
DUAL	LP optimization according to dual pivot selection.
PRIMAL	(also when not specified): Optimization with primal pivot selection rule (and Phase I).
BD	Bound procedure.
BR	Branching rule.
CV	Cut-off value.

Table X.1
Penalty statistics (NS 6 ($\alpha = 0.8$); BR 4; D 1; CV 2)

	A		B				C	
	α_d	α_u	β_d	γ_d	β_u	γ_u	∂	Σ
B-M 24	1.029	1.381	0.661	1.311	33.417	34.555	15	1.015
25	1.029	1.155	0.630	1.238	33.396	34.369	13	1.011
L-S B	1.019	1.017	0.186	1.333	172.613	174.596	04	1.005
C	1.005	5.596	0.152	1.114	87.851	88.077	22	0.998
D2	1.046	1.556	0.264	0.898	17.404	18.214	03	1.010
IBM 5	1.349	1.049	0.315	1.690	15.627	16.091	01	0.941
6	1.537	1.511	0.656	1.741	3.966	5.312	01	1.079
9	1.686	1.720	0.443	1.140	1.467	2.114	27	0.951
P 6	1.514	1.494	0.585	1.204	2.612	3.624	01	1.188
7	2.053	1.322	0.355	0.985	2.773	3.789	01	1.159

Tables A and B compare improved penalties to dichotomous penalties on the one hand and Gomory penalties on the other. In Table C, it is apparent that penalties involve a small percentage of the objective function.

In A, the quantities α_d = ipd/pd and α_u = ipu/pu represent the mean improvement ratio suggested in Example 2 of subsection II.1.2.

In B, the quantities β_d = min {pd, pu}/pg and β_u = max {pd, pu}/pg compare Gomory–Tomlin penalties with dichotomous penalties. Similarly for the improved penalties with γ_d = min {ipd, ipu}/pg and γ_u = max {ipd, ipu}/pg.

In C, the quantity $\partial = (pg/|\bar{x}_o|) \times 10^3$ indicates the relative importance (quantitavely) of Gomory–Tomlin penalties ($\bar{x}_o$ = objective function value at LP optimum). Finally, Σ = |ipd-ipu|/|pd-pu| shows how gaps are modified on the average by this improvement.

Table X.1 (Cont.)
Influence of penalty improvement (NS 6 ($\alpha = 0.8$); BR 4; D 1; CV 2)

	ISEC		NODES		LPITER		MAXFR		DEGREE 0		DEGREE 1	
B-M 24	125	145	133	148	4553	5017	32	52	41	44	552	608
25	375	278	355	270	13759	9485	130	70	72	73	1647	1060
L-S B	6	5	16	9	274	189	9	9	2	0	85	2
C	1	6	9	18	92	212	6	11	0	4	77	179
D2	281	215	375	221	10122	6473	103	60	53	26	3053	2044
IBM 5	60	23	181	56	2982	863	22	38	0	0	66	4
6	92	88	39	32	2679	2353	26	21	0	0	30	0
9	41	46	50	55	1417	1610	19	27	1	0	74	80
p 6	9	7	27	13	581	344	18	12	2	2	129	8
7	15	19	39	35	919	914	22	30	2	0	122	75

First columns : Gomory–Tomlin penalties (BD 4). Second columns: improved (BD 8).

In this table we see that improved penalties do not merely produce better bounds but also influence the tree search and branching process. In most cases, improved penalties are recommended. This is particularly true of more elaborate improvements e.g. BD 9 and BD 10. These rules also involve logical implications, however, so that it is more difficult to evaluate the intrinsic effect of penalty improvement. This is better indicated by the above comparison BD 4 versus BD 8.

Table X.2
Primal versus dual LP optimization (NS 1 + NS 7 (single); BR 7; D 1; RD; CV 2; BD 8)

		ISEC		NODES		LPITER	
		P	D	P	D	P	D
B–M	24	54	48	163	163	2061	1909
	25	105	95	384	381	4133	4045
L–S	B	1	2	2	2	71	79
	C	6	7	76	76	345	391
	D2	132	200	291	352	3729	7459
IBM	5	34	73	189	334	1573	3335
	6	37	15	23	9	544	253
	9	24	29	71	61	824	1159
p	6	7	6	31	31	261	234
	7	5	4	21	20	163	220

It is apparent from the above table that there is no striking difference in performance between a primal and dual approach to LP computations. A more careful study of LP computations reveals, however, that a code's performance can be consistently improved by choosing the reoptimization mode (primal versus dual) of each subproblem according to its structure, m and n. Note also that this option does not only affect LP operations (see NODES count); there are logical implications which appear in one case and not the other due to our special updated row technique; and this can be seen to have a marked effect on the tree.

Table X.3
Influence of logical implications I (NS 1; BR 4; BD 4; CV 2)

		ISEC		NODES		LPITER		MAXFR		DEGREE	
										0	1
B–M	24	224	125	209	133	9818	4553	43	32	41	552
	25	545	375	525	355	23483	13759	95	130	72	1647
L–S	B	20	6	56	16	813	274	37	9	2	85
	C	27	1	156	9	1762	92	9	6	0	77
	D2	1451	281	1102	375	45880	10122	253	103	53	3053
IBM	5	499	60	2224	181	34060	2982	143	22	0	66
	6	302	92	164	39	7491	2679	32	26	0	30
	9	68	41	67	50	2023	1417	15	19	1	74
P	6	13	9	44	27	904	581	26	18	2	129
	7	18	15	36	39	1084	919	17	22	2	122

For each table, the first column corresponds to "no logicals" and the second one to "logicals of degree ≤ 1"; the last two columns indicate how many logicals were used. It is quite apparent that logicals are *effective*. This effect is even more pronounced with other bounding and branching rules which require logicals to generate further improvements.

Table X.4
Influence of logical implications II (NS 1 + NS 7 (single); BR 7; BD 8; RD; CV 2)

	ISEC		NODES		LPITER		MAXFR		DEGREE 0		DEGREE 1		2	CASC
B–M 24	54	67	163	144	2061	1698	37	35	27	38	289	344	661	633
25	105	128	384	329	4133	3274	87	81	70	100	581	737	1091	1400
L–S B	1	3	2	7	71	89	3	5	1	0	24	32	303	113
C	6	8	76	32	345	103	18	12	16	13	306	210	1716	1781
D2	132	152	291	212	3729	2462	71	43	7	26	1231	834	27487	16659
IBM 5	34	68	189	223	1573	1695	24	22	6	10	157	199	789	465
6	37	76	23	54	544	1170	17	32	0	0	9	27	224	127
9	24	26	71	64	824	806	13	14	8	16	104	116	684	393
P 6	7	12	31	44	261	280	13	21	5	2	114	132	358	194
7	5	18	21	44	163	239	9	23	1	0	113	103	151	61

Similar to Table X.3, where "degree ≤ 1" (first column) is now compared to "degree ≤ 2 including *cascades*" (second column). The last two columns indicate how many implications of degree two and higher (cascade type) were used. This table illustrates an interesting phenomenon: by eliminating certain nodes, D2 C fails to discover some powerful D 1 implications and eventually has to search through a larger total number of nodes.

Table X.5
Single versus *double* branching (NS 1; BR 7; BD 8; CV 2; D 1)

	ISEC		NODES		LPITER		DEGREE 0		DEGREE 1		MAXFR	
	S	D	S	D	S	D	S	D	S	D	S	D
B–M 24	61	102	182	113	2125	3448	36	55	322	354	39	23
25	119	189	426	261	4186	6764	83	128	615	695	95	43
L–S B	5	6	20	13	146	168	0	1	12	64	19	13
C	7	21	70	138	288	796	17	64	330	435	21	63
D2	121	300	228	284	3132	8036	6	61	947	2347	52	78
IBM 5	31	75	152	221	1214	3394	4	3	123	99	24	28
6	58	67	61	28	1323	1732	0	0	18	0	37	21
9	34	45	82	51	997	1489	6	2	83	75	20	13
P 6	9	15	62	58	405	569	1	14	23	155	25	34
7	8	33	47	101	279	1502	0	9	107	281	26	39

Table X.6
Influence of feasalties (NS 1 + NS 7 (single); BR 6; BD 8; RD; CV 2; D 1)

	ISEC	NODES	LPITER	MAXFR	DEGREE 0	DEGREE 1
B–M 24	65	206	2193	48	48	439
25	164	464	5610	149	99	1091
L–S B	2	3	82	3	0	27
C	2	15	76	6	0	83
D2	152	277	4049	76	18	1384
IBM 5	35	191	1459	29	9	191
6	13	8	184	7	0	1
9	65	26	756	13	6	94
P 6	3	15	138	8	2	53
7	10	49	465	14	3	127

A comparison with Table X.11 shows comparative advantages of a feasalty strategy. When the objective function is a good predictor for the choice of a branching variable, feasalties are less efficient than penalties. In the case of massive dual degeneracy (IBM 6, P 6), however, penalties are poor and feasalties good. Thus, at each node, penalties and feasalties ought to be combined according to the subproblem structure.

Table X.7
Three typical branching rules (RD; D 1; CV 2)

	ISEC			NODES			LPITER			MAXFR			OPT AT NODE		
B–M 24	52	54	78	111	163	295	2677	2061	3682	30	37	86	2	14	182
25	331	105	301	516	384	1199	16181	4133	14576	171	87	399	356	27	923
L–S B	3	1	3	5	2	8	70	71	105	4	3	5	0	0	0
C	7	6	6	58	76	120	550	345	460	13	18	21	0	0	0
D2	506	132	195	1082	291	618	21823	3729	8375	215	71	152	275	6	352
IBM 5	78	34	68	323	189	701	5789	1513	4643	34	24	109	23	20	46
6	15	37	17	5	23	12	335	544	395	4	17	7	2	20	5
9	27	24	21	46	71	82	1383	824	1208	17	13	22	0	0	0
P 6	4	7	3	14	31	18	224	261	135	8	13	10	3	31	3
7	12	5	5	35	21	33	600	163	197	13	9	17	30	20	27

First columns : NS 1; BR 3; BD 1.
Second columns: NS 1 + NS 7 (single); BR 7; BD 8.
Third columns : NS 1 + NS 7 (single); BR 5; BD 1.

Table X.8

Breadth versus *depth* strategies (NS 6) (BR 7; BD 8; RD; D 1; CV 2; NS 7 (single))

		ISEC			NODES			LPITER			MAXFR			OPT AT NODE		
B–M	24	57	52	52	111	111	111	2809	2677	2737	38	30	28	2	2	2
	25	609	331	324	639	516	525	29378	16181	16127	413	171	166	508	356	368
L–S	B	3	3	1	5	5	5	70	70	70	4	4	4	0	0	0
	C	7	7	6	60	58	58	554	550	537	14	13	12	0	0	0
	D2	369	506	626	706	1082	1305	16977	21823	27373	190	215	248	260	275	459
IBM	5	136	78	63	505	323	262	10352	5789	4675	267	34	26	286	23	19
	6	15	15	15	5	5	5	335	335	335	4	4	4	2	2	2
	9	29	27	26	48	46	45	1561	1383	1309	20	17	14	0	0	0
P	6	3	4	3	14	14	14	211	224	224	7	8	8	3	3	3
	7	15	12	11	40	35	36	799	600	626	13	13	13	31	30	32

First columns : $\alpha = 0$.
Second columns : $\alpha = 0.8$.
Third columns : $\alpha = 1.0$.

Observe that this option is clearly problem dependent. Since the α parameter can be changed at any time without further complication, the art of problem solving consists in divining the proper value for each problem and subproblem; an adaptive strategy with statistical appraisal of its performance is used to modify an initially chosen value. This modification can be local (for one node) or global (for the problem).

Table X.9
Feedback effect for the construction of logical implications (NS 1 + NS 7 (single); BR 7; BD 8; D 1; CV 2)

	ISEC		NODES		LPITER		MAXFR		DEGREE 0		DEGREE 1	
B-M 24	61	46	182	136	2125	1764	39	39	36	31	322	594
25	119	98	426	335	4186	3904	95	97	83	78	615	1103
L-S B	5	5	20	11	146	114	19	12	0	0	12	36
C	7	10	70	69	288	411	21	27	17	16	330	460
D2	121	204	228	234	3132	5925	52	64	6	7	947	2261
IBM 5	31	74	152	257	1214	3546	24	38	4	3	123	1341
6	58	45	61	38	1323	911	37	23	0	0	18	108
9	34	46	82	65	997	1904	20	15	6	8	83	116
P 6	9	8	62	53	405	268	25	24	1	0	123	201
7	8	9	47	43	279	284	26	25	0	0	107	175

The flow of computations is organized here as follows (for each node):

(1) Derive logical implications for the subprogram (in the corresponding subtableau of the initial tableau).

(2) Solve LP.

(3) Compute penalties, bounds and branching variable.

(4) Derive logical implications from special updated rows, in particular the surrogate constraint.

New implications found during steps 3 and 4 may, in turn, lead to new ones if step 1 were repeated; similarly bounds, penalties and branching variable computations also could be improved.

The above table contains in the first columns results obtained by *one* single sweep (1 through 4) while in the second columns we present results obtained by iterating these four steps until no further improvement occurs.

It can be observed that although the number of nodes is generally reduced, the overall performance is frequently degraded because the iteration process is costly.

Table X.10
Five best solutions (NS 1 + NS 7 (single); BR 7; BD 8; CV 2; D 1; RD)

	IMP	LAST AT NODE	ISEC	ISEC B	NODES	LPITER	MAXFR	DEGREE 0	DEGREE 1	SOLUTIONS BEST	2ND	3RD	4TH	5TH
B–M 24	7	579	182	54	741	6813	241	131	661	38	39	39	41	41
25	11	1150	260	105	1150	9707	384	215	1033	43	43	44	44	44
L–S B	9	128	18	1	129	633	75	1	170	550	550	550	550	575
C	11	149	16	6	149	824	71	18	584	73	73	73	73	73
D2	11	854	369	132	886	10108	273	36	3298	−540	−540	−538	−538	−538
IBM 5	6	95	41	34	269	2075	71	5	196	15	15	15	15	15
6	8	239	157	37	239	3594	127	2	140	18	18	18	18	18
9	5	36	26	24	83	934	33	6	103	9	9	9	9	9
P 6	10	238	17	7	238	805	121	4	189	−10618	−10605	−10604	−10602	−10601
7	14	317	30	5	317	1184	168	2	244	−16537	−16524	−16524	−16521	−16520

The column ISEC B indicates the time required to obtain the best solution alone.

Table X.11
Test problems from the literature (NS 1 + NS 7 (single); BD 8; BR 7; CV 2; D 1; RD)

	M	N	Z_{opt}	IMP	OPT AT NODE	ISEC	NODES		LPITER	MAXFR	DEGREE 0	DEGREE 1
B-M 22	20	25	33	1	15	28	97	42	1004	27	23	68
23	20	27	34	1	22	45	147	64	1551	37	30	238
24	20	28	38	1	14	54	163	64	2061	37	27	289
25	20	30	43	1	27	105	384	154	4133	87	70	581
IBM 4	15	30	10	4	12	6	14	10	108	8	1	9
5	15	30	15	2	20	34	189	58	1513	24	6	157
6	31	31	18	3	20	37	23	15	544	17	0	9
9	35	15	9	1	0	24	71	24	824	13	8	104
L-S B	28	35	550	2	0	1	2	1	71	3	1	24
C	12	44	73	1	0	6	76	31	345	18	16	306
D2	37	74	−540	1	6	132	291	113	3729	71	7	1231
E	28	89	1120	4	200	1722	6481	2202	45411	810	874	27766
P 4	10	20	−6120	3	2	2	5	2	67	3	1	21
5	10	28	−12400	7	3	2	8	4	73	6	2	25
6	5	39	−10618	10	31	7	31	17	261	13	5	114
7	5	50	−16537	11	20	5	21	11	163	9	1	113

References: B-M [9], IBM [16], L-S [24], P [31].

The second NODES column indicates the other accounting method described in Table X.0; the same holds in Table X.12. Observe that our performance is admittedly poor in solving the smallest problems; also for problems where the surrogate constraint alone is a sufficiently powerful tool. For hard and/or large problems however, where a product form LP algorithm is necessary, the superiority of our code emerges.

Table X.12
Some ILP problems (see Appendix Y)(NS 1 + NS 7 (single); BR 7; BD 8; CV 2; D 1; RD)

	M	N	Z_{opt}	IMP	OPT AT NODE	ISEC	NODES		LPITER	MAXFR	DEGREE 0	DEGREE 1
P 0	8	15	− 108	1	8	2	18		107	9	3	29
1	10	100	−1689	1	91	60	182		1142	93	12	200
2	10	50	− 765	2	164	74	495		3676	116	118	1024
3	10	75	−1217	2	126	60	260		1986	102	58	120
4	10	100	−1835	3	98	24	99		139	91	4	55
5	10	100	−1921	1	97	15	97		120	99	0	2
6	20	100	−2042	4	118	48	118		749	90	8	502
7	50	100	− 956	2	33	35	45		358	27	3	292
8	10	100	−5555	5	2009	476	2009		12824	1150	25	4052
9	10	100	≤ −5239	9	1741	> 1800						
10	10	200	−3492	3	8	8	11	6	118	6	3	407
11	10	200	−3482	6	14	101	179	110	1871	71	57	1882
12	20	200	−3713	8	66	58	129	66	538	31	15	1033
13	20	200	≤ −5130	14	842	> 900						
14	20	200	−2515	9	213	149	235	125	1380	69	46	4803
15	50	200	−2893	8	179	207	232	101	1405	61	21	2105
16	50	200	−1693	2	4	17	23	10	105	10	6	428
17	50	200	−1662	1	4	24	25	11	163	12	7	604
18	20	200	≤ − 842	4	100	> 900						
19	10	400	−4562	8	168	291	303	147	3619	84	34	4021
20	10	400	≤ −2946	1	134	> 600						
21	20	400	−5946	2	2	3	2	1	37	3	2	784
22	50	50	≤ 1980	3	6	> 300						

For problems with > in the ISEC column, the current best solution is given; optimality was not ascertained in 1800 sec.

Table X.13
Set covering problems (NS 1 + NS 7 (single); BD 8; BR 7; CV 2; DUAL; D 1) [25]

PROBLEM (Salkin)	M	N	Z_{opt}	IMP	ISEC	NODES	LPITER	MAXFR	DEGREE 1	OPT AT NODE
8	30	80	13	1	3	1	62	2	0	1
9	30	90	13	7	2	0	80	1	0	0
10	30	80	13	2	7	8	130	9	32	7
11	30	70	14	1	5	7	106	6	0	4
12	30	80	13	1	2	1	74	3	0	1
13	104	133	1678	1	9	2	189	3	0	2
16	200	500	642	1	60	0	395	1	0	0
18	50	450	24	1	19	5	159	5	0	3
19	36	455	43	1	3	0	41	1	0	0
20	46	683	138	1	38	1	270	2	0	1
18 =[a]	50	450	25	3	57	6	404	6	911	6
19 =	36	455	∞[b]	0	5	0	42	0	1	0
20 =	46	683	∞	0	10	0	67	0	3	0

[a] With equality constraints.
[b] No feasible solution.
Our performance here is generally better than in the literature. The optimal solution is detected immediately.

Appendix Y. A class of zero–one programming test problems

Our motivation for describing below a set of test problems is manifold :

(i) Good test problems are not easy to come by; a great many numerical results are based on problems which are nowhere available for comparative studies (not even from the authors).

(ii) Often running times are given without the *solution* (not even the optimal value) which also makes checks and comparisons impossible.

(ii) The modest set of problems which is readily available in the literature and frequently used for comparative purposes is *totally inadequate* to test the limits and versatility of an efficient code.

The proposed set of problems has the following characteristics :

They are easily reproducible (see attached generating programs).

- Problems of any size can be generated, both in terms of density and/or number of constraints and variables; commonly available test problems are, with few exceptions, too small for efficient codes: the exponential growth of the "difficulty" of integer programming problems renders experimentation with small problems questionable.
- Problems can be made extremely difficult, for any given size, simply by varying certain generating parameters.
- Usually, rounding and cost ranking of the variables does not really help in solving these problems.
- It is often easy to find a good solution; but it is very difficult to find an optimal solution and to prove its optimality. Unlike most other test problems, it often occurs here that good branching strategies do not find an optimal solution in the earlier stages of the execution. This puts an additional *accent on the numerical accuracy* of bound computations.

Finally, in spite of their artificial nature, the proposed problems possess some typical characteristics of integer programming applications: the constraints are either of the *budget restraint* type (multiple knapsack) or of a type specifying *minimal accomplishment* levels (reversed knapsack).

Y.1. *Problem generator*[2]

A generator defines the following number stream (u_k, v_k): Let

$$u_0, \quad p, \quad q, \quad r_0, \quad r \tag{1}$$

be a set of *integral seeds*.

[2] A FORTRAN program of this generator can be obtained from the author.

Set[3]

$$u_{k+1} = p + q\,u_k \,(\mathrm{mod}\, r_0), \tag{2}$$

$$v_{k+1} = \lfloor u_{k+1}(r-1)/(r_0 - 1) + 0.5 \rfloor \tag{3}$$

For all our problems, the following coefficients are used:

$$p = 7, \qquad q = 20\,029,\ r_0 = 100\,000, \qquad u_0 = 39\,527.$$

Each problem is characterized by a set of 8 *parameters*:

$$m, n, r \in \mathbf{N}, \qquad \alpha, \beta, \gamma, \delta, \sigma \in \mathbf{R}^+$$

in the *general form*:

$$\text{minimize} \quad Z = \sum_{j=1}^{n} c_j x_j, \tag{4}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j + (-1)^i s_i = b_i, \tag{5}$$

$$x_j = 0,1, \qquad j = 1, \ldots, n, \tag{6}$$

$$s_i \geq 0 \text{ integer}, \qquad i = 1, \ldots, m. \tag{7}$$

The coefficients a_{ij} and c_j are computed by a 60-line subroutine which is omitted here for brevity.

Y.2. *Some comments*

(1) The matrix (a_{ij}) has a projected *density* σ: every row has approximately σn entries and the matrix therefore has about $\sigma n m$ entries ($\neq 0$).

Non-zero entries are (approximately) found along the diagonal, with band width $\lfloor \sigma n \rfloor$.

(2) All real parameters may take arbitrary values in $\mathbf{R}^+$. However, the following restrictions should be observed:

$1/m \leq \sigma \leq 1$	($\sigma > 1$: matrix full, $\sigma < 1/m$: problem is a knapsack problem),
$\alpha \geq 0, \quad \beta \geq 0,$	
$0 \leq \gamma \leq 2$	($\gamma \gg 2$: no feasible solution),
$0 \leq \delta \leq \frac{4}{3}$	($\gamma \gg \frac{4}{3}$: solution trivial),
$\frac{2}{3}\gamma \leq \delta$	($\gamma \gg \delta$: no feasible solution).

[3] The floor and ceiling functions ($\lfloor\ \rfloor$ and $\lceil\ \rceil$, resp.) are the standard APL notations; a small FORTRAN program is also given in the problem generator to avoid confusion.

(3) $\alpha = \beta = \gamma = \delta = 1$ furnishes an "average" problem (and problem difficulty). For $\gamma < 1, \delta > 1$ the problem becomes easier since more vertices of the unit hypercube become feasible. On the other hand, $\frac{2}{3}\gamma \approx \delta$ usually produces more difficult problems.

For $\alpha > \beta$, the objective function tends to be *positive*, and the "$\geq$" (10a) constraints *active*. In this case approximately $\lceil \frac{1}{2}\gamma \rceil$ of the variables are 1 for feasible solutions.

For $\alpha < \beta$, the objective function tends to be *negative*, and the "$\leq$" (10b) constraints *active*. Approximately $\lfloor \frac{1}{2}\delta \rfloor$ of the variables assume the value 1.

Choosing $\alpha \gg \beta$ or $\beta \gg \alpha$ or $\frac{2}{3}\gamma \approx \delta \leq \frac{1}{2}$ creates very difficult problems (as long as the other parameters do not take trivial simplifying values).

Y.3. *Test problems*

Table Y.1 contains a sample of problems adopted to illustrate the performance of our code.

Table Y.1

Problem	m	n	r	α	β	γ	δ	σ	LP_{opt}	Z_{opt}
P 0	8	15	20	1	1	1	1	$\frac{3}{5}$	−130.62	−108
1	10	100	50	1	1	1	1	$\frac{1}{2}$	−1704.46	−1689
2	10	50	50	1	1	1	1	$\frac{1}{2}$	−797.48	−765
3	10	75	50	1	1	1	1	$\frac{1}{2}$	−1243.58	−1217
4	10	100	50	1	1	1	$\frac{3}{2}$	$\frac{1}{2}$	−1848.11	−1835
5	10	100	50	1	1	$\frac{1}{2}$	1	$\frac{1}{2}$	−1921.50	−1921
6	20	100	50	1	1	$\frac{1}{2}$	$\frac{16}{15}$	$\frac{1}{4}$	−2065.46	−2042
7	50	100	50	1	1	$\frac{1}{2}$	$\frac{16}{15}$	$\frac{1}{10}$	−998.49	−956
8	10	100	50	0	1	$\frac{1}{2}$	$\frac{16}{15}$	$\frac{1}{2}$	−5557.50	−5555
9	10	100	50	0	1	1	1	$\frac{1}{2}$	−5245.07	$\leq$ −5239
10	10	200	50	1	1	1	$\frac{3}{2}$	$\frac{1}{2}$	−3495.94	−3492
11	10	200	50	1	1	1	1	$\frac{1}{2}$	−3493.60	−3482
12	20	200	50	1	1	1	$\frac{3}{2}$	$\frac{1}{4}$	−3726.24	−3713
13	20	200	50	0	1	1	1	$\frac{1}{8}$	−5156.12	$\leq$ −5130
14	20	200	50	1	1	$\frac{1}{2}$	1	$\frac{1}{8}$	−2537.86	−2515
15	50	200	50	1	1	1	$\frac{3}{2}$	$\frac{1}{10}$	−2925.21	−2893
16	50	200	50	1	1	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{1}{20}$	−1716.27	−1693
17	50	200	50	1	1	$\frac{1}{2}$	1	$\frac{1}{20}$	−1687.00	−1662
18	20	200	50	1	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{8}$	−942.68	$\leq$ −842
19	10	400	50	1	1	1	$\frac{3}{2}$	$\frac{1}{4}$	−4568.16	−4562
20	10	400	50	1	1	1	1	$\frac{1}{5}$	−3021.02	$\leq$ −2946
21	20	400	50	1	1	$\frac{1}{2}$	1	$\frac{1}{5}$	−5948.38	−5946
22	50	50	20	1	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1827.83	$\leq$1980

Problem P 0 is *not* a difficult problem. It is given explicitly in Table Y.2 to allow further users to verify their problem generator.

Table Y.2

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	1
22[a]	−9	−11	8	25	−24	−23	−15	−25	−8	5	−5	−15	18	−11	
14	14	14	3	11	3	5	5	2	4	16	0	0	0	0	≥ 40
0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	≤ 7
0	0	0	2	3	2	3	2	2	3	1	1	1	0	0	≥ 8
0	0	0	0	0	15	11	12	9	7	18	9	15	0	0	≤ 68
5	5	7	0	0	0	0	2	1	1	8	6	6	4	1	≥ 20
1	16	17	0	0	0	0	0	0	8	2	7	17	2	18	≤ 68
6	11	12	14	16	0	0	0	0	0	0	9	16	11	8	≥ 44
7	7	11	2	3	3	10	4	8	0	0	0	0	1	2	≤ 41

[a] This and the other numbers of this row are objective function coefficients.
Optimal solution $Z_{opt} = -108$.
$x_j = 1$ for $j = 2, 3, 6, 7, 8, 13, 15$; and 0 otherwise.

References

[1] E.Balas, "An additive algorithm for solving linear programs with zero–one variables", *Operations Research* 13 (1965) 517–546.

[2] E. Balas, "Discrete programming by the filter method", *Operations Research* 15 (1967) 915–957.

[3] E. Balas, "Intersection cuts—A new type of cutting planes for integer programming", *Operations Research* 19 (1) (1971) 19–39.

[4] E. Balas, "Integer programming and convex analysis: Intersection cuts from outer polars", *Mathematical Programming* 2 (1972) 330–382.

[5] E. Balas, "A constraint-activating outer polar method for solving pure or mixed integer 0–1 programs", MSRR # 275, Carnegie-Mellon University, Pittsburgh, Pa. (1972).

[6] E. Balas and R. Jeroslow, "Canonical cuts on the unit hypercube", *SIAM Journal on Applied Mathematics* 23 (1) (1972).

[7] E.M.L. Beale and R.E. Small, "Mixed integer programming by a branch and bound technique" in: W.A. Kalenich, ed., *Proceedings of the IFIP Congress* 1965, Vol. 2, (Spartan Press, East Lansing, Mi.) pp. 450–451.

[8] M. Benichou et al., "Experiments in mixed-integer linear programming", *Mathematical Programming* 1 (1971) 76–94.

[9] B. Bouvier and G. Messoumian, "Programmes linéaires en variables bivalentes—Algorithme de Balas", Université de Grenoble (1965).

[10] C.-A. Burdet and E. Johnson, "A subadditive approach to the group problem of integer programming", IBM Thomas J. Watson Laboratories, Yorktown Heights, N.Y. (1972).

[11] C.-A. Burdet, "Enumerative cuts I", *Operations Research* 21 (1) (1973) 61–89.

[12] C.-A. Burdet, "On cutting planes", Working Paper 73–22, Faculty of Management Sciences, University of Ottawa (1973).

[13] C.-A. Burdet, "On the algebra and geometry of integer cuts", Working Paper 74-8, Faculty of Management Sciences, University of Ottawa (1972, revised and extended 1974).
[14] G.B. Dantzig, "*Linear programming and extensions*" (Princeton University Press, Princeton, N.J., 1963).
[15] J.J.H. Forst, J.P.H. Hirst and J.A. Tomlin, "Practical solution of large mixed integer programming problems with "UMPIRE", *Management Science* 20 (5) (1974).
[16] A.M. Geoffrion, "An improved implicit enumeration approach for integer programming", *Operations Research* 17 (1969) 437–454.
[17] A.M. Geoffrin and R.E. Marsten, "Integer programming algorithms: Framework and State-of-the art survey", *Management Science* 18 (9) (1972).
[18] F. Glover, "Surrogate constraints, *Operations Research* 16 (1968) 741 479.
[19] F. Glover, "Convexity cuts", University of Texas, Austin (December 1969).
[20] R. Gomory, "An algorithm for integer solutions to linear programs", in: R.L. Graves and P. Wolfe, eds., *Recent advances in mathematical programming* (McGraw Hill, N.Y., 1963).
[21] P.L. Hammer, "BABO A boolean approach to bivalent optimization", Publications du Centre de recherches mathématiques de l'Université de Montréal, No. 101 (May 1971).
[22] A.H. Land and A.G. Doig, "An automatic method of solving discrete programming problems", *Econometrica* 28 (1960) 497–520.
[23] E.L. Lawler and D.E. Wood, "Branch-and-bound methods: A survey", *Operations Research* 14 (1966) 699 719.
[24] C.E. Lemke and K. Spielberg, "Direct search algorithms for zero-one and mixed-integer programming", *Operations Research* 15 (1967) 892 914.
[25] C.E. Lemke, H.M. Salkin and K. Spielberg, "Set covering by single branch enumeration with linear programming subproblems", *Operations Research* 19 (4) (1971) 998 1022.
[26] J.D.C. Little et al., "An algorithm for the traveling salesman problem", *Operations Research* 11 (1963) 972 989.
[27] Mathematical Programming System Extended (MPSX), Mixed Integer Programming (MIP), Program Number 5734-XM4 (IBM World Trade Corporation, New York, 1971).
[28] G. Mitra, "Investigation of some branch and bound strategies for the solution of mixed integer linear programs", *Mathematical Programming* 4 (1973) 155 170.
[29] G.L. Nemhauser and R. Garfinkel, *Integer programming* (Wiley, New York, 1972).
[30] OPHELIE MIXED, Mixed Integer Programming System, User Information Manual (SEMA (METRA International), Paris, 1970).
[31] C.C. Petersen, "Computational experience with variants of the Balas algorithm applied to the selection of R & D project", *Management Science* 13 (1967) 736 750.
[32] K. Spielberg, "Minimal preferred variable reduction for zero one programming", Tech. Rept. No. 320 3013 IBM, Philadelphia Scientific Center (1972).
[33] K. Spielberg, "A minimal-inequality branch-bound method, IBM, Philadelphia Scientific Center (October 1972).
[34] J.A. Tomlin, "Branch and bound methods for integer and non convex programming" in: J. Abadie, ed., *Integer and nonlinear programming* (North-Holland, Amsterdam, 1970) pp. 437 450.
[35] UMPIRE System Users' Guide (Scientific Control Systems Ltd., London, 1970).

Mathematical Programming Study 2 (1974) 51–71. North-Holland Publishing Company

A SUBADDITIVE APPROACH TO THE GROUP PROBLEM OF INTEGER PROGRAMMING

Claude-Alain BURDET

Carnegie-Mellon University, Pittsburgh, Pa., U.S.A.

and

Ellis L. JOHNSON

I.B.M. Thomas J. Watson Research Center, Yorktown Heights, N.Y., U.S.A.

Received 11 September 1973
Revised manuscript received 27 June 1974

Solving the linear program associated with an all-integer program gives the group problem $N x \equiv b \pmod 1$, $x \geq 0$ and integer, $z = c x$ (minimize), upon relaxation of non-negativity of the basic variables, where $c \geq 0$ and N is the fractional part of the updated, non-basic columns. A method is given for solving this problem which does not require an explicit group representation and is not dependent on knowing the order of the group. From a *diamond gauge function* the algorithm constructs a continuous function π, which is shown to be subadditive on the unit hypercube. Such continuous functions yield valid inequalities and are used in solving the group problem.

1. Integer programs and their group problems

Given an *integer program*

$$\begin{aligned} &\text{minimize} && z = c\,x, && \text{(1a)}\\ &\text{subject to} && A\,x = b, \\ & && x \geq 0, && \text{(1b)}\\ & && x \quad \text{integer}, && \text{(1c)} \end{aligned}$$

we consider the *associated LP optimal basis* representation of (1a, b):

$$\begin{aligned} &\text{minimize} && (z - z_B) = \bar{c}_N x_N, && \text{(2a)}\\ &\text{subject to} && x_B + (B^{-1}N)\,x_N = B^{-1}b, \\ & && x_B, x_N \geq 0, && \text{(2b)} \end{aligned}$$

where

$$z_B = c_B B^{-1} b, \tag{3a}$$

$$\bar{c}_N = c_N - c_B(B^{-1}N) \geq 0. \tag{3b}$$

As is customary in linear programming, this representation (2), (3) corresponds to a partition of the original matrix A into

$$A = (B, N), \tag{4}$$

where B is a square non-singular matrix (basis). To simplify, the non-basic variables, x_N will be denoted by $x_1, x_2, \ldots, x_n$, and the basic variables x_B by $x_{n+1}, x_{n+2}, \ldots, x_{n+m}$. Written according to the partition (4), the constraints (1b) read

$$B x_B + N x_N = b, \qquad x_B \geq 0, \quad x_N \geq 0.$$

Relaxing $x_B \geq 0$ but imposing the integrality requirement (1c), one obtains

$$N x_N \equiv b \,(\text{mod } B), \tag{5a}$$

$$x_N \geq 0 \text{ and integer.} \tag{5b}$$

One easily verifies that $N x_N \,(\text{mod } B)$ generates an *abelian group.*

We apply a non-singular linear transformation (basis transformation) to obtain an *isomorphic representation* of this group:

$$(B^{-1}N)\, x_N \equiv B^{-1}b \,(\text{mod } 1) \tag{6}$$

$$x_N \geq 0 \text{ and integer.}$$

Now comparing (6) and (2b), we note that the group structure can be defined directly in terms of the *updated LP tableau* corresponding to the LP optimal solution

$$x_B = \bar{b} - \bar{A}\, x_N \geq 0,$$

$$\bar{b} = B^{-1}b, \qquad \bar{A} = (B^{-1}N).$$

Since the congruence (6) is not essentially modified when the entries of the matrix $\bar{A} = (B^{-1}N)$ are changed by integral amounts, we can restrict our attention to the matrix $F = (f_{ij})$ defined as follows. Let

$$f_{i0} = \bar{b}_i - \lfloor \bar{b}_i \rfloor \geq 0, \qquad i \in D \subset B.$$

The index set D denotes a subset of fractional (basic) variables; it corresponds to a set of integrality requirements which are not satisfied by the current LP optimal solution and will be enforced via the group problem 7.

Now define the vectors $f_j \quad (j \in N)$ by

$$f_{ij} = \begin{cases} \bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor & \text{if } 0 \leq \bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor \leq f_{i0}, \\ \bar{a}_{ij} - \lceil a_{ij} \rceil & \text{if } f_{i0} \leq \bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor \leq 1. \end{cases}$$

Note that the unit hypercube $U \subset \mathbf{R}^D$,

$$U = \{u: f_{i0} - 1 \leq u_i \leq f_{i0} \text{ for all } i \in D\},$$

contains the vectors $f_0, f_1, \ldots, f_n$.

One now defines the *group minimization problem* corresponding to the integer program (1) by

$$\text{minimize} \quad (z - z_B) = \sum_{j=1}^{n} \bar{c}_j x_j, \tag{7a}$$

$$\text{subject to} \quad \sum_{j=1}^{n} f_{ij} x_j \equiv f_{i0} \pmod 1, \qquad i \in D, \tag{7b}$$

$$x_j \geq 0 \text{ and integer}, \qquad j = 1, \ldots, n. \tag{7c}$$

Note that the integer program (1) and the group problem (7) are not equivalent since the requirement $x_B = B^{-1}b - B^{-1}N\,x_N \geq 0$ of (1) has been relaxed in (7) to (7b). This paper is primarily concerned with the group problem (7); thus with respect to the original integer program (1), our results merely reflect a *necessary condition*; of course, necessary conditions of the problem (7) are also necessary conditions for (1), while the (optimal) solution of (7) is only an (optimal) solution to (1) when it passes the sufficiency test $x_B \geq 0$.

There exists a variety of methods in the literature to derive necessary conditions for integer programs (1): our approach is to use the group problem as a vehicle to this end. One may also consider only a *subgroup* defined by a subset of the rows $i = 1, \ldots, d$; historically, this is the initial approach adopted by Gomory [3] where only *one row* is used to generate a necessary condition (*cutting plane*). Another way to form a subproblem would be to distinguish between rows stemming from *slack* and *structural* variables in the original LP formulation, since these two types of variables usually have a very different cost configuration.

In the theory presented below, we have adopted the basic philosophy which consists in pursuing the determination of the "*most restrictive*" necessary conditions which can be obtained for the group problem (7); *typically, our approach is capable of producing necessary and sufficient conditions for* (7). In fact, it can be seen to provide a direct method for finding an optimal solution to the group problem which *does not require any explicit knowledge of the group structure* (like its factorization into cyclic subgroups via the Smith normal form) *and which is not affected by a large group order.*

In [5], a method was given which was only applicable to cyclic group problems. This paper gives a method based on *diamond gauge functions* which make it possible to do away with this assumption that the group is cyclic; this study also results in a different algorithm for the cyclic group case.

As far as the original integer program is concerned, the present theory also opens some new paths:

(i) The *cutting planes* and *bounds* which are described in this paper can be used in a branch and bound context.

(ii) An extension of the present result furnishes a new type of approach for finding directly the optimal solution to integer programs; this is the object of a follow-up paper.

2. Valid inequalities

Definition 2.1. The inequality $\sum_{j=1}^{n} \pi_j x_j \geq \pi_0$ is called *valid* if it is satisfied by all feasible integer solutions to (1).

Of course, an inequality which is satisfied by all the solutions to the group problem (7) is valid. The following theorem indicates how valid inequalities can be obtained.

Theorem 2.2. *Let π be a function defined on the d-dimensional unit hypercube U satisfying*

$$\pi(u) \geq 0 \quad \textit{for all } u \in U, \tag{8a}$$

$$\pi(0) = 0, \tag{8b}$$

$$\pi(u) + \pi(v) \geq \pi(u + v) \quad \textit{(subadditivity)} \tag{8c}$$

for all u and v in U, where $(u + v)$ is taken modulo 1 so that $(u + v) \in U$.

Then the inequality

$$\sum_{j=1}^{n} \pi(f_j)\, x_j \geq \pi(f_0) \tag{9}$$

is valid.

Proof. Take a (feasible) solution u' to the group problem

$$u' = \sum_{j=1}^{n} f_j x_j \equiv f_0, \qquad x_j \geq 0 \text{ and integer};$$

we show that

$$\sum_{j=1}^{n} \pi(f_j) x_j \geq \pi\left(\sum_{j=1}^{n} f_j x_j\right),$$

where f_j is assumed for all $j \in N$ to belong to U. The proof is straightforward, starting with $u = f_{j_1}$ (j_1 is such that $x_{j_1} > 0$) and $v = f_{j_1}$, one applies (8c) to obtain

$$\pi(u) + \pi(v) = \pi(f_{j_1}) + \pi(f_{j_1}) \geq \pi(2f_{j_1}). \qquad (*)$$

This can be repeated (with $u = 2f_{j_1}$)

$$\pi(2f_{j_1}) + \pi(f_{j_1}) \geq \pi(3f_{j_1}), \qquad (*)$$

$$\vdots$$

until the point $u = x_{j_1} f_{j_1}$ is reached; now choose $v = f_{j_2}$ ($j_2 \neq j_1$, such that $x_{j_2} > 0$)

$$\pi(x_{j_1} f_{j_1}) + \pi(f_{j_2}) \geq \pi(x_{j_1} f_{j_1} + f_{j_2}), \qquad (*)$$

$$\vdots$$

until $u = x_{j_1} f_{j_1} + x_{j_2} f_{j_2}$ is reached; and so on until the last inequality

$$\pi(u' - f_{j_n}) + \pi(f_{j_n}) \geq \pi(u'). \qquad (*)$$

Now summing up all (*) inequalities, one obtains

$$\sum_{j=1}^{n} x_j \pi(f_j) \geq \pi\left(\sum_{j=1}^{n} x_j f_j\right) = \pi(u')$$

because the right-hand side of each inequality cancels out with the first term of the next inequality.

Subadditivity and other properties of valid inequalities are also developed in [4] and [6].

3. Subadditivity and diamond gauge functions

Consider the d-dimensional vector space $\mathbf{R}^d$ corresponding to the constraints (7b) of the group problem (7); we have

$$f_0 = (f_{10}, \ldots, f_{d0}) \in \mathbf{R}^d.$$

For the unit hypercube U, one knows that

(a) $0 \in U \subset \mathbf{R}^d$,

(b) for any vertex V of U, one has

$$V_i = f_{i0} \quad \text{or} \quad f_{i0} - 1 \quad \text{for all } i = 1, \ldots, d.$$

Definition 3.1. Given a set of $(d + 1)$ parameters $\alpha_0, \alpha_1, \ldots, \alpha_d$ satisfying

$$\alpha_i \geq 0 \quad \text{for all } i = 0, 1, \ldots, d, \tag{10a}$$

$$\sum_{i=1}^{d} \alpha_i f_{i0} \bar{f}_{i0} = \alpha_0, \quad \text{with } \bar{f}_{i0} = 1 - f_{i0}. \tag{10b}$$

We define the *diamond gauge function* D by

$$D(u) = \sum_{i=1}^{d} \alpha_i \sigma_i(u_i)\, u_i, \tag{11a}$$

with

$$\sigma_i(u_i) = \begin{cases} \bar{f}_{i0} & \text{if } 0 \leq u_i, \\ -f_{i0} & \text{if } u_i \leq 0. \end{cases} \tag{11b}$$

Property 3.2. There are d degrees of freedom in the definition of the function D; the d parameters $\alpha_1, \ldots, \alpha_d$ (plus one degree for the scaling factor α_0 which is compensated for by the equation (10b)). For the time being, these parameters are kept unspecified; but an algorithm (see Section 4, remark 3) will try to take advantage of such freedom.

Property 3.3. For any vertex V of U, one has

$$D(V) = \sum_{i=1}^{d} \alpha_i f_{i0} \bar{f}_{i0} = \alpha_0;$$

furthermore,

$$D(0) = 0,$$

$$0 \leq D(u) \leq \alpha_0 \quad \text{for all } u \in U.$$

Property 3.4. Consider one of the 2^d (truncated) orthants Q

$$Q = \left\{ u: \begin{array}{ll} 0 \leq u_i \leq f_{i0} & \text{for all } i \in \Delta^+ \\ -\bar{f}_{i0} \leq u_i \leq 0 & \text{for all } i \in \Delta^- \end{array} \right\} \subset U, \tag{12}$$

where Δ^+ and Δ^- are given disjoint index sets such that $\Delta^+ \cup \Delta^- = \{1, \ldots, d\}$ which characterize Q.

For any $u \in Q$, one has

$$D(u) = \sum_{i \in \Delta^+} \alpha_i \bar{f}_{i0} u_i - \sum_{i \in \Delta^-} \alpha_i f_{i0} u_i$$

$$= \sum_{i=1}^{d} \delta_i u_i \tag{13a}$$

with

$$\delta_i = \begin{cases} \alpha_i \bar{f}_{i0} & \text{for } i \in \Delta^+ \quad (u_i \geq 0), \\ -\alpha_i f_{i0} & \text{for } i \in \Delta^- \quad (u_i \leq 0). \end{cases} \tag{13b}$$

Thus we have proved the following lemma.

Lemma 3.5. *The diamond function D is linear in Q.*

Furthermore, one has:

Lemma 3.6. *The level set* $\text{lev}_\alpha D = \{u: D(u) \leq \alpha\}$ *is convex.* (See also [2]).

Proof. For $\alpha < 0$, one has $\text{lev}_\alpha D = \emptyset$ because $D(u) \geq 0$ by construction (see Property 3.3). For $0 \leq \alpha \leq \alpha_0$, the set $\text{lev}_\alpha D$ can be seen to be defined by the intersection of 2^d halfspaces of the form

$$\sum_{i=1}^{d} \delta_i u_i \leq \alpha_0$$

generated by all the partitions Δ^+, Δ^- of the index set $\{1, \ldots, d\}$; hence $\text{lev}_\alpha D$ is convex.

Property 3.7. The function D has been named after the set $\text{lev}_{\alpha_0} D$ which is a *diamond polyhedron centered at the origin* (see [1, 2]); for each vertex V of U there corresponds one $(d-1)$-dimensional facet of $\text{lev}_{\alpha_0} D$.

Lemma 3.8. *D is a convex gauge function on U.*

Proof. We must show that D is a gauge and that it is convex.

(a) $D(u) \geq 0$, and $D(0) = 0$ (see Property 3.3).

(b) Lemma 3.5 shows that D is linear in every orthant Q; thus it is linear along every ray starting from the origin.

(c) Since the level sets $\text{lev}_\alpha D$ are convex, D is quasi-convex; then (a) and (b) imply convexity of D.

Lemma 3.9. *D is subadditive on U.*

Proof. Subadditivity follows from convexity. One has for all $\lambda \in [0, 1]$,

$$D(\lambda u + (1 - \lambda) v) \leq \lambda D(u) + (1 - \lambda) D(v) \quad \text{for all } u, v \in U.$$

Take $\lambda = \frac{1}{2}$, then

$$D(\tfrac{1}{2}(u + v)) \leq \tfrac{1}{2}(D(u) + D(v)).$$

But D is linear on the ray passing through $(u + v) \in U$; thus

$$D(\tfrac{1}{2}(u + v)) = \tfrac{1}{2}D(u + v) \geq \tfrac{1}{2}D(w),$$

where $w \in U$ and $w \equiv u + v \pmod 1$ by Lemma 3.10.

Lemma 3.10. *For $u \in U$,*

$$D(u) \leq D(t) \quad \textit{for all } t \equiv u \pmod 1.$$

Proof. For a given $u \in U$, let $t = u + k$, where k_i integer, $i = 1, \ldots, d$. From (11),

$$\begin{aligned} D(t) &= \sum_{i=1}^{d} \alpha_i \sigma_i(t_i)\, t_i \\ &= \sum_{i=1}^{d} \alpha_i \sigma_i(t_i)\,(u_i + k_i). \end{aligned}$$

Thus, by a coordinate-wise argument, we show

$$\sigma_i(t_i)\,(u_i + k_i) \geq \sigma_i(u_i)\, u_i \quad \text{for each } i.$$

Consider the two cases:

(a) $u_i \geq 0$. The inequality is true if $k_i \geq 0$ since then $\sigma_i(t_i) = \sigma_i(u_i) = \bar{f}_{i0} \geq 0$. We need only consider $k_i \leq -1$. Furthermore, $u_i \leq f_{i0}$ and also $u_i + k_i \leq -\bar{f}_{i0} = f_{i0} - 1$. Therefore,

$$\begin{aligned} \sigma_i(u_i)\, u_i = \bar{f}_{i0} u_i &\leq \bar{f}_{i0} f_{i0} \\ &\leq -f_{i0}(u_i + k_i) = \sigma_i(t_i)(u_i + k_i). \end{aligned}$$

(b) $u_i < 0$. Similarly, we need only consider $k_i \geq 1$. There, one has $-\bar{f}_{i0} = f_{i0} - 1 \leq u_i$ and $u_i + k_i \geq f_{i0}$. Hence,

$$\begin{aligned} \sigma_i(u_i)\, u_i = -f_{i0} u_i &\leq \bar{f}_{i0} f_{i0} \\ &\leq \bar{f}_{i0}(u_i + k_i) = \sigma_i(t_i)(u_i + k_i). \end{aligned}$$

The basic tool of this paper is the following function π:

$$\pi(u) = \min_{e \in E} \{d_e + D(u - e)\} \quad \text{for all } u \in U, \tag{17}$$

where E is any finite subset of U called *generator set* with given quantities $d_e \geq 0$ for all $e \in E$.

Theorem 3.11. *π is subadditive on U, provided it is subadditive on E, that is, if $\pi(e_1) + \pi(e_2) \geq \pi(e_1 + e_2)$ holds true for every pair $e_1, e_2 \in E$.*

Note that $(e_1 + e_2)$ need not be in E.

Proof (by contradiction). Suppose $\pi(u) + \pi(v) < \pi(u + v)$ for some $u, v \in U$. Now one has by construction, $\pi(u) = d_e + D(u - e)$ for some $e \in E$. Consider the line between u and e,

$$w(\lambda) = u + \lambda(e - u) \qquad \lambda \in [0, 1];$$

and one has

$$\pi(w(1)) + \pi(v) = d_e + \pi(v) = \pi(u) + \pi(v) - D(u - e).$$

In a similar way, we know that $\pi(e + v) = d_f + D(e + v - f)$ for some $f \in E$; we will next show that $\pi(e) + \pi(v) < \pi(e + v)$, that is,

$$d_e + \pi(v) < d_f + D(e + v - f).$$

Recalling that the definition (17) of π stipulates that

$$\pi(u + v) \leq d_f + D(u + v - f)$$

and that we previously assumed (for the sake of contradiction) that

$$\pi(u) + \pi(v) = d_e + D(u - e) + \pi(v) < \pi(u + v),$$

we now obtain:

$$\begin{aligned} \pi(u) + \pi(v) - D(u - e) = d_e + \pi(v) &< \pi(u + v) - D(u - e) \\ &\leq d_f + D(u + v - f) - D(u - e), \end{aligned}$$

while we need show

$$d_e + \pi(v) < d_f + D(e + v - f);$$

thus it is sufficient to show that

$$D(u + v - f) - D(u - e) \leq D(e + v - f),$$

i.e.,

$$D(u - e) + D(e + v - f) \geq D(u + v - f)$$

that is

$$D(u') + D(v') \geq D(u' + v'), \quad \text{with } u' = u - e, \quad v' = e + v - f.$$

Lemma 3.9 can now be seen to establish the desired result.

In summary, we have shown that

$$\pi(u) + \pi(v) < \pi(u + v) \quad \text{implies} \quad \pi(e) + \pi(v) < \pi(e + v) \quad \text{for some } e \in E,$$

where u and v are arbitrary points in U. Let us now choose v as a new point u and let e be the new point v and apply the above implication with these new points; one obtains

$$\pi(v) + \pi(e) < \pi(v + e) \quad \text{implies} \quad \pi(e') + \pi(e) < \pi(e' + e),$$

where e' is some element of E such that

$$\pi(v) = d_{e'} + D(v - e').$$

The proof is now complete because

$$\pi(e') + \pi(e) < \pi(e' + e)$$

exhibits a contradiction to the subadditivity hypothesis of π on the generator set E.

To conclude, we note that Sections 2 and 3 contain the necessary theoretical foundations for a method to construct valid inequalities: given any generator set E with the described properties, the relations (9) and (17) will deliver a valid inequality.

In the next section, we show that:

(i) The classical mixed-integer Gomory cuts correspond to a special (trivial) generator set E.

(ii) Better (i.e., more restrictive) inequalities can be obtained in a constructive process which defines non-trivial generator sets E.

(iii) This process terminates finitely in a natural way when a generator set E containing the point $f_0 \in U$ is identified; together with a "best inequality", this approach also furnishes an optimal solution to the group problem.

4. Solving the group problem with diamond gauge functions and a generator set E

4.1. *Gomory and diamond cuts* [1, 2, 3]

Let $E = \{0\}$, with $d_0 = 0$; the definition (17) and the expression (9) then become

$$\sum_{j=1}^{n} \sum_{i=1}^{d} \alpha_i \sigma_{ij} x_j \geq \alpha_0, \tag{18a}$$

where

$$\sigma_{ij} = \begin{cases} f_{ij}\bar{f}_{i0} & \text{if } 0 \leq f_{ij} \ (\leq f_{i0}), \\ \bar{f}_{ij} f_{i0} & \text{if } (-\bar{f}_{i0} \leq) \ f_{ij} \leq 0. \end{cases} \tag{18b}$$

This inequality (18) represents a diamond cut which can be shown (see [2]) to be a non-negative linear combination of the mixed-integer Gomory cuts

$$\sum_{j=1}^{n} \sigma_{ij} x_j \geq f_{i0}\bar{f}_{i0}$$

generated from the i^{th} row with coefficients $\alpha_i \geq 0$, $i = 1, \ldots, d$. (Recall $\sum_{i=1}^{d} \alpha_i f_{i0}\bar{f}_{i0} = \alpha_0$ (10b).)

4.2. *Bounds*

Suppose the parameters $\alpha_1, \ldots, \alpha_d$ are chosen to satisfy

$$\pi(f_j) = \sum_{i=1}^{d} \alpha_i \sigma_{ij} \leq \bar{c}_j,$$

where $\bar{c}_j \geq 0$ is the reduced cost associated with the non-basic variable x_j; in the LP tableau one has

$$z - z_B = \sum_{j=1}^{n} \bar{c}_j x_j \geq \sum_{j=1}^{n} \pi(f_j) x_j \geq \alpha_0,$$

that is, the parameter α_0 provides a *lower bound* $(z_B + \alpha_0)$ for the answer to the group minimization problem (7).

Thus by defining a process (see subsection 4.3) which gradually raises the value of α_0 above its initial level $\alpha_0 = 0$, one generates bigger (i.e., better) bounds for the group problem; note that $(z_B + \alpha_0)$ is also a bound for the original integer program (1). Eventually the best bound (i.e., the

optimal value for the group problem) can be found, along with the optimal solution to (7).

4.3. *A process to solve the group problem*

In order to increase the numerical value of α_0 while maintaining subadditivity of the function π, one has to build up a generator set E, adjoining step by step some new additional points to the trivial generator set $E = \{0\}$.

Since the basic idea in this process is to increase the value $\pi(f_0) = \alpha_0$, one verifies that the value $\pi(u)$ also increases (in the weak sense) for all $u \in U$.

But such modification of the entire function π on U is constrained by the two following simultaneous conditions:

(i) Subadditivity of π must be maintained.

(ii) $\pi(f_0) = \pi(\sum_{j=1}^{n} f_j x_j) \leq \sum_{j=1}^{n} \bar{c}_j x_j$ must hold true so that α_0 always represents a lower bound.

Clearly as α_0 is initially increased (from $\alpha_0 = 0$) the function π remains subadditive (it keeps the same conical shape); for some value $\alpha_0 = \tilde{\alpha}_0$, the constraint (ii) will then become active, for some $j = \tilde{j}$ (see Fig. 1), i.e., we must have $\pi(f_{\tilde{j}}) = \bar{c}_{\tilde{j}}$ for all $\alpha_0 \geq \tilde{\alpha}_0$; this means that a further increase of α_0 will destroy the conical shape of the function π: testing subadditivity now becomes a non-trivial task; at this stage one has the generator set $E = \{0, f_{\tilde{j}}\}$ with $d_{\tilde{j}} = \bar{c}_{\tilde{j}}$. Examination of the definition (17) convinces one that the function π now "consists of two cones" (see Fig. 1) because one has

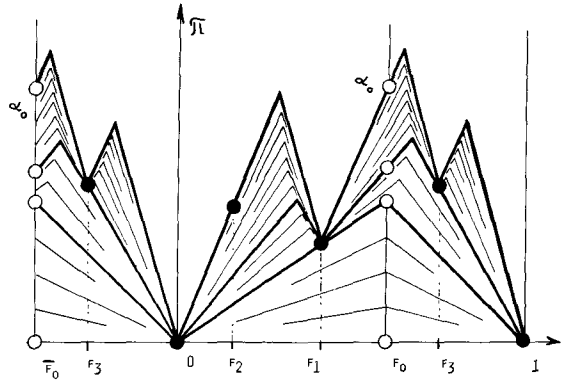

Fig. 1. A one-dimensional illustration of the process defined by raising the value of α_0, and of the definition of the function π as the minimum of several diamond functions D.

to take the diamond function (translated gauge) $\bar{c}_j + D(u - f_j)$ into account in the $\min_{e \in E}$ calculation (17); and as new elements are introduced into the generator set E, more cones need be considered.

In order to organize the verification of the constraints (i) and (ii), as this construction develops, we set up two lists E and F:

F is a list of points in U which have been identified as those points where the constraint (ii) may become active.

E is the current generator set; it contains all the points where the constraint (ii) has become active while lifting the π function; originally one has $E = \{0\}$, then $E = \{0, f_j\}$, ..., and so on until $E = \{0, f_j, \ldots, f_0\}$.

Every element of E and F is a *non-negative integer combination* (taken modulo 1) of the original f_j, $j = 1, \ldots, n$, i.e., for all $e \in E$,

$$e = \mathscr{F}\left(\sum_{j=1}^{n} \lambda_j^e f_j\right), \qquad \lambda_j^e \geq 0 \text{ integer} \tag{19a}$$

with

$$\pi(e) = d_e = \sum_{j=1}^{n} \lambda_j^e \bar{c}_j \geq 0. \tag{19b}$$

($\mathscr{F}$ is the fractional part operator (modulo 1) which maps $\mathbf{R}^d$ onto U.) From Theorem 3.11, we see that every $f \in F$ is the sum of two elements e, e' in E; thus one has

$$\lambda_j^f = \lambda_j^e + \lambda_j^{e'}, \qquad j = 1, \ldots, n,$$

$$d_f = d_e + d_{e'},$$

$$f = \mathscr{F}\left(\sum_{j=1}^{n} \lambda_j^f f_j\right).$$

4.4. *Remarks*

(1) Termination of this process (defined by increasing $\alpha_0 = \pi(f_0)$) will occur when the element f_0 must be included into the generator set E; indeed this means that *a least cost non-negative integer combination*

$$f_0 = \mathscr{F}\left(\sum_{j=1}^{n} \lambda_j^{f_0} f_j\right), \qquad \lambda_j^{f_0} \geq 0, \text{ integer}$$

with

$$\pi(f_0) = \alpha_0 = \sum_{j=1}^{n} \lambda_j^{f_0} \bar{c}_j$$

has been uncovered, exhibiting an optimal solution to the group problem (7). Since $f_0 \in E$, the value of $\alpha_0 = \pi(f_0)$ can no longer be increased without cutting off this optimal solution.

Note that nowhere in this process do we need any information on the group structure other than the fact that the left-hand side of the congruence (7b) generates all group elements.

Note also that the order Δ of the group (recall that $\Delta = \det B$) does not intervene explicitly in the computations.

Finally, it should be clear that since both lists E and F only contain distinct group elements, the finiteness of Δ implies finite termination of the process.

(2) From Theorem 3.11 one may be led to believe that the set F must contain the points $(e + e')$ corresponding to all pairs (e, e') of elements e and e' in the generator set E. Fortunately, Theorem 4.3 below indicates that only a "small" subset of these pairs need be considered.

Definition 4.1. A pair (e, f_{j_0}) of elements in the generator set E, with

$$e = \sum_j \lambda_j f_j, \qquad \lambda_j \geq 0 \text{ integer} \tag{19}$$

is called *extremal* if

(a) $(e + f_{j_0}) \notin E$,

(b) for all other pairs $(e' = \sum_j \lambda'_j f_j, f_{j'_0})$ such that $(e' + f'_{j'_0}) \notin E$, one has

$$\lambda_j \leq \lambda'_j \quad \text{for all } j \neq j_0, j'_0,$$

$$\lambda_{j_0} + 1 \leq \lambda'_{j_0}, \qquad \lambda_{j'_0} \leq \lambda'_{j'_0} + 1.$$

Lemma 4.2. *Let π be subadditive on U; if there exist two elements $e_1 = \sum_j \lambda^1_j f_j$ and $e_2 = \sum_j \lambda^2_j f_j$ in E such that*

$$\pi(e_1) + \pi(e_2) = \pi(e_1 + e_2), \tag{20}$$

then

$$\pi(e) + \pi(f_{j_0}) = \pi(e + f_{j_0}) \tag{21}$$

holds true for any $e = \sum_j \lambda_j f_j$ and f_{j_0} such that

$$0 \leq \lambda_j \leq \lambda^1_j + \lambda^2_j \quad \text{for all } j \neq j_0, \tag{22a}$$

$$0 \leq \lambda_{j_0} \leq \lambda^1_{j_0} + \lambda^2_{j_0} - 1. \tag{22b}$$

Proof. Consider the following list of inequalities due to the subadditivity of π on U.

$$\begin{aligned}
&\pi(f_1) && + \pi(f_1) \geq \pi(2f_1),\\
&\pi(2f_1) && + \pi(f_1) \geq \pi(3f_1),\\
&\quad\vdots && \quad\vdots \quad\quad \vdots\\
&\pi(\lambda_1 f_1) && + \pi(f_2) \geq \pi(\lambda_1 f_1 + f_2),\\
&\quad\vdots && \quad\vdots \quad\quad \vdots\\
&\pi(e) && + \pi(f_{j_0}) \geq \pi(e + f_{j_0}),\\
&\quad\vdots && \quad\vdots \quad\quad \vdots\\
&\pi(e_1 + e_2 - f_n) && + \pi(f_n) \geq \pi(e_1 + e_2).
\end{aligned} \tag{23}$$

The condition (22b) guarantees that the inequality (23) will appear in the above list. Summing all inequalities, we obtain

$$\sum_j (\lambda_j^1 + \lambda_j^2)\, \pi(f_j) \geq \pi(e_1 + e_2),$$

because the right-hand side term of each inequality cancels out with the first term on the left-hand side of the next inequality. But since e_1 and e_2 belong to E, we know (from (19b)) that

$$\pi(e_1) = \sum_j \lambda_j^1 \pi(f_j),$$

and similarly for e_2; hence

$$\pi(e_1) + \pi(e_2) = \sum_j (\lambda_j^1 + \lambda_j^2)\, \pi(f_j) \geq \pi(e_1 + e_2).$$

By hypothesis of Lemma 3.10, however, we know that equality holds in the latter inequality; therefore, equality must also hold for each inequality in the list, in particular for (23).

Remark. All points in E (and F) can be expressed by a non-negative (integer) combination of the original f_j, $j = 1, \ldots, n$,

$$e = \sum_{j \in N} \lambda_j f_j, \qquad \lambda_j \geq 0, \quad \lambda_j \equiv 0 \ (\mathrm{mod}\ 1).$$

This expression can be interpreted as a *path* from the origin to the point e using each f_j, λ_j times; a subpath is then defined by $\sum_{j \in N} \lambda'_j f_j = e'$ with $\lambda'_j \leq \lambda_j$ for all j. Definition 4.1 can now be seen to state that a pair (e, f_{j_0}) is extremal iff the point $f = (e + f_{j_0}) \notin E$, while *every subpath* f' of f belongs to E (i.e., $f' \in E$). But for every pair $e_1, e_2 \in E$ such that the point $f = (e_1 + e_2) \notin E$, one has a subpath f' with $f' = (e + f_{j_0})$, where (e, f_{j_0}) is extremal.

The condition of Theorem 3.11 can be relaxed to include only extremal pairs as stated below.

Theorem 4.3. *The function π is subadditive on U if*

$$\pi(e) + \pi(f_{j_0}) \geq \pi(e + f_{j_0})$$

for all extremal pairs (e, f_{j_0}), $e \in E, f_{j_0} \in E$.

Proof (by contradiction). Suppose that, for some value $\tilde{\alpha}_0$ of the parameter α_0, one has

$$\tilde{\pi}(e_1) + \tilde{\pi}(e_2) < \tilde{\pi}(e_1 + e_2)$$

for some e_1, e_2 in E. ($\tilde{\pi}$ denotes the π-function corresponding to the value $\tilde{\alpha}_0$ in (10b), (11a) and (17)). By construction of the function $\tilde{\pi}$ we know that if $(e_1 + e_2) \in E$, then we must have $\tilde{\pi}(e_1) + \tilde{\pi}(e_2) = \tilde{\pi}(e_1 + e_2)$; hence we can assume $(e_1 + e_2) \notin E$.

Now, at the level $\alpha_0 = 0$, Theorem 4.3 trivially holds true; thus, by gradually increasing the value of α_0 we will eventually find a value $\bar{\alpha}_0$ and a pair (e_1, e_2) such that:

(i) $0 \leq \bar{\alpha}_0 < \tilde{\alpha}_0$,

(ii) $\bar{\pi}$ is still subadditive on U,

(iii) $\bar{\pi}(e_1) + \bar{\pi}(e_2) = d_{e_1} + d_{e_2} = \bar{\pi}(e_1 + e_2)$;
$d_{e_1} + d_{e_2} < \tilde{\pi}(e_1 + e_2)$.

Thus, we can apply Lemma 4.2 to the function $\bar{\pi}$ to obtain

$$\bar{\pi}(e) + \bar{\pi}(f_{j_0}) = \bar{\pi}(e + f_{j_0})$$

for all pairs (e, f_{j_0}) satisfying (22). Furthermore, for the function $\tilde{\pi}$, we must have

$$\tilde{\pi}(e) + \pi(f_{j_0}) < \tilde{\pi}(e + f_{j_0}) \tag{24}$$

for at least one of these pairs, otherwise we would have

$$d_{e_1} + d_{e_2} = \tilde{\pi}(e_1) + \tilde{\pi}(e_2) \geq \tilde{\pi}(e_1 + e_2)$$

which is contrary to our assumption (iii).

We now show that, among the pairs which satisfy (24), there exists an extremal one, i.e., there exist

$$e \in E \quad \text{and} \quad f_{j_0} \in E \tag{25a}$$

such that

$$(e + f_{j_0}) \notin E \tag{25b}$$

while

$$\tilde{\pi}(e) + \tilde{\pi}(f_{j_0}) < \tilde{\pi}(e + f_{j_0}). \tag{24}$$

For this purpose we consider an index j_1 such that $\lambda^1_{j_1} > 0$ (this implies that $f_{j_1} \in E$) and set $e = f_{j_1}$. Two cases can arise:

(i) Either (25) holds true for some j_0 such that $\lambda^1_{j_0} + \lambda^2_{j_0} \geq 1$ and we have found an extremal pair satisfying (24) because

$$\tilde{\pi}(e) + \tilde{\pi}(f_{j_0}) = \bar{\pi}(e) + \bar{\pi}(f_{j_0}) = \bar{\pi}(e + f_{j_0}) < \tilde{\pi}(e + f_{j_0})$$

since ($\tilde{\pi}(u) > \bar{\pi}(u)$ for all $u \in U - \{E\}$, by construction).

(ii) Else one has

$$\tilde{\pi}(f_{j_1}) + \tilde{\pi}(f_{j_0}) = \tilde{\pi}(f_{j_1} + f_{j_0})$$

for all j_0 such that $\lambda^1_{j_0} + \lambda^2_{j_0} \geq 1$; since by construction $\tilde{\pi}(u) > \bar{\pi}(u)$ for all $u \in U - \{E\}$, the point $(f_{j_1} + f_{j_0})$ must then belong to E, and we may choose $e = f_{j_1} + f_{j_0}$.

This can be repeated until an extremal pair (e, f_{j_0}) which satisfies (24) is finally identified; such a pair must exist, for otherwise we would ultimately reach $(e_1 + e_2)$ with equality $\tilde{\pi}(e_1) + \tilde{\pi}(e_2) = \tilde{\pi}(e_1 + e_2)$ which is contrary to hypothesis.

But, now the identified extremal pair satisfies (24) which contradicts the hypothesis of the theorem. Thus, our assumption must be rejected and we have shown

$$\pi(e_1) + \pi(e_2) = \pi(e_1 + e_2) \quad \text{for all } e_1, e_2 \in E.$$

The proof can be completed by applying Theorem 3.11.

(3) Nothing yet has been said concerning the (free) choice of the parameters α_i, $i = 1, \ldots, d$; our theory, (in particular Theorems 3.11 and 4.3) applies to a fixed set of values α_i. However, it can readily be seen that, for a given generator set E (i.e., at any stage the computations) these values α_i can be changed, provided the following conditions remain satisfied:

(a) the new values α_i satisfy (10),

(b) the new function π, corresponding to the new values α_i, remains subadditive on all extremal pairs in E.

Indeed, in this case, the new function π is again subadditive on U (Theorem 4.3).

In practice, one would be interested in finding the values α_i which produce the best function π, i.e., the function π which dominates uniformly all the other ones; this is obtained by solving the optimization problem: Find the values α_i which

$$\text{maximize} \quad z(\alpha) = \min_{e \in E} \{d_e + D(f_0 - e)\}, \tag{26a}$$

$$\text{subject to} \quad d_{f_{j_0}} \geq \min_{e \in E} \{d_e + D(f_{j_0} - e)\} \tag{26b}$$

for all extremal pairs e, f_{j_0} in E.

This problem (26) is generally non-convex so that the determination of a (globally) optimal set of values α_i may be impractical. One should remember, however, that this optimal solution to (26) is not *required* and that any feasible solution can be used. In practice, one will conveniently choose a "good feasible set of values α_i" (typically a local optimum of (26) which can be achieved by linear programming).

The use of such a freedom of choice for the parameters α_i in an algorithmic implementation is based on the *hope* that "good" values will lead to an earlier termination of the process defined by increasing the α_0 value; as can be seen in (26b) the meaning of the word "good" here depends on the set E (and in particular on its extremal pairs) so that it changes with every iteration where the set E is augmented. This unfortunate situation forbids a simple evaluation of the "long range effect" of a particular set of values α_i. This may be considered as an open area for applied research; it is related to the old question of choosing the "best" row to generate a Gomory cut.

Appendix

Example. We briefly outline the first steps of the process described in Section 4.3.

Consider the group problem

$$\begin{aligned} \text{minimize} \quad & x_1 + \tfrac{3}{2}x_2, \\ \text{subject to} \quad & \tfrac{1}{4}x_1 + \tfrac{1}{2}x_2 = \tfrac{3}{4}, \\ & -\tfrac{1}{2}x_1 + \tfrac{1}{6}x_2 \equiv \tfrac{1}{3}, \\ & x_1, x_2 \equiv 0 \,(\text{mod } 1). \end{aligned}$$

The general from of the diamond gauge reads

$$D(u_1, u_2) = \alpha_1 \begin{Bmatrix} \frac{1}{4} \\ -\frac{3}{4} \end{Bmatrix} u_1 + \alpha_2 \begin{Bmatrix} \frac{2}{3} \\ -\frac{1}{3} \end{Bmatrix} u_2$$

for $u \in U = \{u: -\frac{1}{4} \leq u_1 \leq \frac{3}{4}, -\frac{2}{3} \leq u_2 \leq \frac{1}{3}\}$.

The *Gomory (mixed integer) cuts* are obtained as follows:

(1) $\alpha_1 = 12, \alpha_2 = 0$; $E = \{e_0 = 0\}$; $F = \{f_1 = (\frac{1}{4}, -\frac{1}{2}), f_2 = (\frac{1}{2}, \frac{1}{6})\}$,

$$D(u) = 12\begin{Bmatrix} \frac{1}{4} \\ -\frac{3}{4} \end{Bmatrix} u_1.$$

Thus one has

$$\tfrac{3}{4}x_1 + \tfrac{3}{2}x_2 \geq \tfrac{7}{4} = \alpha_0,$$

where α_0 is the bound delivered by this Gomory cut.

(2) $\alpha_1 = 0, \alpha_2 = 6$; E and F as above.

The cut reads

$$x_1 + \tfrac{2}{3}x_2 \geq \tfrac{4}{3}.$$

If one keeps E and F as above, the bound $\alpha_0 = \max\{\frac{7}{4}, \frac{4}{3}\} = \frac{7}{4}$ can be improved by modifying the α parameters; an optimal value in this case is the following.

Diamond cut ($\alpha_1 = 10, \alpha_2 = \frac{9}{4}$):

$$x_1 + \tfrac{3}{2}x_2 \geq \tfrac{19}{8}.$$

The process now consists in extending the E set, where one has to include the following e points for which the constraints (ii) become active:

$$e_1 = f_1 = (\tfrac{1}{4}, -\tfrac{1}{2}), \qquad e_2 = f_2 = (\tfrac{1}{2}, \tfrac{1}{6}).$$

For the set F we must now consider the following points:

$$2f_1 = (\tfrac{1}{2}, 0), \qquad 2f_2 = (0, \tfrac{1}{3}),$$

$$f_1 + f_2 = (-\tfrac{1}{4}, -\tfrac{1}{3}).$$

Since both f_1 and f_2 belong to E, the coefficients of the (subadditive) cut are known to remain:

$$\pi(f_1) = d_1 = 1, \qquad \pi(f_2) = d_2 = \tfrac{3}{2}.$$

Thus one only need compute $\pi(f_0)$ to determine the new right-hand side $\pi_0 = \alpha_0$ (which is also a new bound). One has

$$E = \{e_0 = 0, e_1 = f_1, e_2 = f_2\},$$

$$d_0 = 0, \qquad d_1 = 1, \qquad d_2 = \tfrac{3}{2},$$

$$F = \{2f_1, 2f_2, f_1 + f_2\},$$

$$\pi(f_0) = \pi(\tfrac{3}{4}, \tfrac{1}{3}) = \min_{e \in E} \{d_e + D(u - e)\}.$$

In this phase, we seek to improve the bound by increasing α_1 and α_2; for this illustrative example, we set

$$\alpha_1 = 10q, \qquad \alpha_2 = \tfrac{9}{4}q$$

and the parameter q is now increased until the constraint (ii) of Section 4.3 becomes active for one of the three points in F.

Hence q must satisfy

$$\min\{D_q(2f_1),\ 1 + D_q(2f_1),\ \tfrac{3}{2} + D_q(2f_1)\} = 2,$$

where the right-hand side 2 is the cost of the point $2f_1$, i.e.,

$$\min\{\tfrac{5}{4}q,\ 1 + q,\ \tfrac{3}{2} + \tfrac{1}{8}q\} = 2$$

which can be satisfied by the value $q = 4$.

Similarly one has

$$\min\{D_q(2f_2),\ 1 + D_q(2f_2),\ \tfrac{3}{2} + D_q(2f_2)\} = 3,$$

i.e.,

$$\min\{\tfrac{1}{2}q,\ 1 + 2q,\ \tfrac{3}{2} + \tfrac{3}{2}q\} = 3$$

which gives $q = 6$; and finally,

$$\min\{D_q(f_1 + f_2),\ 1 + D_q(f_1 + f_2),\ \tfrac{3}{2} + D_q(f_1 + f_2)\} = 2\tfrac{1}{2},$$

i.e.,

$$\min\{\tfrac{17}{8}q,\ 1 + \tfrac{3}{2}q,\ \tfrac{3}{2} + q\} = 2\tfrac{1}{2}$$

and therefore $q = \frac{20}{17}$.

Now one should take the smallest among the above values of q, in order to obtain a function π which does not violate subadditivity at any element of F, i.e., $q = \frac{20}{17}$. The resulting bound is $\alpha_0 = \pi(f_0) = 2\frac{9}{17}$.

For simplicity we do not attempt here to improve this bound by (local) optimization of the cut with respect to α_1, α_2; instead we will augment the generator set E and thereby increase the value of q. The points where the constraint (ii) is active must be introduced into E, i.e.,

$$e_3 = f_1 + f_2 = (-\tfrac{1}{4}, -\tfrac{1}{3}), \qquad d_3 = 2\tfrac{1}{2}.$$

The F-list must also be augmented accordingly:

$$F = \{2f_1, 2f_2, 2f_1 + f_2, f_1 + 2f_2, 2(f_1 + f_2)\}.$$

Here we shall use Theorem 4.3 to reduce the number of pairs to be checked.

The *extremal* pairs are

$$(e_1, f_1) \quad \text{and} \quad (e_2, f_2).$$

But we already have computed the value q for these pairs, namely

$$(e_1, f_1)\colon q = 4; \qquad (e_2, f_2)\colon q = 6.$$

Thus we may now set $q = \min\{4, 6\} = 4$ and compute the bound

$$\pi(f_0) = \alpha_0 = \min\{4(\tfrac{19}{8}),\ 1 + 4(\tfrac{11}{8}),\ \tfrac{3}{2} + 4(\tfrac{7}{8})\} = 5.$$

One sees that extending the generator set E has allowed a bound improvement from the diamond cut ($\alpha_0 = \frac{19}{8}$) to $\alpha_0 = 5$; i.e., one has the valid inequality

$$x_1 + \tfrac{3}{2}x_2 \geq 5.$$

The optimal solution of the group problem is ($x_1 = 1$, $x_2 = 5$) with objective value $8\frac{1}{2}$; and 5 can be considered a "good" bound.

References

[1] C.A. Burdet, "Some enumerative inequalities and their application in integer programming algorithms", *Mathematical Programming* 2 (1972) 32–64.

[2] C.A. Burdet, "Enumerative cuts I", *Operations Research* 21 (1) (1973) 61–89.

[3] R.E. Gomory, "An algorithm for integer solutions to linear programs", in: R.L. Graves and P. Wolfe, eds., *Recent advances in mathematical programming* (McGraw-Hill, New York, 1963) 269–302.

[4] R.E. Gomory and E.L. Johnson, "Some continuous functions related to convex polyhedra, II", *Mathematical Programming* 3 (1972) 359–389.

[5] E.L. Johnson, "Cyclic groups, cutting planes and shortest paths", in: T.C. Hu and S. Robinson, eds., *Mathematical programming* (Academic Press, New York, 1973).

[6] E.L. Johnson, "A group problem for mixed integer programs", IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y. (1972).

[7] R.T. Rockafellar, "*Convex analysis*" (Princeton University Press, Princeton, N.J., 1970).

Mathematical Programming Study 2 (1974) 72–81. North-Holland Publishing Company

TWO COMPUTATIONALLY DIFFICULT SET COVERING PROBLEMS THAT ARISE IN COMPUTING THE 1-WIDTH OF INCIDENCE MATRICES OF STEINER TRIPLE SYSTEMS*

D.R. FULKERSON, G.L. NEMHAUSER

Cornell University, Ithaca, N.Y., U.S.A.

L.E. TROTTER, Jr.**

Mathematics Research Center, University of Wisconsin

Received 7 November 1973

Two minimum cardinality set covering problems of similar structure are presented as difficult test problems for evaluating the computational efficiency of integer programming and set covering algorithms. The smaller problem has 117 constraints and 27 variables, and the larger one, constructed by H.J. Ryser, has 330 constraints and 45 variables. The constraint matrices of the two set covering problems are incidence matrices of Steiner triple systems. An optimal solution to the problem that we were able to solve (the smaller one) gives some new information on the 1-widths of members of this class of (0,1)-matrices.

1. Introduction

The purpose of this note is two-fold. First, we supply data for two integer programming (set covering) problems which we believe are computationally hard. Our experience indicates that optimal solutions to these problems are very tedious to compute and verify, even though they have far fewer variables than numerous solved problems in the literature. Thus these problems seem to be appropriate difficult test problems for evaluating the computational efficiency of integer programming and set covering algorithms.

* This research was supported, in part, by National Science Foundation Grants GK-32282X and GP-32316X and Office of Naval Research Contract No. N00014-67-A-0077-0028 to Cornell University. A substantial amount of the computer time was provided by the Mathematics Research Center of the University of Wisconsin under U.S. Army Contract No. DA-31-124-ARO-D-462.

** Currently at Yale University, New Haven, Conn., U.S.A.

The constraint matrices of the two set covering problems are incidence matrices of Steiner triple systems. An optimal solution to the problem that we were able to solve (the smaller one) gives some new information on the 1-widths of members of this particular class of (0,1)-matrices. This new information is the second purpose of this note.

2. Origin of the problems

The α-width of a (0,1)-matrix A is the minimum number of columns that can be selected from A such that all row sums of the resulting submatrix of A are at least α. Here α is an integer parameter ranging from zero to the smallest row sum of the matrix A. This notion was introduced and studied by Fulkerson and Ryser in a series of papers [3, 4, 5]. Henceforth we restrict attention to the case $\alpha = 1$ and denote the 1-width of an $m \times n$ (0,1)-matrix A having no zero rows by $w(A)$. The integer $w(A)$ can thus be determined from an optimal solution to the set covering problem

$$\begin{aligned} &w(A) = \min 1_n x, \\ &A x \geq 1_m, \qquad x \geq 0 \text{ and integral.} \end{aligned} \tag{2.1}$$

Here 1_k is the k-vector all of whose components are 1, and we are viewing A as the incidence matrix of m elements (rows) versus n subsets (columns) of the m-set. (Alternatively, and this point of view is perhaps more appropriate for the discussion to follow, we can view (2.1) as the problem of determining the least number of elements of an n-set required to represent all members of a family of m subsets of the n-set.)

A Steiner triple system on n elements is a pair $(S, \mathcal{T})$, where $S = \{1, 2, \ldots, n\}$ and $\mathcal{T} = (S_1, \ldots, S_m)$ is a family of triples (subsets of S of cardinality 3) such that every pair of elements of S (every subset of S of cardinality 2) is contained in precisely one of the triples. It is well known that Steiner triple systems exist if and only if $n \geq 3$ and $n \equiv 1,3 \pmod 6$, in which case $m = n(n-1)/6$, and each element of S appears in precisely $\frac{1}{2}(n-1)$ of the triples. (Steiner triple systems are particular cases of combinatorial configurations known as balanced incomplete block designs, which are used in the statistical design of experiments. For example, if n drugs are to be tested on m patients, and each patient is to be given three drugs, a Steiner triple system provides a design in which each pair of drugs is tested on one patient.)

The incidence matrix $A = (a_{ij})$ of a Steiner triple system is a (0,1)-matrix whose rows correspond to the triples and whose columns correspond to

the elements of S. Thus $a_{ij} = 1$ if and only if $j \in S_i$. Corresponding to each of the parameters $n = 3, 7, 9$ there is a unique Steiner triple system; for $n = 13$ there are two distinct systems, and for $n = 15$, there are eighty distinct systems [14]. The unique system for $n = 9$ is given by the 12×9 incidence matrix

$$A_9 = \begin{bmatrix} Z & I & 0 \\ 0 & Z & I \\ I & 0 & Z \\ I & I & I \end{bmatrix}, \tag{2.2}$$

where 0 is the zero matrix of order 3, I is the identity matrix of order 3, and

$$Z = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

For the system (2.2) we have $w(A_9) = 5$, since the rows of A_9 are covered by its first five columns, but by no smaller set of columns. Both systems corresponding to $n = 13$ have 1-width 7; for $n = 15$, the 1-width varies from 7 to 9 [5]. Indeed, in their study of 1-widths of Steiner triple systems, Fulkerson and Ryser [5] derived the lower bound

$$w(A) \geq \tfrac{1}{2}(n - 1) \tag{2.3}$$

and determined conditions under which equality holds in (2.3). They remarked, however, that good upper bounds seem difficult to obtain. Since all triple systems on 15 or fewer elements have 1-widths at most $\frac{2}{3}n - 1$, they at one time speculated that this might be an upper bound. Ryser subsequently constructed a 45 element system as a potential counter-example, and conjectured that it had 1-width 30. Ryser's construction of the 45-element system starts with a particular 15-element system that has 1-width 9. This 15-element system has the 35×15 incidence matrix

$$A_{15} = \begin{bmatrix} Z & E & 0 \\ 0 & Z & E \\ E & 0 & Z \\ I & I & I \end{bmatrix}, \tag{2.4}$$

where

$$Z = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}, \qquad E = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

and I is the 5×5 identity matrix. He then constructs a 45-element system having a 330×45 incidence matrix of the following form:

$$A_{45} = \begin{bmatrix} A_{15} & 0 & 0 \\ 0 & A_{15} & 0 \\ 0 & 0 & A_{15} \\ C^1 & I & P^1 \\ \vdots & \vdots & \vdots \\ C^{15} & I & P^{15} \end{bmatrix} \tag{2.5}$$

In (2.5), I is the 15×15 identity matrix; for $k = 1, \ldots, 15$, C^k is a 15×15 matrix whose k^{th} column contains all ones and whose other columns contain all zeros, and P^k is a 15×15 permutation matrix with $\sum_{k=1}^{15} P^k = J$, where J is the matrix of all ones.

We attempted to determine $w(A_{45})$ by solving the set covering problem (2.1), but did not succeed. We then used the matrix A_9 from (2.2) to construct a 27-element system structurally similar to A_{45}. The resulting 117×27 incidence matrix is

$$A_{27} = \begin{bmatrix} A_9 & 0 & 0 \\ 0 & A_9 & 0 \\ 0 & 0 & A_9 \\ \tilde{C}^1 & I & \tilde{P}^1 \\ \vdots & \vdots & \vdots \\ \tilde{C}^9 & I & \tilde{P}^9 \end{bmatrix}. \tag{2.6}$$

We solved (2.1) for A_{27} and found that $w(A_{27}) = 18$. Thus $\frac{2}{3}n - 1$ is not an upper bound on 1-widths of Steiner triple systems on n elements. The system (2.5), constructed by Ryser, has $w(A_{45}) \leq 30$, and we feel reasonably sure that Ryser's speculation that $w(A_{45}) = 30$ is correct.

More detailed descriptions of A_{27} and A_{45} are given in Table 1 and Table 2.

Table 1
The triples of A_{27}

(1)	2 3 4	(2)	1 3 5	(3)	1 2 6	(4)	5 6 7
(5)	4 6 8	(6)	4 5 9	(7)	1 8 9	(8)	2 7 9
(9)	3 7 8	(10)	1 4 7	(11)	2 5 8	(12)	3 6 9
(13)	11 12 13	(14)	10 12 14	(15)	10 11 15	(16)	14 15 16
(17)	13 15 17	(18)	13 14 18	(19)	10 17 18	(20)	11 16 18
(21)	12 16 17	(22)	10 13 16	(23)	11 14 17	(24)	12 15 18
(25)	20 21 22	(26)	19 21 23	(27)	19 20 24	(28)	23 24 25
(29)	22 24 26	(30)	22 23 27	(31)	19 26 27	(32)	20 25 27
(33)	21 25 26	(34)	19 22 25	(35)	20 23 26	(36)	21 24 27
(37)	1 10 19	(38)	1 11 24	(39)	1 12 23	(40)	1 13 25
(41)	1 14 21	(42)	1 15 20	(43)	1 16 22	(44)	1 17 27
(45)	1 18 26	(46)	2 10 24	(47)	2 11 20	(48)	2 12 22
(49)	2 13 21	(50)	2 14 26	(51)	2 15 19	(52)	2 16 27
(53)	2 17 23	(54)	2 18 25	(55)	3 10 23	(56)	3 11 22
(57)	3 12 21	(58)	3 13 20	(59)	3 14 19	(60)	3 15 27
(61)	3 16 26	(62)	3 17 25	(63)	3 18 24	(64)	4 10 25
(65)	4 11 21	(66)	4 12 20	(67)	4 13 22	(68)	4 14 27
(69)	4 15 26	(70)	4 16 19	(71)	4 17 24	(72)	4 18 23
(73)	5 10 21	(74)	5 11 26	(75)	5 12 19	(76)	5 13 27
(77)	5 14 23	(78)	5 15 25	(79)	5 16 24	(80)	5 17 20
(81)	5 18 22	(82)	6 10 20	(83)	6 11 19	(84)	6 12 27
(85)	6 13 26	(86)	6 14 25	(87)	6 15 24	(88)	6 16 23
(89)	6 17 22	(90)	6 18 21	(91)	7 10 22	(92)	7 11 27
(93)	7 12 26	(94)	7 13 19	(95)	7 14 24	(96)	7 15 23
(97)	7 16 25	(98)	7 17 21	(99)	7 18 20	(100)	8 10 27
(101)	8 11 23	(102)	8 12 25	(103)	8 13 24	(104)	8 14 20
(105)	8 15 22	(106)	8 16 21	(107)	8 17 26	(108)	8 18 19
(109)	9 10 26	(110)	9 11 25	(111)	9 12 24	(112)	9 13 23
(113)	9 14 22	(114)	9 15 21	(115)	9 16 20	(116)	9 17 19
(117)	9 18 27						

Table 2
The triples of A_{45}

(1)	3 4 6	(2)	4 5 7	(3)	1 5 8	(4)	1 2 9
(5)	2 3 10	(6)	2 5 6	(7)	1 3 7	(8)	2 4 8
(9)	3 5 9	(10)	1 4 10	(11)	8 9 11	(12)	9 10 12
(13)	6 10 13	(14)	6 7 14	(15)	7 8 15	(16)	7 10 11
(17)	6 8 12	(18)	7 9 13	(19)	8 10 14	(20)	6 9 15
(21)	1 13 14	(22)	2 14 15	(23)	3 11 15	(24)	4 11 12
(25)	5 12 13	(26)	1 12 15	(27)	2 11 13	(28)	3 12 14
(29)	4 13 15	(30)	5 11 14	(31)	1 6 11	(32)	2 7 12
(33)	3 8 13	(34)	4 9 14	(35)	5 10 15	(36)	18 19 21
(37)	19 20 22	(38)	16 20 23	(39)	16 17 24	(40)	17 18 25
(41)	17 20 21	(42)	16 18 22	(43)	17 19 23	(44)	18 20 24
(45)	16 19 25	(46)	23 24 26	(47)	24 25 27	(48)	21 25 28
(49)	21 22 29	(50)	22 23 30	(51)	22 25 26	(52)	21 23 27
(53)	22 24 28	(54)	23 25 29	(55)	21 24 30	(56)	16 28 29
(57)	17 29 30	(58)	18 26 30	(59)	19 26 27	(60)	20 27 28
(61)	16 27 30	(62)	17 26 28	(63)	18 27 29	(64)	19 28 30
(65)	20 26 29	(66)	16 21 26	(67)	17 22 27	(68)	18 23 28
(69)	18 24 29	(70)	20 25 30	(71)	33 34 36	(72)	34 35 37
(73)	31 35 38	(74)	21 32 39	(75)	32 33 40	(76)	32 35 36
(77)	31 33 37	(78)	32 34 38	(79)	33 35 39	(80)	31 34 40
(81)	38 39 41	(82)	39 40 42	(83)	36 40 43	(84)	36 37 44
(85)	37 38 45	(86)	37 40 41	(87)	36 38 42	(88)	37 39 43
(89)	38 40 44	(90)	36 39 45	(91)	31 43 44	(92)	32 44 45
(93)	33 41 45	(94)	34 41 42	(95)	35 42 43	(96)	31 42 45
(97)	32 41 43	(98)	33 42 44	(99)	34 43 45	(100)	35 41 44
(101)	31 36 41	(102)	32 37 42	(103)	33 38 43	(104)	34 39 44
(105)	35 40 45	(106)	1 16 31	(107)	1 17 39	(108)	1 18 37
(109)	1 19 40	(110)	1 20 38	(111)	1 21 41	(112)	1 22 33
(113)	1 23 35	(114)	1 24 32	(115)	1 25 34	(116)	1 26 36
(117)	1 27 45	(118)	1 28 44	(119)	1 29 43	(120)	1 30 42
(121)	2 16 39	(122)	2 17 32	(123)	2 18 40	(124)	2 19 38
(125)	2 20 36	(126)	2 21 35	(127)	2 22 42	(128)	2 23 34
(129)	2 24 31	(130)	2 25 33	(131)	2 26 43	(132)	2 27 37
(133)	2 28 41	(134)	2 29 45	(135)	2 30 44	(136)	3 16 37
(137)	3 17 40	(138)	3 18 33	(139)	3 19 36	(140)	3 20 39
(141)	3 21 34	(142)	3 22 31	(143)	3 23 43	(144)	3 24 35
(145)	3 25 32	(146)	3 26 45	(147)	3 27 44	(148)	3 28 38
(149)	3 29 42	(150)	3 30 41	(151)	4 16 40	(152)	4 17 38
(153)	4 18 36	(154)	4 19 34	(155)	4 20 37	(156)	4 21 33
(157)	4 22 35	(158)	4 23 32	(159)	4 24 44	(160)	4 25 31
(161)	4 26 42	(162)	4 27 41	(163)	4 28 45	(164)	4 29 39
(165)	4 30 43	(166)	5 16 38	(167)	5 17 36	(168)	5 18 39
(169)	5 19 37	(170)	5 20 35	(171)	5 21 32	(172)	5 22 34
(173)	5 23 31	(174)	5 24 33	(175)	5 25 45	(176)	5 26 44
(177)	5 27 43	(178)	5 28 42	(179)	5 29 41	(180)	5 30 40
(181)	6 16 41	(182)	6 17 35	(183)	6 18 34	(184)	6 19 33
(185)	6 20 32	(186)	6 21 36	(187)	6 22 44	(188)	6 23 42
(189)	6 24 45	(190)	6 25 43	(191)	6 26 31	(192)	6 27 38
(193)	6 28 40	(194)	6 29 37	(195)	6 30 39	(196)	7 16 33

Table 2 (continued)

(197)	7 17 42	(198)	7 18 31	(199)	7 19 35	(200)	7 20 34
(201)	7 21 44	(202)	7 22 37	(203)	7 23 45	(204)	7 24 43
(205)	7 25 41	(206)	7 26 40	(207)	7 27 32	(208)	7 28 39
(209)	7 29 36	(210)	7 30 38	(211)	8 16 35	(212)	8 17 34
(213)	8 18 43	(214)	8 19 32	(215)	8 20 31	(216)	8 21 42
(217)	8 22 45	(218)	8 23 38	(219)	8 24 41	(220)	8 25 44
(221)	8 26 39	(222)	8 27 36	(223)	8 28 33	(224)	8 29 40
(225)	8 30 37	(226)	9 16 32	(227)	9 17 31	(228)	9 18 35
(229)	9 19 44	(230)	9 20 33	(231)	9 21 45	(232)	9 22 43
(233)	9 23 41	(234)	9 24 39	(235)	9 25 42	(236)	9 26 38
(237)	9 27 40	(238)	9 28 37	(239)	9 29 34	(240)	9 30 36
(241)	10 16 34	(242)	10 17 33	(243)	10 18 32	(244)	10 19 31
(245)	10 20 45	(246)	10 21 43	(247)	10 22 41	(248)	10 23 44
(249)	10 24 42	(250)	10 25 40	(251)	10 26 37	(252)	10 27 39
(253)	10 28 36	(254)	10 29 38	(255)	10 30 35	(256)	11 16 36
(257)	11 17 43	(258)	11 18 45	(259)	11 19 42	(260)	11 20 44
(261)	11 21 31	(262)	11 22 40	(263)	11 23 39	(264)	11 24 38
(265)	11 25 37	(266)	11 26 41	(267)	11 27 34	(268)	11 28 32
(269)	11 29 35	(270)	11 30 33	(271)	12 16 45	(272)	12 17 37
(273)	12 18 44	(274)	12 19 41	(275)	12 20 43	(276)	12 21 38
(277)	12 22 32	(278)	12 23 36	(279)	12 24 40	(280)	12 25 39
(281)	12 26 34	(282)	12 27 42	(283)	12 28 35	(284)	12 29 33
(285)	12 30 31	(286)	13 16 44	(287)	13 17 41	(288)	13 18 38
(289)	13 19 45	(290)	13 20 42	(291)	13 21 40	(292)	13 22 39
(293)	13 23 33	(294)	13 24 37	(295)	13 25 36	(296)	13 26 32
(297)	13 27 35	(298)	13 28 43	(299)	13 29 31	(300)	13 30 34
(301)	14 16 43	(302)	14 17 45	(303)	14 18 42	(304)	14 19 39
(305)	14 20 41	(306)	14 21 37	(307)	14 22 36	(308)	14 23 40
(309)	14 24 34	(310)	14 25 38	(311)	14 26 35	(312)	14 27 33
(313)	14 22 31	(314)	14 29 44	(315)	14 30 32	(316)	15 16 42
(317)	15 17 44	(318)	15 18 41	(319)	15 19 43	(320)	15 20 40
(321)	15 21 39	(322)	15 22 38	(323)	15 23 37	(324)	15 24 36
(325)	15 25 25	(326)	15 26 33	(327)	15 27 31	(328)	15 28 34
(329)	15 29 32	(330)	15 30 45				

3. Computational experience

An optimal linear programming solution in both problems is, of course, to set all variables equal to $\frac{1}{3}$, giving values of 9 and 15, respectively, for the sum of variables in the two problems. Optimal linear programming bases are not unique for A_{27} and A_{45}; the computer produced optimal bases with determinants of magnitude 27^3 and 45^3, respectively. An optimal solution to (2.1) with $A = A_{27}$ is given by columns 1, 2, 3, 4, 5, 6, 10, 11, 12, 13, 14, 15, 19, 20, 21, 25, 26 and 27.

Both problems were attempted by an implicit enumeration algorithm

[17] and a cutting plane algorithm based on Gomory's method of integer forms [7]. All computing was done on a Univac 1108. We also sent A_{45} to several people who have obtained computational experience with integer programming codes different from those available to us. Of particular interest would be results obtained from codes based on the group-theoretic approach to integer programming [2, 8, 9, 10],[1] and codes based on specialized algorithms for set covering problems [1, 11, 13, 16]. Nobody has informed us that they have solved (2.1) for A_{45}. There were a few negative responses. These generally did not report unequivocal failure, but difficulty in treating a problem of this size with an experimental code, etc.

Using the cutting plane code, we solved (2.1) for $A = A_9$ in about 20 seconds after adding 44 cuts. The cutting plane code was unsuccessful on the larger triple systems A_{15}, A_{27} and A_{45}. Typically, the first few iterations yielded cuts that caused an increase in the objective value, but this was followed by long sequences of cuts that failed to produce a significant change in the value of the objective function. For example, in treating A_{15}, the initial sequence of 19 cuts raised the objective value from 5.0 to 5.425, after which 32 cuts left the value unaltered. (Recall that $w(A_{15}) = 9$.) This particular run took 82 seconds. In contrast, this problem was solved in 5 seconds using implicit enumeration. This enumeration algorithm is very similar to one developed and tested by Geoffrion [6]. It uses simple tests for infeasibilities, and uses linear programming relaxations to obtain lower bounds and thus a necessary condition on whether a given partial solution to (2.1) has a feasible completion of smaller value than the best available solution. The linear programming solutions also generate surrogate constraints. A dual algorithm is used to solve the linear programs. When fathoming occurs, the next partial solution considered is determined by backtracking. Branching is done by fixing a free variable at zero or one.

In our successful run with A_{27} we began with the partial solution corresponding to the optimal solution, which was found by inspection. About 6000 partial solutions were considered before optimality was verified. The run consumed about 16 minutes of computer time. Most of the fathoming came from the LP bounds, but this bounding test was quite ineffective for partial solutions having a small number of fixed variables. Several attempts

[1] Shapiro [15] attempted to solve (2.1) with A_{27} using IPA [10]. The enumeration was abandoned after about 1 minute of computer time on the IBM 360/67. Optimal solutions found for the group problems were highly infeasible and little progress was being made towards achieving feasibility in the enumeration phase.

at solving the problem for A_{45} failed, although we began with what is probably an optimal solution.

In contrast with our experience on these problems, others have reported considerable success in solving set covering problems with a variety of algorithms [1, 6, 9, 10, 11, 13, 16]. For example, Geoffrion [6] has reported solving, with an algorithm essentially identical to our implicit enumeration algorithm, several set covering problems with $m = 30$ and $30 \leq n \leq 90$ in times varying from one to twelve seconds on the IBM 7044, a much slower computer. Lemke et al. [13] have solved considerably larger problems in a reasonable amount of time, e.g., a problem with $m = 50$, $n = 450$ was solved in about 2 minutes on an IBM 360/50. Gorry et al. [10] have solved some airline crew scheduling problems; one with $m = 313$ and $n = 482$ was solved in about 3 minutes on the IBM 360/85.

Compared to most of the covering problems considered in the literature, the problems A_{27} and A_{45} have a relatively small number of variables but a large number of constraints. Supposedly, however, of the two parameters, the number of variables is the more significant in solving a covering problem by implicit enumeration. Furthermore, the densities (number of ones/mn) of A_{27} and A_{45} are close to the densities in the problems considered in [6] and [13] mentioned above.

Why, then, are these two problems A_{27} and A_{45} difficult? We don't really know, but some plausible explanations might be:

(1) The symmetries in the problems no doubt tend to increase the amount of enumeration required.

(2) The rather large determinants of optimal LP bases contribute to the unattractiveness of cutting plane methods and indicate that group-theoretic methods may encounter difficulties.

(3) The optimal value in the integer problem is large compared to the optimal value in the real (or rational) problem, thus emasculating the power of linear programming.

Recently, Jeroslow [12] has constructed a simple family of n variable, (0, 1)-integer programs that cannot be solved by implicit enumeration, even using linear programming for fathoming, without enumerating at least $2^{n/2}$ possibilities. Although A_{27} and A_{45} are not in this family, they show that problems that arise naturally can be nasty. We hope that a reasonable set of such hard problems can be accumulated for the purpose of evaluating the efficiency of proposed integer programming algorithms.

References

[1] M. Bellmore and H.D. Ratliff, "Set covering and involutory bases", *Management Science* 18 (1971) 194–206.

[2] G.H. Bradley and P.N. Wahi, "An algorithm for integer linear programming: A combined algebraic and enumeration approach", *Operations Research* 21 (1973) 45–60.

[3] D.R. Fulkerson and H.J. Ryser, "Widths and heights of (0,1)-matrices", *Canadian Journal of Mathematics* 13 (1961) 239–255.

[4] D.R. Fulkerson and H.J. Ryser, "Multiplicities and minimal widths for (0,1)-matrices", *Canadian Journal of Mathematics* 14 (1962) 498–508.

[5] D.R. Fulkerson and H.J. Ryser, "Width sequences for special classes of (0,1)-matrices", *Canadian Journal of Mathematics* 15 (1963) 371–396.

[6] A.M. Geoffrion, "An improved implicit enumeration approach for integer programming", *Operations Research* 17 (1969) 437–454.

[7] R.E. Gomory, "An algorithm for integer solutions to linear programs", in: R.L. Graves and P. Wolfe, eds., *Recent advances in mathematical programming* (McGraw-Hill, New York, 1963) pp. 269–302.

[8] R.E. Gomory, "On the relation between integer and non-integer solutions to linear programs", *Proceedings of the National Academy of Sciences* 53 (1965) 260–265.

[9] G.A. Gorry and J.F. Shapiro, "An adaptive group theoretic algorithm for integer programming problems", *Management Science* 17 (1971) 285–306.

[10] G.A. Gorry, W.D. Northrup and J.F. Shapiro, "Computational experience with a group theoretic integer programming algorithm", *Mathematical Programming* 4 (1973) 171–192.

[11] R.W. House, L.D. Nelson and J. Rado, "Computer studies of a certain class of linear integer problems", in: A. Lavi and T. Vogl, eds., *Recent advances in optimization techniques* (Wiley, New York, 1966) pp. 241–280.

[12] R.G. Jeroslow, "Trivial integer programs unsolvable by branch-and-bound", *Mathematical Programming* 6 (1974) 105–109.

[13] C.E. Lemke, H.M. Salkin and K. Spielberg, "Set covering by single branch enumeration with linear programming subproblems", *Operations Research* 19 (1971) 998-1022.

[14] H.J. Ryser, *Combinatorial mathematics*, Carus Mathematical Monograph No. 14 (Wiley, New York, 1963).

[15] J.F. Shapiro, Private communication (September 1973).

[16] H. Thiriez, "The set covering problem: A group theoretic approach", *Revue Francaise d'Informatique et de Recherche Operationelle* V3 (1971) 83–104.

[17] L.E. Trotter, Jr. and C.M. Shetty, "An algorithm for the bounded variable integer programming problem", *Journal of the Association for Computing Machinery* 21 (1974) 505–513.

Mathematical Programming Study 2 (1974) 82–114. North-Holland Publishing Company

LAGRANGEAN RELAXATION FOR INTEGER PROGRAMMING *

A.M. GEOFFRION

University of California, Los Angeles, Calif., U.S.A.

Received 25 January 1974
Revised manuscript received 26 August 1974

Taking a set of "complicating" constraints of a general mixed integer program up into the objective function in a Lagrangean fashion (with fixed multipliers) yields a "Lagrangean relaxation" of the original program. This paper gives a systematic development of this simple bounding construct as a means of exploiting special problem structure. A general theory is developed and special emphasis is given to the application of Lagrangean relaxation in the context of LP-based branch-and-bound.

1. Introduction

The general integer linear programming problem can be written as

$$\text{(P)} \qquad \underset{x \geqq 0}{\text{minimize}} \; c\,x,$$
$$\text{subject to } A\,x \geqq b, \qquad B\,x \geqq d,$$
$$x_j \text{ integer}, \qquad j \in I,$$

where b, c and d are vectors, A and B are matrices of conformable dimensions, and the index set I denotes the variables required to be integer. The reason for distinguishing two types of constraints is that the second of these, $B\,x \geqq d$, is supposed to have special structure.

We define the *Lagrangean relaxation* of (P) relative to $A\,x \geqq b$ and a conformable nonnegative vector λ to be

$$(\text{PR}_\lambda) \qquad \underset{x \geqq 0}{\text{minimize}} \; c\,x + \lambda(b - A\,x),$$
$$\text{subject to } B\,x \geqq d,$$
$$x_j \text{ integer}, \qquad j \in I.$$

* An earlier version of this paper was presented at the IBM International Symposium on Discrete Optimization, Wildbad, Germany, October 30–November 1, 1972. This research was supported by the Office of Naval Research under Contract Number N00014-69-A-0200-4042 and by the National Science Foundation under Grants GP-26294 and GP-36090X.

The fruitful application of (PR_λ) in specific cases requires judicious partitioning of the constraints into the two types $A x \geqq b$ and $B x \geqq d$, and an appropriate choice of $\lambda \geqq 0$.

Lagrangean relaxation has been used by Held and Karp [20,21] in their highly successful work on the traveling-salesman problem; by Fisher [6] in his promising algorithm for scheduling in the presence of resource constraints and in his efficient machine scheduling algorithm [7]; by Fisher and Schrage [9] in their proposed algorithm for scheduling hospital admissions; by Ross and Soland [26] in their remarkably efficient algorithm for the generalized assignment problem; and by Shapiro [27] and Fisher and Shapiro [10] in the context of a group theoretic approach to pure integer programming. See also [8]. Other authors have also made special application of Lagrangean relaxation ideas implicitly if not explicitly in their work. Not to be forgotten is the general relevance of the literature on Lagrangean methods for nonconvex optimization (e.g., [2, 5, 18]).

The purpose of this paper is to develop the theory and explore the usefulness of Lagrangean relaxation in the context of branch-and-bound or implicit enumeration methods for (P). In contrast with most of the references just cited, our emphasis is on LP-based branch-and-bound algorithms rather than those whose bounding constructs do not involve linear programming. This is not to deny the great value of non-LP-based techniques for special problems, but rather to stress the as yet untapped potential of Lagrangean relaxation as a means of making the most widely used general purpose approach more efficient for problems with special structure. The development is intended for use at two levels. Pedagogically it strives for a unified exposition of a number of old and new developments in integer programming. As a research effort it aims to develop what appears to be a potent general approach to the design of improved algorithms for special classes of integer programs. Although the algorithmic context of this paper is the branch-and-bound approach to integer linear programs, it is clear that these ideas can also be applied to other classes of algorithms and problems.

The paper is organized as follows. The basic results concerning the relation between (P), (PR_λ) and related problems are collected in Section 2. Lagrangean duality theory turns out to play a surprisingly major role. In Section 3, a generic LP-based branch-and-bound approach for (P) is reviewed, and the basic uses and strategies of Lagrangean relaxation in this context are described. Section 4 derives the standard penalties of Driebeek [4] and Tomlin [28] from the viewpoint of Lagrangean relaxation, and

several new penalties are developed. In Section 5, the concept of surrogate constraints as developed by Glover [17] and the author [12] is shown to be subsumed by the Lagrangean relaxation viewpoint. Section 6 derives cutting-planes based on Lagrangean relaxation, including some which utilize the penalties of Section 4. Concluding comments are given in Section 7.

Three simple examples for the special constraints $Bx \geqq d$ will now be introduced. They will serve in the sequel to illustrate general ideas and to emphasize that Lagrangean relaxation is intended to be specialized to particular problem structures. The final subsection of this Introduction summarizes the special notations and assumptions commonly used in the sequel.

1.1. *Three examples*

The Lagrangean relaxation (PR_λ) must be much simpler to solve than (P) itself in order for it to yield any computational advantage. It should admit a closed form solution or be solvable by an efficient specialized algorithm. Thus the constraints $Bx \geqq d$ must possess considerable special structure. Three of the simplest possible examples of such structure are as follows. They will be referred to repeatedly in the sequel.

Example 1. The constraints $Bx \geqq d$ specify only upper bounds on some or all of the variables. For instance, in 0–1 programming problems the integer variables possess upper bounds of unity. It is easy to see that the optimal solution of (PR_λ) can be written down by inspection of the signs of the collected coefficient vector of x, namely $(c - \lambda A)$.

Example 2. The constraints $Bx \geqq d$ are as in Example 1 but also include some generalized upper bounding constraints of the form

$$\sum_{j \in J_k} x_j = 1, \qquad k = 1, 2, \ldots, K, \tag{1}$$

where $J_1, \ldots, J_K$ are disjoint subsets of I. Such constraints perform a "multiple choice" function. The optimal solution of (PR_λ) can again be written down by inspection, with a search for the smallest $(c - \lambda A)_j$ now being necessary over each subset J_k.

Example 3. The constraints $Bx \geqq d$ are as in Example 1 but also include some constraints of the form

$$\sum_{j \in J_k} \beta_{kj} x_j \leqq \beta_{kk} x_k, \qquad k = 1, \ldots, K, \tag{2}$$

where the K subsets $\{k, J_k\}$ are disjoint, $x_1, \ldots, x_K$ are 0–1 variables, the variables in J_k are continuous-valued, and all β coefficients are strictly positive. This type of constraint typically arises in location and expansion models. In the familiar capacitated plant location problem for example, x_k is 1 or 0, according to whether or not a plant of capacity β_{kk} is built at the k^{th} site, x_j for $j \in J_k$ corresponds to the amounts shipped from plant site k to various destinations, and the β_{kj}'s are all unity. The Lagrangean relaxation (PR_λ) can be solved easily because it separates into K independent problems of the form

$$\begin{aligned} &\text{minimize} \sum_{j \in J_k} (c - \lambda A)_j x_j + (c - \lambda A)_k x_k, \\ &\text{subject to} \sum_{j \in J_k} \beta_{kj} x_j \leqq \beta_{kk} x_k, \\ &\qquad 0 \leqq x_j \leqq u_j, \qquad j \in J_k, \\ &\qquad x_k = 0 \text{ or } 1, \end{aligned} \tag{3^k_λ}$$

where u_j is the upper bound on variable x_j. If $x_k = 0$, it follows from the positivity of β_{kj} that $x_j = 0$ must hold for all $j \in J_k$. If $x_k = 1$, (3^k_λ) becomes a trivial "continuous knapsack problem" with bounded variables. The best of the solutions obtained under the two cases $x_k = 0$ and $x_k = 1$ yields the true optimal solution of (3^k_λ). From these K solutions one may directly assemble the optimal solution of (PR_λ).

These three examples are among the simplest types of special constraints $Bx \geqq d$ for which the associated Lagrangean relaxation can be optimized very efficiently. Whereas closed form solutions are available for these examples, other applications may call for specialized algorithms of a less trivial sort. In most practical applications of integer programming there are several obvious and tractable choices for the constraints to be designated as $Bx \geqq d$. In Held and Karp's excellent work on the traveling-salesman problem [20, 21], for example, (PR_λ) is a minimum spanning "1-tree" problem for which highly efficient algorithms are available. And in Fisher and Schrage's algorithm for hospital admissions scheduling [9], (PR_λ) separates into a relatively simple scheduling problem for each patient.

1.2. *Notation and assumptions*

Notation and terminology is generally standard and consistent with that of [14], a survey paper containing additional background material. However, *the reader should memorize the following peculiar notations:* if $(\cdot)$ is an optimization problem, then $v(\cdot)$ is its optimal value, $F(\cdot)$ is its set of feasible solutions, and $\bar{(\cdot)}$ refers to the same problem with all integrality conditions on the variables dropped; the vector $\bar{\lambda}$ denotes an optimal multiplier vector (dual solution) associated with the constraints $Ax \geqq b$ for the ordinary linear program $(\bar{\mathrm{P}})$.

We adopt the convention that the optimal value of an infeasible optimization problem is $+\infty$ (resp. $-\infty$) if it is a minimizing (resp. maximizing) problem. The inner product of two vectors, be they row or column, is denoted simply by their juxtaposition.

Two benign assumptions are made throughout this paper in the interest of decluttering the exposition, except where explicitly stated to the contrary. The first is that the nonspecial constraints $Ax \geqq b$ are all inequality constraints. If some of these constraints were given as *equalities*, then the corresponding components of λ would not be required to be nonnegative. This is the only change required to accommodate equality constraints. The second assumption is that the special constraints $Bx \geqq d$ include upper bounds on all variables. This obviates the need for special treatment of the case where (P) or one of its relaxations has optimal value equal to $-\infty$, and also permits certain notational economies. This assumption is consistent with the vast majority of potential applications. It is a simple exercise to allow for its absence in all of the results to follow.

2. Theory of Lagrangean relaxation

The term *relaxation* is used in this paper in the following sense: a minimizing problem (Q) is said to be a relaxation of a minimizing problem (P) if $F(\mathrm{Q}) \supseteq F(\mathrm{P})$ and the objective function of (Q) is less than or equal to that of (P) on $F(\mathrm{P})$. Clearly (PR_λ) is a relaxation in this sense for all $\lambda \geqq 0$, for the extra Lagrangean term $\lambda(b - Ax)$ in the objective function of (PR_λ) must be nonpositive when $Ax \geqq b$ is satisfied. Notice that the common practice of relaxation by simply throwing away some of the constraints is equivalent to Lagrangean relaxation with $\lambda = 0$. Permitting $\lambda \neq 0$ (but always $\geqq 0$) allows the relaxation to be tighter.

The potential usefulness of any relaxation of (P), and of a Lagrangean relaxation in particular, is largely determined by how near its optimal

value is to that of (P). This furnishes a criterion by which to measure the "quality" of a particular choice for λ. The ideal choice would be to take λ as an optimal solution of the (concave) program

$$\text{(D)} \qquad \underset{\lambda \geqq 0}{\text{maximize}}\ v(\text{PR}_\lambda),$$

which is designated by (D) because it coincides with the formal Lagrangean dual of (P) with respect to the constraints $A x \geqq b$ (see, e.g., [13]). This problem in turn is intimately linked to the following relaxation of (P):

$$\text{(P*)} \qquad \underset{x}{\text{minimize}}\ c x,$$

$$\text{subject to } A x \geqq b,$$

$$x \in \text{Co}\{x \geqq 0\colon B x \geqq d \text{ and } x_j \text{ integer}, j \in I\},$$

where Co denotes the convex hull of a set. It may be difficult to express the convex hull in (P*) as an explicit set of linear constraints, but in principle this is always possible and so (P*) may be regarded as a linear program. In fact, as we shall see, (P*) and (D) are essentially LP duals. An optimal multiplier vector corresponding to $A x \geqq b$ will be denoted by λ^* when (P*) has finite optimal value.

Theorem 1 describes some of the basic relationships between (P), (PR_λ), (D), (P*), and ($\bar{\text{P}}$) (the usual LP relaxation which drops the integrality requirements).

Theorem 1. (a)

$$\text{(a)} \qquad F(\bar{\text{P}}) \supseteq F(\text{P*}) \supseteq F(\text{P}), \qquad F(\text{PR}_\lambda) \supseteq F(\text{P}),$$

$$v(\bar{\text{P}}) \leqq v(\text{P*}) \leqq v(\text{P}), \qquad v(\text{PR}_\lambda) \leqq v(\text{P}) \quad \textit{for all } \lambda \geqq 0.$$

(b) *If* ($\bar{\text{P}}$) *is feasible, then* $v(\bar{\text{P}}) \leqq v(\text{PR}_{\bar{\lambda}})$.

(c) *If for a given* λ *a vector* x *satisfies the three conditions*

(i) x *is optimal in* (PR_λ),

(ii) $A x \geqq b$,

(iii) $\lambda(b - A x) = 0$,

then x *is an optimal solution of* (P). *If* x *satisfies* (i) *and* (ii) *but not* (iii), *then* x *is an* ε*-optimal solution of* (P) *with* $\varepsilon = \lambda(A x - b)$.

(d) *If* (P*) *is feasible, then*

$$v(\text{D}) \equiv \max_{\lambda \geqq 0} v(\text{PR}_\lambda) = v(\text{PR}_{\lambda^*}) = v(\text{P*}).$$

Proof. Parts (a) and (c) are very easy. Let $(\overline{\mathrm{P}})$ be feasible. Then it has finite optimal value, for $Bx \geqq d$ includes upper bounds on all variables, and

$$v(\overline{\mathrm{P}}) = \max_{\lambda \geqq 0} v(\overline{\mathrm{PR}_\lambda}) \quad \text{(by the dual theorem of linear programming)}^1,$$

$$= v(\overline{\mathrm{PR}_{\bar{\lambda}}}) \quad \text{(by the definition of } \bar{\lambda}),$$

$$\leqq v(\mathrm{PR}_{\bar{\lambda}}) \quad \text{(because } F(\overline{\mathrm{PR}_{\bar{\lambda}}}) \supseteq F(\mathrm{PR}_{\bar{\lambda}}).$$

This proves part (b). An identical argument (the third portion is not needed) applied to (P*) yields the conclusion of part (d) if one uses the following observation in the obvious way:

$$v(\mathrm{PR}_\lambda) = [\min_x c\,x + \lambda(b - A\,x),$$

$$\text{s.t. } x \in \mathrm{Co}\,\{x \geqq 0 \colon B\,x \geqq d \text{ and } x_j \text{ integer}, j \in I\}],$$

which holds because the minimum value of a linear function over any compact set is not changed if the set is replaced by its convex hull.

A few comments are in order. Part (a) simply records the most obvious relations between (P) and its relaxations $(\overline{\mathrm{P}})$, (P*) and (PR_λ). Part (b) shows that $\bar{\lambda}$, an immediate by-product of optimally solving the standard LP relaxation $(\overline{\mathrm{P}})$, yields a Lagrangean relaxation that is at least as good as $(\overline{\mathrm{P}})$ itself (hopefully it will be better). Part (c) indicates the well-known conditions under which a solution of a Lagrangean relaxation is also optimal or near-optimal in (P). This is in recognition of the fact that Lagrangean relaxation is of interest not only for the lower bounds it yields on $v(\mathrm{P})$, but also for the possibility that it may actually yield an optimal or near-optimal solution of (P). It follows, incidentally, that (PR_λ) can yield in this manner a proven ε-optimal solution ($\varepsilon \geqq 0$) of (P) only if $v(\mathrm{PR}_\lambda) \geqq v(\mathrm{P}) - \varepsilon$. Thus the provable quality of the feasible solutions obtainable from Lagrangean relaxation by invoking part (c) is limited by the gap (if any) between $v(\mathrm{P})$ and $v(\mathrm{D})$. Part (d) establishes that Lagrangean relaxation can do as well as, but no better than (P*). Thus, the position of $v(\mathrm{P}^*)$ in the interval $[v(\overline{\mathrm{P}}), v(\mathrm{P})]$ is the question of central concern when analyzing the potential value of Lagrangean relaxation applied to a particular problem class.

[1] We have taken here the "partial" dual of $(\overline{\mathrm{P}})$ with respect to the constraints $Ax \geqq b$, rather than the "full" dual customarily used in linear programming. See [13] (especially Sec. 6.1) for an account of this generalization of the traditional duality theory. It is easily verified that $\bar{\lambda}$ is a bona fide optimal solution of the partial dual even though it may be defined in terms of the full dual.

It bears emphasis that the conclusion of part (d) is really a simple consequence of the fact that (P*) and (D) are essentially LP duals of one another. The true dual of (P*) when multipliers are introduced just for the $Ax \geqq b$ constraints is

$$\underset{\lambda \geqq 0}{\text{maximize}} \left[\min_{x} cx + \lambda(b - Ax),\right.$$
$$\left.\text{s.t. } x \in \text{Co}\{x \geqq 0: Bx \geqq d \text{ and } x_j \text{ integer}, j \in I\}\right].$$

But, as observed in the proof of part (d), the maximand of this problem equals $v(\text{PR}_\lambda)$, so (D) *must have the same optimal value and optimal solution set as the true dual of* (P*). Thus one may invoke most of the rich optimality/duality theory for linear programming to say much more about the relationship between (P*) and (D) than is said in Theorem 1 (d). For instance, one may assert when (P*) is feasible that the set of its optimal multipliers coincides with the set of optimal solutions of (D) and also with the negative of the set of subgradients at $y = 0$ of its (convex) b-perturbation function

$$\phi_b^*(y) \overset{\text{d}}{=} \left[\begin{array}{l}\text{inf. } cx, \\ \text{s.t. } Ax \geqq b - y, \\ \quad x \in \text{Co}\{x \geqq 0: Bx \geqq d \text{ and } x_j \text{ integer}, j \in I\}\end{array}\right],$$

(see, e.g., [13, Th. 1 and 3]).

Theorem 1 leaves open two important questions: the relations between $v(\overline{\text{P}})$ and $v(\text{P}^*)$ and between $v(\text{P}^*)$ and $v(\text{P})$. These relations are taken up in the next two theorems.

Theorem 2 shows that $v(\overline{\text{P}}) = v(\text{P}^*)$ when the following holds:

Integrality Property. The optimal value of (PR_λ) is not altered by dropping the integrality conditions on its variables, i.e., $v(\text{PR}_\lambda) = v(\overline{\text{PR}_\lambda})$ for all $\lambda \geqq 0$.

Theorem 2. *Let* $(\overline{\text{P}})$ *be feasible and* (PR_λ) *have the Integrality Property. Then* (P*) *is feasible and*

$$v(\overline{\text{P}}) = v(\text{PR}_{\bar{\lambda}}) = v(\text{D}) = v(\text{PR}_{\lambda_*}) = v(\text{P}^*).$$

Proof. In view of Theorem 1 (b), (d), it is enough to show that $v(\overline{\text{P}}) = v(\text{P}^*)$. We have

$$v(\overline{\text{P}}) = \max_{\lambda \geqq 0} v(\overline{\text{PR}_\lambda}) \quad \text{(by duality)},$$
$$= \max_{\lambda \geqq 0} v(\text{PR}_\lambda) \quad \text{(by the Integrality Property)},$$

$$= \max_{\lambda \geqq 0} \left[\min_{x} c\,x + \lambda\,(b - A\,x), \right.$$

$$\text{s.t.} \quad x \in \text{Co}\,\{x \geqq 0 \colon B\,x \geqq d \text{ and } x_j \text{ integer}, j \in I\}\Big]$$

(by the observation used in the proof of Theorem 1 (d)),

$$= v(\text{P}^*) \qquad \text{(by duality)}.$$

Notice that the feasibility of (P*) is a consequence of the fact that its dual has finite optimal value.

Thus Lagrangean relaxation can do no better than the standard LP relaxation ($\overline{\text{P}}$) when the constraint partition $A\,x \geqq b$, $B\,x \geqq d$ is such that the Integrality Property holds.[2] The best choice of λ for (PR_λ) is then $\bar{\lambda}$ from ($\overline{\text{P}}$). In this circumstance, Lagragean relaxation seems to be of questionable value unless a near-optimal solution of (D) can be found by specialized means more rapidly than ($\overline{\text{P}}$) can be solved by linear programming methods. Generally it is more promising to use Lagrangean relaxations for which the Integrality Property does not hold.

The Integrality Property clearly holds for Examples 1 and 2 (one may assume without loss of generality that the upper bounds on the integer variables are integers), but it does not hold for Example 3. The presence or absence of the Integrality Property is evident in many applications upon inspection of (PR_λ) in light of the special structure of the constraints $B\,x \geqq d$. In other applications one may be able to appeal to the total unimodularity characterization of natural integer solutions of linear programming problems (e.g., [30]).

We now turn to the relationship between $v(\text{P}^*)$ [or $v(\text{D})$] and $v(\text{P})$. A sufficient condition for $v(\text{P}^*) = v(\text{P})$ obviously is

$$F(\text{P}^*) = \text{Co}\,[F(\text{P})],$$

but this is likely to be difficult to verify in specific cases because the "integer polyhedron" is a notoriously difficult object to study.

Most of what is known about the relationship in question is a consequence of the fact that (D) is the formal Lagrangean dual of (P) with respect to the constraints $A\,x \geqq b$. Careful examination of Lagrangean duality theory shows that many of the results do not require convexity of the primal

[2] This fact has been noted by Nemhauser and Ullman [25] in the special context of Example 1.

problem. For instance, convexity is not used in the proofs of the key Lemmas 3, 4 and 5 of [13]. These results yield Theorem 3, which uses the following definitions. The *b-perturbation function* associated with (P) is defined as

$$\phi_b(y) \triangleq [\inf_{x \geqq 0} c\,x, \quad \text{s.t. } A\,x \geqq b - y, \quad B\,x \geqq d, \quad x_j \text{ integer}, \quad j \in I].$$

A vector γ conformable with y is said to be a *global subgradient* of ϕ_b at $y = 0$ (assuming $\phi_b(0) \equiv v(\mathrm{P})$ is finite) if

$$\phi_b(y) \geqq v(\mathrm{P}) + \gamma\,y \quad \text{for all } y.$$

The adjective "global" is used to emphasize that the subgradient definition used here relates to a global rather than local aspect of ϕ_b (which is generally nonconvex).

Theorem 3. *Assume that* (P) *is feasible* (*and therefore has an optimal solution, since all variables are bounded*).

(a) *The following are equivalent:*
  (1) $v(\mathrm{P}) = v(\mathrm{D})$.
  (2) *There exists a global subgradient of* ϕ_b *at* $y = 0$.
  (3) *There exists a pair* (x, λ) *satisfying* $\lambda \geqq 0$ *and conditions* (i), (ii) *and* (iii) *of Theorem* 1(c).

(b) *If* $v(\mathrm{P}) = v(\mathrm{D})$, *then each optimal solution of* (D) *is the negative of a global subgradient of* ϕ_b *at* $y = 0$ *and conversely, and any such solution* λ^* *yields the set of all optimal solutions of* (P) *as the vectors* x *which satisfy conditions* (i), (ii) *and* (iii) *of Theorem* 1 (c) *with* $\lambda = \lambda^*$.

The most interesting aspect of Theorem 3 is the criterion for the equality $v(\mathrm{P}) = v(\mathrm{D})$ in terms of the existence of a global subgradient of ϕ_b at the origin and the identification of these subgradients with the solutions of (D). The theorem also confirms that Lagrangean relaxation does indeed yield the optimal solutions of (P) when $v(\mathrm{P}) = v(\mathrm{D})$, via the optimality conditions of Theorem 1 (c).

The b-perturbation function ϕ_b thus emerges as a key object for study if one wishes to understand when $v(\mathrm{P}) = v(\mathrm{D})$ is likely to hold. What is known about ϕ_b? Clearly it is nonincreasing. It can also be shown to be lower semicontinuous. It is piecewise-linear and convex over any region

where the optimal values of the integer variables stay constant, for in such a region the perturbed (P) reduces to a perturbed linear program. And ϕ_b is obviously bounded below by the piecewise-linear convex perturbation function ϕ_b^* defined earlier for (P*). In fact, this last observation can be strengthened to assert that ϕ_b^* is actually the *best* possible convex function which nowhere exceeds ϕ_b in value. This geometrically obvious but important result is stated more precisely as follows.

Theorem 4. *The b-perturbation function ϕ_b^* associated with* (P*) *is precisely the lower convex envelope of the b-perturbation function ϕ_b associated with* (P).

Proof. An alternative way of phrasing the result is to say that the epigraph of ϕ_b^* is the convex hull of the epigraph of ϕ_b; that is, $\text{Epi}[\phi_b^*] = \text{Co}\{\text{Epi}[\phi_b]\}$. Clearly,

$$\begin{aligned}\text{Epi}[\phi_b] &\triangleq \{(\mu, y): \mu \geqq \phi_b(y)\} \\ &= \{(\mu, y): \mu \geqq c\,x \text{ and } b - A\,x \leqq y \text{ for some } x \in X\},\end{aligned}$$

where

$$X \triangleq \{x \geqq 0: B\,x \geqq d \text{ and } x_j \text{ integer for } j \in I\},$$

and similarly for $\text{Epi}[\phi_b^*]$ with X replaced by $\text{Co}\{X\}$. Suppose that $(\bar{\mu}, \bar{y}) \in \text{Epi}[\phi_b^*]$. Then $\bar{\mu} \geqq c\,\bar{x}$ and $b - A\,\bar{x} \leqq \bar{y}$ for some $\bar{x} \in \text{Co}\{X\}$. Let $\bar{x} = \sum_h \theta_h x^h$, where $x^h \in X$, $\theta_h \geqq 0$ for all h and $\sum_h \theta_h = 1$. Define $\mu^h = c\,x^h$ and $y^h = b - A\,x^h$ for all h. Clearly $(\mu^h, y^h) \in \text{Epi}[\phi_b]$ for all h. Hence $\sum_h \theta_h(\mu^h, y^h) \in \text{Co}\{\text{Epi}[\phi_b]\}$. But

$$\begin{aligned}\sum_h \theta_h(\mu^h, y^h) &= (\textstyle\sum_h \theta_h \mu^h, \sum_h \theta_h y^h) \\ &= (\textstyle\sum_h \theta_h c\,x^h, \sum_h \theta_h(b - A\,x^h)) = (c\,\bar{x}, b - A\,\bar{x}) \leqq (\bar{\mu}, \bar{y})\end{aligned}$$

and so $(\bar{\mu}, \bar{y})$ must also be in $\text{Co}\{\text{Epi}[\phi_b]\}$. This shows that $\text{Epi}[\phi_b^*] \subseteq \text{Co}\{\text{Epi}[\phi_b]\}$. Now suppose $(\bar{\mu}, \bar{y}) \in \text{Co}\{\text{Epi}[\phi_b]\}$. Let $(\bar{\mu}, \bar{y}) = \sum_h \theta_h (\mu^h, y^h)$, where $(\mu^h, y^h) \in \text{Epi}[\phi_b]$, $\theta_h \geqq 0$ for all h and $\sum_h \theta_h = 1$. Let x^h be any point in X satisfying $\mu^h \geqq c\,x^h$ and $b - A\,x^h \leqq y^h$. Then

$$\sum_h \theta_h \mu^h \geqq \sum_h \theta_h c\,x^h = c\,\bar{x},$$

$$\sum_h \theta_h y^h \geqq \sum_h \theta_h(b - A\,x^h) = b - A\,\bar{x},$$

where $\bar{x} \overset{d}{=} \sum_h \theta_h x^h$. Thus $(\bar{\mu}, \bar{y}) \geqq (c\,\bar{x}, b - A\,\bar{x})$ with $\bar{x} \in \text{Co}\,\{X\}$, which shows that $(\bar{\mu}, \bar{y}) \in \text{Epi}\,[\phi_b^*]$. This completes the proof.

This is the central connection between (P) and (P*) – actually, between two parameterized families of problems of which (P) and (P*) are members of special significance. The duality gap (if any), $v(P) - v(D)$, is precisely equal to the difference between the b-perturbation function of (P) and its lower convex envelope, both evaluated at the origin. This characterization provides the basis for a qualitative understanding of duality gaps—and hence of the potential of Lagrangean relaxation—when applied to specific classes of problems with reference to salient characteristics of the data.

Some of these ideas are illustrated in Fig. 1 for a hypothetical mixed integer program with but a single A-type constraint (so that y is a scalar). Suppose that only two sets of values for the integer variables enter into an optimal solution of (P) as b varies. The piecewise-linear and convex b-perturbation functions for the two corresponding linear programs (with the integer variables fixed) are drawn as light lines. One of these linear programs becomes infeasible for $y \leqq y^1$, while the other becomes infeasible for $y \leqq y^2$. The pointwise minimum of these two functions is $\phi_b(y)$, which is superimposed as a heavy line. The lower convex envelope of $\phi_b(y)$, namely $\phi_b^*(y)$, is superimposed as a line with alternating dots and dashes. It is clear that there is no duality gap (difference between $\phi_b(y)$ and $\phi_b^*(y)$) for $y^2 \leqq y \leqq y^3$ or $y^4 \leqq y$. A global subgradient of ϕ_b at $y = 0$ will exist

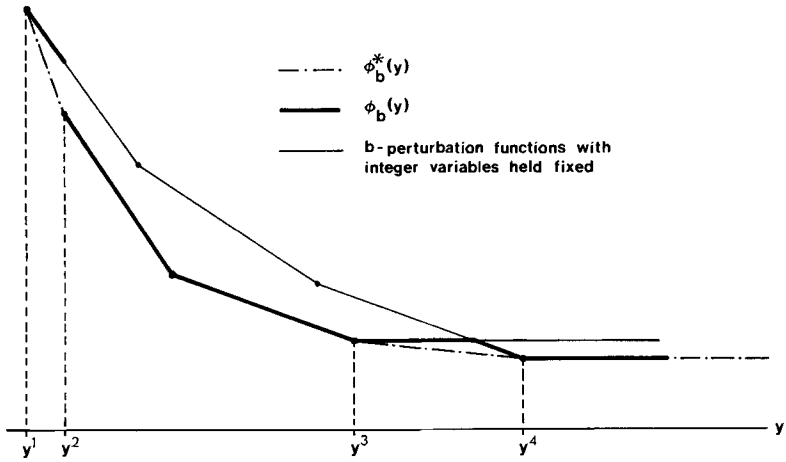

Fig. 1. Hypothetical illustration of b-perturbation functions.

(any subgradient of ϕ_b^* at $y = 0$ will do) if $y = 0$ falls in either of these intervals. If $y = 0$ falls between y^1 and y^2 or between y^3 and y^4, on the other hand, there will be a gap and no global subgradient of ϕ_b at $y = 0$ will exist.

We note in closing that the duality gap tends to be rather small for the class of problems with which we have numerical experience, namely capacitated facility location problems with additional constraints. The special constraints are as in Example 3. For four practical problems the values averaged as follows (after normalization via division by 0.01 $v(\mathrm{P})$):

$$\begin{aligned} v(\mathrm{P}) &= 100.00, \\ v(\mathrm{D}) &= 99.93, \\ v(\mathrm{PR}_{\bar{\lambda}}) &= 98.97, \\ v(\bar{\mathrm{P}}) &= 97.46. \end{aligned}$$

Notice that the duality gap is small by comparison with the gap between the integer problem and its usual LP relaxation, and that the LP multipliers $\bar{\lambda}$ yield a Lagrangean relaxation quite a bit better than the LP relaxation itself. See [15] for further details.

3. The use of Lagrangean relaxation in LP-based branch-and-bound

Virtually all of the current generally successful integer linear programming algorithms are of the branch-and-bound type with linear programming as the primary source of bounds [14]. This section and those to follow discuss the use of Lagrangean relaxation as a device for possibly improving the efficiency of such algorithms for special classes of problems.

A brief review of the usual LP-based branch-and-bound approach to (P) is necessary at this point. The terminology adopted is that of [14] which can be consulted for further details. At any given time there is a list of so-called *candidate problems*, each of which is simply (P) with certain additional "separation" constraints appended. The union of the feasible regions of the candidate problems constitutes a partition of the unenumerated portion of the feasible region of (P). There is also a number z^* representing the objective value of the *incumbent*, the best currently known feasible solution of (P) (initially z^* can be taken to be a suitably large number). The primary iterative step is to select one of the candidate problems, say (CP), and to examine it for the existence of a feasible solution of (P) with value better than z^*. The examination may be conclusive or inconclusive, depending on how much effort is expended; the usual practice involves solving the linear program ($\overline{\mathrm{CP}}$), which ignores all integrality conditions on the

variables of (CP). A conclusive examination is one for which the outcome is

(i) that (CP) is infeasible (e.g. $(\overline{\text{CP}})$ is infeasible), or
(ii) that $v(\text{CP}) \geqq z^*$ (e.g. $v(\overline{\text{CP}}) \geqq z^*$), or
(iii) that $v(\text{CP}) < z^*$ and an optimal solution of (CP) is at hand (e.g. the optimal solution $\bar{x}$ of $(\overline{\text{CP}})$ happens to satisfy the integrality conditions); this solution replaces the current incumbent and z^* is updated.

Then (CP) is said to be *fathomed* and is deleted from the list of candidate problems. Otherwise, (CP) is not fathomed and must be separated into two or more simpler candidate (sub)problems to be added to the list. This is accomplished via mutually exclusive and exhaustive separation constraints. The usual practice (cf. [3]) is to select a particular *separation variable* $j_0 \in I$ and to invoke an interval dichotomy on its range. For instance, for $j_0 = 3$ one subproblem might receive the new constraint $x_3 \leqq 2$ and the other the new constraint $x_3 \geqq 3$. Candidate problems continue to be examined in this fashion, with fathoming or separation occurring each time, until the list of candidate problems is exhausted.

It should be evident that a Lagrangean relaxation of (CP), say (CPR_λ), is just as amenable as the usual linear programming relaxation $(\overline{\text{CP}})$ as a device for examining candidate problems: the infeasibility of (CPR_λ) implies that of (CP); $v(\text{CPR}_\lambda) \geqq z^*$ implies $v(\text{CP}) \geqq z^*$; and an optimal solution of (CPR_λ), say x^{R}, is optimal in (CP) if it is feasible in (CP) and satisfies complementary slackness (see Theorem 1 (c)). Note that if x^{R} is feasible in (CP) but does not satisfy complementary slackness, it may still improve on the incumbent, in which case it should be used to update the incumbent and z^* even though (CP) is not fathomed. In cases where x^{R} is not feasible in (CP) it may be worth trying to adjust it in some problem-specific manner so as to gain feasibility and, hopefully, to improve thereby on the incumbent. This is exactly the same tactic as is commonly used with $(\overline{\text{CP}})$ when the (fractional) LP solution is rounded to satisfy integrality in the hope of obtaining an improved feasible solution.

The usual linear programming relaxation $(\overline{\text{CP}})$ is also used commonly to derive conditional bounds for use in guiding separation, for tagging newly created candidate subproblems with lower bounds on their optimal value, and for reducing the range restrictions on integer variables without sacrificing optimality. Lagrangean relaxations of (CP) can be used for these same purposes. Suppose that some variable $j \in I$ has a fractional value in the LP solution $\bar{x}$ of $(\overline{\text{CP}})$. We are interested in lower bounds on $v(\text{CP} \mid x_j \leqq \lfloor \bar{x}_j \rfloor)$ and $v(\text{CP} \mid x_j \geqq \lfloor \bar{x}_j \rfloor + 1)$, where " $\mid$ " signifies that the constraint following

it is appended to the problem, and $[\bar{x}_j]$ stands for the integer part of $\bar{x}_j$. Such conditional bounds are given, respectively, by

$$\begin{aligned} v_D(j) &\triangleq v(\mathrm{CPR}_\lambda \mid x_j \leqq [\bar{x}_j]), \\ v_U(j) &\triangleq v(\mathrm{CPR}_\lambda \mid x_j \geqq [\bar{x}_j] + 1). \end{aligned} \tag{4}$$

If $v_D(j) \geqq z^*$ holds, then the lower limit for x_j obviously can be tightened to $[\bar{x}_j] + 1$. Similarly, $v_U(j) \geqq z^*$ implies that the upper range restriction can be lowered to $[\bar{x}_j]$. It is even possible that both $v_D(j) \geqq z^*$ and $v_U(j) \geqq z^*$ hold, in which case it is clear that (CP) is fathomed. The bounds (4) can also be used to guide separation in the event that (CP) is not fathomed. Let $V_D(j)$ and $V_U(j)$ be computed for every $j \in I$ such that $\bar{x}_j$ is fractional. One appealing choice for the separation variable would be the one which maximizes the larger of $V_D(j)$ and $V_U(j)$ over all eligible j. Several successful integer programming codes have employed an analogous criterion based on $(\overline{\mathrm{CP}})$ rather than (CPR_λ). Once a separation variable j_0 is selected, $V_D(j_0)$ and $V_U(j_0)$ yield lower bounds for future reference on the newly created candidate subproblems.

The computation of conditional bounds like (4) is taken up in more detail in Section 4. We note here only that the bounding problems have the same structure as (CPR_λ) since we have assumed that range restrictions on all variables are incorporated into the special constraints $Bx \geqq d$, just as (CPR_λ) will have the same structure as (PR_λ) if, as is usually the case, the separation constraints employed are simple range restrictions on the variables.

Thus we see that Lagrangean relaxation can be used for the standard branch-and-bound tasks of fathoming, generating improved feasible solutions, range reduction, and guiding separation. It can also be used to derive surrogate constraints and cutting-planes. These uses are taken up in Section 5 and 6.

We turn now to a discussion of the strategy questions which arise in connection with the use of (CPR_λ) as an adjunct to $(\overline{\mathrm{CP}})$. The two main questions concern how λ is to be chosen and whether (CPR_λ) should be used before or after or even in place of $(\overline{\mathrm{CP}})$. These questions cannot be answered definitively in general, but an obviously important consideration is whether or not the Integrality Property defined in Section 2 holds for the particular constraint partition under consideration.

Suppose the Integrality Property does hold. Then (CPR_λ) can be infeasible only if $(\overline{\mathrm{CP}})$ is infeasible, and if $(\overline{\mathrm{CP}})$ is feasible, then by Theorem 2 it

must yield the best possible choice of λ for (CPR_λ) and $v(\mathrm{CPR}_{\lambda^*}) = v(\overline{\mathrm{CP}})$. Thus (CPR_λ) cannot fathom (CP) by infeasibility or by value unless $(\overline{\mathrm{CP}})$ would also do so. One can also show that at least one of the conditional bounds $V_{\mathrm{D}}(j)$ and $V_{\mathrm{U}}(j)$ must coincide with $v(\overline{\mathrm{CP}})$ for each variable that is fractional in an optimal solution of $(\overline{\mathrm{CP}})$ when the natural choice $\bar{\lambda}$ from $(\overline{\mathrm{CP}})$ is used in (4). Moreover, *both* of the bounds coincide with $v(\overline{\mathrm{CP}})$ in the special case of Examples 1 and 2 and perhaps in other cases as well. These facts argue against the use of a Lagrangean relaxation for which the Integrality Property holds. It has little to offer that cannot already be achieved by $(\overline{\mathrm{CP}})$, though it may possibly prove to be more fruitful than $(\overline{\mathrm{CP}})$ as a source of improved feasible solutions. It is important to recognize, however, that this negative conclusion rests on the implicit assumption that $(\overline{\mathrm{CP}})$ is of manageable size as a linear program. If this is not the case, then (CPR_λ) may be a comparatively attractive computational alternative. A beautiful illustration is provided by Held and Karp's work on the traveling-salesman problem. Here $(\overline{\mathrm{CP}})$ has such an enormous number of constraints that it is not practical to solve directly. Of course, the omission of $(\overline{\mathrm{CP}})$ necessitates the introduction of some method for computing a near optimal λ (see below). And even if $(\overline{\mathrm{CP}})$ *is* of manageable size it may still be sufficiently burdensome computationally that (CPR_λ) is attractive as a surrogate to be invoked *prior* to $(\overline{\mathrm{CP}})$ during the examination of a candidate problem. The hope is that the Lagrangean relaxation will permit (CP) to be fathomed without having to resort to the more expensive linear program $(\overline{\mathrm{CP}})$. The best choice for λ is likely to be a multiplier vector saved from the linear program corresponding to the prior candidate problem most closely related to the current one. Section 5 indicates how this tactic coincides in special cases with the use of surrogate constraints – a device which has proven quite effective computationally in some applications (cf. [14, Sec. 3.1.5]).

Now suppose that the Integrality Property does *not* hold. Then $(\overline{\mathrm{CP}})$ does not necessarily yield the best choice for λ, and (CPR_λ) may succeed in fathoming where $(\overline{\mathrm{CP}})$ fails. It makes strategic sense to invoke (CPR_λ) either before or after $(\overline{\mathrm{CP}})$ or even in lieu of it, depending on the relative tightness and computational expense of the two relaxations. The most effective strategy also depends on the role played by $(\overline{\mathrm{CP}})$ in generating the λ to be used by (CPR_λ), since $(\overline{\mathrm{CP}})$ can be used to generate a starting (or even final) value of λ which can then be improved upon by some independent method. To indicate the possible methods for finding a suitable λ we shall consider for the sake of notational convenience the situation encountered before any

branching has taken place. Then (CP) is (P) itself and (CPR_λ) is just (PR_λ). The general situation is entirely analogous.

There are two broad approaches to computing a suitable λ for (PR_λ): (sub)optimization of the concave Lagrangean dual problem (D) and (sub) optimization of the linear program (P*). The first approach yields λ directly, whereas the second yields λ indirectly as the multiplier vector associated with the $A\,x \geqq b$ constraints in (P*). The distinction should not be thought of as a rigid one; some methods can be described naturally from either viewpoint.

Consider the first approach. One of the most promising methods for seeking an optimal solution of (D) is via the Agmon–Motzkin–Schoenberg method as revived by Held and Karp [21]. See also the recent and extensive study of this method by Held, Wolfe and Crowder [22]. The idea is very simple. Let $\lambda^\nu \geqq 0$ be the current estimate of an optimal solution of (D) and let x^ν be an optimal solution of (PR_{λ^ν}). Then the new estimate is

$$\lambda^{\nu+1} = \max\{\lambda^\nu + \theta^\nu(b - A\,x^\nu), 0\},$$

where the max operator is applied component-wise and θ^ν is a positive step size satisfying certain requirements [22]. The vector $(b - A\,x^\nu)$ is a subgradient of $v(PR_\lambda)$ at $\lambda = \lambda^\nu$ but the sequence $\langle v(PR_{\lambda^\nu})\rangle$ is not necessarily monotone. Favorable computational experience has been reported for several different applications [8, 21, 22]. An alternative is to carry out an ascent method for (D); see [8, 10, 20]. Still another method is to optimize (D) by tangential approximation (outer linearization/relaxation) making use of the fact that the evaluation of $v(PR_\lambda)$ for a given λ yields a linear support at that point. The available evidence [20, 24] suggests that convergence is slow in some applications. A combination of ascent and tangential approximation is possible with the BOXSTEP method of Hogan, Marsten and Blankenship [23].

Consider now the indirect approach via (P*). Perhaps the most obvious method is to apply generalized programming (Dantzig–Wolfe decomposition, inner linearization/restriction) with the convex hull portion of the constraints of (P*) represented in terms of its extreme points. The column-generation problems are precisely of the form (PR_λ). Since this method is equivalent to the tangential approximation method for (D), however, its efficiency is suspect. Another possibility is to apply the primal-dual simplex method to (P*) with special provisions to accomodate the convex hull. This method, developed by Fisher and by Fisher and Shapiro, can also be interpreted as an ascent method for (D). Some encouraging computational

experience has been reported [8]. In some applications the form of the constraints describing the convex hull in (P*) is known. Then it may be possible to apply the dual simplex method to (P*) with (most) violated constraints generated as needed. This is probably one of the best methods for obtaining a near-optimal λ fairly quickly when it applies. It has the added advantage of yielding valid constraints that may be appended to $(\overline{P})$ to make it a tighter relaxation of (P).

Other specialized techniques, both exact and heuristic, can be devised for (D) or (P*) in particular applications.

4. Penalties

The so-called "penalty" concept in integer programming was propelled to prominence by Driebeek [4], although the essential notion was used earlier by Dakin [3] and Healy [19]. The original idea was to underestimate the amount by which the optimal value of the LP relaxation of the current candidate problem would increase if separation were carried out using a particular separation variable. The estimates of change, referred to as penalties, can be used to help guide separation and may also permit fathoming or range reduction. An important subsequent refinement of this original idea was the recognition that it is the candidate problems and subproblems themselves, and not their LP relaxations, which are central to the underlying enumerative process. Tomlin [28, 29] showed how to modify the penalty formulae so as to take at least partial account of the integrality conditions. The resulting penalties are underestimates of the difference between $v(\overline{CP})$ and the optimal value of a candidate subproblem derived from (CP). See [14] for a discussion of current practice in the computation and use of penalties.

Lagrangean relaxation furnishes a convenient setting for deriving the simple and strengthened penalties alluded to above. This is done in subsection 4.1. More importantly, it leads naturally to extensions and specializations which do not follow as easily from the more traditional viewpoints. These are illustrated in subsections 4.2–4.4 for Examples 1–3. It is hoped that these improved penalties and their counterparts for other structures will add new vitality to the penalty concept by overcoming the limitations of standard penalties pointed out so clearly by Forrest, Hirst and Tomlin [11].

4.1. *Basic results: $B\,x \geqq d$ vacuous*

The first task is to show how the formulae for simple and strengthened penalties are related to Lagrangean relaxation. This requires taking $\lambda = \bar{\lambda}$ and specializing $B\,x \geqq d$ to be vacuous (in contrast to our usual convention, in this subsection, $B\,x \geqq d$ will not include upper bounds on the variables). Define I_f to be the indices in I such that $\bar{x}_j$ is fractional ($\bar{x}$ is the optimal solution of $(\overline{\mathrm{CP}})$). It is easy to verify that the objective function coefficient of $(\mathrm{CPR}_{\bar{\lambda}})$ vanishes for all such $j \in I_f$, and hence for all such j we have

$$V_{\mathrm{D}}(j) \stackrel{\mathrm{d}}{=} v(\mathrm{CPR}_{\bar{\lambda}} \mid x_j \leqq [\bar{x}_j]) = v(\overline{\mathrm{CP}}),$$
$$V_{\mathrm{U}}(j) \stackrel{\mathrm{d}}{=} v(\mathrm{CPR}_{\bar{\lambda}} \mid x_j \geqq [\bar{x}_j] + 1) = v(\overline{\mathrm{CP}}).$$

Thus the Lagrangean relaxation $(\mathrm{CPR}_{\bar{\lambda}})$ appears to yield zero "down" and "up" penalties for separation on x_j.

A simple remedy is to employ an alternative representation for x_j in terms of variables whose objective function coefficients in $(\mathrm{CPR}_{\bar{\lambda}})$ do not vanish. Such a representation is available from the final tableau of the linear program $(\overline{\mathrm{CP}})$ since $j \in I_f$ must be basic therein:

$$x_j = \bar{x}_j - \sum_{i \in N} \bar{a}_{ji} x_i,$$

where N is the set of nonbasic variables. The use of this representation in the definition of $V_{\mathrm{D}}(j)$ and $V_{\mathrm{U}}(j)$ leads to the following conditional bounds: for $j \in I_f$,

$$\begin{aligned} V_{\mathrm{D}}^*(j) &\stackrel{\mathrm{d}}{=} v(\mathrm{CPR}_{\bar{\lambda}} \mid \bar{x}_j - \textstyle\sum_{i \in N} \bar{a}_{ji} x_i \leqq [\bar{x}_j]), \\ V_{\mathrm{U}}^*(j) &\stackrel{\mathrm{d}}{=} v(\mathrm{CPR}_{\bar{\lambda}} \mid \bar{x}_j - \textstyle\sum_{i \in N} \bar{a}_{ji} x_i \geqq [\bar{x}_j] + 1). \end{aligned} \tag{5}$$

Clearly,

$$V_{\mathrm{D}}^*(j) \leqq v(\mathrm{CP} \mid x_j \leqq [\bar{x}_j]),$$
$$V_{\mathrm{U}}^*(j) \leqq v(\mathrm{CP} \mid x_j \geqq [\bar{x}_j] + 1)$$

for all $j \in I_f$; that is, these conditional bounds really do underestimate the optimal value of the candidate problems that would result if (CP) were separated using x_j as the separation variable.

Unfortunately the computation of $V_{\mathrm{D}}^*(j)$ and $V_{\mathrm{U}}^*(j)$ may be onerous if $\bar{a}_{ji} \neq 0$ for some variables $i \in N \cap I$. The computation then requires solving a knapsack-type problem with some integer variables. Hence it is natural

to think of estimating $V_D^*(j)$ and $V_U^*(j)$ from below by simply dropping all integrality conditions:

$$\begin{aligned} V_D^{*0}(j) &\triangleq v(\overline{\mathrm{CPR}}_{\bar\lambda} \mid \bar x_j - \textstyle\sum_{i\in N} \bar a_{ji} x_i \leqq [\bar x_j]), \\ V_U^{*0}(j) &\triangleq v(\overline{\mathrm{CPR}}_{\bar\lambda} \mid \bar x_j - \textstyle\sum_{i\in N} \bar a_{ji} x_i \geqq [\bar x_j] + 1). \end{aligned} \tag{6}$$

The computation of each of these conditional bounds merely requires minimizing a linear function with all nonnegative coefficients [$(c - \bar\lambda A) \geqq 0$, by duality] subject to a single linear constraint and $x \geqq 0$. This is sometimes referred to as a "continuous knapsack" type problem and it is easy to write down an explicit solution:

$$V_D^{*0}(j) = v(\overline{\mathrm{CP}}) + (\bar x_j - [\bar x_j]) \underset{i\in N:\, \bar a_{ji} > 0}{\text{minimum}} \{(c - \bar\lambda A)_i / \bar a_{ji}\}, \tag{7}$$

$$V_U^{*0}(j) = v(\overline{\mathrm{CP}}) + ([\bar x_j] + 1 - \bar x_j) \underset{i\in N:\, \bar a_{ji} < 0}{\text{minimum}} \{(c - \bar\lambda A)_i / (-\bar a_{ji})\}$$

(we have used the fact that $\bar\lambda b = v(\overline{\mathrm{CP}})$ by LP duality). These conditional bounds are identical to those associated with the simplest penalties mentioned earlier (cf. (5) in [4]).

The strengthened penalties of Tomlin can also be recovered from this viewpoint by retaining the condition that x_j must not be in the open interval $(0, 1)$ for $j \in N \cap I$. Then we obtain

$$\begin{aligned} V_D^{*1}(j) &\triangleq v(\overline{\mathrm{CPR}}_{\bar\lambda} \mid \bar x_j - \textstyle\sum_{i\in N} \bar a_{ji} x_i \leqq [\bar x_j] \text{ and } x_i \notin (0,1) \text{ for all } i \in N \cap I), \\ V_U^{*1}(j) &\triangleq v(\overline{\mathrm{CPR}}_{\bar\lambda} \mid \bar x_j - \textstyle\sum_{i\in N} \bar a_{ji} x_i \geqq [\bar x_j] + 1 \text{ and } x_i \notin (0,1) \text{ for all } i \in N \cap I). \end{aligned} \tag{8}$$

It is not difficult to see that at most one variable need be at a positive level in an optimal solution of the modified continuous knapsack problems defined in (8). This observation leads to the explicit formulae:

$$V_D^{*1}(j) = v(\overline{\mathrm{CP}}) + \underset{i\in N:\, \bar a_{ji} > 0}{\text{minimum}} \begin{cases} (c - \bar\lambda A)_i (\bar x_j - [\bar x_j]) / \bar a_{ji} & \text{if } i \notin I, \\ (c - \bar\lambda A)_i \max\{(\bar x_j - [\bar x_j])/\bar a_{ji}, 1\} & \text{if } i \in I, \end{cases} \tag{9}$$

$$V_U^{*1}(j) = v(\overline{\mathrm{CP}}) + \underset{i\in N:\, \bar a_{ji} < 0}{\text{minimum}} \begin{cases} (c - \bar\lambda A)_i ([\bar x_j] + 1 - \bar x_j) / (-\bar a_{ji}) & \text{if } i \notin I, \\ (c - \bar\lambda A)_i \max\{([\bar x_j] + 1 - \bar x_j)/(-\bar a_{ji}), 1\} & \text{if } i \in I. \end{cases}$$

These formulae are identical with the strengthened penalties of Tomlin (cf. (10) and (11) of [28] or (3.5) and (3.6) of [29]). It is evident from the very definitions (5), (6) and (8) that

$$V_D^{*0}(j) \leqq V_D^{*1}(j) \leqq V_D^*(j),$$
$$V_U^{*0}(j) \leqq V_U^{*1}(j) \leqq V_U^*(j) \tag{10}$$

for all $j \in I_f$.

This completes the recovery of known formulae for Driebeek and Tomlin penalties for $j \in I_f$. Exactly the same type of analysis holds for penalties associated with *basic* variables of $I - I_f$. Such penalties are of interest as a means of obtaining tighter ranges on integer variables which happen to be naturally integer in the optimal solution of $(\overline{\text{CP}})$. For a *nonbasic* variable variable x_j in $I - I_f$ the quantity of interest is $v(\text{CPR}_{\bar{\lambda}} \mid x_j \geqq 1)$; no alternative representation in terms of nonbasic variables is possible. Evidently,

$$v(\text{CPR}_{\bar{\lambda}} \mid x_j \geqq 1) = v(\overline{\text{CP}}) + (c - \bar{\lambda} A)_j. \tag{11}$$

Again this is a standard result.

Another technique for strengthening (6) makes use of the following elementary and well-known result.

Theorem 5. *Let* (IP) *be a minimizing integer linear program in which exactly one variable, say* x_h, *is declared to be integer-valued. Suppose that* $\bar{x}_h$, *the optimal level of* x_h *when* (IP) *is solved ignoring the integrality requirement, is fractional. Then the optimal value of* (IP) *is given by*

$$v(\text{IP}) = \min\{v(\overline{\text{IP}} \mid x_h = [\bar{x}_h]),\ v(\overline{\text{IP}} \mid x_h = [\bar{x}_h] + 1)\}.$$

The possibility that $(\overline{\text{IP}} \mid x_h = [\bar{x}_h])$ or $(\overline{\text{IP}} \mid x_h = [\bar{x}_h] + 1)$ or both are infeasible is not excluded (recall that our convention is to define a minimum over an empty set as $+\infty$).

Let $i_D(j)$ be the minimizing nonbasic i in the formula for $V_D^{*0}(j)$ given in (7). The index $i_U(j)$ is defined similarly. Then application of Theorem 5 in the obvious way permits the following improvement on (6) to be computed with only a little extra effort:

$$V_D^{*2}(j) \overset{d}{=} v(\overline{\text{CPR}}_{\bar{\lambda}} \mid \bar{x}_j - \textstyle\sum_{i \in N} \bar{a}_{ji} x_i \leqq [\bar{x}_j] \text{ and } x_{i_D(j)} \text{ integer}),$$
$$V_U^{*2}(j) \overset{d}{=} v(\overline{\text{CPR}}_{\bar{\lambda}} \mid \bar{x}_j - \textstyle\sum_{i \in N} \bar{a}_{ji} x_i \geqq [\bar{x}_j] + 1 \text{ and } x_{i_U(j)} \text{ integer}). \tag{12}$$

Neither (12) nor (8) necessarily dominates the other; one may verify the following relationship for $j \in I_f$:

$$(\bar{x}_j - [\bar{x}_j])/\bar{a}_{ji_D(j)} \begin{Bmatrix} \leqq \\ \geqq \end{Bmatrix} 1 \Rightarrow V_D^{*1}(j) \begin{Bmatrix} \geqq \\ \leqq \end{Bmatrix} V_D^{*2}(j),$$
$$([\bar{x}_j] + 1 - \bar{x}_j)/(-\bar{a}_{ji_U(j)}) \begin{Bmatrix} \leqq \\ \geqq \end{Bmatrix} 1 \Rightarrow V_U^{*1}(j) \begin{Bmatrix} \geqq \\ \leqq \end{Bmatrix} V_U^{*2}(j). \tag{13}$$

We are unable to supply a reference to the conditional bounds (12) in the published literature. However, Armstrong and Sinha [1] have independently and very recently proposed a precisely analogous strengthening of (8) for the mixed integer 0–1 case. They report favorable computational experience.

So far we have required $Bx \geqq d$ to be vacuous; that is, all upper bounds and other special constraints are treated as general A-type constraints. Analogs of the previous penalty results as well as new penalty results emerge easily by allowing $Bx \geqq d$ to be nonvacuous. This will now be illustrated for the three examples.

4.2. *Penalties for Example* 1

Example 1 differs from the previous development only in that ($\text{CPR}_{\overline{\lambda}}$) now has upper-bounded variables. As indicated in Section 3, it can be shown that both $V_{\text{D}}(j)$ and $V_{\text{U}}(j)$ equal $v(\overline{\text{CP}})$ for all $j \in I_f$ due to the vanishing of the corresponding objective function coefficients in ($\text{CPR}_{\overline{\lambda}}$). The remedy for these vanishing penalties is again to invoke the representation for x_j which is available from the final LP tableau of ($\overline{\text{CP}}$). This representation will be written as

$$x_j = \alpha_{j0} - \sum_{i \neq j} \alpha_{ji} x_i \quad \text{for } j \in I_f, \tag{14}$$

where, of course, many of the coefficients α_{ji} may be 0. The resulting strengthened conditional lower bounds on $v(\text{CP} \mid x_j \leqq [\overline{x}_j])$ and $v(\text{CP} \mid x_j \geqq [\overline{x}_j] + 1)$ for $j \in I_f$ are

$$\begin{aligned} V_{\text{D}}^*(j) &\triangleq v(\text{CPR}_{\overline{\lambda}} \mid x_j = \alpha_{j0} - \textstyle\sum_{i \neq j} \alpha_{ji} x_i \leqq [\overline{x}_j]), \\ V_{\text{U}}^*(j) &\triangleq v(\text{CPR}_{\overline{\lambda}} \mid x_j = \alpha_{j0} - \textstyle\sum_{i \neq j} \alpha_{ji} x_i \geqq [\overline{x}_j]) + 1). \end{aligned} \tag{15}$$

We have used the notations V_{D}^* and V_{U}^* as in (5) because (15) is an exact counterpart of (5). Like (5), (15) could be too expensive computationally because each estimate requires solving a knapsack-type problem in integer variables. The fact that the knapsack problem now has upper-bounded variables is a dubious advantage. The most easily computed lower approximation to (15) is obtained by dropping the integrality requirements as in (6):

$$\begin{aligned} V_{\text{D}}^{*0}(j) &\triangleq v(\overline{\text{CPR}}_{\overline{\lambda}} \mid x_j = \alpha_{j0} - \textstyle\sum_{i \neq j} \alpha_{ji} x_i \leqq [\overline{x}_j]), \\ V_{\text{U}}^{*0}(j) &\triangleq v(\overline{\text{CPR}}_{\overline{\lambda}} \mid x_j = \alpha_{j0} - \textstyle\sum_{i \neq j} \alpha_{ji} x_i \geqq [\overline{x}_j] + 1). \end{aligned} \tag{16}$$

The notations V_D^{*0} and V_U^{*0} have again been carried over. The differences

$$V_U^{*0}(j) - v(\overline{\overline{CP}}), \qquad v_D^{*0}(j) = v(\overline{CP}) \tag{17}$$

are Driebeek-like up and down penalties for Example 1. The computation of (16) requires only slightly more effort than the computation of (6). A "continuous knapsack" problem with upper-bounded variables must now be solved. Explicit formulae for V_D^{*0} and V_U^{*0} are slightly more cumbersome than expression (7), but are easily programmed for a computer.

To strengthen (16) one may formally write the counterpart of (8), but unfortunately explicit calculation may be nearly as costly as that of (15) itself. This is because the upper bounds generally invalidate the key property of (8) that at most one variable need be at a positive level in an optimal solution of each associated optimization problem. Thus the strengthened penalties of Tomlin do not generalize usefully to $Bx \geqq d$ when it includes upper bounds on variables.

But the other technique based on Theorem 5 for strenthening $V_D^{*0}(j)$ and $V_U^{*0}(j)$ *does* generalize nicely. Let $i_D(j)$ and $i_U(j)$ be respectively the fractional-valued variables in the solutions of the optimizations corresponding to $V_D^{*0}(j)$ and $V_U^{*0}(j)$. It is easy to see that at most one variable need be fractional in each of these solutions; if none is, then that penalty cannot be strengthened by the present device. The strengthened conditional bounds analogous to (12) are:

$$\begin{aligned} V_D^{*2}(j) &\overset{d}{=} v(\overline{\overline{CP}}R_{\overline{x}} \mid x_j = \alpha_{j0} - \textstyle\sum_{i \neq j} \alpha_{ji} x_i \leqq [\overline{x}_j] \text{ and } x_{i_D(j)} \text{ integer}), \\ V_U^{*2}(j) &\overset{d}{=} v(\overline{\overline{CP}}\overline{R}_{\overline{x}} \mid x_j = \alpha_{j0} - \textstyle\sum_{i \neq j} \alpha_{ji} x_i \\ &\qquad \geqq [\overline{x}_j] + 1 \text{ and } x_{i_U(j)} \text{ integer}). \end{aligned} \tag{18}$$

The required optimizations are inexpensive to carry out. Clearly,

$$\begin{aligned} V_D^{*0}(j) \leqq V_D^{*2}(j) \leqq V_D^{*}(j) \leqq v(CP \mid x_j \leqq [\overline{x}_j]), \\ V_U^{*0}(j) \leqq V_U^{*2}(j) \leqq V_U^{*}(j) \leqq v(CP \mid x_j \geqq [\overline{x}_j] + 1). \end{aligned} \tag{19}$$

Exactly the same types of penalties can be constructed for $j \in I - I_f$ when the objective function coefficient of x_j vanishes in $(CPR_{\overline{x}})$.

4.3. *Penalties for Example* 2

The development of penalties for Example 2 closely parallels that for Example 1. For $j \in I_f$, both up and down penalties again vanish, and it is

necessary to use the final LP tableau representation of the form (14).[3] The resulting conditional bounds $V_D^*(j)$ and $V_U^*(j)$ defined in (15) may still be too expensive computationally to use in general, but the multiple choice constraints do tend to make the computation easier by comparison with Example 1. There are nontrivial situations where $V_D^*(j)$ and $V_U^*(j)$ can be computed relatively economically by a simple enumerative procedure. But in general one may have to fall back on the Driebeek-like penalties defined by (16). The required computations are no longer simple continuous knapsack problems with upper-bounded variables, but they can still be carried out efficiently by specialized techniques (e.g. by parametric optimization applied to the dual of $(\overline{\text{CPR}}_{\bar{\lambda}})$ with respect to the added constraint). Strengthening these penalties along the lines suggested by Tomlin as in (8) appears to be no easier in general than (15) itself. But again, as with Example 1, the strengthening of (18) based on Theorem 5 is attractive. The indices $i_D(j)$ and $i_U(j)$ may be selected to be any of the fractional-valued variables in the solutions of the optimizations corresponding to $V_D^{*0}(j)$ and $V_U^{*0}(j)$. The implementation of (18) on a computer is only slightly more expensive than that of (16). Naturally, (19) continues to hold. The reader should have no difficulty seeing what to do if penalties are desired for variables in $I - I_f$.

The special nature of the multiple choice constraints (1) makes it possible to define "cumulative" conditional bounds on the "upward" problems as follows:

$$V_U^{*0}(j; J_k) \triangleq \max \{ V_U^{*0}(j), V_D^{*0}(i) \quad \text{for } i \in \{J_k - j\} \} \tag{20}$$

where it is understood that j is in J_k in these definitions. That this provides true lower bounds on $v(\text{CP} \mid x_j = 1)$ follows from the fact that $x_j = 1$ implies $x_i = 0$ for all $i \neq j$ in the same multiple choice set. Similar cumulative bounds hold for V^{*2} and V^*.

4.4. *Penalties for Example* 3

For Example 3 we must distinguish between the "switching" (x_k) versus the "nonswitching" variables in I_f. The up and down penalties associated with $V_D(j)$ and $V_U(j)$ are highly unlikely to vanish for the fractional switching variables. In fact, one can argue that they are likely to be quite large. Our experience with the practical facility location problems mentioned at the end of Section 2 has been that these penalties tend to be at least *an order of*

[3] Numbered displays from the discussion of Example 1 will be used here with the understanding that (CP), etc., have the structure of Example 2 rather than Example 1.

magnitude greater than the standard Tomlin penalties when $\bar{\lambda}$ is used, and yet take less time to compute [15]. For the nonswitching variables in I_f, however, it is easy to see that the naive penalties vanish and thus that alternative representations from the final LP tableau may be useful. The detailed discussion would be so close to that for Example 2 that it will not be given here.

5. Surrogate constraints

Consider the case where (P) is a pure 0–1 integer program with $Bx \geqq d$ consisting solely of unit upper bounds on all variables. The present author proposed [12] the use of "surrogate" constraints (after Glover [16]) of the form

$$cx + \lambda(b - Ax) < z^*, \tag{21}$$

with the prescription that $\lambda \geqq 0$ be chosen as the optimal dual vector corresponding to $Ax \geqq b$ in $(\overline{\mathrm{P}})$ or some $(\overline{\mathrm{CP}})$. Clearly such a constraint must be satisfied by every feasible solution to (P) with lower objective value than that of the incumbent. Two uses of this type of surrogate constraint were proposed in connection with the examination of a typical candidate problem: as a possible means of fathoming via the easy test

$$\underset{x=0,1}{\text{minimum}} \{cx + \lambda(b - Ax)\} \overset{?}{\geqq} z^* \tag{22}$$

and as a possible means of range reduction via the following easy tests applied to a typical (say the jth) variable:

$$\underset{x=0,1}{\text{minimum}} \{cx + \lambda(b - Ax) \text{ s.t. } x_j = 0\} \overset{?}{\geqq} z^*, \tag{23a}$$

$$\underset{x=0,1}{\text{minimum}} \{cx + \lambda(b - Ax) \text{ s.t. } x_j = 1\} \overset{?}{\geqq} z^*. \tag{23b}$$

If (22) holds, then (P) is fathomed. If (23a) [resp. (23b)] holds, then x_j must be 1 [resp. 0] in any feasible solution of (P) which is superior in value to the current incumbent. It is understood, naturally, that all separation constraints must also be honored in taking the minima in (22) and (23) when examining a candidate problem subsequent to (P). If all separation constraints involve only additional range restrictions on the variables, as is usually the case, then (22) and (23) remain computationally trivial.

It is easy to interpret (22) and (23) from the viewpoint of Lagrangean relaxation (remember that $Bx \geqq d$ consists of just the upper bound constraints $x_j \leqq 1$). Test (22) can be rewritten as

$$v(\mathrm{PR}_\lambda) \geqq z^*, \tag{24}$$

which is precisely the elementary fathoming criterion normally associated with (PR_λ). Similarly, (23) can be rewritten as

$$v(\mathrm{PR}_\lambda \mid x_j = 0) \geqq z^*, \tag{25a}$$

$$v(\mathrm{PR}_\lambda \mid x_j = 1) \geqq z^*. \tag{25b}$$

This is precisely the ordinary range reduction criterion described in Section 3. And the injunction to obtain λ from the usual linear programming relaxation is a consequence of Theorem 2, which implies that the strongest tests are obtained in this way.

Thus the surrogate constraint (21) and the tests based on it are seen to be completely subsumed by the simplest Lagrangean relaxation techniques for the special case of Example 1. Generalizations of (21)–(23) when (P) is not a pure 0-1 program or when $Bx \geqq d$ includes more than simple upper bounds can be obtained without difficulty. Some such generalizations were developed several years ago by this author in unpublished lecture notes and by Glover [17] using the surrogate constraint viewpoint, but in each case the same results may be obtained easily as special cases of more general and powerful results based on Lagrangean relaxation.

6. Cutting planes

For present purposes, a *cutting-plane* is any linear inequality which must be satisfied by all of the feasible solutions of a candidate problem but is violated by an optimal solution of its usual linear programming relaxation $(\overline{\mathrm{CP}})$. Appending cutting-planes to $(\overline{\mathrm{CP}})$ makes it a tighter relaxation of (CP) and thereby yields better bounds for use in a hybrid branch-and-bound algorithm (cf. [14, Sec. IV]). Cutting-planes may also, of course, be used in a purely cutting-plane approach.

This section explores the uses of Lagrangean relaxation as a source of cutting-planes. For notational convenience we only consider cuts relative to the initial candidate problem (P) itself. It is a simple matter to apply the ideas developed below to any candidate problem.

The simplest type of cutting-plane for (P) is (here $\lambda \geqq 0$)

$$v(\mathrm{PR}_\lambda) \leqq cx + \lambda(b - Ax). \tag{26}$$

A special case of this cut was proposed by Shapiro [27], who showed that it can be at least as strong as all of the cuts in a well-known group theoretic

class. The validity of this constraint for any feasible solution of (P) follows from the definition of $v(\mathrm{PR}_\lambda)$ and the fact that the feasible region of (P) is contained in that of (PR_λ). It will be violated at $\bar{x}$, an optimal solution of $(\bar{\mathrm{P}})$, if $v(\mathrm{PR}_\lambda) > v(\bar{\mathrm{P}})$ holds, because $\lambda \geqq 0$ and $A\bar{x} \geqq b$ imply

$$v(\bar{\mathrm{P}}) = c\bar{x} \geqq c\bar{x} + \lambda(b - A\bar{x}).$$

The condition $v(\mathrm{PR}_\lambda) > v(\bar{\mathrm{P}})$ will hold when $v(\bar{\mathrm{P}}) < v(\mathrm{D})$ and λ is sufficiently near optimal in (D). Of course this condition is impossible when the Integrality Property holds; in fact, the Integrality Property implies that (26) cannot be violated by any solution of $(\bar{\mathrm{P}})$ whatever, because then

$$v(\mathrm{PR}_\lambda) = v(\overline{\mathrm{PR}}_\lambda) \leqq cx + \lambda(b - Ax)$$

for all x feasible in $(\overline{\mathrm{PR}}_\lambda)$ and thus for all x feasible in $(\bar{\mathrm{P}})$. Thus (26) can be a true cutting-plane only when the Integrality Property does not hold. Appending it to $(\bar{\mathrm{P}})$ must increase the optimal value of $(\bar{\mathrm{P}})$ at least to $v(\mathrm{PR}_\lambda)$ because (26) implies

$$cx \geqq v(\mathrm{PR}_\lambda) - \lambda(b - Ax) \geqq v(\mathrm{PR}_\lambda) \quad \text{for all } x \text{ feasible in } (\bar{\mathrm{P}}).$$

An improvement of (26) is obtained by replacing $v(\mathrm{PR}_\lambda)$ with $v(\mathrm{PR}_\lambda \mid x_{j_1}, \ldots, x_{j_P})$, which denotes the optimal value of (PR_λ) as a function of specified values for the distinguished *cut variables* $x_{j_1}, \ldots, x_{j_P}$. (If the values of the cut variables are such that no completion exists which is feasible in (PR_λ)—e.g., if an integer cut variable takes on a fractional value—then by convention, $v(\mathrm{PR}_\lambda \mid x_{j_1}, \ldots, x_{j_P})$ is defined to be $+\infty$ at such a point.) The constraint

$$v(\mathrm{PR}_\lambda \mid x_{j_1}, \ldots, x_{j_P}) \leqq cx + \lambda(b - Ax) \tag{27}$$

is valid by an argument similar to that for (26) and is uniformly at least as tight because

$$v(\mathrm{PR}_\lambda) \leqq v(\mathrm{PR}_\lambda \mid x'_{j_1}, \ldots, x'_{j_P}) \tag{28}$$

obviously holds for every feasible solution x' of (P). Strict inequality holds in (28) except when $x'_{j_1}, \ldots, x'_{j_P}$ happens to be part of an optimal solution of (PR_λ). This fact also renders (27) less susceptible to neutralization by the Integrality Property.

The difficulty with (27), of course, is that $v(\mathrm{PR}_\lambda \mid x_{j_1}, \ldots, x_{j_P})$ need not be a linear function. It depends upon the structure of (PR_λ) and the choice of cut variables. One source of nonlinearity has to do with the domain on which it is $+\infty$. Fortunately, (27) need only hold for *feasible* solutions of

(P), and so $v(\mathrm{PR}_\lambda \mid x_{j_1}, \ldots, x_{j_P})$ can be redefined arbitrarily wherever it is $+\infty$. It is clear that this redefinition should be linearly interpolative in nature. Of course, this still may not render (27) linear. It may be necessary to determine a linear lower bounding function $l_\lambda(x_{j_1}, \ldots, x_{j_P})$,

$$l_\lambda(x_{j_1}, \ldots, x_{j_P}) \leqq v(\mathrm{PR}_\lambda \mid x_{j_1}, \ldots, x_{j_P}) \quad \text{for all } x \text{ feasible in (P).} \tag{29}$$

Clearly, l_λ should be as "tight" as possible, especially in the vicinity of $\bar{x}$. Thus the linear constraint to be appended to $(\overline{\mathrm{P}})$ is of the form

$$l_\lambda(x_{j_1}, \ldots, x_{j_P}) \leqq c\,x + \lambda(b - A\,x), \tag{30}$$

where $(j_1, \ldots, j_P)$ is an arbitrary set of cut variable indices, $\lambda \geqq 0$, and (29) must hold.

The above ideas can be illustrated with reference to the three examples of Section 1. Examples 1 and 2 satisfy the Integrality Property and so constraint (26) cannot be violated at $\bar{x}$. Furthermore, it can be shown for these examples that no constraint of the form (30) can be violated at $\bar{x}$, no matter what λ or cut variables are chosen. Example 3, on the other hand, does lend itself to the derivation of useful cutting-planes. Cut (26) tends to be quite good, even when $\bar{\lambda}$, an immediate by-product of $(\overline{\mathrm{P}})$, is used. We see from the computational experience cited at the end of Section 2 that, in the four practical problems studied, a single cut of the form (26) with $\lambda = \bar{\lambda}$ raised the optimal value of $(\overline{\mathrm{P}})$ an average of at least 59.6% of the distance from $v(\overline{\mathrm{P}})$ to $v(\mathrm{P})$. If the effort to find an optimal λ were expended, (26) would raise the optimal value an average of at least 97.1% of the way to $v(\mathrm{P})$. It should also be noted that a cut of the form (30) is available as an immediate by-product of the evaluation of (PR_λ). Recall that (PR_λ) separates into independent subproblems of the form (3^k_λ):

$$\begin{aligned} v(\mathrm{PR}_\lambda) = \lambda\,b + \sum_{k=1}^{K} v(3^k_\lambda) \\ + \underset{x_j,\ j \in T}{\text{minimum}} \{ \textstyle\sum_{j \in T} (c - \lambda A)\,x_j \text{ s.t. } 0 \leqq x_j \leqq u_j, \\ j \in T \text{ and } x_j \text{ integer}, j \in T \cap I \}, \end{aligned} \tag{31}$$

where T comprises the indices of all variables not appearing in any of the subproblems of type (3^k_λ). To evaluate $v(\mathrm{PR}_\lambda)$ one makes use of the fact that

$$v(3^k_\lambda) = \min \{ v(3^k_\lambda \mid x_k = 0), v(3^k_\lambda \mid x_k = 1) \} \tag{32}$$

and of the fact that the last term involving $j \in T$ in (31) is trivially evaluated

by inspection. Thus the binary variables x_k are obvious choices for cut variables. We have

$$v(\mathrm{PR}_\lambda \mid x_1, \ldots, x_K) = \mathrm{CON}_\lambda + \sum_{k=1}^{K} v(3_\lambda^k \mid x_k), \tag{33}$$

where the constant CON_λ equals the first and last terms of (31). The binary nature of the variables makes it easy to write down a linear function l_λ satisfying (29) with equality in this case:

$$\mathrm{CON}_\lambda + \sum_{k=1}^{K} v(3_\lambda^k \mid x_k = 1)\, x_k = v(\mathrm{PR}_\lambda \mid x_1, \ldots, x_K)$$

for all binary $(x_1, \ldots, x_K)$. Thus (30) becomes

$$\mathrm{CON}_\lambda + \sum_{k=1}^{K} v(3_\lambda^k \mid x_k = 1)\, x_k \leqq c\,x + \lambda\,(b - A\,x). \tag{34}$$

Our experience with the same four practical problems as mentioned above is that a single cut of this type raised $v(\overline{\mathrm{P}})$ an average of 69.7% of the way from $v(\overline{\mathrm{P}})$ to $v(\mathrm{P})$ when $\bar{\lambda}$ was used [15].

The derivation of a type (30) cut for Example 3 generalizes easily to the frequent situation where (PR_λ) separates into a number of independent subproblems involving 0–1 variables. Suppose

$$v(\mathrm{PR}_\lambda) = \lambda\, b + \sum_{k=1}^{P} v(\mathrm{PR}_\lambda^k),$$

where (PR_λ^k) involves the variables J_k ($J_1, \ldots, J_P$ is a mutually exclusive and exhaustive partition) among which is a 0–1 variable j_k. Suppose further that both $v(\mathrm{PR}_\lambda^k \mid x_{j_k} = 0)$ and $v(\mathrm{PR}_\lambda^k \mid x_{j_k} = 1)$ can be obtained inexpensively in the course of evaluating $v(\mathrm{PR}_\lambda^k)$. Then j_k is a natural choice for a cut variable and a type (30) constraint is

$$\lambda\, b + \sum_{k=1}^{P} v(\mathrm{PR}_\lambda^k \mid x_{j_k} = 0)(1 - x_{j_k}) + v(\mathrm{PR}_\lambda^k \mid x_{j_k} = 1)\, x_{j_k} \leqq$$
$$\leqq c\,x + \lambda\,(b - A\,x). \tag{35}$$

We have made use of the relations

$$v(\mathrm{PR}_\lambda \mid x_{j_1}, \ldots, x_{j_P}) = \lambda\, b + \sum_{k=1}^{P} v(\mathrm{PR}_\lambda^k \mid x_{j_k}),$$

$$v(\mathrm{PR}_\lambda^k \mid x_{j_k}) = v(\mathrm{PR}_\lambda^k \mid x_{j_k} = 0)(1 - x_{j_k}) + v(\mathrm{PR}_\lambda^k \mid x_{j_k} = 1)\, x_{j_k} \quad \text{for } x_{j_k} = 0,1.$$

The latter relation furnishes the required l_λ function with equality in (29). Constraint (35) is likely to improve on the counterpart of (26), namely

$$\lambda b + \sum_{k=1}^{P} v(\mathrm{PR}_\lambda^k) \leqq c x + \lambda (b - A x), \tag{36}$$

because

$$\begin{aligned} v(\mathrm{PR}_\lambda^k) &= \min \{ v(\mathrm{PR}_\lambda^k \mid x_{j_k} = 0),\ v(\mathrm{PR}_\lambda^k \mid x_{j_k} = 1) \} \\ &\leqq v(\mathrm{PR}_\lambda^k \mid x_{j_k} = 0)(1 - x_{j_k}) \\ &\quad + v(\mathrm{PR}_\lambda^k \mid x_{j_k} = 1)\, x_{j_k} \quad \text{for } 0 \leqq x_{j_k} \leqq 1. \end{aligned}$$

It should also be pointed out that (35) and (36) can be decomposed into P component inequalities:

$$v(\mathrm{PR}_\lambda^k \mid x_{j_k} = 0)(1 - x_{j_k}) + v(\mathrm{PR}_\lambda^k \mid x_{j_k} = 1)\, x_{j_k} \leqq \sum_{j \in J_k} (c - \lambda A)_j x_j, \tag{35_k}$$

$$v(\mathrm{PR}_\lambda^k) \leqq \sum_{j \in J_k} (c - \lambda A)_j x_j. \tag{36_k}$$

The validity of (35_k) and (36_k) should be evident. Their sum over all k yields (35) and (36), respectively.

Other types of cutting-planes can be devised with the help of the penalty formulae of Section 4. In particular, useful cutting-planes for Examples 1 and 2 can be determined (recall that neither (26) nor (30) were useful in this context). Both the simple penalties based on (16) and the strengthened penalties based on (18) can be used to generate cuts violated by $\bar{x}$ so long as at least one of these penalties is nonzero. This may be done as follows. Consistency of notation requires that we let (CP) equal (P) when applying the results of Section 4.

Consider first the simple conditional bounds (16). Select any $j \in I_f$ such that at least one of the penalties is strictly positive and take this j to be the one cut variable. Clearly

$$\begin{aligned} &v(\mathrm{PR}_\lambda \mid x_j = \alpha_{j0} - \textstyle\sum_{i \neq j} \alpha_{ji} x_i) \leqq \\ &\leqq c x + \bar{\lambda} \leqq c x + \bar{\lambda}(b - A x) \quad \text{for all } x \text{ feasible in (P).} \end{aligned} \tag{37}$$

The left-hand side of (37) is convex as a function of x_j, and thus the unique linear function passing through it at the points $[\bar{x}_j]$ and $[\bar{x}_j] + 1$ does not overestimate it for any integer value of x_j:

$$\begin{aligned} &V_{\mathrm{D}}^{*0}(j) + (V_{\mathrm{U}}^{*0}(j) - V_{\mathrm{D}}^{*0}(j))(x_j - [\bar{x}_j]) \leqq \\ &\leqq v(\overline{\mathrm{PR}}_{\bar{\lambda}} \mid x_j = \alpha_{j0} - \sum_{i \neq j} \alpha_{ji} x_i) \quad \text{for all integer } x_j. \end{aligned} \tag{38}$$

Together, (37) and (38) imply that

$$V_D^{*0}(j) + (V_U^{*0}(j) - V_D^{*0}(j))(x_j - [\bar{x}_j]) \leqq c\,x + \bar{\lambda}\,(b - A\,x) \quad (39)$$

is a legitimate cut. Notice that if there are several $j \in I_f$ for which $V_D^{*0}(j)$ and $V_U^{*0}(j)$ are computed, then it is an easy matter to select j so as to yield the cut of type (39) which is most violated by $\bar{x}$.

Now consider the strengthened conditional bounds (18). An analog of inequality (37) holds, but the analog of (38) does not because of the added integrality requirement in (18). It appears necessary to require that $j \in I_f$ be a 0–1 variable if a cut is to be based on (18) with j as the single cut variable. Then

$$V_D^{*2}(j) + (V_U^{*2}(j) - V_D^{*2}(j))\,x_j \leqq c\,x + \bar{\lambda}\,(b - A\,x) \quad (40)$$

is a legitimate cut. By (19), (40) is clearly a superior cut to (39). It is a simple matter to select j so as to yield the cut which is deepest at $\bar{x}$ among those of the form (40).

For Example 2 one should of course use in (39) and (40) the cumulative penalties defined in (20) in place of $V_U^{*0}(j)$ or $V_U^{*2}(j)$ if the necessary quantities are at hand. One may further improve cuts (39) and (40) when j is a multiple choice variable by using one of the obvious cuts

$$\sum_{j \in J_k} V_U^{*0}(j; J_k)\,x_j \leqq c\,x + \bar{\lambda}\,(b - A\,x), \qquad k = 1, \ldots, K \quad (41)$$

or the still stronger cuts

$$\sum_{j \in J_k} V_U^{*2}(j; J_k)\,x_j \leqq c\,x + \bar{\lambda}\,(b - A\,x), \qquad k = 1, \ldots, K. \quad (42)$$

Each of these cuts takes all of J_k as the set of cut variables. It is easy to verify that cuts of the form (41) [resp. (42)] are at least as strong as those of the form (39) [resp. (40)] for all x feasible in (P).

A cut similar to (41) was proposed by Healy [19]. To be precise, for the kth cut he omitted the term $\bar{\lambda}\,(b - A\,x)$ and used $V_U^{*0}(j)$ as the coefficient of x_j, where $V_U^{*0}(j)$ is computed with $B\,x \geqq d$ taken to consist of only the kth multiple choice constraint (no upper bounds or other multiple choice constraints are included). This cut is dominated by (41).

We note in closing that penalty-based cuts with more than one cut variable can often be obtained for Examples 1 and 2 and other structures by: (i) adding to ($\mathrm{PR}_{\bar{\lambda}}$) relations of the form (14) for any subset of j's in I_f so long as no variable appears with a nonzero coefficient in more than one of these relations, and then (ii) exploiting separability.

7. Conclusion

Lagrangean relaxation is a systematic exploitation of the formal Lagrangean dual problem in integer programming. This dual problem need not be solved optimally and need not be devoid of a duality gap in order to be useful. It provides a means for fathoming, range reduction, generating improved feasible solutions, and guiding separation (Sec. 3). It also provides new penalties (Sec. 4) and cutting-planes (Sec. 6) and supplants the narrower notion of surrogate constraints (Sec. 5). All of these functions usually can be tailored to the special structure of the particular problem class at hand, beginning with the judicious choice of the subset of constraints to play the role of $Bx \geqq d$. This has been carried out in detail for three of the simplest structures. Some of the uses of Lagrangean relaxation have been explored by other authors for several more complex structures [6], [7], [8], [9], [10], [20], [21], [26]. Yet it remains to work out the full import of Lagrangean relaxation even for these structures and for many others of importance. It is hoped that the framework of this paper will facilitate this effort and encourage new applications.

Acknowledgment

This paper has benefited from careful readings by Marshall L. Fisher and Roy E. Marsten.

References

[1] R.D. Armstrong and P. Sinha, "Improved penalty calculations for a mixed integer branch-and-bound algorithm", *Mathematical Programming* 6 (1974) 212–223.

[2] R. Brooks and A. Geoffrion, "Finding Everett's Lagrange multipliers by linear programming", *Operations Research* 14 (1966) 1149–1153.

[3] Dakin, R.J., "A tree search algorithm for mixed integer programming problems", *Computer Journal*, 8 (1965) 250–255.

[4] N.J. Driebeek, "An algorithm for the solution of mixed integer programming problems", *Management Science* 12 (1966) 576–587.

[5] Everett, H.M., "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources", *Operations Research* 11 (1966) 399–417.

[6] M.L. Fisher, "Optimal solution of scheduling problems using Lagrange multipliers: Part I", *Operations Research* 21 (1973) 1114–1127.

[7] M.L. Fisher, "A dual algorithm for the one-machine scheduling problem", Graduate School of Business Rept., University of Chicago, Chicago, Ill. (1974).

[8] M.L. Fisher, W.D. Northup and J.F. Shapiro, "Using duality to solve discrete optimization problems: Theory and computational experience", Working Paper OR 030-74, Operations Research Center, M.I.T. (1974).

[9] M.L. Fisher and L. Schrage, "Using Lagrange multipliers to schedule elective hospital admissions", Working Paper, University of Chicago, Chicago, Ill. (1972).
[10] M.L. Fisher and J.F. Shapiro, "Constructive duality in integer programming", *SIAM Journal on Applied Mathematics*, to appear.
[11] J.J.H. Forrest, J.P.H. Hirst and J.A. Tomlin, "Practical solution of large mixed integer programming problems with UMPIRE", *Management Science* 20 (1974) 733–773.
[12] A.M. Geoffrion, "An improved implicit enumeration approach for integer programming", *Operations Research* 17 (1969) 437–454.
[13] A.M. Geoffrion, "Duality in nonlinear programming", *SIAM Review* 13 (1971) 1–37.
[14] A.M. Geoffrion and R.E. Marsten, "Integer programming algorithms: A framework and state-of-the-art survey", *Management Science* 18 (1972) 465–491.
[15] A.M. Geoffrion and R.D. McBride, "The capacitated facility location problem with additional constraints", paper presented to the Joint National Meeting of AIIE, ORSA, and TIMS, Atlantic City, November 8–10, 1972.
[16] F. Glover, "A multiphase-dual algorithm for the zero-one integer programming problem", *Operations Research* 13 (1965) 879–919.
[17] F. Glover, "Surrogate constraints", *Operations Research* 16 (1968) 741–749.
[18] H.J. Greenberg and T.C. Robbins, "Finding Everett's Lagrange multipliers by Generalized Linear Programming, Parts I, II, and III", Tech. Rept. CP-70008, Computer Science/Operations Research Center, Southern Methodist University, Dallas, Tex. revised (June 1972).
[19] W.C. Healy, Jr., "Multiple choice programming", *Operations Research* 12 (1964) 122–138.
[20] M. Held and R.M. Karp, "The traveling salesman problem and minimum spanning trees", *Operations Research* 18 (1970) 1138–1162.
[21] M. Held and R.M. Karp, "The traveling salesman problem and minimum spanning trees: Part II", *Mathematical Programming* 1 (1971) 6–25.
[22] M. Held, P. Wolfe and H.P. Crowder, "Validation of subgradient optimization", Mathematical Sciences Department, IBM Watson Research Center, Yorktown Heights, N.Y. (August 1973).
[23] W.W. Hogan, R.E. Marsten and J.W. Blankenship, "The BOXSTEP method for large scale optimization", Working Paper 660-73, Sloan School of Management, M.I.T. (December 1973).
[24] R.E. Marsten, private communication (August 22, 1973).
[25] G.L. Nemhauser and Z. Ullman, "A note on the generalized Lagrange multiplier solution to an integer programming problem", *Operations Research* 16 (1968) 450–452.
[26] G.T. Ross and R.M. Soland, "A branch and bound algorithm for the generalized assignment problem", *Mathematical Programming*, to appear.
[27] J.F. Shapiro, "Generalized Lagrange multipliers in integer programming", *Operations Research* 19 (1971) 68–76.
[28] J.A. Tomlin, "An improved branch and bound method for integer programming", *Operations Research* 19 (1971) 1070–1075.
[29] J.A. Tomlin, "Branch and bound methods for integer and non-convex programming", in: J. Abadie, ed., *Integer and nonlinear programming* (North-Holland, Amsterdam, 1970).
[30] A.F. Veinott and G.B. Dantzig, "Integral extreme points", *SIAM Review* 10 (1968) 371–372.

Mathematical Programming Study 2 (1974) 115–136. North-Holland Publishing Company

A HEURISTIC ALGORITHM FOR MIXED-INTEGER PROGRAMMING PROBLEMS*

Toshihide IBARAKI
Kyoto University, Kyoto, Japan.

Tateaki OHASHI
Sumitomo Electric Industry, LTD., Osaka, Japan.

Hisashi MINE
Kyoto University, Kyoto, Japan.

Received 6 November 1972
Revised manuscript received 24 April 1974

A heuristic algorithm for solving mixed-integer programming problems is proposed. The basic idea is to search good feasible solutions located near the LP optimal solution. It consists of four phases : Phase 0, computation of LP optimal solution; Phase 1, computation of the central trajectory T of the feasible region; Phase 2, search for (integer) feasible solutions along T; Phase 3, improvements of feasible solutions. The computational results are encouraging. For example, randomly generated problems with 50 constraints and 400 variables consumed $2 \sim 3$ minutes on a FACOM 230/60. The quality of the obtained solutions seem to be quite high. In fact, for many problems with known optimal solutions, our algorithm was successful in obtaining exact optimal solutions.

1. Introduction

Motivated by the limited success of integer programming algorithms guaranteeing optimal solutions, various heuristic algorithms have been investigated by Reiter and Rice [12], Echols and Cooper [2], Senju and Toyoda [14], Roth [13], Hillier [5] and possibly by others. A heuristic algorithm aims at obtaining a good feasible solution relatively quickly.

The results obtained by Hillier [5] motivated our research. His algorithm deals with all-integer programs and, for example, could obtain suboptimal solutions of problems of the size 60×300 (constraints $\times$ variables) in about 5 minutes on an IBM 360/67. The quality of the obtained solutions seems to be quite high.

* This paper is a slightly shortened version of the working paper [8], which is available from the authors. A FORTRAN list of the entire code is also available upon request.

In this paper, we will give a heuristic algorithm for mixed-integer programs. The basic idea is to search integer feasible solutions located near the LP optimal solution. It is similar to Hillier's but differs in many respects. The main difference consists in the search method for obtaining initial integer feasible solutions, which will be explained later. The computational results for various test problems are encouraging. For example, solutions of problems of the size 50 × 400 with 50 -400 integer variables are obtained in 2 3 minutes on a FACOM 230/60. Judging from the computational experience for test problems with known optimal solutions, the obtained integer feasible solutions are usually very close to (and frequently even equal to) exact optimal solutions. It is concluded that the heuristic approach of this type can be considered as one of the most promising practical algorithms for mixed-integer programs.

2. Preliminaries

Let a mixed-integer programming (MIP) problem P be written as follows:

$$\text{P:} \quad \text{maximize } z = \sum_{j=1}^{n_1} c_j x_j + \sum_{j=1}^{n_2} e_j y_j, \tag{1}$$

$$\text{subject to } x_{n_1+i} = b_i - \sum_{j=1}^{n_1} a_{ij} x_j - \sum_{j=1}^{n_2} d_{ij} y_j, \qquad i = 1, 2, \ldots, m, \tag{2}$$

$$x_j \geqslant 0, \qquad j = 1, 2, \ldots, n_1 + m, \tag{3}$$

$$y_j \geqslant 0, \qquad j = 1, 2, \ldots, n_2, \tag{4}$$

$$y_j \text{ integer}, \qquad j = 1, 2, \ldots, n_2, \tag{5}$$

where x_{n_1+i}, $i = 1, 2, \ldots, m$ are slack variables. Let

$$n = n_1 + n_2. \tag{6}$$

Without loss of generality we assume that coefficients in (2) are normalized so that

$$\sum_{j=1}^{n_1} a_{ij}^2 + \sum_{j=1}^{n_2} d_{ij}^2 = 1, \qquad i = 1, 2, \ldots, m \tag{7}$$

holds.

Let $\bar{\text{P}}$ denote the linear programming (LP) problem obtained from P

by neglecting the integer constraint (5). Let $\eta = (\eta_1, \eta_2, \ldots, \eta_{n_2})$, where η_i are nonnegative integers, and let

$$b_i(\eta) = b_i - \sum_{j=1}^{n_2} d_{ij}\eta_j, \qquad i = 1, 2, \ldots, m. \tag{8}$$

$\bar{\mathrm{P}}(\eta)$ denotes the LP problem $\bar{\mathrm{P}}$ with each b_i replaced by $b_i(\eta)$. In addition, if all variables y_j are fixed to 0 in $\bar{\mathrm{P}}(\eta)$, the resulting problem is denoted $\mathrm{P}(\eta)$. $\mathrm{P}(\eta)$ is the LP problem obtained from P by fixing each integer variable y_j to η_j.

In describing our algorithm, it is sometimes required to use the simplex tableau of $\bar{\mathrm{P}}(\eta)$ with integer variables y_j being restricted to be nonbasic (i.e., fixed to 0). Let continuous variables $x_1, x_2, \ldots, x_{n_1+m}$ be partitioned into basic variables $u_1, u_2, \ldots, u_m$ and nonbasic variables $t_1, t_2, \ldots, t_{n_1}$. The corresponding simplex tableau is written as follows.

$$\begin{aligned} z &= \alpha_{00} + \sum_{j=1}^{n_1} \alpha_{0j}(-t_j) + \sum_{j=1}^{n_2} \beta_{0j}(-y_j), \\ u_i &= \alpha_{i0} + \sum_{j=1}^{n_1} \alpha_{ij}(-t_j) + \sum_{j=1}^{n_2} \beta_{ij}(-y_j), \qquad i = 1, 2, \ldots, m. \end{aligned} \tag{9}$$

3. Outline of the algorithm

Our algorithm consists of four phases, Phases 0–3. We now give an outline of each phase. The details will follow in subsequent sections.

In Phase 0, the LP problem $\bar{\mathrm{P}}$ is solved. If $\bar{\mathrm{P}}$ is infeasible, so is P. Thus computation terminates. If $\bar{\mathrm{P}}$ has an optimal solution, it is denoted $(\bar{x}, \bar{y})$. In this case, if $\bar{y}$ is an integer vector, $(\bar{x}, \bar{y})$ is an optimal solution of P, and the computation terminates. Otherwise, Phase 1 is entered. Finally, if $\bar{\mathrm{P}}$ is unbounded, it may be the case that P is unbounded. However, we omit this case from consideration. If necessary, we can add constraints $x_j, y_j \leqq M$, where M is a large positive number, to eliminate unboundedness.

In Phase 1, trajectory T (defined in Section 5) is calculated. T starts from $(\bar{x}, \bar{y})$ obtained in Phase 0 and moves to the interior of the feasible region of $\bar{\mathrm{P}}$. In some sense, T is considered as the trajectory of the center of the feasible region.

Phase 2 tries to find integer feasible solutions of P. The search starts from $(\bar{x}, \bar{y})$ and proceeds along T calculated in Phase 1. The basic idea is to round each vector on T to its nearest integer vector, and to perform a search for an integer feasible solution in its neighborhood in case the roun-

ded vector is not a feasible solution. When the prespecified number of integer feasible solutions are obtained or all search procedure along T is completed, Phase 2 terminates. If no integer feasible solution is found in this computation, Phase 2 is concluded to be a failure. Otherwise, it proceeds to Phase 3.

In Phase 3, an attempt is made to improve each integer feasible solution obtained in Phase 2. The search for improvement is guided by coefficients β_{0j} in the simplex tableau (9) which serve as a measure of the amount of the increase (or decrease) of the objective value incurred by changing y_j by 1 or -1. After carrying out the search of the prespecified depth, the best integer feasible solution obtained is provided as a suboptimal solution of P.

Each phase in the above algorithm contains various program parameters, such as the depth of each search. These parameters should be determined prior to execution. Since they are directly related to the quality of the obtained solution and the computation time, it is crucial for achieving a good performance to have reasonable values. These are usually determined empirically, and some typical values are given in Section 8. A list of all program parameters is given in the Appendix.

4. Phase 0

Phase 0 is straightforward. The LP problem $\bar{\mathrm{P}}$ is solved by the simplex method. In our code of Section 8, the following strategy is taken. If $\bar{\mathrm{P}}$ is initially neither primal nor dual feasible, the so-called two-phase method is applied. If $\bar{\mathrm{P}}$ is initially primal feasible, only the second phase of the two-phase method is applied. On the other hand, if $\bar{\mathrm{P}}$ is initially dual feasible, the dual simplex method is applied instead of the two-phase method. In the rest of this paper, we assume that an optimal solution $(\bar{x}, \bar{y})$ of $\bar{\mathrm{P}}$ with its objective value $\bar{z}$ has been obtained.

5. Phase 1

Consider the following LP problem.

$$\mathrm{Q}(u): \quad \text{maximize } \lambda, \tag{10}$$

$$\text{subject to } x_{n_1+i} = b_i - \sum_{j=1}^{n_1} a_{ij}x_j - \sum_{j=1}^{n_2} d_{ij}y_j - \lambda, \quad i = 1, 2, \ldots, m, \tag{11}$$

$$\sum_{j=1}^{n_1} c_j x_j + \sum_{j=1}^{n_2} e_j y_j = u, \tag{12}$$

$$x_j \geqslant 0, \qquad j = 1, 2, \ldots, n_1 + m, \tag{13}$$

$$y_j \geqslant 0, \qquad j = 1, 2, \ldots, n_2, \tag{14}$$

$$\lambda \geqslant 0, \tag{15}$$

where λ is a new variable and u is a given constant.

Let $\bar{z}$ and $\underline{z}$ respectively denote the objective values of an optimal solution of $\bar{\mathrm{P}}$ and of an optimal solution of the LP problem obtained by regarding $\bar{\mathrm{P}}$ as a minimization problem ($\underline{z}$ could be $-\infty$). Q(u) has a feasible solution if

$$\underline{z} \leq u \leq \bar{z} \tag{16}$$

is satisfied.

The optimal solution of Q(u) is equal to the point in the intersection of the feasible region of $\bar{\mathrm{P}}$ and the hyperplane (12), that maximizes the minimum of distances to the hyperplanes defined by

$$b_j = \sum_{j=1}^{n_1} a_{ij} x_j + \sum_{j=1}^{n_2} d_{ij} y_j, \qquad i = 1, 2, \ldots, m.$$

From this geometrical interpretation, the following property is easily proved. $\bar{\lambda}(u)$ and $(\bar{x}(u), \bar{y}(u))$ denote the maximum value of λ and an optimal solution of Q(u). Let $\langle \bar{y}(u) \rangle$ be the vector obtained from $\bar{y}(u)$ by rounding each element to its nearest integer. Then if

$$\bar{\lambda}(u) \geqslant \tfrac{1}{2}\sqrt{n_2} \tag{17}$$

is satisfied, $(\bar{x}(u), \langle \bar{y}(u) \rangle)$ is an integer feasible solution of P, since the sphere with radius $\frac{1}{2}\sqrt{n_2}$ in the n_2-dimensional space obtained by fixing continuous variables to $\bar{x}(u)$ necessarily contains an integer point.

Even if (17) is not satisfied, it would be reasonable to search a feasible solution around $(\bar{x}(u), \bar{y}(u))$. This idea was originally used by Huard in his "method of centers" [7]. Although Huard started his search from the point $(\bar{x}(u), \bar{y}(u))$ with u maximizing $\bar{\lambda}(u)$, we choose rather to concentrate our search to the neighborhood of $(\bar{x}, \bar{y})$ (LP optimal solution), since it is expected with relatively high probability that integer feasible solutions with good objective values are located around $(\bar{x}, \bar{y})$. For this purpose it is helpful to calculate the trajectory of $(\bar{x}(u), \bar{y}(u))$ when u is continuously

decreased from z (see (16)). This trajectory is denoted T, and calculated in Phase 1 of our algorithm.

The calculation of T is done by parametric linear programming (see [8]). T consists of a series of line segments connected consecutively. In our code of Section 8, only the first CUT1 (program parameter) line segments from $(\bar{x}, \bar{y})$ are actually computed. Their junction points are numbered as

$$(x_i, y_i), \qquad i = 0, 1, \ldots, \text{CUT1} \tag{18}$$

in the increasing order of the distance from $(x_0, y_0) = (\bar{x}, \bar{y})$.

6. Phase 2

Let $\langle y \rangle$ denote the integer vector obtained by rounding each element of y to its nearest integer value. Starting from (x_0, y_0) and moving inward on trajectory T, $\langle y \rangle$ for each point (x, y) on T is calculated. Let y^i denote the i^{th} integer vector obtained. To each y^i, $i = 1, 2, \ldots, \text{CUT2}$ (program parameter), a search is then applied to find an integer feasible solution of P. The search consists of two stages:

(i) Solve LP problem $P(y^i)$. If $P(y^i)$ is feasible, its optimal solution is an integer feasible solution of P. If $P(y^i)$ is infeasible, go to (ii).

(ii) Change one of the elements of y^i by $+1$ or -1 as discussed below. Solve $P(\hat{y}^i)$, where $\hat{y}^i$ is the resulting integer vector. This process is repeated until

(a) a new integer feasible solution of P is found,

(b) a solution already obtained as an integer feasible solution of P is again generated, or

(c) the search is concluded to be a failure, i.e., the prespecified number DEPTH2 of changes are completed.

In repeating the above process, each changed variable y_j is recorded together with its direction, and it is prohibited to change y_j back to its original value.

This search applied to $y^1, y^2, \ldots$ is terminated when NINT2 (program parameter) integer feasible solutions of P are generated or all $y^1, y^2, \ldots, y^{\text{CUT2}}$ are tested. In the latter case the search of Phase 2 is concluded to be a failure if no integer feasible solution is found, and the computation terminates. This indicates that either P is infeasible or the feasible region of P is too small for a feasible solution to be found by this type of search. However, no decisive conclusion is usually drawn. If the search is considered

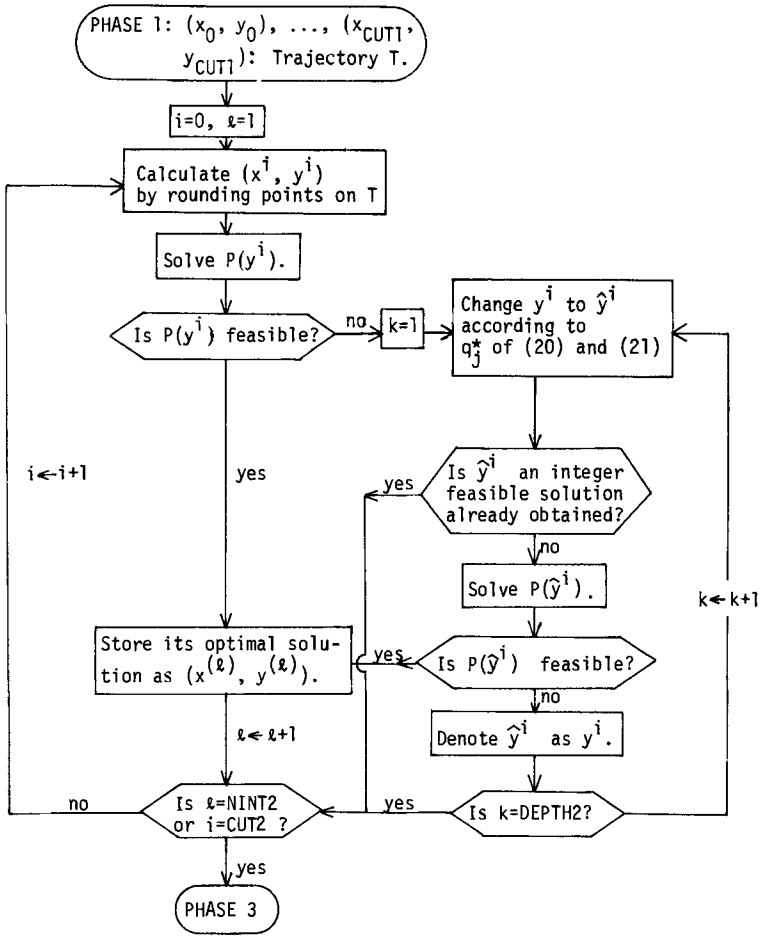

Fig. 1. Flowchart of Phase 2.

to be too shallow, the computation of Phase 1 and Phase 2 may be again tried with larger values of CUT1, CUT2 and DEPTH2. The flowchart of Phase 2 is given in Fig. 1.

Integer feasible solutions obtained in the above computation are denoted

$$(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), (\boldsymbol{x}^{(2)}, \boldsymbol{y}^{(2)}), \ldots, (\boldsymbol{x}^{(\mathrm{NINT2})}, \boldsymbol{y}^{(\mathrm{NINT2})}) \tag{19}$$

and Phase 3 is then entered.

Now the detail of stage (ii) is described. Let (9) denote the simplex tableau when $P(\boldsymbol{y}^i)$ is found to be infeasible. Define q_j^+, q_j^- as follows (α_{k0}, β_{kj} are coefficients of the simplex tableau (9)):

$$q_j^- = \begin{cases} \sum_{k=1}^{m} \min\{0, \alpha_{k0} + \beta_{kj}\} & \text{if } y_j^i > 0, \\ -\infty & \text{if } y_j^i = 0, \end{cases}$$
$$q_j^+ = \sum_{k=1}^{m} \min\{0, \alpha_{k0} - \beta_{kj}\}, \qquad j = 1, 2, \ldots, n_2. \tag{20}$$

q_j^* ($*$ denotes $+$ or $-$) is a measure of infeasibility after the variable y_j is changed by $*1$. (Note that $\alpha_{k0} + \beta_{kj}$ and $\alpha_{k0} - \beta_{kj}$ respectively denote the first column of the simplex tableau when y_j is changed by -1 and $+1$.) Let

$$q_{j_0}^* = \max\{q_j^+, q_j^- : j = 1, 2, \ldots, n_2\}, \tag{21}$$

then y_{j_0} is changed by $*1$ in stage (ii) (this new solution is denoted $\boldsymbol{y}^i$) since this is the most promising change according to the above measure. This completes the description of Phase 2. A slightly modified definition of q_j^* was also tested (see [8]) to speed up the computation of these parameters.

7. Phase 3

Let $z^{(1)}, z^{(2)}, \ldots, z^{(\mathrm{NINT2})}$ be the objective values of integer feasible solutions $(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \ldots, (\boldsymbol{x}^{(\mathrm{NINT2})}, \boldsymbol{y}^{(\mathrm{NINT2})})$ obtained in Phase 2. Rearranging superscripts if necessary, we assume without loss of generality that

$$z^{(1)} \geq z^{(2)} \geq \ldots \geq z^{(\mathrm{NINT2})}$$

holds. In Phase 3, we attempt to improve the first NINT3 ($\leq$NINT2) feasible solutions.

Before proceeding to the details, we outline the improvement procedure applied to $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$. First one of integer variables y_j is selected and changed by $+1$ or -1 so that the objective value may increase. If the resulting solution is still feasible, $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ (and $z^{(i)}$) is replaced by it and an attempt for improvement is again initiated from the new solution. On the other hand, if the resulting solution is not feasible, the search for a feasible solution with a better objective value takes place in a manner similar to Phase 2. If a feasible solution is found by this search, $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ (and $z^{(i)}$) is also replaced by it, and an attempt for improvement is again initiated.

This attempt is repeated for WIDTH3 (program parameter) integer variables of each of $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$, $i = 1, 2, \ldots,$ NINT3.

The selection of the integer variable y_j to increase the objective value is done according to the simplex tableau (9) of $\mathrm{P}(\boldsymbol{y}^{(i)})$. If we do not change the basis, the objective value α_{00} ($= z^{(i)}$) becomes

$$\alpha_{00} \pm \beta_{0j} \tag{22}$$

by the change of y_j by ∓ 1, where β_{0j} is the modified cost of y_j given in the simplex tableau (9) of $P(y^{(i)})$. Thus if β_{0j} is positive (negative), we can temporarily increase the objective value by decreasing (increasing) y_j by 1.

It is desirable to determine the integer variable and its direction of the change so that the increase in the objective value (22) may be as large as possible while keeping the displacement from the feasibility condition

$$\alpha_{k0} \pm \beta_{kj} \geq 0, \qquad k = 1, 2, \ldots, m \tag{23}$$

of the resulting tableau as small as possible. To achieve this objective, the following parameters s_j and r_j are used. They are similar to those used by Hillier [5].

First define h_{kj} and h'_{kj} for $k = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n_2$, by

$$h_{kj} = \begin{cases} 0 & \text{if } \beta_{0j} = 0, \text{ or } \beta_{0j} > 0 \text{ and } y_j^{(i)} = 0, \\ \alpha_{k0}/|\beta_{kj}| & \text{if } \beta_{0j}\beta_{kj} < 0, \\ \infty & \text{if } \beta_{kj} = 0 \text{ or } \beta_{0j}\beta_{kj} > 0, \end{cases} \tag{24}$$

$$h'_{kj} = \begin{cases} h_{kj} & \text{if } h_{kj} < 1, \\ [h_{kj}] & \text{if } h_{kj} \geq 1 \end{cases} \tag{25}$$

([,] denotes the integer part),

$$s_j = \min \{h'_{kj}: k = 1, 2, \ldots, m\}, \qquad j = 1, 2, \ldots, n_2, \tag{26}$$

$$r_j = |\beta_{0j}| s_j, \qquad j = 1, 2, \ldots, n_2. \tag{27}$$

Each s_j (≥ 0) gives the amount of y_j to be changed from $y_j^{(i)}$ in the direction of increasing the objective value without destroying the feasibility condition (23). r_j indicates the amount of the objective value increased by this change.

Now if $s_j \geq 1$ for some j, it implies that y_j can be changed by 1 in the direction of increasing the objective value. Thus y_j (with the largest r_j among those satisfying $s_j \geq 1$) is so changed and $(x^{(i)}, y^{(i)})$ is replaced by the new solution. On the other hand if

$$s_j < 1, \qquad j = 1, 2, \ldots, n_2$$

holds, r_j is arranged in the decreasing order

$$r_{j_1} \geq r_{j_2} \geq \ldots \geq r_{j_{n_2}}, \tag{28}$$

and the first WIDTH3 (program parameter) indices are selected. (If $r_{j_k} = 0$ for some j_k, they are arranged in the decreasing order of $|\beta_{0j_k}|$.)

For each j_k, let $\boldsymbol{y}^{(i)}(j_k)$ denote the integer vector obtained from $\boldsymbol{y}^{(i)}$ by changing y_{j_k} as follows:

$$y_{j_k} \leftarrow y_{j_k}+1 \quad \text{if } \beta_{0j_k} < 0,$$

$$y_{j_k} \leftarrow y_{j_k}-1 \quad \text{if } \beta_{0j_k} > 0.$$

$\mathrm{P}(\boldsymbol{y}^{(i)}(j_k))$ is then solved. Three outcomes are possible.

(i) $\mathrm{P}(\boldsymbol{y}^{(i)}(j_k))$ is feasible and its optimal solution has the objective value greater than $z^{(i)}$.

(ii) It is known that $\mathrm{P}(\boldsymbol{y}^{(i)}(j_k))$ cannot have the objective value greater than $z^{(i)}$, since α_{00} of the tableau satisfies $\alpha_{00} \leq z^{(i)}$. (The tableau is dual feasible).

(iii) $\mathrm{P}(\boldsymbol{y}^{(i)}(j_k))$ is infeasible.

If (i) occurs, an improved integer feasible solution of P is obtained, and the solution $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ (and $z^{(i)}$) is replaced by the new solution. In case of (ii), the search is concluded to be a failure, and

(1) moves to the computation for j_{k+1} if $k <$ WIDTH3.

(2) moves to the improvement of $(\boldsymbol{x}^{(i+1)}, \boldsymbol{y}^{(i+1)})$ if $k =$ WIDTH3 and $i <$ NINT3,

(3) terminates the whole computation if $k =$ WIDTH3 and $i =$ NINT3.

In case of (iii), the search for a feasible solution is carried out. The detail is however omitted, since it is similar to that used in Phase 2. This search is also based on q_j^* of (20), (21) and the search depth (the maximum number of variables to be changed) in this case is prespecified by the program parameter DEPTH3. (A slightly modified definition of q_j^* is also used, and seems to be effective; for details, see [8].) The only difference is that the objective value α_{00} of the simplex tableau is always observed and the search is abandoned whenever it is no greater than $z^{(i)}$. The treatment of this case is the same as case (ii) above. If an improved solution is obtained, it is treated in the same manner as case (i). Finally, if the search is completed and no feasible solution is found, the same procedure as case (ii) takes place.

The entire procedure of Phase 3 is shown by the flowchart of Fig. 2.

8. Computational results

The algorithm described was coded in FORTRAN and run on the FACOM 230/60 of Kyoto University. This machine very roughly corresponds to the IBM 360/65 and the UNIVAC 1108. The computation was carried out all in core unless otherwise stated.

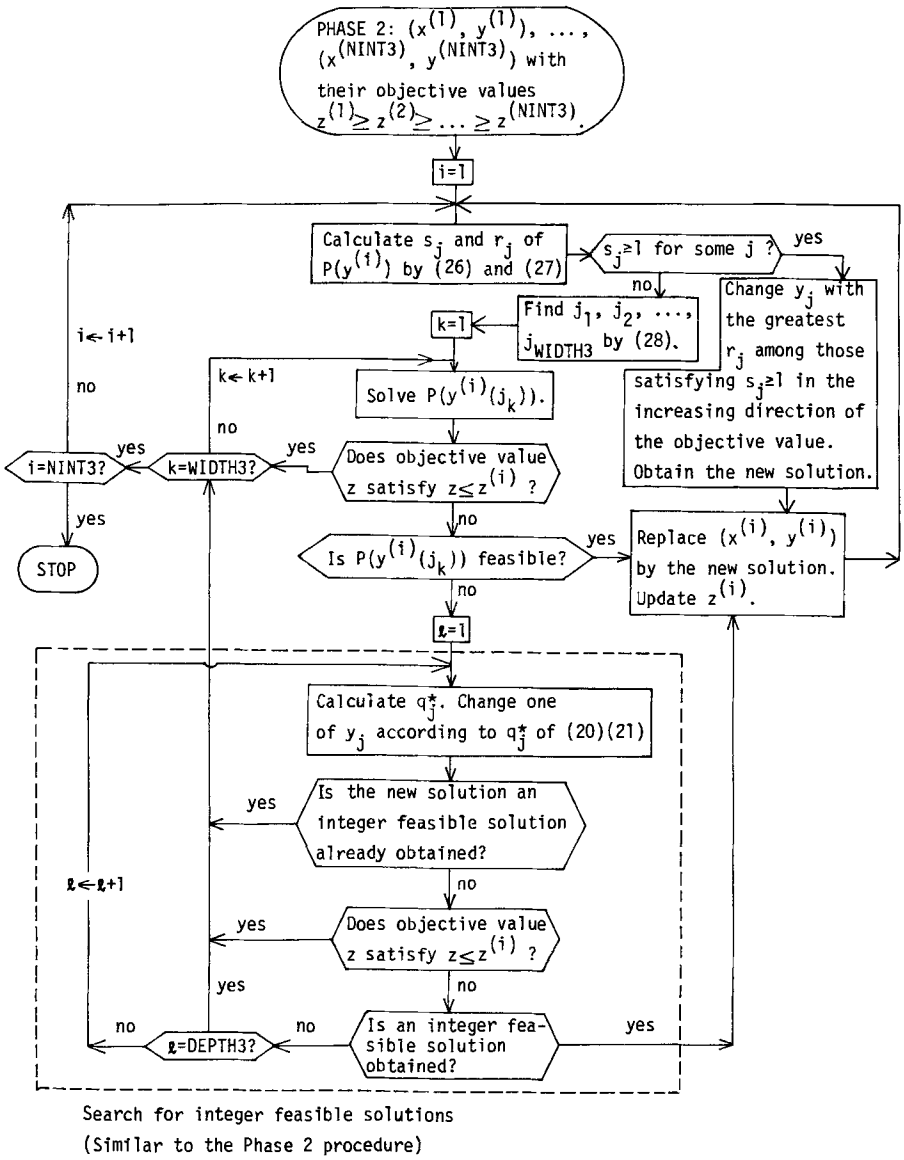

Fig. 2. Flowchart of Phase 3.

Table 1 shows the computational results of problems taken from the literature, for which optimal solutions are known. The results in Table 1 were obtained for program parameters such that

$$\text{CUT1} = 5, \quad \text{CUT2} = 200, \quad \text{NINT2} = 1, \quad \text{NINT3} = 1,$$
$$\text{DEPTH2} = 3, \quad \text{WIDTH3} = \min(10, n_2),$$
$$\text{DEPTH3} = 3, \quad \text{YM} = 0.1.$$

This set of parameters may be classified as a "shallow search". To see the effect of the value of parameters, Table 2 shows results obtained for some problems in Table 1 with different sets of parameters. It should be mentioned that some problems in Table 1 satisfies $\bar{\lambda}(u) = 0$ for all u. Even for them, our code was successful in obtaining feasible solutions.

Since it is possible to control the computation time to some extent by adjusting program parameters, it may be useful to have a standard for a reasonable amount of computation time. One possibility is to base the standard on the LP time (T_0) for solving $\bar{P}$ (i.e., Phase 0). Practically, it may be desirable to have the total computation time (T_{total}) satisfy

$$T_{\text{total}} = k\, T_0, \tag{29}$$

where k is a specified constant (e.g., $k = 10$).

It is important that (29) holds independent of problem size so that we may be able to estimate the computation time in advance with some accuracy.

The results of Table 1 and Table 2 appear to roughly satisfy the above standard. The constant k, however, seems to vary from one type of problem to another. The estimation of k is usually done empirically based on a certain amount of computational experience for similar problems.

To see the behavior of our algorithm for problems without particular structure, a number of problems have been randomly generated and solved. Each coefficient of Type A problems is an integer generated randomly from a uniform distribution with intervals

$$\begin{aligned} &a_{ij}, \quad d_{ij}, \quad e_j : [0, 99], \\ &b_i : \ [1000, 1999]. \end{aligned} \tag{30}$$

(The normalization (7) takes place after the generation of all coefficients.) These conditions are the same as those used by Hillier [5] for his Type II problems. Type B problems differ from Type A in that coefficients a_{ij} and d_{ij} with nonzero values are first selected randomly with probability 0.1 under the condition that at least one coefficient in each column is to be selected, and then values of nonzero coefficients are randomly determined according to (30).

Table 1
Computational results for problems taken from the literature, with parameters CUT1=5, CUT2=200, NINT2=1, NINT3=1, DEPTH2=3, WIDTH3=min (10, $\tilde{n}_2$), DEPTH3=3 and YM=0.1 (Shallow search)

Problem	m[a]	n[b]	n_2[c]	n_3[d]	Time (sec.)					Solution					k of eq. (29)
					Ph.0	Ph.1	Ph.2	Ph.3	Total	Ph.2[e]	Ph.3[e]	n.d.[f]	Opt.[g]	Optimal?	
Haldi 7 [4]	4	5	2	0	0.01	0.07	0.08	0.13	0.30	75	76.7	6.89	76.7	yes	30.0
Haldi 10	10	12	12	0	0.1	0.1	0.1	0.4	0.8	17	17	0.70	17	yes	8.0
Haldi 10	10	12	6	0	0.1	0.1	0.1	0.1	0.5	17.6	17.6	0.47	17.6	yes	5.0
Land–Doig [9]	6	21	21	0	0.1	0.1	0.1	1.3	1.6	531	575	0.25	594	no	16.0
Thompson 9 [15]	10	7	7	0	0.1	0.1	2.4	0.1	2.6	468	468	11.58	464	no	26.0
Logic design 1	28	21	4	4	0.1	0.4	0.3	0.6	1.4	6	6	2.00	6	yes	14.0
[11] 2	56	41	8	8	0.3	1.1	0.9	1.5	3.8	8	8	2.14	8	yes	12.7
Investment 1	12	44	20	20	0.1	0.3	0.3	0.7	1.4	33.0	29.5	0.26	28.0	no	14.0
planning [10] 2	12	44	20	20	0.1	0.3	0.3	0.7	1.4	27.5	24.8	0.26	23.3	no	14.0
3	12	44	20	20	0.1	0.3	0.2	2.0	2.6	33.3	23.4	0.16	23.4	yes	26.0
4	12	44	20	20	0.1	0.3	0.3	1.8	2.5	28.4	19.8	0.16	19.8	yes	25.0
Jobshop[h] 3 × 7	84	43	21	21	0.3	0.4	0.4	0.8	1.9	51	47	6.00	47	yes	6.3
(Makespan) 5 × 4	140	61	40	40	1.4	1.2	1.8	6.0	10.4	40	38	10.00	37	no	7.4
5 × 5	175	76	50	50	2.0	1.3	2.0	7.8	13.1	42	41	6.00	40	no	6.6
Jobshop[h] 3 × 7	84	43	21	21	0.2	0.4	0.5	1.1	2.2	121	115	6.93	115	yes	11.0
(Flow-time) 5 × 4	140	61	40	40	0.5	0.7	1.5	18.0	20.6	146	139	24.60	129	no	41.2
5 × 5	175	76	50	50	0.7	1.2	1.4	21.3	24.6	169	156	22.36	133	no	35.2
Fixed charge 9	110	96	48	48	10.3	3.5	4.7	–	18.6	–	–	–	314	(k)	1.8
transportation [3] 9a	110	96	48	48	32.3	3.5	6.6	13.2	55.6	2842	2578	0.15	2357	no	1.7

Table 1 (Contd.)

Problem		m[a]	n[b]	n_2[c]	n_3[d]	Time (sec.)					Solution					k of eq. (29)
						Ph.0	Ph.1	Ph.2	Ph.3	Total	Ph.2[e]	Ph.3[e]	n.d.[f]	Opt.[g]	Optimal?	
Project planning[i]		93	120	41	41	2.2	3.1	1.8	12.5	19.5	1191	1189	0.0004	1189	yes	8.8
Blending[j]		194	99	30	30	27.6	5.5	227.2	244.9	505.1	124	124	0.26	?	?	18.3
Graph partition[v]	A	75	57	29	29	0.7	2.7	2.4	4.7	10.5	3.0	3.0	0.57	2.0	no	15.0
	B	190	117	39	39	48.4	4.9	11.4	42.2	106.9	41.0	41.0	0.34	41.0	yes	2.2
	C	158	126	126	126	19.2	3.4	42.6	–	65.2	–	–	–	41.0	(k)	3.4

[a] The number of constraints, including those of the type $x_j \leq b_i$ or $y_j \leq b_i$.
[b] The number of variables including both continuous and integer variables.
[c] The number of integer variables including 0–1 variables.
[d] The number of 0–1 variables.
[e] The objective values of the best solutions obtained in Phase 2 and Phase 3 respectively. Each problem is either minimization or maximization problem.
[f] The normalized deviation (31) of the best solution obtained in Phase 3 from the LP optimal value.
[g] The objective value of an optimal solution obtained by some other means.
[h] Due to H. Kise of Kyoto Institute of Technology.
[i] Due to Y. Hama.
[j] Due to H. Watada. This problem was solved on the double-precision basis using an auxiliary memory. Thus the computation speed is considerably slower than that for other problems.
[k] Failed to obtain a feasible solution.
[v] Due to T. Nishimoto of Hitachi, LTD. Three problems represent different formulations of the same problem that partitions a graph (with 39 nodes) in certain optimal way.

Table 2.
Computational results for problems of Table 1 with different sets of program parameters

Problem		$m^{(a)}$	$n^{(b)}$	Time (sec.)					Solution				k of eq. (29)	Parameters[l]
				Ph.0	Ph.1	Ph.2	Ph.3	Total	Ph.2[e]	Ph.3[e]	n.d.[f]	Optimal?		
Land–Doig [9]		6	21	0.1	0.1	0.3	5.8	6.3	531	594	0.18	yes	63	DEPTH2=10 NINT2=3
Thompson [15]	9	10	7	0.1	0.1	6.2	0.7	7.0	468	468	11.58	no	70	NINT3=3 DEPTH3=10
Investment planning [10]	1	12	44	0.1	0.3	0.5	1.7	2.5	33.0	29.5	0.26	no	25	DEPTH2=5 NINT2=2
	2	12	44	0.1	0.3	0.4	3.5	4.4	27.5	23.3	0.23	yes	44	NINT3=2 DEPTH3=5
	3	12	44	0.1	0.3	0.5	6.1	7.0	28.5	23.4	0.16	yes	70	
	4	12	44	0.1	0.3	0.5	4.8	5.7	24.9	19.8	0.16	yes	57	
Fixed charge transportation [3]	9	110	96	10.3	3.7	16.1	14.2	44.3	427	361.4	0.65	no	4.3	DEPTH2=5 NINT2=4
	9a	110	96	32.3	3.6	8.7	65.2	109.8	3104	2357	0.03	yes	3.4	NINT3=2 DEPTH3=5 YM=0.25
Graph partition	A	75	57	0.7	2.7	2.5	7.3	13.2	3.0	2.0	0.38	yes	18	DEPTH2=10 DEPTH3=10
	B	190	117	48.7	5.0	11.4	146.1	211.2	41.0	41.0	0.34	yes	4.3	
	C	158	126	160.	6.5	27.9	65.5	259.6[w]	44.0	44.0	0.56	no	1.6	

[a], [b], [e], [f] See notes of Table 1.
[l] Only those different from Table 1 are shown.
[w] The computation was cut off before termination.

Table 3 lists the computational results for random problems. The results for Type A problems are the average of five problems. All nine problems of Type B in Table 3 have the same coefficients (with 11.5 % nonzero coefficients) but differ in the number of variables specified to be integers. The optimality of these problems cannot be indicated because optimal solutions are not known. However, the normalized deviations from the LP optimal solutions $(\bar{x}, \bar{y})$, i.e.,

$$\text{n. d.} = (\bar{z} - z)\left(\sum_{j=1}^{n_1} c_j^2 + \sum_{j=1}^{n_2} e_j^2\right)^{-1/2} \tag{31}$$

are listed, where z is the objective value of the best suboptimal solution. Since n.d. for these problems are quite small, they are expected to be very good suboptimal solutions.

It is of course possible to use our algorithm for all-integer problems. To see the performance of our algorithm for all-integer problems and to compare it with Hillier's, some random problems whose data are given in [6] are also solved. The results are listed in Table 4 together with Hillier's results.

Although it is not possible to draw a decisive conclusion, our algorithm seems to be at least competitive with his. Roughly speaking, our algorithm tends to use more computation time than his but give solutions of higher quality. For example, the best solution obtained in [5] for Thompson 9 with the minimal objective value 464 (which seems to be a very difficult problem) has the objective value 474, while ours has 468. This may suggest the effectiveness of using the trajectory T in Phase 2.

To get certain information about the quality of integer feasible solutions obtained in Phase 2, some problems are solved with large NINT2. Table 5 summarizes the results when the best solution among those obtained in Phase 2 is found. From Table 5, we may conclude that NINT2 = 2 (or at most NINT2 = 3 $\sim$ 4) is a reasonable value to be used.

Table 6 shows how many improved solutions are found in Phase 3 for INIT3 initial solutions. The data are taken for problems and parameters of Tables 1 and 3. It is observed that the direct improvement due to $s_j \geq 1$ for some j occurs less frequently than other types of improvement. (Thus for most problems the improvement is attained after the search is once made into the outside of the feasible region.) This tendency is even stronger for problems with more integer variables. It should also be noted that NINT3 = 1 is sufficient for most problems. For problems listed in Tables 1–4, the best suboptimal solutions in Phase 3 were always derived from the

Table 3
Computational results for randomly-generated problems

Problem	m[a]	n[b]	n_2[c]	n_3[d]	Time (sec.)					Solution			Parameters[ℓ]
					Ph.0	Ph.1	Ph.2	Ph.3	Total	Ph.2[e]	Ph.3[e]	n.d.[f]	
Type A[q] 1	30	60	10	0	1.9	0.7	2.2	1.4	6.4	21.4	21.4	0.01	DEPTH2=2
2	30	60	30	0	2.4	0.7	2.2	5.6	10.9	20.7	20.8	0.03	NINT2=2, NINT3=2
3	30	60	60	0	2.5	0.7	0.7	7.4	11.2	20.5	20.7	0.16	DEPTH3=3
Type B[r] 0	50	400	0	0	46.1[m]	–	–	–	46.1	615.0	–	–	DEPTH2=2
1	50	400	50	0	46.4	5.9[m]	34.8	21.6	108.8	614.8	614.9	0.01	NINT2=1, NINT3=1
2	50	400	100	0	46.6	6.0	25.8	20.1	98.4	613.5	613.7	0.11	DEPTH3=2
3	50	400	150	0	46.1	6.0	24.8	37.9	114.8	612.4	613.1	0.17	
4	50	400	200	0	47.6	6.1	27.7	49.3	130.8	611.8	612.6	0.21	
5	50	400	250	0	47.6	6.1	35.0	61.3	150.1	609.0	611.8	0.29	
5	50	400	300	0	47.6	6.1	37.8	71.2	162.8	609.4	610.6	0.38	
7	50	400	350	0	47.6	6.1	68.7	85.5	208.0	604.6	608.7	0.56	
8	50	400	400	0	47.6	6.1	98.6	167.3	319.6	598.0	603.5	1.01	

[a]–[f], [ℓ] See notes of Tables 1 and 2.

[m] Theoretically all figures in columns Ph.0 and Ph.1 must be the same respectively. The observed errors are due to the inaccuracy of the system measurement.

[q] Each figure in the table is the average of five problems.

[r] All Type B problems have the same coefficients except that n_2 are different.

Table 4
Computational results for Hillier's random problems with program parameters CUT1=5, CUT2=200, NINT2=4, DEPTH2=10, NINT3=2, WIDTH3=10, DEPTH3=5 and YM=0.25

Problem [6]	m[a]	n[b]	n_2[c]	Time (sec.)					Solution		Hillier's[o]		Modified[p]
				Ph.0	Ph.1	Ph.2	Ph.3	Total	Best sol.[n]	Optimal?	Total Time	Optimal?	Optimal?
I-5	15	15	15	0.2	0.1	1.0	2.7	4.0	6133	no	1.3	no	no
I-6	15	15	15	0.2	0.1	0.5	2.9	3.7	2231	no	1.5	no	yes
II-1	15	15	15	0.1	0.1	0.4	3.7	4.3	1864	yes	0.9	no	no
II-3	15	15	15	0.2	0.1	0.3	1.9	2.5	1983	yes	1.1	no	yes
II-11	30	15	15	0.1	0.2	0.4	3.4	4.0	1488	yes	3.6	no	no
II-12	30	15	15	0.2	0.3	0.3	3.3	4.1	1424	yes	3.9	yes	yes

[a]–[c] See notes of Table 1.
[n] The objective values of best solutions obtained in Phase 3.
[o] Results of Hillier's standard algorithm. See [5], Table IV. The machine is IBM 360/67.
[p] Results of Hillier's multiple solution algorithm. This algorithm requires considerably longer computation time than the standard algorithm.

Table 5
Computational results about when the best among feasible solutions in Phase 2 is obtained[t]

Time when the best among feasible solutions in Phase 2 is obtained	1	2	3	4
Problems in Table 1[s]	14	3	1	0
Type A Problems (Table 3)	8	6	1	0
Total	22	9	2	0

[s] Excluding Fixed Charge Transportation 9, Project Planning, Blending, and Graph Partition.
[t] Each entry in the i^{th} column shows the number of problems with the best solution obtained as the i^{th} feasible solution.

Table 6
The number of improvements attained in Phase 3 for computational results of Table 1 and Table 3[u]

Number of improvements	0	1	2	3	4	5	6 ~ 10	11 ~ 15	16 ~ 20
Problems in Table 1	10	8	1	5	0	0	0	0	0
Type A Problems (Table 3)	6	4	0	1	1	1	2	0	0
Type B Problems (Table 3)	0	0	1	1	3	1	1	0	1
Total	16	12	2	7	4	2	3	0	1

[u] Each entry shows the number of problems with the corresponding number of improvements.

best feasible solutions $(x^{(1)}, y^{(1)})$ obtained in Phase 2, with only one exception—the best suboptimal solution of the fixed-charge transportation problem 9 of Table 2 was derived from the second best solution $(x^{(2)}, y^{(2)})$ of Phase 2.

9. Discussion

Since our LP code used in the algorithm of the previous section is very primitive, at least the following capabilities should be added.

(1) The incorporation of the upper bounding technique or the generalized upper bounding technique.

(2) The use of revised simplex method, which takes advantage of the sparseness of nonzero coefficients in practical problems.

(3) The use of auxiliary memory so that larger problems may be handled.

With these modifications, our algorithm is expected to be able to solve much larger problems. In particular, those problems with larger number of continuous variables but with relatively small number of integer variables (at most about 100 as those solved in the previous computational experimentation) seem to be readily solved, since the total number of pivot operations tends to be rather insensitive to the number of continuous variables.

Reviewing the original manuscript, one of the referees notified us that F.S. Hillier was also developing a similar heuristic procedure for MIP problems as an extension of his procedure [5] for all-integer problems. By private communication with Hillier, it turned out that he also developed the same idea of using the trajectory T for initiating the search for integer feasible solutions, about the same time as the submission of this paper. (However, our idea was originally presented at the ORSJ Conference in 1971 [8].) Hillier's idea was based on the suggestion by Faaland that the optimal solution of $Q(u)$ without constraint (12) should be used as $x^{(2)}$ defined in [5]. The property stated after (17) was also noticed by Faaland. Hillier's procedure corresponding to our Phase 2 and Phase 3 is significantly different from ours. He treats the MIP problem as an all-integer problem by fixing continuous variables to some appropriate values, while we treat the MIP problem as an LP problem by considering integer variables as parameters. It should be noted, however, that both procedures partially use very similar ideas, and furthermore some computation steps used in one procedure can be substituted for the ones used in the other. A direction of future research would be to clarify all the advantages of each approach (empirically or theoretically), and then to combine them into a single more effective procedure.

It seems that Hillier is still attempting the improvement of his approach [5] and conducting an intensive computational experiment (mainly for all-integer problems). Therefore the date taken from [5] such as used in Table 4 should be properly corrected upon the completion of his experiment.

Acknowledgement

The authors wish to express their appreciation to Professor F.S. Hillier of Stanford University for his cooperation in exchanging information,

and to the anonymous referee for pointing out the existence of Hillier's work.

Appendix

List of program parameters

Phase 1

CUT1 : The maximum number of line segments of T to be calculated.

YM : The margin that is used to perturb T so that more number of promising integer points may be tested in Phase 2; y_j satisfying $[y_j] + 0.5 - \text{YM} \leq y_j \leq [y_j] + 0.5 + \text{YM}$ is treated as a number which can be rounded up and/or down. (See [8] for details.)

Phase 2

CUT2 : The maximum number of integer test points obtained by rounding the points on T.

DEPTH2 : The search depth for an integer feasible solution initiated from each integer point obtained above.

NINT2 : The total number of integer feasible solutions to be obtained in Phase 2.

Phase 3

NINT3 : The number of integer feasible solutions for which the improvement will be attempted in Phase 3. (NINT3 $\leq$ NINT2).

WIDTH3 : The number of integer variables, each of which is tentatively changed by 1 or -1 to initiate the search for the improvement of an integer feasible solution obtained in Phase 2. (WIDTH3 $\leq n_2$).

DEPTH3 : The depth of the above mentioned search initiated by changing one integer variable.

References

[1] E. Balas, "An additive algorithm for solving linear programs with zero–one variables", *Operations Research* 13 (1965) 517–546.

[2] R.E. Echols and L. Cooper, "Solution of integer linear programming problems by direct search", *Journal of the Association for Computing Machinery* 15 (1968) 75–84.

[3] P. Gray, "Mixed integer programming algorithms for site selection and other fixed charge problems having capacity constraints", SED-Special Rept., Stanford Research Institute, Menlo Park, Calif. (1967).

[4] J. Haldi, "25 integer programming test problems", Rept. of Graduate School of Business, Stanford University, Stanford, Calif. (1964).

[5] F.S. Hillier, "Efficient heuristic procedures for integer linear programming with an interior", *Operations Research* 17 (1969) 600–636.
[6] F.S. Hillier, "A bound-and-scan algorithm for pure integer linear programming with general variables", Rept. No. 3, Department of Operations Research, Stanford University, Stanford, Calif. (1969).
[7] Huard, "Programmes mathematiques nonlineares a variables bivalentes", in: H.W. Kuhn, ed., *Proceedings of the Princeton symposium on mathematical programming* (Princeton University Press, Princeton, N.J., 1970).
[8] T. Ibaraki, T. Ohashi and H. Mine, "A heuristic algorithm for mixed-integer programming problems", Fall Conference of Operations Research Society of Japan, 1971 and 1972; Working paper, Department of Applied Mathematics and Physics, Kyoto University, Kyoto, Japan (1972).
[9] A.H. Land and A.G. Doig, "An automatic method of solving discrete programming problems", *Econometrica* 28 (1960) 497–520.
[10] A.S. Manne, "A mixed integer algorithm for project evaluation", Memorandom 71–3, Development Research Center, International Bank for Reconstruction and Development, Washington, D.C. (1971).
[11] S. Muroga and T. Ibaraki, "Logical design of an optimum network by integer linear programming", Rept. No. 264, Department of Computer Science, University of Illinois, Urbana, Ill. (1968).
[12] S. Reiter and D.B. Rice, "Discrete optimizing solution procedures for linear and nonlinear integer programming problems", *Management Science* 12 (1966) 829–850.
[13] R.H. Roth, "An approach to solving linear discrete optimization problems", *Journal of the Association for Computing Machinery* 17 (1970) 300–313.
[14] S. Senju and Y. Toyoda, "An approach to linear programming with 0–1 variables, *Management Science* 15 (1968) B196–B207.
[15] G.L. Thompson, "The stopped simplex method: 1. Basic theory for mixed integer programming", *Revue Francaise de Recherche Operationnelle* 8 (1964) 159–182.

Mathematical Programming Study 2 (1974) 137–179. North-Holland Publishing Company

ON THE GROUP PROBLEM FOR MIXED INTEGER PROGRAMMING

Ellis L. JOHNSON

IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y., U.S.A.

Received 11 September 1973
Revised manuscript received 26 August 1974

A theory is developed for a group problem arising from mixed integer programming. This theory gives descriptions of functions on the unit hypercube from which cutting planes can be constructed for any mixed integer program. Methods for generating such functions are given.

1. Introduction

For a pure integer programming problem

$$\begin{aligned} &\text{minimize} && Z = c\,x, \\ &\text{subject to} && A\,x = b, \qquad x \geq 0 \text{ an integer,} \end{aligned}$$

the group problem is obtained by relaxing the non-negativity restriction on the basic variables (usually corresponding to an optimal basis). The group problem is, thus,

$$\sum_{j \in J_N} \bar{a}_{ij} x_j \equiv \bar{b}_i \,(\text{mod } 1), \qquad i = 1, \ldots, m,$$

$$x_j \geq 0 \text{ and integer}, \qquad j \in J_N,$$

where J_N is the index set of non-basic variables and $\bar{a}_{ij}$, $\bar{b}_i$ are the updated coefficients (see [3, 5, 6] for further details).

In the mixed integer case, we obtain a system of congruences:

$$\sum_{j \in J_N} \bar{a}_{ij} x_j \equiv \bar{b}_i \,(\text{mod } 1),$$

with one such congruence for each basic variable which is required to be integer. That is, from an updated linear programming tableau (usually corresponding to an optimal linear programming basis), drop the rows which have basic variables not required to be integer-valued. Then, convert the remaining rows into congruences modulo 1.

This congruence, or group, problem is a relaxation of the original problem obtained by deleting the non-negativity restrictions on the basic variables. We retain the integrality requirement on those basic variables which should be integer and also on the non-basic variables which are required to be integer. Non-negativity of all of the non-basic variables is also imposed in formulating this group problem.

Any inequality satisfied by all solutions of this group problem (referred to subsequently as valid inequalities) is a legitimate cutting plane for the original integer program. In addition, any bound Z_B satisfying $Z_* \geq Z_B$ for Z_* the optimal solution of the minimization problem

$$\text{minimize} \quad Z = \sum_{j \in J_N} \bar{c}_j x_j,$$

$$\text{subject to} \quad x_j,\ j \in J_N, \text{ satisfying the group problem}$$

is also a bound for the integer program. That is, the integer optimum Z_I satisfies $Z_I \geqq Z_L + Z_B$, where Z_L is the linear programming optimum. Such bounds are useful in branch and bound algorithms or implicit enumeration and can be obtained easily from one cut or by solving a linear program over several cuts.

The difficulty in the mixed integer case is that the additional restrictions on x_j for $j \in J_N$, the non-basic variables, are $x_j \geq 0$ for all $j \in J_N$ and x_j integer only for a subset of the $j \in J_N$; say for $j \in J_I \cap J_N$, where J_I denotes the subset of the variable indices corresponding to variables required to be integer-valued. The case where all of the variables x_j are required to be integer-valued, i.e., $j \in J_I$ for all j, is well understood in theory although in practice there may be difficulties (see [5, 8]). It is striking that dropping the integrality restriction on some non-basic variables should make the problem more difficult to handle. Nevertheless, cutting plane methods seem to be more successful in the pure integer than the mixed integer case. Some work has been done with no integrality restriction on the non-basic variables [1]. This paper studies the general mixed integer group problem in an attempt to develop a basic understanding of the interaction of the integer variables and the continuous variables. Previous work [6, 7] in that direction concentrated on group problems with only one constraint. That case is well-understood but seems not to capture enough of the difficulty of the overall problem to be very useful in the mixed integer case. We now consider general mixed integer group problems having any number of rows.

First some notational conventions need to be mentioned. In a system of congruences of the form

$$\sum_{j \in J_N} \bar{a}_{ij} x_j \equiv \bar{b}_i \,(\mathrm{mod}\ 1)$$

for $j \in J_N \cap J_1$, $\bar{a}_{ij}$ can be reduced to its fractional part $F(\bar{a}_{ij})$ with $0 \leq F(\bar{a}_{ij}) < 1$ without changing the congruence. Similarly, $\bar{b}_i$ can be changed to $F(\bar{b}_i)$. In what follows, we let $u = (F(\bar{a}_{1j}), \ldots, F(\bar{a}_{mj})$ be the vector of fractional parts of the coefficients for some x_j, $j \in J_N \cap J_1$ and let U denote the entire set of such vectors. Let W denote the set of vectors $w = (\bar{a}_{1j}, \ldots, \bar{a}_{mj})$ for $j \in J_N \setminus J_1$. Let u_0 be the vector $(F(\bar{b}_1), \ldots, F(\bar{b}_m))$. Then, the system of congruences can be written:

$$\sum_{u \in U} u\, t(u) + F\left(\sum_{w \in W} w\, s(w)\right) = u_0.$$

Each u is a vector of fractional parts of coefficients for some variable x_j, $j \in J_N \cap J_1$, and now that x_j is written simply $t(u)$. Similarly, x_j, $j \in J_N \setminus J_1$, with coefficient column w becomes $s(w)$. The $t(u)$, $u \in U$, are required to be non-negative integers, and $s(w)$, $w \in W$, are required to be non-negative real numbers. We drop the $\equiv$ and (mod 1), and consider that any equation of fractional parts is always modulo 1.

2. Problem definition

Let I^m be the group of real vectors $u = (u_1, \ldots, u_m)$ in the unit cube $[0, 1] \times \ldots \times [0, 1]$ with addition taken modulo 1 in each component. Let U be any subset of I^m. Let S^m be the set of real numbers $w = (w_1, \ldots, w_m)$ having $\max \{|w_i| : i = 1, \ldots, m\} = 1$. That is, S^m is the boundary of the cube $-1 \leq x_i \leq 1$, $i = 1, \ldots, m$, in $\mathbf{R}^m$. Let W be any subset of S^m.

Let t be a function on U such that (i) $t(u)$ is an integer for all $u \in U$, (ii) $t(u) \geq 0$ for all $u \in U$, and (iii) t has *finite support*; that is, $t(u) > 0$ only for a finite subset U_t of U. Correspondingly, let s be a non-negative real-valued function on W with finite support, but s is not required to be integer valued. Let $T(U, W, u_0)$, $u_0 \in I^m \setminus \{0\}$, be the set of all such pairs (t, s) such that

$$\sum_{u \in U} u\, t(u) + F\left(\sum_{w \in W} w\, s(w)\right) = u_0. \tag{1}$$

Here, $\sum w\, s(w)$ is a real number, and F is the mapping from $\mathbf{R}^m$ to I^m consisting of taking the fractional part of each component of a real vector.

We do not consider I^m as a subset of $\mathbf{R}^m$ but instead as an additive group. Thus, (1) is written as equality in the group I^m. The mapping $|u|$ is used to denote the vector in $\mathbf{R}^m$ corresponding to $u \in I^m$. We can write (1) equivalently as

$$\sum_{u \in U} |u|\, t(u) + \sum_{w \in W} w\, s(w) \equiv |u_0| \text{ (modulo 1)}.$$

We say that $(t, s) \in T(U, W, u_0)$ is a *solution to the problem* $P(U, W, u_0)$.

One definition used throughout will be made here. If f and g are real-valued functions on a set X, and if A is a subset of X, then $f(A) < g(A)$ means $f(x) \leq g(x)$ for all $x \in A$, and $f(x) < g(x)$ for at least one $x \in A$. For a scalar, such as zero, $0 < f(A)$ means $Z(A) < f(A)$, where $Z(x) = 0$ for all $x \in A$. When $A = X$, we usually omit it and write $f \leq g$ or $0 \leq f$.

3. Inequalities

A valid inequality for the problem $P(U, W, u_0)$ is a pair of non-negative real-valued functions (π, μ), π defined on I^m and μ on S^m, such that

$$\sum_{u \in U} \pi(u)\, t(u) + \sum_{w \in W} \mu(w)\, s(w) \geq 1 \tag{2}$$

for all $(t, s) \in T(U, W, u_0)$. We will frequently use the notation $(\pi, \mu) \cdot (t, s) \geq 1$ to mean (2).

A *minimal valid inequality* for $P(U, W, u_0)$ is a valid inequality (π, μ) for $P(U, W, u_0)$ such that there is no other valid inequality (ρ, ν) for $P(U, W, u_0)$ with $(\rho, \nu) < (\pi, \mu)$; that is, $\rho \leq \pi$ and $\nu \leq \mu$, and at least one of $\rho < \pi$ or $\nu < \mu$ holds.

An *extreme valid inequality* for $P(U, W, u_0)$ is a valid inequality (π, μ) such that

$$(\pi, \mu) = \tfrac{1}{2}(\rho_1, \nu_1) + \tfrac{1}{2}(\rho_2, \nu_2),$$

for (ρ_1, ν_1) and (ρ_2, ν_2) valid inequalities for $P(U, W, u_0)$, implies $(\rho_1, \nu_1) = (\rho_2, \nu_2) = (\pi, \mu)$.

Theorem 3.1. *The extreme valid inequalities are minimal valid inequalities.*

Proof. If (π, μ) is not minimal, then there is a valid inequality (ρ, ν) with $(\rho, \nu) < (\pi, \mu)$. Clearly,

$$(\pi, \mu) = \tfrac{1}{2}(\rho, \nu) + \tfrac{1}{2}(\pi + (\pi - \rho), \mu + (\mu - \nu)).$$

Since (π, μ) is a valid inequality and since $\pi - \rho \geq 0$ and $\mu - v \geq 0$, it follows that $(\pi + (\pi - \rho), \mu + (\mu - v))$ is also a valid inequality. Therefore, (π, μ) is exhibited as a midpoint of two different valid inequalities, contradicting extremality.

The next theorem is a limited converse of Theorem 3.1. Since the extreme valid inequalities are among the minimal valid inequalities, they are obviously among the minimal valid inequalities which can not be written as the midpoint of two other minimal valid inequalities. The next theorem says that the only such minimal valid inequalities are the extreme valid inequalities. In fact, every minimal valid inequality which is extreme among the minimal valid inequalities is an extreme inequality.

Before stating the theorem, let us remark that the set of valid inequalities is a convex set, and our definition of an extreme valid inequality is a standard way of defining extreme point of a convex set. The set of minimal valid inequalities is not necessarily a convex set, but we can define an extreme minimal valid inequality in the same way, that is, a minimal valid inequality which can not be written as the midpoint of two different minimal valid inequalities. Theorem 6.1 shows that the minimal valid inequalities are a convex set in the important special case where $U = I^m$ and $W = S^m$. In other cases, the minimal valid inequalities are not a convex set. The theorem to follow is of interest in either case.

Theorem 3.2. *If (π, μ) is an extreme minimal valid inequality, then (π, μ) is an extreme valid inequality.*

Proof. Suppose that such a (π, μ) is not an extreme valid inequality. Then $(\pi, \mu) = \frac{1}{2}(\rho_1, v_1) + \frac{1}{2}(\rho_2, v_2)$ for $(\rho_1, v_1) \neq (\rho_2, v_2)$. We show that minimality of (π, μ) implies minimality of (ρ_1, v_1) and (ρ_2, v_2), contradicting (π, μ) being extreme among the minimal valid inequalities.

Suppose (ρ_1, v_1) is not minimal. Then there is a valid inequality $(\rho_3, v_3) < (\rho_1, v_1)$. Because the valid inequalities are a convex set, $(\pi_1, \mu_1) = \frac{1}{2}(\rho_3, v_3) + \frac{1}{2}(\rho_2, v_2)$ is a valid inequality. But, $(\pi_1, \mu_1) < (\pi, \mu)$, contradicting minimality of (π, μ) and completing the proof.

A *subadditive valid inequality* for $P(U, W, u_0)$ is a valid inequality (π, μ) for $P(U, W, u_0)$ such that

$$\pi(u) + \pi(v) \geq (u + v), \quad \text{whenever all three of } u, v \text{ and } u + v \text{ are in } U; \tag{3}$$

$$\pi(u) + \sum_{w \in W} \mu(w)\, s(w) \geq \pi(v), \quad \text{whenever } u, v \text{ in } U, \text{ and } u + F\left(\sum_{w \in W} w\, s(w)\right) = v, \tag{4}$$

$$\sum_{w \neq x} \mu(w)\, s(w) \geq \mu(x), \quad \text{whenever } x = \sum_{w \neq x} w\, s(w) \in W. \tag{5}$$

In (4) and (5), s is assumed to be a non-negative real-valued function on W with finite support.

Theorem 3.3 *The minimal valid inequalities are subadditive valid inequalities.*

Proof. Suppose (π, μ) is a minimal valid inequality for $P(U, W, u_0)$ but is not subadditive. Then at least one of (3), (4), (5) is not satisfied.

Suppose first that (3) is violated. The argument in this case is practically the same as the proof of [6, Theorem 1.2] and will not be repeated. The same idea is used when (4) or (5) are assumed to be violated.

Suppose (4) is violated. Then, for some v_1, v_2 and $s_1(w)$, $w \in W$,

$$v_1 + F\left(\sum_{w \in W} w\, s_1(w)\right) = v_2,$$

$$\pi(v_1) + \sum_{w \in W} \mu(w)\, s_1(w) < \pi(v_2).$$

Define ρ on I^m by

$$\rho(u) = \begin{cases} \pi(u) & \text{if } u \neq v_2, \\ \pi(v_1) + \sum_{w \in W} \mu(w)\, s_1(w) & \text{if } u = v_2. \end{cases}$$

Given $(t, s) \in T(U, W, u_0)$, define

$$t^*(u) = \begin{cases} t(v_1) + t(v_2) & \text{if } u = v_1, \\ 0 & \text{if } u = v_2, \\ t(u) & \text{otherwise,} \end{cases}$$

$$s^*(w) = s(w) + t(v_2)\, s_1(w), \qquad w \in W.$$

Then $(t^*, s^*) \in T(U, W, u_0)$ and $(\rho, \mu) \cdot (t, s) = (\pi, \mu) \cdot (t^*, s^*) \geq 1$. Therefore, (ρ, μ) is a valid inequality for $P(U, W, u_0)$ contradicting minimality of (π, μ).

The case when (5) is violated is also similar. If (5) is violated, then for some $x \in W$,

$$\sum_{w \neq x} \mu(w)\, s_1(w) < \mu(x), \quad \text{where } x = \sum_{w \neq x} w\, s_1(w).$$

Let v be defined on S^m by

$$v(w) = \begin{cases} \mu(w) & \text{if } w \neq x, \\ \sum_{y \neq x} \mu(y)\, s_1(y) & \text{if } w = x. \end{cases}$$

Now, for $(t, s) \in T(U, W, u_0)$, define

$$s^*(w) = \begin{cases} s(w) + s_1(w)\, s(x) & \text{if } w \neq x, \\ 0 & \text{if } w = x. \end{cases}$$

Then $(t, s^*) \in T(U, W, u_0)$ and (π, v) is a valid inequality contradicting minimality of (π, μ). The proof is completed.

The following theorem is concerned with (π, μ) which are extreme in the set of subadditive valid inequalities. The set of subadditive valid inequalities is a convex subset of the set of valid inequalities. We have shown that the extreme points of the sets of valid inequalities are precisely the extreme points of the set of minimal inequalities and are therefore in the set of subadditive valid inequalities. They are, then, extreme points of the set of subadditive valid inequalities. There may be other extreme points of this latter set, and the next theorem addresses that question.

Theorem 3.4. *If (π, μ) is extreme in the set of subadditive valid inequalities and if (π, μ) is a minimal valid inequality, then it is also an extreme valid inequality.*

Proof. The theorem is actually a corollary of the Theorem 3.2 since if (π, μ) is not extreme, then by minimality of (π, μ) it must be a midpoint of minimal valid inequalities. But minimal valid inequalities are subadditive, contradicting the hypothesis of the theorem.

4. Subadditive functions on subgroups U

Let U be a non-empty subgroup of I^m. Then $0 \in U$ in particular. Define (π, μ) to be a *subadditive function* on (U, W) if π and μ are non-negative real valued functions satisfying (3) and both of

$$\sum_{w \in L} \mu(w)\, s(w) \geq \pi(v), \quad \text{whenever } v = F\left(\sum_{w \in L} w\, s(w) \right) \tag{6}$$

$$\sum_{w \in L} \mu(w)\, s(w) \geq \mu(x), \quad \text{whenever } x = \sum_{w \in L} w\, s(w). \tag{7}$$

In both (6) and (7), L denotes a linearly independent subset of W and s is a positive real-valued function on L with finite support. In (7), we may as well exclude the case $x \in L$. The restriction of linear independence on L does not change the class of functions satisfying (6) and (7), as the next lemma proves. The restriction of linear independence is intended as a limitation on the inequalities (4) and (5) which must be satisfied in order for all of them to be satisfied.

The purpose of this section is to start with (3), (4) and (5) and substitute weaker conditions for (4) and (5). To begin, (3), (4) and (5) will be shown to be always equivalent to the weaker conditions (3), (6) and (7). Then, in various special cases, (3), (6) and (7) will be shown equivalent to other conditions which are either weaker or more descriptive.

Lemma 4.1. *If U is a subgroup and if (π, μ) is a subadditive function on (U, W), then* (4) *and* (5) *hold.*

Proof. In the first place, in (4) and (5) we can restrict s to being positive on a linearly independent subset L of W since otherwise there is a function λ on the subset of W for which $s(w) > 0$ and such that

$$\sum_{w \in W} w\, \lambda(w) = 0.$$

Then

$$\sum_{w \in W} \mu(w)\,(s(w) + \varepsilon\, \lambda(w)) = \sum_{w \in W} \mu(w)\, s(w) + \varepsilon \sum_{w \in W} \mu(w)\, \lambda(w).$$

By increasing ε from 0 when $\sum \mu(w)\, \lambda(w) \leq 0$ or decreasing ε from 0 when $\sum \mu(w)\, \lambda(w) > 0$ we have $\sum \mu(w)\, s(w) \geq \sum \mu(w)\, s'(w)$, where $s'(w) = s(w) + \varepsilon\, \lambda(w)$. The ε can be changed in this way until some $s'(w)$ becomes zero. In this way, the w for which $s'(w) > 0$ can be reduced to a linearly independent set. Recall that the s we started with had finite support, so $s(w) > 0$ on only a finite set.

Secondly, if x has $s(x) > 0$ in (5), then the inequality in (5) holds automatically if $s(x) \geq 1$. If $s(x) < 1$, then an equivalent inequality is obtained from

$$x = \sum_{\substack{w \in W \\ w \neq x}} w \frac{s(w)}{1 - s(x)}.$$

Thirdly, if

$$u + F\left(\sum_{w \in L} w\, s(w)\right) = v,$$

then $v - u \in U$, so $F(\sum w\, s(w)) \in U$ and by (6) and (3),

$$\pi(u) + \sum_{w \in L} \mu(w)\, s(w) \geq \pi(u) + \pi\left(F\left(\sum_{w \in L} w\, s(w)\right)\right) \geq \pi(v).$$

Hence (3) and (6) imply (4), completing the proof.

Conditions (6) and (7) will be shown equivalent to weaker conditions in several special cases including U a finite subgroup and $U = I^m$.

Before proceeding, two useful lemmas will be shown.

Lemma 4.2. *If π is a function on I^m satisfying $\pi(u) + \pi(v) \geq \pi(u + v)$ for u, $v \in U$, U a subgroup of I^m, then*

$$\sum_{u \in U} \pi(u)\, t(u) \geq \pi\left(\sum_{u \in U} u\, t(u)\right)$$

for any non-negative integer valued function t on U with finite support.

Proof. See [5, 6]; particularly [6, Theorem 1.5].

By definition, $\pi(0) = 0$ for subadditive π. The next lemma is concerned with the slope of $\pi(u)$ as u approaches 0.

Lemma 4.3. *If π is a real valued function on I^m satisfying $\pi(u) + \pi(v) \geq \pi(u + v)$ for all u, v in I^m and if*

$$\lim_{h \downarrow 0} \sup \frac{\pi(F(h\, w))}{h} = \beta < \infty,$$

for some $w \in S^m$, then

$$\lim_{h \downarrow 0} \frac{\pi(F(h\, w))}{h} = \beta.$$

Proof. By the lim sup being β, for any $\varepsilon > 0$ there is an $h < 1$ with $\pi(F(h\, w))/h > \beta - \varepsilon$ and with $\pi(F(h'w))/h' < \beta + \varepsilon$ for $0 < h' < h$. If the limit does not exist, then there is an h_1, $0 < h_1 < h$, with $\pi(F(h_1 w))/h_1 < \beta - \varepsilon$.

Then for some r satisfying $0 \leq r < h_1$,

$$h = \lfloor h/h_1 \rfloor h_1 + r,$$

where $\lfloor x \rfloor$ means the largest integer below or equal to x. By (3),

$$\begin{aligned}\pi(F(h\,w)) &\leq \pi(F(\lfloor h/h_1 \rfloor h_1 w)) + \pi(F(r\,w)) \\ &\leq \lfloor h/h_1 \rfloor\, \pi(F(h_1 w)) + \pi(F(r\,w) \\ &< \lfloor h/h_1 \rfloor\, (\beta - \varepsilon)\, h_1 + \pi(F(r\,w)) \\ &= (\beta - \varepsilon)\, h - (\beta - \varepsilon)\, r + \pi(F(r\,w)).\end{aligned}$$

By the lim sup being β, h can be chosen small enough that $\pi(F(h'w))/h' < \beta + \varepsilon$ for $0 < h' < h$. Then

$$\pi(F(r\,w)) < (\beta + \varepsilon)\, r.$$

Hence,

$$\begin{aligned}\pi(F(h\,w)) &< (\beta - \varepsilon)\, h - (\beta - \varepsilon)\, r + (\beta + \varepsilon) r \\ &< (\beta - \varepsilon)\, h + 2\varepsilon\, r.\end{aligned}$$

Now, we can fix h and let h_1 and, hence, r be arbitrarily small. Therefore,

$$\pi(F(h\,w)) \leq (\beta - \varepsilon)\, h,$$

a contradiction. The lemma is proven.

We now prove for several cases that (6) and (7) can be replaced by equivalent, weaker statements. That is, a non-negative real valued function (π, μ) is a subadditive function on (U, W), U a subgroup of I^m, provided (3) holds, i.e., $\pi(u) + \pi(v) \geq \pi(u + v)$, and provided some weakening of (6) and (7) hold. We first consider $U = I^m$ and any $W \subseteq S^m$.

Lemma 4.4. *If $U = I^m$, then (6) can be replaced by*

$$\mu(w) \geq \lim_{h \downarrow 0} \frac{\pi(F(h\,w))}{h}, \qquad w \in W. \tag{8}$$

Proof. The proof consists of showing that (3), (6) and (7) imply (8), and conversely that (3), (8) and (7) imply (6).

First, it is clear from (6) that

$$\mu(w)\, h \geq \pi(F(h\,w)) \quad \text{for } 0 \leq h \quad \text{and} \quad w \in W.$$

Hence,

$$\mu(w) \geq \limsup_{h \downarrow 0} \frac{\pi(F(h\,w))}{h}.$$

By Lemma 4.3, (8) follows.

Next, let us show that (3), (8) and (7) imply (6). Suppose not. Then

$$\sum_{w\in L} \mu(w)\, s_1(w) = \pi(v_1) - \varepsilon, \qquad \varepsilon > 0, \quad v_1 = F\left(\sum_{w\in L} w\, s_1(w)\right),$$

where $s_1(w) > 0$, $w \in L$. Let

$$\varepsilon' = \frac{\varepsilon}{2\sum_{w\in L} s_1(w)} > 0.$$

By (8), for each $w \in L$ there is an $h_0(w) > 0$ such that

$$\mu(w) > \frac{\pi(F(h(w)\, w))}{h(w)} - \varepsilon' \quad \text{for } 0 < h(w) < h_0(w).$$

Let $h(w) = s_1(w)/n(w)$ for $n(w)$ an integer satisfying $n(w) > s_1(w)/h_0(w)$. Then $0 < h(w) < h_0(w)$, and

$$\sum_{w\in L} \mu(w)\, s_1(w) \geq \sum_{w\in L} n(w)\, \pi\left(F\left(\frac{s_1(w)}{n(w)}\, w\right)\right) - \tfrac{1}{2}\varepsilon.$$

Substituting $\pi(v_1) - \varepsilon = \sum \mu(w)\, s_1(w)$ gives

$$\pi(v_1) \geq \sum_{w\in L} n(w)\, \pi\left(F\left(\frac{s_1(w)}{n(w)}\, w\right)\right) + \tfrac{1}{2}\varepsilon.$$

Now, by Lemma 1.6,

$$\begin{aligned}\pi(v_1) &\geq \pi\left(\sum_{w\in L} n(w)\, F\left(\frac{s_1(w)}{n(w)}\, w\right)\right) + \tfrac{1}{2}\varepsilon \\ &\geq \pi\left(\sum_{w\in L} s_1(w)\, w\right) + \tfrac{1}{2}\varepsilon = \pi(v_1) + \tfrac{1}{2}\varepsilon.\end{aligned}$$

Thus, a contradiction is reached, proving the lemma.

The next lemma concerns the case $W = S^m$. Taken together, these two lemmas give a characterization of the subadditive functions on (I^m, S^m).

First, a definition is needed. For a function μ on S^m, define the *homogenous extension of* μ *to* $\mathbf{R}^m$ to be the function f on $\mathbf{R}^m$ given by

$$f(y) = \begin{cases} 0 & \text{if } y = 0, \\ \max_i \{|y_i|\}\, \mu\left(\dfrac{v}{\max_i \{|y_i|\}}\right) & \text{if } y \neq 0. \end{cases} \tag{9}$$

Lemma 4.5. *If $W = S^m$, then (7) may be replaced by the requirement that the homogenous extension of μ to $\mathbf{R}^m$ be a convex function.*

Proof. The proof consists of showing that (7) implies f is convex and conversely. Clearly, f defined by (9) is positively homogenous. If f is convex, then (7) holds (see [11, Theorem 4.7 and Corollary 4.7.1]).

Suppose now that (7) holds for $W = S^m$. Let f be defined by (9). Let x and y be in $\mathbf{R}^m$ and let $0 < \lambda < 1$. We wish to show that

$$\lambda f(x) + (1 - \lambda) f(y) \geq f(\lambda x + (1 - \lambda) y).$$

If x (or y) is zero, then the inequality holds by $(1 - \lambda) f(y) = f((1 - \lambda) y)$. Hence, let $x \neq 0$ and $y \neq 0$. Let $M_x = \max_i \{|x_i|\}$ and $M_y = \max_i \{|y_i|\}$. Then $M_x > 0$ and $M_y > 0$. If $\lambda x + (1 - \lambda) y = 0$, then the inequality holds by non-negativity of f and by $f(0) = 0$. Let $\lambda x + (1 - \lambda) y \neq 0$ and let $M_{xy} = \max_i \{\lambda x_i + (1 - \lambda) y_i\}$.

By definition (9) of f, convexity of f is now equivalent to

$$\lambda M_x \mu\left(\frac{x}{M_x}\right) + (1 - \lambda) M_y \mu\left(\frac{y}{M_y}\right) \geq M_{xy} \mu\left(\frac{\lambda x + (1 - \lambda) y}{M_{xy}}\right),$$

or

$$\frac{\lambda M_x}{M_{xy}} \mu\left(\frac{x}{M_x}\right) + \frac{(1 - \lambda) M_y}{M_{xy}} \mu\left(\frac{y}{M_y}\right) \geq \mu\left(\frac{\lambda x + (1 - \lambda) y}{M_{xy}}\right).$$

But the above holds by (7) and by

$$\frac{\lambda M_x}{M_{xy}} \frac{x}{M_x} + \frac{(1 - \lambda) M_y}{M_{xy}} \frac{y}{M_y} = \frac{\lambda x + (1 - \lambda) y}{M_{xy}}.$$

The proof is, thus, completed.

Rockafellar [11] defines a *gauge* to be a non-negative, positively homogenous convex function which is zero at the origin. For $\mu(w) \geq 0$, $w \in S^m$, the homogenous extension f of μ to $\mathbf{R}^m$ is obviously non-negative and positively homogenous. Hence, Lemma 4.5 says that (7) holds for $W = S^m$ if and only if f given by (9) is a gauge.

A gauge f is characterized by the convex set $C = \{x : f(x) \leq 1\}$. Hence μ satisfying (7) is also characterized by such a convex set C, where f is the gauge given by (9). In terms of that C,

$$\mu(w) = \max \{\lambda : \lambda w \in C\}$$

since for λ^* such that $f(\lambda^* w) = 1$, $\mu(w) = \lambda^*$. This observation suggests a close connection between the μ of a subadditive function (π, μ) and the intersection cut of Balas [1]. When we come to discussion of subadditive valid inequalities, this connection will be pursued further.

We now turn to the case where U is a finite subgroup of I^m. A result analogous to Lemma 4.4 is given below.

Lemma 4.6. *If U is a finite subgroup of I^m, then* (6) *can be weakened by assuming that L, in addition to linear independence, has the property that there is no $v' \in U$, and $s'(w)$, $w \in L$, with $0 < s'(L) < s(L)$ such that*

$$v' = F\left(\sum_{w \in L} w\, s'(w)\right).$$

Proof. Suppose (6) holds for all linearly independent subsets L and all s satisfying the conditions of this lemma. We, then, show that (6) holds in general.

If L is linearly independent, $v = F(\sum_{w \in L} w\, s(w))$, and if the conditions of the lemma do not hold, then

$$v' = F\left(\sum_{w \in L} w\, s'(w)\right), \qquad 0 < s'(L) < s(L).$$

Let us choose s' minimal with respect to this property; we can do so because U is a finite subgroup. Then

$$\sum_{w \in L} \mu(w)\, s'(w) \geq \pi(v')$$

by (6) holding in this special case. But now,

$$\begin{aligned} \sum_{w \in L} \mu(w)\, s(w) &= \sum_{w \in L} \mu(w)\, s'(w) + \sum_{w \in L} \mu(w)\,(s(w) - s'(w)) \\ &\geq \pi(v') + \sum_{w \in L} \mu(w)\,(s(w) - s'(w)). \end{aligned}$$

We can now repeat for $s(w) - s'(w)$, $w \in L$, and use (3) to eventually show $\sum \mu(w)\, s(w) \geq \pi(v)$ because $\sum w(s(w) - s'(w))$ also belongs to U. All we need to prove is that eventually $s(w) - s'(w)$ will not allow an $s'(w)$, $0 < s'(L) < s(L) - s'(L)$ with $\sum w\, s'(w) \in U$. We started with a fixed linearly independent subset L of W and $s(w) > 0$ for $w \in L$. In the rectangle $x = \sum_{w \in L} w\, s'(w)$, $0 < s'(L) < s(L)$, there are only a finite number of x such that $F(x) \in U$. It is this number which is reduced by at least one every time we replace $s(w)$ by $s(w) - s'(w)$.

Part of the next lemma is analogous to Lemma 4.5 in the case of a finite set $W \subseteq S^m$. However, a result about (6) is also included. This result is actually included in Lemma 4.4 when $U = I^m$. Hence, the main interest in the weakening of (6) is when U is a finite subgroup of I^m.

Lemma 4.7. *If W is a finite subset of S^m, then* (6) *and* (7) *can be weakened by requiring that L, in addition to linear independence, has the property that there is no $w' \in W \setminus L$, $w' \neq \sum_{w \in L} w\, s(w)$ such that*

$$w' = \sum_{w \in L} w\, s'(w)$$

for $s'(L) \geq 0$.

Proof. The proofs of (6) and (7) will be developed in parallel. In (6), let $x = \sum_{w \in L} w\, s(w)$ and $v = F(x)$. Here x need not be in W but $v \in U$ is required.

Suppose that (6) and (7) hold for all x and L subject to the conditions of the lemma. Then (6) and (7) must be proven for all linearly independent L and x with $x = \sum_{w \in L} w\, s(w)$, $s(w) > 0$ and $x \notin L$. The proof will be by induction on the number of $w' = \sum_{w \in L} w\, s'(w)$ in $W \setminus L$ with $s'(w) \geq 0$ for $w \in L$. By the conditions of the lemma, we have assumed that (6) and (7) hold when there are no such $w' \neq x$.

Suppose, as induction hypothesis, that (6) and (7) hold when there are $k - 1$ or fewer points w'. Suppose now that there are k such points and let them be denoted $w^1, w^2, \ldots, w^k$, where

$$\sum_{w \in L} w\, s_i(w), \qquad i = 0, 1, \ldots, k,$$

with $s_i(w) \geq 0$ and $w^i \notin L$. For consistency, let $w^0 = x$ and $s_0(w) = s(w) > 0$ for $w \in L$. We then have $w^0, w^1, \ldots, w^k$ in $\mathbf{R}^m$ which are not in L but are representable as non-negative linear combinations of $w \in L$. We want to show that $\pi(v) \leq \sum \mu(w)\, s_0(w)$ whenever $v = F(w^0) \in U$ and that $\mu(w^0) \leq \sum \mu(w)\, s_0(w)$ whenever $w^0 \in W$. Here, $w^1, \ldots, w^k$ are in W.

Let $x^1 \in L$ be chosen from $w \in L$ having $s_1(w) > 0$ so that

$$\frac{s_0(w)}{s_1(w)} \geq \frac{s_0(x_1)}{s_1(x_1)} \quad \text{for all } w \in L \text{ having } s_1(w) > 0.$$

Since $w^1 = \sum_{w \in L} w\, s_1(w)$ and $x^1 \in L$ with $s_1(x^1) > 0$, the set $L_1 = (L \setminus \{x^1\}) \cup \{w^1\}$ is a linearly independent set whose non-negative cone (i.e.,

the set of all non-negative linear combinations) is contained in the non-negative cone of L. Furthermore, x^1 is not in the non-negative cone of L_1 since

$$w^1 = x^1 s_1(x^1) + \sum_{w \in L \setminus \{x^1\}} w\, s_1(w),$$

so

$$x^1 = w^1 \frac{1}{s_1(x^1)} - \sum_{w \in L \setminus \{x^1\}} w \frac{s_1(w)}{s_1(x^1)}$$

and $-s_1(w) < 0$ for at least one $w \in L \setminus \{x^1\}$. This latter fact is due to x^1 and w^1 being different points of S^m, so they can not be scalar multiplies of each other.

Next, w^0 will be shown to be in L_1 by the choice of x^1. We know $w^0 = \sum_{w \in L} w\, s_0(w)$ with $s_0(w) > 0$ for $w \in L$. Hence,

$$\begin{aligned} w^0 &= x^1 s_0(x^1) + \sum_{w \in L \setminus \{x^1\}} w\, s_0(w) \\ &= \left(w^1 \frac{1}{s_1(x^1)} - \sum_{w \in L \setminus \{x^1\}} w \frac{s_1(w)}{s_1(x^1)} \right) s_0(x^1) + \sum_{w \in L \setminus \{x^1\}} w\, s_0(w) \\ &= w^1 \frac{s_0(x^1)}{s_1(x^1)} + \sum_{w \in L \setminus \{x^1\}} w \left(s_0(w) - s_1(w) \frac{s_0(x^1)}{s_1(x^1)} \right). \end{aligned}$$

The above representation of w^0 as a linear combination of $w \in L_1$ is a non-negative linear combination if

$$s_0(w) - s_1(w) \frac{s_0(x^1)}{s_1(x^1)} \geq 0, \qquad w \in L \setminus \{x^1\},$$

which is true when $s_1(w) = 0$ by $s_0(w) > 0$ and when $s_1(w) > 0$ by the choice of x^1.

We are now in a position to apply the induction hypothesis. The point $x = w^0$ is representable as a non-negative linear combination of $w \in L_1$, and there are $k - 1$ or fewer points (in fact, such points will be among w^2, ..., w^k) of $W \setminus L_1$ in the non-negative cone of L_1. We first consider (7). If $w^0 \in W$, then the induction hypothesis assures that

$$\mu(w^0) \leq \sum_{w \in L_1} \mu(w)\, r_0(w),$$

where

$$r_0(w^1) = \frac{s_0(x^1)}{s_1(x^1)},$$

$$r_0(w) = s_0(w) - s_1(w)\frac{s_0(x^1)}{s_1(x^1)}, \qquad w \in L \setminus \{x^1\}.$$

We wish to show that

$$\mu(w^0) \leq \sum_{w \in L} \mu(w)\, s_0(w).$$

In order to do so we must reverse the role of w^0 and w^1 in the above development. There we obtained a set L_1 containing w^1 and including w^0 in its non-negative cone. Here we find a set L_0 containing w^0 and including w^1 in its non-negative cone. To do so, let $x^0 \in L$ be chosen such that

$$\frac{s_1(w)}{s_0(w)} \geq \frac{s_1(x^0)}{s_0(x^0)}, \qquad w \in L.$$

Here, $s_0(w) > 0$, $w \in L$, so the restriction $s_0(w) > 0$ need not be stated. Let $L_0 = (L \setminus \{x^0\}) \cup \{w^0\}$. Then, as before, x^0 is not in the non-negative cone of L_0, w^1 is in the non-negative cone of L_0, and the induction hypothesis applies to give

$$\mu(w^1) \leq \sum_{w \in L_0} \mu(w)\, r_1(w),$$

where

$$w^1 = \sum_{w \in L_0} w\, r_1(w),$$

and $r_1(w)$, $w \in L_0$, is given by

$$r_1(w^0) = \frac{s_1(x^0)}{s_0(x^0)},$$

$$r_1(w) = s_1(w) - s_0(w)\frac{s_1(x^0)}{s_0(x^0)}, \qquad w \in L \setminus \{x^0\}.$$

As before, $r_1(w) \geq 0$, $w \in L_0$, *by the choice of* x^0.

We are now in a position to prove (7), that is, $\mu(w^0) \leq \sum_{w \in L} \mu(w)\, s_0(w)$. We have proven two inequalities:

$$\mu(w^0) \leq \sum_{w \in L_1} \mu(w)\, r_0(w),$$

$$\mu(w^1) \leq \sum_{w \in L_0} \mu(w)\, r_1(w).$$

Multiplying the first by $s_1(x^1) > 0$ and the second by $s_0(x^1) > 0$ and adding gives

$$\mu(w^0)\left(s_1(x^1) - s_0(x^1)\frac{s_1(x^0)}{s_0(x^0)}\right) \leq$$

$$\leq s_1(x^1) \sum_{w \in L \setminus \{x^1\}} \mu(w)\, r_0(w) + s_0(x^1) \sum_{w \in L \setminus \{x^1\}} \mu(w)\, r_1(w)$$

because $r_0(w^1) = s_0(x^1)/s_1(x^1)$ and $r_1(w^0) = s_1(w^0) = s_1(x^0)/s_0(x^0)$. Simplifying the above inequality using the expressions derived for $r_0(w)$, $w \in L \setminus \{x^1\}$, and $r_1(w)$, $w \in L \setminus \{x^0\}$, gives the desired inequality provided

$$s_1(x^1) - s_0(x^1)\frac{s_1(x^0)}{s_0(x^0)} > 0.$$

Certainly, the left-hand side is non-negative by the choice of x^0. If it were zero, then the ratios $s_1(w)/s_0(w)$ would all be the same by the choice of x^1. Then, w^0 and w^1 would be scalar multiples of each other, a contradiction because they are different points of $W \subseteq S^m$. The proof is completed.

There are four different cases which will be considered in more detail: (i) $U = I^m$ and $W = S^m$; (ii) $U = I^m$ and W a finite subset of S^m; (iii) U a finite subgroup and $W = S^m$; and (iv) U a finite subgroup and W a finite subset of S^m. Case (i) is treated by Lemmas 4.4 and 4.5, case (ii) by Lemmas 4.4 and 4.7, case (iii) by Lemmas 4.6 and 4.5, and case (iv) by Lemmas 4.6 and 4.7. In case (iii), however, condition (6) is only weakened in Lemma 4.6 by requiring that there be no $v' = F(\sum_{w \in L} w\, s'(w))$, $0 < s'(L) < s(L)$. There is more that can be said.

Lemma 4.8. *If U is a finite subgroup and $W = S^m$, then* (6) *is equivalent to*

$$\pi(v) \leq \lambda\, \mu(w), \tag{10}$$

when $F(\lambda w) = v$, $\lambda > 0$, and $F(\lambda' w) \notin U$ for $0 < \lambda' < \lambda$.

Proof. Clearly (10) is a special case of (6).

The remainder of the proof is to show that (3), (10) and (7) imply (3), (6) and (7). Suppose not. Then,

$$\pi(v) > \sum_{w \in L} \mu(w)\, s(w), \qquad v = F\left(\sum_{w \in L} w\, s(w)\right),$$

and no $v' = F(\sum_{w \in L} w\, s'(w))$ for $v' \in U$ and $0 < s'(L) < s(L)$. Let

$$x = \sum_{w \in L} w\, s(w).$$

Then, $x \in \mathbf{R}^m \setminus \{0\}$ so $\lambda = \max_i |x_i|$ is positive and $w' = (1/\lambda)\, x \in S^m$. By (7) and $w' = (1/\lambda)\, x = (1/\lambda) \sum_{\in L} w\, s(w)$,

$$\mu(w') \leq 1/\lambda \sum_{w \in L} \mu(w)\, s(w).$$

Combining the above gives

$$\pi(v) > \lambda\, \mu(w'), \qquad v = F(\lambda\, w'),$$

a contradiction to (10). In addition, $F(\lambda' w') \notin U$ for $0 < \lambda' < \lambda$ since $F(\sum_{w \in L} w\, s'(w))$ is not in U for $0 < s'(L) < s(L)$.

5. Subadditive valid inequalities

Let us now establish the close connection between subadditive valid inequalities and subadditive functions. Any subadditive function (π, μ) can be scaled by multiplying both π and μ by a positive number λ. The next theorem says that a properly scaled subadditive function is a subadditive valid inequality.

Theorem 5.1. *If (π, μ) is a subadditive function on a subgroup U of I^m and if*

$$\pi(u) + \sum_{w \in L} \mu(w)\, s(w) \geq 1, \quad \text{whenever } u_0 = u + F\left(\sum_{w \in L} w\, s(w) \right) \tag{11}$$

for L a linearly independent subset of W and $s(L) > 0$, then (π, μ) is a subadditive valid inequality for $\mathrm{P}(U, W, u_0)$. Conversely, a subadditive valid inequality (π, μ) for $\mathrm{P}(U, W, u_0)$, where U is a subgroup of I^m is a subadditive function satisfying (11). *In condition* (11) *we can clearly exclude $s(w)$ for which*

$$u_0 = u + F\left(\sum_{w \in L} w\, s'(w) \right) \quad \text{for } 0 < s'(L) < s(L).$$

Proof. The converse part of the theorem and the last assertion are obvious.

Lemma 4.1 showed that (π, μ) satisfies (4) and (5), so if (π, μ) is a valid inequality, then it is a subadditive valid inequality.

We now prove that (π, μ) is a valid inequality. First, if we know (11) for linearly independent subsets L of W, then in fact (11) is true for any subset of W. As always, s must have a finite support. The proof of this assertion is exactly the same as proving (4) in Lemma 4.1.

Secondly, from (3) we know

$$\sum_{u\in U} \pi(u)\, t(u) \geq \pi\left(\sum_{u\in U} u\, t(u)\right)$$

by Lemma 4.2. Hence, for any $(t, s) \in T(U, W, u_0)$,

$$\sum_{u\in U} \pi(u)\, t(u) + \sum_{w\in W} \mu(w)\, s(w) \geq$$

$$\geq \pi\left(\sum_{u\in U} u\, t(u)\right) + \sum_{w\in W} \mu(w)\, s(w) \geq 1,$$

completing the proof of the theorem.

When U is the zero subgroup, $U = \{0\}$, $\pi(u) = 0$ and (11) becomes

$$\sum_{w\in L} \mu(w)\, s(w) \geq 1, \quad \text{whenever } u_0 = F\left(\sum_{w\in L} w\, s(w)\right).$$

The above is the condition on the scaling of μ in order for it to be a valid inequality. In this special case, the only content of the theorem is that

$$\sum_{w\in W} \mu(w)\, s(w) \geq 1$$

will be true for all non-negative s with finite support for which $u_0 = F(\sum w\, s(w))$ provided it is true for all non-negative s with linearly independent support.

In the above case, it is interesting to return to the gauge f defined from μ by (9) and the convex set $C = \{x: f(x) \leq 1\}$. This convex set has $w\, \mu(w)$ on its boundary, as mentioned in the discussion following Lemma 4.5. The condition for μ to be a valid inequality is that $\sum \mu(w)\, s(w) \geq 1$, whenever $u_0 = F(\sum w\, s(w))$. If μ is subadditive, f given by (9) is convex and $f(w) = \mu(w)$, so

$$\sum \mu(w)\, s(w) = \sum f(w)\, s(w) = \left(\sum s(w)\right) \sum f(w) \frac{s(w)}{\sum s(w)}$$

$$\geq \left(\sum s(w)\right) f\left(\sum w \frac{s(w)}{\sum s(w)}\right) = f\left(\sum w\, s(w)\right).$$

If $\sum w\, s(w)$ is outside of C or on its boundary, then $f(\sum w\, s(w)) \geq 1$ and $\sum \mu(w)\, s(w) \geq 1$ is satisfied. Thus it suffices to require that $\sum w\, s(w)$ lie outside

or on the boundary of C whenever $u_0 = F(\sum w\, s(w))$. This argument is similar to the reasoning used by Balas in defining his intersection cut [1] and by Glover in developing the convexity cut [4]. For further details of this comparison, see [2].

When U is not the zero subgroup, μ is closely related to π whenever (π, μ) is a minimal subadditive valid inequality as will be shown in the next section. Only when U is the zero subgroup will our π correspond to an arbitrary convex set C.

The next corollary further limits the conditions which must hold for (π, μ) to be a valid inequality for the case where U is a finite subgroup and W is a finite set.

Corollary 5.2. *If (π, μ) is a subadditive function on (U, W), where U is a finite subgroup of I^m and W is a finite subset of S^m, and if (11) holds whenever there is neither any $v \in U$, $v = u + F(\sum_{w \in L} w\, s'(w))$, where $0 < s'(L) < s(L)$, nor any $w' \in W$, $w' = \sum_{w \in L} w\, s'(w)$, where $0 < s'(L)$, then (π, μ) is a subadditive valid inequality for $P(U, W, u_0)$.*

Proof. The proof closely parallels the proofs of Lemma 4.6 and 4.7. There, we were proving (6) and (7) under similar restrictions on w' and L. Roughly speaking, we transfer those arguments here by shifting the origin to a fixed u; that is, by considering $w' = \sum_{w \in L} w\, s(w)$ with $F(w') = u_0 - u$ and then considering the inequality $\sum_{w \in L} \mu(w)\, s(w) \geq 1 - \pi(u)$. We, of course, use (6) and (7) in the proof here.

Corollary 5.3. *If (π, μ) is a subadditive function on (U, S^m), where U is a finite subgroup of I^m and if*

$$\pi(u) + \lambda\mu(w) \geq 1, \tag{12}$$

whenever $u_0 = u + F(\lambda w)$, where $u + F(\lambda' w)$ is not in U and is not u_0 for $0 < \lambda' > \lambda$, then (π, μ) is a subadditive valid inequality for $P(U, S^m, u_0)$.

The proof of this corollary closely parallels that of Lemma 4.8.

Corollary 5.4. *If $u_0 \in U$ in Theorem 5.1, then (11) can be weakened to $\pi(u_0) \geq 1$. In particular, when $U = I^m$, $\pi(u_0) \geq 1$ is necessary and sufficient to insure that a subadditive function is a subadditive valid inequality for $P(U, W, u_0)$.*

In principal, we now have a constructive definition of the subadditive valid inequalities when U is a finite subgroup of I^m and W is a finite subset of S^m. That is, (π, μ) must be non-negative, satisfy (3), (6), (7), and (11) where (6), (7), and (11) need only be required in the cases given in Lemmas 4.6, 4.7, and 5.2. These restrictions on (π, μ) are linear inequalities, and there are only a finite number of such inequalities provided the $w \in W$ have rational numbers for each component. In that case, the extreme subadditive (π, μ) can be found. In order to find the extreme valid inequalities, we need to know which of these extreme subadditive inequalities are minimal. The next section addresses that question.

6. Minimal inequalities

The previous development provides necessary and sufficient conditions for (π, μ) to be a subadditive valid inequality for $P(U, W, u_0)$ when U is a subgroup of I^m. Since minimal valid inequalities are subadditive by Theorem 3.3, we know some necessary conditions for minimal valid inequalities. In this section, necessary and sufficient conditions will be developed for some cases. The case $U = I^m$ and any $W \subseteq S^m$ will be treated first.

Theorem 6.1. *Let π be a function on I^m to the non-negative reals and let μ be a function on S^m to the non-negative reals. For any $u_0 \in I^m \setminus \{0\}$ and any $W \subseteq S^m$, (π, μ) is a minimal valid inequality for $P(I^m, W, u_0)$ if and only if*

$$\pi(u) + \pi(v) \geq \pi(u + v), \qquad u, v \in I^m, \tag{13}$$

$$\mu(w) = \lim_{h \downarrow 0} \frac{\pi(F(h\, w))}{h}, \qquad w \in W \subseteq S^m, \tag{14}$$

$$\pi(u) + \pi(u_0 - u) = 1 \quad \text{for all } u \in I^m. \tag{15}$$

Before proving the theorem, a lemma will be shown. Conditions (13) and (14) here are almost enough to assure that (π, μ) is a subadditive function. The next lemma says that they are, in fact, enough.

Lemma 6.2. *If π and μ are functions as given in Theorem* 6.1 *and satisfy* (13) *and* (14), *then (π, μ) is a subadditive function on (I^m, W).*

Proof. By assuming (14), we assume that the limit there exists. By Lemma 4.4, we need only show (7); that is,

$$\mu(x) \leq \sum_{w \in L} \mu(w)\, s(w), \qquad x = \sum_{w \in L} w\, s(w),$$

where L is a linearly independent subset of W. Since L is linearly independent in $\mathbf{R}^m$, there are m or fewer elements in L. Fix L and $s(w) > 0$, $w \in L$, with $x = \sum w\, s(w) \in W$. For any $\varepsilon > 0$, let $\varepsilon' = \varepsilon/(2\ |L|)$. By (14), there is an $h_0(w) > 0$ for $w \in L$ such that

$$\mu(w)\, s(w) = \mu(w\, s(w)) \geq \frac{\pi(F(h\, w\, s(w)))}{h} - \varepsilon', \qquad 0 < h < h_0(w).$$

Similarly there is an $h_x > 0$ such that $\mu(x) \leq \pi(F(h\, x))/h + \frac{1}{2}\varepsilon$ for $0 < h < h_x$. Let $h_\varepsilon = \min\{h_x, h_0(w): w \in L\}$. Then $h_\varepsilon > 0$ and for $0 < h < h_\varepsilon$,

$$\begin{aligned}\sum_{w \in L} \mu(w)\, s(w) &= \sum_{w \in L} \mu(w\, s(w)) \\ &\geq \frac{1}{h} \sum_{w \in L} \pi(F(h\, w\, s(w))) - \tfrac{1}{2}\varepsilon \\ &\geq \frac{1}{h}\pi\left(\sum_{w \in L} F(h\, w\, s(w))\right) - \tfrac{1}{2}\varepsilon\end{aligned}$$

by subadditivity of π. Clearly, $\sum F(h\, w\, s(w)) = F(h \sum w\, s(w)) = F(h\, x)$ since the first summation is actually modulo 1. Hence,

$$\begin{aligned}\sum_{w \in L} \mu(w)\, s(w) &\geq \frac{1}{h}\pi(F(h\, x)) - \tfrac{1}{2}\varepsilon \\ &\geq \mu(x) - \varepsilon.\end{aligned}$$

Since $\varepsilon > 0$ is arbitrary, the inequality (7) follows.

Proof of theorem 6.1. Suppose (13), (14) and (15) hold for (π, μ). By the previous lemma, (π, μ) is a subadditive function on (I^m, W). By (15) with $u = 0$, $\pi(u_0) = 1$. By Corollary 5.4, (π, μ) is a subadditive valid inequality. To show it minimal, we must show that there is no other valid inequality (ρ, ν) for $P(I^m, S^m, u_0)$ with $(\rho, \nu) < (\pi, \mu)$. Clearly, (15) excludes $\rho < \pi$ since then $\rho(u) < \pi(u)$ for some $u \in U$ and $\rho(u_0 - u) \leq \pi(u_0 - u)$, so $\rho(u) + \rho(u_0 - u) < 1$. We next show that (14) excludes $\nu < \mu$.

Let $\nu(w) < \mu(w)$ and $\varepsilon = \mu(w) - \nu(w) > 0$ for some $w \in W$. By (14), there is an h, $0 < h < 1$, with $\mu(w) < \pi(F(h\, w))/h + \varepsilon$. Then $(u_0 - F(h\, w)) + F(h\, w) = u_0$ so (ρ, ν) being valid implies $\rho(u_0 - F(h\, w)) + h\, \nu(w) \geq 1$. But by the above,

$$h\, \nu(w) = h\, \mu(w) - h\, \varepsilon < \pi(F(h\, w)) + h\, \varepsilon - h\, \varepsilon \leq \rho(F(h\, w)).$$

Hence,

$$\rho(u_0 - F(h\,w)) + h\,v(w) < \rho(u_0 - F(h\,w)) + \rho(F(h\,w)) = 1,$$

a contradiction. Therefore, (π, μ) is a minimal valid inequality for $P(I^m, W, u_0)$ whenever (13), (14) and (15) hold.

Suppose now that (π, μ) is a minimal valid inequality for $P(I^m, W, u_0)$. We show that (13), (14) and (15) hold. By Theorem 3.3, (π, μ) is a subadditive valid inequality. By Theorem 5.1, (π, μ) is a subadditive function, and by Corollary 5.4 it satisfies $\pi(u_0) \geq 1$. By definition of subadditive function, (13) holds. By Lemma 4.4, the limit in (14) exists and

$$\mu(w) \geq \lim_{h \downarrow 0} \frac{\pi(F(h\,w))}{h}, \qquad w \in W.$$

By Lemma 6.2 and minimality of μ, equality must hold because otherwise $(\pi, \lim \pi(F(h\,w))/h)$ is a valid inequality less than (π, μ). Hence (14) holds.

We next show that (15) holds. The proof is similar to the proof of [6, Theorem 1.6]. If (15) does not hold, then for some $v \in I^m$,

$$\pi(v) + \pi(u_0 - v) = 1 + \delta, \qquad \delta > 0.$$

At least one of $\pi(v)$, $\pi(u_0 - v)$ is positive, so suppose $\pi(v) > 0$.

Define ρ on I^m by

$$\rho(u) = \begin{cases} \dfrac{1}{1+\delta}\pi(v) & \text{if } u = v, \\ \pi(u) & \text{if } u \neq v, \quad u \in I^m. \end{cases}$$

Since $\delta > 0$ and $\pi(v) > 0$, it follows that $\rho < \pi$. A contradiction to minimality of π will be reached if (ρ, π) can be shown to be a valid inequality for $P(I^m, W, u_0)$.

Let $(t, s) \in T(I^m, W, u_0)$. We wish to show that $(\rho, \mu) \cdot (t, s) \geq 1$; i.e.,

$$\sum_{u \in I^m} \rho(u)\,t(u) + \sum_{w \in W} \mu(w)\,s(w) \geq 1.$$

By definition of ρ,

$$\sum_{u \in I^m} \rho(u)\,t(u) + \sum_{w \in W} \mu(w)\,s(w) =$$

$$= \sum_{\substack{u \in I^m \\ u \neq v}} \pi(u)\,t(u) + \sum_{w \in W} \mu(w)\,s(w) + \frac{1}{1+\delta}\pi(v)\,t(v).$$

If $t(v) \geq (1 + \delta)/\pi(v)$, then clearly $(p, \mu) \cdot (t, s) \geq 1$ holds. If $t(v) = 0$, then it holds by (π, μ) being a valid inequality. Hence, suppose

$$1 \leq t(v) < \frac{1 + \delta}{\pi(v)}.$$

Regrouping the term $(1/(1 + \delta))\,\pi(v)\,t(v)$ above gives

$$\begin{aligned}
&\sum_{u \in I^m} \rho(u)\,t(u) + \sum_{w \in W} \mu(w)\,s(w) = \\
&\quad = \left(\sum_{\substack{u \in I^m \\ u \neq v}} \pi(u)\,t(u) + \pi(v)\,(t(v) - 1) \right) + \sum_{w \in W} \mu(w)\,s(w) \\
&\qquad\qquad + \pi(v) - \frac{\delta}{1 + \delta}\pi(v)\,t(v) \\
&\quad \geq \pi \left(\sum_{\substack{u \in I^m \\ u \neq v}} u\,t(u) + v(t(v) - 1) \right) + \sum_{w \in W} \mu(w)\,s(w) \\
&\qquad\qquad + \pi(v) - \frac{\delta}{1 + \delta}\pi(v)\,t(v),
\end{aligned}$$

by subadditivity and Lemma 4.2. Subadditivity also implies (4), so the above is greater than or equal to

$$\pi(v') + \pi(v) - \frac{\delta}{1 + \delta}\pi(v)\,t(v) \geq \pi(v') + \pi(v) - \delta$$

by $t(v) < (1 + \delta)/\pi(v)$. Here

$$v' = \sum_{\substack{u \in I^m \\ u \neq v}} u\,t(u) + v(t(v) - 1) + F\left(\sum_{w \in W} w\,s(w) \right).$$

But $(t, s) \in T(I^m, W, u_0)$, so

$$\sum_{u \in I^m} u\,t(u) + F\left(\sum_{w \in W} w\,s(w) \right) = u_0.$$

Hence $v' = u_0 - v$, and we get

$$\sum_{u \in I^m} \rho(u)\,t(u) + \sum_{w \in W} \mu(w)\,s(w) \geq \pi(u_0 - v) + \pi(v) - \delta = 1.$$

Hence, (ρ, μ) is a valid inequality for $\mathrm{P}(I^m, W, u_0)$, a contradiction is reached and the theorem is proven.

In this case, where $U = I^m$, the set of minimal valid inequalities, is a convex set and the extreme points of the set of minimal valid inequalities are precisely the extreme valid inequalities by Theorem 3.2.

The following corollary is a consequence of Lemma 6.2 but is interesting in itself. Recall the definition (9) of the homogenous extension f of μ to $\mathbf{R}^m$.

Corollary 6.3. *If π is subadditive function on I^m (that is, $\pi(u) + \pi(v) \geq \pi(u + v)$) and if the limit*

$$\lim_{h \downarrow 0} \frac{\pi(F(h\, w))}{h}$$

exists for all $w \in S^m$, then the homogenous extension f of μ for μ defined by

$$\mu(w) = \lim_{h \downarrow 0} \frac{\pi(F(h\, w))}{h}, \qquad w \in S^m,$$

is convex.

Proof. Lemma 6.2 shows that (π, μ) is a subadditive function on (I^m, S^m), and, hence, (7) holds. By Lemma 4.5, the homogenous extension of μ to $\mathbf{R}^m$ is a convex function. Thus, the corollary is proven.

We next consider finite subgroups U of I^m and finite subsets W of S^m.

Theorem 6.4. *Let π be a function on I^m to the non-negative reals and let μ be a function on S^m to the non-negative reals. Let U be a finite subgroup of I^m and let W be a finite subset of S^m such that each element $w \in W$ has rational components. For any $u_0 \in I^m \setminus \{0\}$, (π, μ) is a minimal valid inequality if and only if all of* (16)–(19) *below hold:*

$$\pi(u) + \pi(v) \geq \pi(u + v), \qquad u, v \in U, \tag{16}$$

$$\sum_{w \in L} \mu(w)\, s(w) \geq \pi(v), \quad \text{whenever } v = F\left(\sum_{w \in L} w\, s(w)\right), \tag{17}$$

$$\sum_{w \in L} \mu(w)\, s(w) \geq \mu(x), \quad \text{whenever } x = \sum_{w \in L} w\, s(w), \tag{18}$$

$$\pi(v) + \sum_{w \in L} \mu(w)\, s(w) \geq 1, \quad \text{whenever } u_0 = F\left(v + \sum_{w \in L} w\, s(w)\right), \tag{19}$$

where in (17), (18) *and* (19) *the set L is a linearly independent subset of W, $s(w) \geq 0$, and we can assume the condition of Lemma* 4.7: *there is no $y \in W$,*

$y \notin L$, $y \neq \sum w\, s(w)$, such that $y = \sum_{w \in L} w\, s'(w)$, for some $s'(w) \geq 0$, $w \in L$. In (17) we can assume there is no $v' \in U$ and $s'(w)$, $0 \leq s'(w) \leq s(w)$ for $w \in L$, such that $v' = F(\sum_{w \in L} w\, s'(w))$. Let

$$U_0 = \{v \in U\colon \pi(v) + \sum_{w \in L} \mu(w)\, s(w) = 1 \text{ for some } L \text{ and } s(L) \text{ with } u_0 = v + \sum_{w \in L} w\, s(w)\}$$

and let

$$W_0 = \{x \in W\colon \pi(v) + \sum_{w \in L} \mu(w)\, s(w) = 1 \text{ for some } v \in U_0 \text{ and } L \text{ with } x \in L \text{ and } s(x) > 0\}.$$

For minimality, we need

for every $u \in U$, either $u \in U_0$ or there exists $v \in U_0$ such that $\pi(u) + \pi(v - u) = \pi(v)$, (20)

for every $w \in W$, either $\mu(w) = 0$ or $w \in W_0$ or there exists $x_0 \in W_0$ with $\mu(x_0) = \sum_{x \in L} \mu(x)\, s(x)$, where L is a linearly independent subset of W, $w \in L$, $s(w) > 0$, and $x_0 = \sum_{x \in L} x\, s(x)$. (21)

Proof. Conditions (16)–(19) are necessary and sufficient for (π, μ) to be a subadditive valid inequality by Theorems 4.6, 4.7, and 5.2. By Theorem 3.3, the minimal valid inequalities are subadditive. Thus we need only prove that (20) and (21) are necessary and sufficient for a subadditive valid inequality to be minimal.

Suppose, first, that (π, μ) is a subadditive valid inequality and that (20) and (21) hold. Clearly, (20) implies that no $\pi(u)$ can be lowered and still have (π, μ) be valid. Similarly, (21) implies that no $\mu(w)$ can be decreased and still have (π, μ) valid.

Finally, suppose (π, μ) is a minimal valid inequality. We must show that (20) and (21) hold. Clearly, U_0 and W_0 are both non-empty for a minimal valid inequality.

We first prove that if (20) does not hold for some u, then (20) does not hold for some u for which $\pi(u) > 0$. Suppose $\pi(u) = 0$ and (20) does not hold for u. Then,

$$\pi(u_1 - u) = \pi(u_1) + \delta_1, \qquad \delta_1 > 0, \quad \text{for all } u_1 \in U_0.$$

Clearly $\pi(u_1 - u) > 0$ for all $u_1 \in U_0$. If (20) does not hold for some $u_1 - u$, we are done. Otherwise, (20) does hold for every $u_1 - u$, so that for every

$u_1 \in U_0$ either $u_1 - u \in U_0$ or there exists $u_2 \in U_0$, depending on u_1, for which

$$\pi(u_1 - u) + \pi(u_2 - (u_1 - u)) = \pi(u_2).$$

Clearly this $u_2 \neq u_1$, since if $u_2 = u_1$, then the previous inequality becomes $\pi(u_1 - u) + \pi(u) = \pi(u_1)$, contradicting the fact that (20) does not hold for u.

Consider first the case that for some $u_1 \in U_0$, $u_1 - u \in U_0$. Let $u_2 = u_1 - u$. Then by (20) not holding for u and by $\pi(u) = 0$,

$$\pi(u_1 - u) = \pi(u_1) + \delta_1, \qquad \delta_1 > 0,$$

$$\pi(u_2 - u) = \pi(u_2) + \delta_2, \qquad \delta_2 > 0.$$

But $u_2 = u_1 - u$ and, hence, $u_2 - u = u_1$, so adding the two above equations and cancelling gives

$$0 = \delta_1 + \delta_2,$$

a contradiction.

Suppose now that for every $u_1 \in U_0$, $u - u_1 \notin U$ and there exists a $u_2 \in U_0$, depending on u_1, for which $\pi(u_2) = \pi(u_1 - u) + \pi(u_2 - (u_1 - u))$. This $u_2 \neq u_1$. Begin with any particular $u_1 \in U_0$. Then we get u_2 as above, $u_2 \neq u_1$. Now from u_2 we can similarly find $u_3 \neq u_2$. Given u_3 we can find u_4. Since U is finite, eventually one of the u_j must be one of the previous u_i's. Then

$$\begin{aligned} &\pi(u_i - u) + \pi(u_{i+1} - (u_i - u)) = \pi(u_{i+1}), \\ &\qquad\qquad \vdots \\ &\pi(u_{j-1} - u) + \pi(u_j - (u_{j-1} - u)) = \pi(u_j), \end{aligned}$$

where $u_i = u_j$. Now, from u not satisfying (20) and $\pi(u) = 0$, $\pi(u_k - u) = \pi(u_k) + \delta_k$, $\delta_k > 0$. Substituting in the above gives

$$\begin{aligned} &\pi(u_i) + \delta_i + \pi(u_{i+1} - (u_i - u)) = \pi(u_{i+1}), \\ &\pi(u_{i+1}) + \delta_{i+1} + \pi(u_{i+2} - (u_{i+1} - u)) = \pi(u_{i+2}), \\ &\qquad\qquad \vdots \\ &\pi(u_{j-1}) + \delta_{j-1} + \pi(u_j - (u_{j-1} - u)) = \pi(u_j). \end{aligned}$$

Cancelling $\pi(u_i) = \pi(u_j)$ and $\pi(u_{i+1}) = \pi(u_{j+1})$, ..., from the equations gives

$$\sum_{k=i}^{j-1} \delta_k + \sum_{k=i+1}^{j} \pi(u_k - (u_{k-1} - u)) = 0,$$

a contradiction since $\delta_k > 0$ all k.

Now assume (20) is violated by u for which $\pi(u) > 0$. Let

$$\delta_1 = \min\{\pi(u) + \pi(u_1 - u) - \pi(u_1): u_1 \in U_0\}.$$

Then $\delta_1 > 0$ because (20) does not hold and because U_0 is finite. Let

$$\delta_2 = \min_{v \notin U_0} \left\{ \min\{\pi(v) + \sum_{w \in L} \mu(w)\, s(w) - 1\} \right\},$$

where the interior min is over all $s \geq 0$ for which

$$v + F\left(\sum_{w \in L} w\, s(w)\right) = u_0, \qquad L \text{ linearly independent.}$$

Now, $\delta_2 > 0$ because it can be shown to be a minimum of a finite number of terms each of which is positive. Since $v \notin U_0$, each term is clearly positive. Also, there are only a finite number of $v \notin U_0$ since U is finite. Since W is finite, there are clearly only a finite number of linearly independent subsets L of W. Thus, we can fix $v \notin U_0$ and $L \subseteq W$, L linearly independent. There are a discrete number of points congruent modulo 1 to $u_0 - v$ in $\mathbf{R}^m$. Each such point can be written in exactly one way as a linear combination $\sum w\, s(w)$ of the $w \in L$ because L is linearly independent. Each w has rational coordinates by assumption, say $w = (p_i/q_i, i = 1, \ldots, m)$. Let

$$s(w) \geq \text{l.c.m.}\{q_1, \ldots, q_m\} = D(w),$$

where l.c.m. means least common multiple. Then, $w\, s(w)$ is congruent to $w(s(w) - D(w))$. Since $\mu(w) \geq 0$, we know in the min defining δ_2 that $s(w) < D(w)$. For a given $u_0 - v$ and a given L, there are only a finite number of points in $\mathbf{R}^m$ congruent to $u_0 - v$ which can be written as $\sum w\, s(w)$ with $0 \leq s(w) < D(w)$. Each of these points can only be formed in one way as such a linear combination. Hence, only a finite number of terms

$$\pi(v) + \sum_{w \in L} \mu(w)\, s(w) - 1$$

need be considered in defining δ_2 and $\delta_2 > 0$.

Let $\delta = \min\{\delta_1, \delta_2\}$, so $\delta > 0$. Let

$$\rho(v) = \begin{cases} \dfrac{1}{1+\delta}\,\pi(u) & \text{if } v = u, \\ \pi(u) & \text{if } v \neq u. \end{cases}$$

By $\pi(u) > 0$, $\rho(U) < \pi(U)$. If (ρ, μ) is shown to be a valid inequality, then a contradiction is reached. Let (t, s) be a solution to $\mathrm{P}(U, W, u_0)$. Then

$$\sum_{v \in U} \rho(v)\, t(v) = \rho(u)\, t(u) + \sum_{v \neq u} \pi(v)\, t(v)$$

$$= \frac{\pi(u)\, t(u)}{1 + \delta} + \sum_{v \neq u} \pi(v)\, t(v).$$

If $t(u) = 0$, then $\sum \rho(v)\, t(v) = \sum \pi(v)\, t(v)$, so clearly

$$\sum_{v \in U} \rho(v)\, t(v) + \sum_{w \in W} \mu(w)\, s(w) \geq 1.$$

If $t(u) \geq (1 + \delta)/\pi(u)$, then $\sum \rho(v)\, t(v) \geq 1$ already. Otherwise $1 \leq t(u) < (1 + \delta)/\pi(u)$. Let us regroup:

$$\sum_{v \in U} \rho(v)\, t(v) = \sum_{v \neq u} \pi(v)\, t(v) + \pi(v)\, t(u) - \frac{\delta\, \pi(u)\, t(u)}{1 + \delta}.$$

Hence, $\sum \rho(v)\, t(v) > \sum \pi(v)\, t(v) - \delta$ and $t(u) \geq 1$. Hence,

$$\sum_{v \in U} \rho(v)\, t(v) > \left\{ \sum_{v \neq u} \pi(v)\, t(v) + \pi(u)\, (t(u) - 1) \right\} + \pi(u) - \delta$$

$$\geq \pi(u_1 - u) + \pi(u) - \delta,$$

by subadditivity, where $u_1 = \sum v\, t(v)$. If $u_1 \in U_0$, then by $\delta \leq \delta_1$,

$$\sum_{v \in U} \rho(v)\, t(v) + \sum_{w \in W} \mu(w)\, s(w) \geq \pi(u_1) + \sum_{w \in W} \mu(w)\, s(w) \geq 1.$$

If $u_1 \notin U_0$, then the same result follows from $\delta \leq \delta_2$.

We now turn to (21). The proof of (21) is somewhat similar to that of (20). Suppose $w \in W$ and (21) does not hold for w. Then $w \notin W_0$, $\mu(w) > 0$, and if any $x_0 \in W_0$ with $x_0 = \sum_{x \in L} x\, s(x)$, $w \in L$, and $s(w) > 0$, then

$$\mu(x_0) < \sum_{x \in L} \mu(x)\, s(x).$$

Here, L is linearly independent. Since W is finite, there are only finitely many subsets L and finitely many $x_0 \in W_0$. For each $x_0 \in W_0$ and L with $w \in L$, there is a unique $s(x)$, $x \in L$, such that $x_0 = \sum_{x \in L} x\, s(x)$. Define

$$\delta_1 = \min \left\{ \sum_{x \in L} \mu(x)\, s(x) - \mu(x_0) \colon w \in L,\ s(w) > 0,\ s(x) \geq 0 \text{ for } x \in L, \right.$$

$$\left. x_0 = \sum_{x \in L} x\, s(x), \text{ and } L \text{ is a linearly independent subset of } W \right\}.$$

Clearly $\delta_1 > 0$ since it is the minimum of a finite number of positive numbers.

Just as in proving (20), $\delta_2 > 0$, where

$$\delta_2 = \min\left\{\pi(v) + \sum_{x\in L} \mu(x)\, s(x) - 1 : v + F\left(\sum_{x\in L} x\, s(x)\right) = u_0,\ w \in L,\right.$$
$$\left. s(w) > 0,\ s(x) \geq 0 \text{ for } x \in L,\ L \text{ linearly independent}\right\}.$$

As in proving (20), δ_2 is the minimum of a finite set of positive numbers because we need only consider $s(x) \leq D(x)$.

Hence $\delta = \min\{\delta_1, \delta_2\} > 0$. Form v by letting

$$v(x) = \begin{cases} \mu(x) & \text{if } x \neq w, \\ \dfrac{1}{1+\delta}\mu(w) & \text{if } x = w. \end{cases}$$

Since (21) is violated, $\mu(w) > 0$ and hence $v(w) < \mu(w)$. It remains to show that (π, v) is valid. Let $(t, s) \in T(U, W, u_0)$. Then, as in showing (20),

$$\sum_{u\in U} \pi(u)\, t(u) + \sum_{x\in W} v(x)\, s(x) =$$

$$= \sum_{u\in U} \pi(u)\, t(u) + \sum_{x\in W} \mu(x)\, s(x) - \frac{\delta\, \mu(w)}{1+\delta}\, s(w).$$

If $s(w) \geq (1+\delta)/\mu(w)$, then clearly $v(w)\, s(w) \geq 1$ so (π, v) is valid. Otherwise, $0 < s(w) < (1+\delta)/\mu(w)$. Then

$$\sum_{u\in U} \pi(u)\, t(u) + \sum_{x\in W} v(x)\, s(x) > \sum_{u\in U} \pi(u)\, t(u) + \sum_{x\in W} \mu(x)\, s(x) - \delta.$$

As in showing (20), the proof now follows by the way δ was defined.

7. A fill-in procedure

This section gives a way to form a subadditive valid inequality (π, μ) for the problem $P(I^m, S^m, u_0)$ from a subadditive valid inequality for the problem $P(U, W, u_0)$, where U is a finite subgroup and W is a finite subset of S^m such that any $x \in \mathbf{R}^m$ can be written as $x = \sum_{w\in W} w\, s(w)$ for some $s(W) \geq 0$. The assumption that every $x \in \mathbf{R}^m$ can be written in this way is a restriction on W used below.

The initial valid inequality for $P(U, W, u_0)$ can be obtained from com-

puting extreme valid inequalities using Theorem 6.4. In Section 8, some extreme valid inequalities are given for some particular cases with $m = 2$.

The method given here is analogous to the "two-slope fill-in" of [6] (see Theorem 3.3).

We begin by extending μ from W to S^m. For each $x \in S^m$, let

$$\mu(x) = \min\left\{\sum_{w \in L} \mu(w)\, s(w) : x = \sum_{w \in L} w\, s(w)\right\},$$

where L is a linearly independent subsrt of W and $s(L) \geq 0$. By our assumption that each $x \in \mathbf{R}^m$ can be written as a non-negative combination of $w \in W$, we know that $\mu(x)$ is finite for all $x \in S^m$.

Now, let g be the gauge function which is the homogenous extension of μ to the entire space $\mathbf{R}^m$. Extend π to I^m by letting

$$\pi(v) = \min_{u \in U} \{\pi(u) + g(x) : v = u + F(x)\},$$

In this way, a function (π, μ) has been defined on (I^m, S^m). It remains to show that the function is a valid inequality for the problem $\mathrm{P}(I^m, S^m, u_0)$. We will show (π, μ) to be a subadditive valid inequality. By Lemmas 4.4 and 4.5 by Corollary 5.4, we need to show four conditions:

$$\pi(u) + \pi(v) \geq \pi(u + v) \quad \text{for all } u, v \in I^m,$$

g is convex,

$$\mu(w) \geq \lim_{h \downarrow 0} \frac{\pi(F(h\, w))}{h} \quad \text{for all } w \in S^m,$$

$$\pi(u_0) \geq 1.$$

We first show g to be convex. It suffices to show that the set of $\{x : g(x) \leq 1\}$ is convex since g is a gauge function. In fact, $\{x : g(x) \leq 1\} = C$, where

$$C = \text{convex hull}\left\{\frac{w}{\mu(w)} : w \in W\right\}.$$

Clearly,

$$g(x) = \min\left\{\sum_{w \in W} \mu(w)\, s(w) : x = \sum_{w \in W} w\, s(w),\ s(W) \geq 0\right\},$$

so $g(x) \leq 1$ if and only if there is an $s(W) \geq 0$ such that

$$\sum_{w \in W} \mu(w)\, s(w) \leq 1 \text{ with } x = \sum_{w \in W} w\, s(w).$$

Now, $0 \in C$ so

$$\sum_{w \in W} \lambda(w) \frac{w}{\mu(w)} \in C,$$

provided $\sum \lambda(w) \leq 1$. Letting $\lambda(w) = \mu(w)\, s(w)$, the above condition for $g(x) \leq 1$ shows that $g(x) \leq 1$ implies $x \in C$. On the other hand, if $x \in C$, then let

$$x = \sum_{w \in W} \lambda(w) \frac{w}{\mu(w)}, \quad \text{where } \sum_{w \in W} \lambda(w) = 1.$$

Now, let $s(w) = \lambda(w)/\mu(w)$, and we see that $g(x) \leq 1$.

The above arguments assumed that $\mu(w) > 0$. If $\mu(w) = 0$, then C includes the ray λw for $\lambda \geq 0$ and so does the set of x with $g(x) \leq 1$. The above arguments are, thus, easily extended to this case.

Thus, g is convex. Before showing the other three conditions on (π, μ), an observation is needed. By definition of subadditive valid inequality, (3) holds so for $v \in U$,

$$\pi(v) \leq \min_{u \in U} \{\pi(u) + g(x) : v = u + F(x)\}.$$

Hence, our π on I^m is a true extension in that $\pi(u)$ remains unchanged for $u \in U$. Similarly, (4) assures that $\mu(w)$ remains unchanged for $w \in W$.

To show $\pi(u_0) \geq 1$ is easy using Theorem 5.1 since (11) there implies $\pi(u_0) \geq 1$. Also, to show

$$\mu(w) \geq \lim_{h \to 0} \frac{\pi(F(h\, w))}{h}, \qquad w \in S^m,$$

is easy since $\pi(F(h\, w)) \leq g(h\, w) = h\, \mu(w)$, or $\pi(F(h\, w))/h \leq \mu(w)$.

Subadditivity remains to be shown; that is, $\pi(u) + \pi(v) \geq \pi(u + v)$ for all $u, v \in I^m$. We know that subadditivity holds whenever u and v are in U, by assumption. For v not in U, $\pi(v) = \pi(u) + g(x)$ for some $u \in U$, where $v = u + F(x)$. The fact that the minimum used in defining $\pi(v)$ is actually achieved for some u and x is the same as used earlier in proving Theorem 6.4. Let $\pi(v_1) = \pi(u_1) + g(x_1)$, and $\pi(v_2) = \pi(u_2) + g(x_2)$ for $u_1, u_2 \in U$. Then

$$\begin{aligned} \pi(v_1) + (v_2) &= \pi(u_1) + \pi(u_2) + g(x_1) + g(x_2) \\ &\geq \pi(u_1 + u_2) + g(x_1 + x_2) \end{aligned}$$

by subadditivity of π on U and convexity of g. Since

$$\begin{aligned} v_1 + v_2 &= u_1 + u_2 + F(x_1) + F(x_2) \\ &= u_1 + u_2 + F(x_1 + x_2), \end{aligned}$$

it follows that

$$\pi(v_1 + v_2) \leq \pi(u_1 + u_2) + g(x_1 + x_2).$$

Here, the fact that U is a group implies $u_1 + u_2 \in U$, and we also use the definition of $\pi(v_1 + v_2)$ as a minimum over such terms. Thus, subadditivity is proven.

The next section gives the extreme valid inequalities for particular finite U and W and shows the fill-in described here.

8. Computing some extreme valid inequalities

For certain problems $P(U, W, u_0)$, the extreme valid inequalities have been computed. All were for the two-dimensional case $m = 2$. The simplest was $U = \{(0,0)\}$ and $W = \{1,0), (0,1), (-1,0), (0,-1)\}$. The problem $P(U, W, u_0)$ is then

$$F\left(\begin{pmatrix}1\\0\end{pmatrix} s(1,0) + \begin{pmatrix}0\\1\end{pmatrix} s(0,1) + \begin{pmatrix}-1\\0\end{pmatrix} s(-1,0) + \right.$$
$$\left. + \begin{pmatrix}0\\-1\end{pmatrix} s(0,-1)\right) = \begin{pmatrix}u_{01}\\u_{02}\end{pmatrix},$$

where the variables $s(1,0)$, $s(0,1)$, $s(-1,0)$, $s(0,-1)$ must be non-negative real numbers. A valid inequality is given by numbers $\mu(1,0)$, $\mu(0,1)$, $\mu(-1,0)$, $\mu(0,-1)$ such that

$$\mu(1,0)\, s(1,0) + \mu(0,1)\, s(0,1) + \mu(-1,0)\, s(-1,0)$$
$$+ \mu(0,-1)\, s(0,-1) \geq 1$$

for any solutions.

Using Theorem 6.4, only conditions (19) apply, and the extreme valid inequalities are given by the extreme solutions to

$$u_{01}\, \mu(1,0) + u_{02}\, \mu(0,1) \geq 1,$$
$$(1 - u_{01})\, \mu(-1,0) + u_{02}\, \mu(0,1) \geq 1,$$
$$(1 - u_{01})\, \mu(-1,0) + (1 - u_{02})\, \mu(0,-1) \geq 1,$$
$$u_{01}\, \mu(1,0) + (1 - u_{02})\, \mu(0,-1) \geq 1.$$

In addition, (21) must hold. Here (21) amounts to requiring that of these four inequalities the following hold with equality: one or two and two

or three and three or four and one or four. For example, if one and three hold with equality, then (21) is satisfied. Essentially, (21) means that for any $x \in W$ there is some solution $s(W)$ such that

$$u_0 = F\left(\sum_{w \in W} w\, s(w)\right)$$

with $1 = \sum \mu(w)\, s(w)$ such that $s(x) > 0$.

There are two extreme valid inequalities for this problem. They can be obtained by taking the Gomory mixed integer cut [3, p. 528] for each row of the problem. They were obtained by hand calculation using the double description method [10] and treating (u_{01}, u_{02}) symbolically. The values of μ are given in Table 1 in terms of $u_1 = u_{01}$ and $u_2 = u_{02}$. By symbolic calculation in u_1 and u_2 is meant that the symbols u_1 and u_2 are carried along so that the computation remains valid for all values within a given range.

Table 1

Face	$\mu(1, 0)$	$\mu(0, 1)$	$\mu(-1, 0)$	$\mu(0, -1)$
1	$\frac{1}{u_1}$	0	$\frac{1}{1 - u_1}$	0
2	0	$\frac{1}{u_2}$	0	$\frac{1}{1 - u_2}$

The fill-in procedure of Section 7 gives a function for each of the two above extreme valid inequalities. These two functions are

$$\mu_1(x_1, x_2) = \begin{cases} \dfrac{x_1}{u_1} & \text{if } 0 \le x_1 \le u_1, \\ \dfrac{1 - x_1}{1 - u_1} & \text{if } u_1 < x_1 < 1, \end{cases}$$

$$\mu_2(x_1, x_2) = \begin{cases} \dfrac{x_2}{u_2} & \text{if } 0 \le x_2 \le u_2, \\ \dfrac{1 - x_2}{1 - u_2} & \text{if } u_2 < x_2 < 1. \end{cases}$$

These two functions are pictured in Fig. 1.

A more interesting problem is obtained by taking $U = \{(0, 0), (\frac{1}{2}, 0), (0, \frac{1}{2}), (\frac{1}{2}, \frac{1}{2})\}$. This U is congruent to the direct product $C_2 \times C_2$. When W

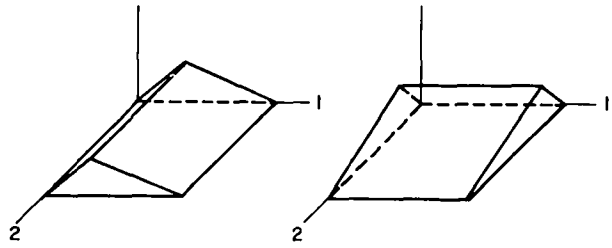

Fig. 1.

is empty and u_0 is in U, the faces are given in [5]. Consider now the mixed problem $P(U, W, u_0)$, where W is as before and for this U. The problem is to find $s(w) \geq 0$ and integer $t(U) \geq 0$ such that

$$\begin{pmatrix} \frac{1}{2} \\ 0 \end{pmatrix} t(\tfrac{1}{2}, 0) + \begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix} t(0, \tfrac{1}{2}) + \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} t(\tfrac{1}{2}, \tfrac{1}{2}) + \begin{pmatrix} 1 \\ 0 \end{pmatrix} s(1, 0) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} s(0, 1) + \begin{pmatrix} -1 \\ 0 \end{pmatrix} s(-1, 0) + \begin{pmatrix} 0 \\ -1 \end{pmatrix} s(0, -1)) = \begin{pmatrix} u_{01} \\ u_{02} \end{pmatrix}.$$

The extreme valid inequalities for this problem were obtained using SCRATCHPAD [9] to execute the double description method treating u_{01} and u_{02} symbolically. These calculations are similar to those described in the appendix of [6] where we actually carried out the same kind of symbolic calculation for the one dimensional case $m = 1$. In this case, the u_{01} and u_{02} were restricted to $0 \leq u_{01} \leq \frac{1}{2}$ and $0 \leq u_{02} \leq u_{01}$ since any other problem can be changed to this form using the following observation. We get the same problem (with a different u_0 and a rearrangement of the variables) by performing any or all of the two following types of operations:

(i) reversing either or both axis;

(ii) swapping axes.

The extreme valid inequalities are given in Table 2. They are scaled to $\pi(u_0) = 1$ and are given in terms of $u_1 = u_{01}$ and $u_2 = u_{02}$.

The filled-in functions obtained from the first two extreme valid inequalities are the same as in Fig. 1. The next two are pictured in Fig. 2. They can be obtained by multiplying each row of the problem through by two and then generating Gomory's mixed integer cut. The last two extreme valid inequalities are more interesting. The filled-in function for the first of these two is pictured in Fig. 3 for $u_0 = (0.2, 0.1)$. The last one, that is

Table 2

	$\mu(1, 0)$	$\mu(0, 1)$	$\mu(-1, 0)$	$\mu(0, -1)$	$\pi(\frac{1}{2}, 0)$	$\pi(0, \frac{1}{2})$	$\pi(\frac{1}{2}, \frac{1}{2})$
1	$\frac{1}{u_1}$	0	$\frac{1}{1-u_1}$	0	$\frac{1}{2(1-u_1)}$	0	$\frac{1}{2(1-u_1)}$
2	0	$\frac{1}{u_2}$	0	$\frac{1}{1-u_2}$	0	$\frac{1}{2(1-u_2)}$	$\frac{1}{2(1-u_2)}$
3	$\frac{1}{u_1}$	0	$\frac{1}{\frac{1}{2}-u_1}$	0	0	0	0
4	0	$\frac{1}{u_2}$	0	$\frac{1}{\frac{1}{2}-u_2}$	0	0	0
5	$\frac{1-u_1}{u_1(1-u_1+u_2)}$	$\frac{1}{1-u_1+u_2}$	$\frac{1}{1-u_1+u_2}$	$\frac{\frac{1}{2}+u_2}{(\frac{1}{2}-u_2)(1-u_1+u_2)}$	$\frac{1}{2(1-u_1+u_2)}$	$\frac{1}{2(1-u_1+u_2)}$	0
6	$\frac{1}{1+u_1-u_2}$	$\frac{1-u_2}{u_2(1+u_1-u_2)}$	$\frac{\frac{1}{2}+u_1}{(\frac{1}{2}-u_1)(1+u_1-u_2)}$	$\frac{1}{1+u_1-u_2}$	$\frac{1}{2(1+u_1-u_2)}$	$\frac{1}{2(1+u_1-u_2)}$	0

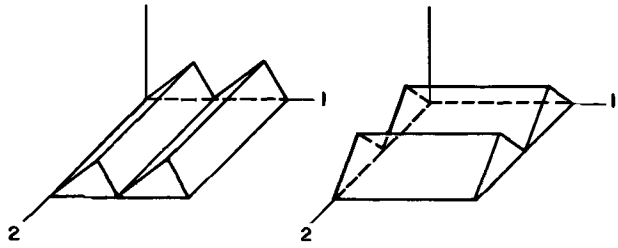

Fig. 2.

extreme valid inequality number six, is obtained from number five by reversing the axes.

We will now describe in some detail how a cut can be derived from the filled-in function from inequality 5. First, the change in variables necessary to adjust the right-hand side u_0 so that $0 \leq u_{02} \leq u_{01} \leq \frac{1}{2}$ will be further described. The basic fact being used is an extension of [5, Theorem 14], and the proof is essentially the same as there, so is deleted. The next two theorems state the results.

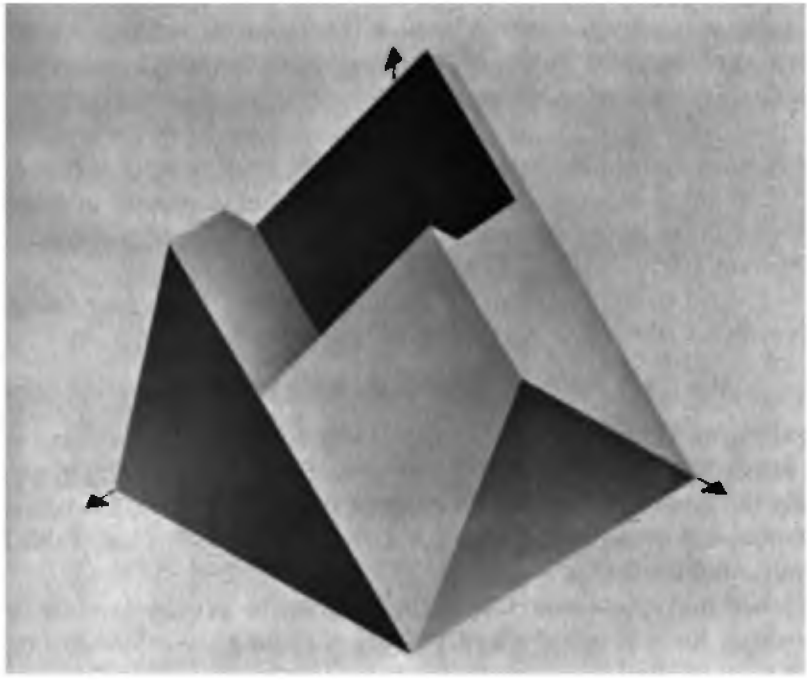

Fig. 3.

Theorem 8.1. *There are two types of automorphisms of the problem* $P(I^m, S^m, u_0)$. *They are:*

(i)
$$\varphi(w_1, \ldots, w_m) = (w_1, \ldots, w_{i-1}, w_j, w_{i+1}, \ldots, w_{j-1}, w_i, w_{j+1}, \ldots),$$
$$\varphi(u_1, \ldots, u_m) = (u_1, \ldots, u_{i-1}, u_j, u_{i+1}, \ldots, u_{j-1}, u_i, u_{j+1}, \ldots);$$

(ii)
$$\varphi(w_1, \ldots, w_m) = (w_1, \ldots, w_{i-1}, -w_i, w_{i+1}, \ldots),$$
$$\varphi(u_1, \ldots, u_m) = (u_1, \ldots, u_{i-1}, 1 - u_i, u_{i+1}, \ldots).$$

The first swaps axes i and j, and the second reverses axis i. These two types of automorphisms can be combined sequentially to produce other automorphisms.

Theorem 8.2. *If* (π, μ) *is a valid inequality for* $P(I^m, S^m, u_0)$ *and if* φ *is an automorphism of* $P(I^m, S^m, u_0)$, *then* (π', μ') *defined by*

$$\mu'(u) = \pi(\varphi^{-1}u), \qquad \mu'(w) = \mu(\varphi^{-1}w),$$

is a valid inequality for $P(I^m, S^m, \varphi(u_0))$. *The theorem remains true if we replace valid inequality by any of subadditive valid inequality, minimal valid inequality, or extreme valid inequality.*

Both theorems remain true if instead of $U = I^m$ we have $U = C \times C \times \ldots \times C$, a direct product of finite cyclic groups or if instead of $W = S^m$ we have a finite W closed under the two types of mappings φ described in Theorem 8.1.

The functions obtained from inequalities 5 and 6 of Table 2 are examples of these two theorems. If we start with inequality 5 and let

$$\varphi(w_1, w_2) = (w_2, w_1), \qquad \varphi(u_1, u_2) = (u_2, u_1),$$

then applying Theorem 8.2 gives inequality 6. In particular, if we start with a u_0 satisfying $0 \leq u_{01} \leq u_{02} \leq \frac{1}{2}$ and apply the above mapping φ, we get exactly the same faces as in table 2 except that 1 becomes 2, 2 becomes 1, 3 becomes 4, 4 becomes 3, 5 becomes 6, and 6 becomes 5. Thus, Table 2 is actually valid for $0 \leq u_{01}, u_{02} \leq \frac{1}{2}$.

We will now show how to use the theorem to get the extreme valid inequalities for $0 \leq u_{01} \leq \frac{1}{2}$ and $\frac{1}{2} \leq u_{02} \leq 1$ using these two theorems. Let $\varphi(w_1, w_2) = (w_1, -w_2)$ and $\varphi(u_1, u_2) = (u_1, 1 - u_2)$. Then $\varphi(u_1, u_2) = (u_1', u_2') = (u_1, 1 - u_2)$ satisfies $0 \leq u_1', u_2', \leq \frac{1}{2}$ whenever $0 \leq u_1 \leq \frac{1}{2}$ and

Table 3

	$\mu(1,0)$	$\mu(0,1)$	$\mu(-1,0)$	$\mu(0,-1)$	$\pi(0,\frac{1}{2})$	$\pi(0,\frac{1}{2})$	$\pi(\frac{1}{2},\frac{1}{2})$
1	$\frac{1}{u_1}$	0	$\frac{1}{1-u_1}$	0	$\frac{1}{2(1-u_1)}$	0	$\frac{1}{2(1-u_1)}$
2	0	$\frac{1}{u_2}$	0	$\frac{1}{1-u_2}$	0	$\frac{1}{2u_2}$	$\frac{1}{2u_2}$
3	$\frac{1}{u}$	0	$\frac{1}{\frac{1}{2}-u_1}$	0	0	0	0
4	0	$\frac{1}{u_2-\frac{1}{2}}$	0	$\frac{1}{1-u_2}$	0	0	0
5	$\frac{1-u_1}{u_1(2-u_1-u_2)}$	$\frac{\frac{3}{2}-u_2}{(u_2-\frac{1}{2})(2-u_1-u_2)}$	$\frac{1}{2-u_1-u_2}$	$\frac{1}{2-u_1-u_2}$	$\frac{1}{2(2-u_1-u_2)}$	$\frac{1}{2(2-u_1-u_2)}$	0
6	$\frac{1}{u_1+u_2}$	$\frac{1}{u_1+u_2}$	$\frac{\frac{1}{2}+u_1}{(\frac{1}{2}-u_1)(u_1+u_2)}$	$\frac{u_2}{(1-u_2)(u_1+u_2)}$	$\frac{1}{2(u_1+u_2)}$	$\frac{1}{2(u_1+u_2)}$	0

$\frac{1}{2} \leq u_2 \leq 1$. Thus we can use Theorem 8.2 and Table 2. Let $u_0' = (u_{01}', u_{02}')$ with $0 \leq u_{01}' \leq \frac{1}{2}$ and $\frac{1}{2} \leq u_{02}' \leq 1$, and let $u_0 = (u_{01}, u_{02}) = (u_{01}', 1 - u_{02}')$. Table 2 gives us inequalities for u_0 in terms of u_{01}, u_{02}. To change them to inequalities for $P(U, W, u_0')$, we must substitute $u_{02} = 1 - u_{02}'$ into those expressions. Also, $\mu'(0, 1)$ is now given by $\mu(0, -1)$, $\mu(0, -1)$ by $\mu(0, 1)$, while the remainder are unchanged. The result is shown in Table 3.

As an example, six cuts from this table will be derived for a particular two row integer program. Let

$$x_1 + 0.4x_3 + 1.3x_4 - 0.01x_5 + 0.07x_6 = 0.2,$$

$$x_2 - 0.3x_3 + 0.4x_4 - 0.04x_5 + 0.1x_6 = 0.9,$$

where x_1, x_2, x_3, and x_4 are integer variables. The cuts from Table 3 are given from the filled-in functions (π, μ) on $I^m \times S^m$ by

$$\pi(0.4, 0.7)\, x_3 + \pi(0.3, 0.4)\, x_4 + 0.04\mu(-0.25, -1)\, x_5 + 0.1\mu(0.7, 1)\, x_6 \geq 1.$$

Here they are:

$$\frac{0.6}{0.8} x_3 + \frac{0.7}{0.8} x_4 + \frac{0.01}{0.8} x_5 + \frac{0.07}{0.2} x_6 \geq 1,$$

$$\frac{0.7}{0.9} x_3 + \frac{0.4}{0.9} x_4 + \frac{0.04}{0.1} x_5 + \frac{0.1}{0.9} x_6 \geq 1,$$

$$\frac{0.1}{0.3} x_3 + \frac{0.2}{0.3} x_4 + \frac{0.01}{0.3} x_5 + \frac{0.07}{0.2} x_6 \geq 1,$$

$$\frac{0.2}{0.4} x_3 + \frac{0.4}{0.4} x_4 + \frac{0.04}{1} x_5 + \frac{0.1}{0.4} x_6 \geq 1,$$

$$\frac{0.4}{0.9} x_3 + \frac{0.3}{0.9} x_4 + \frac{0.05}{0.9} x_5 + \frac{0.43}{0.9} x_6 \geq 1,$$

$$\frac{0.433}{1.1} x_3 + \frac{0.7}{1.1} x_4 + \frac{0.38}{1.1} x_5 + \frac{0.17}{1.1} x_6 \geq 1$$

or,

$$0.75\, x_3 + 0.875x_4 + 0.0125x_5 + 0.35\, x_6 \geq 1,$$

$$0.778x_3 + 0.444x_4 + 0.40\, x_5 + 0.11\, x_6 \geq 1,$$

$$0.333x_3 + 0.667x_4 + 0.0333x_5 + 0.35\, x_6 \geq 1,$$

$$0.5\ x_3 + \quad x_4 + 0.40\ x_5 + 0.25\ x_6 \geq 1,$$

$$0.444x_3 + 0.333x_4 + 0.0555x_5 + 0.478x_6 \geq 1,$$

$$0.394x_3 + 0.636x_4 + 0.346x_5 + 0.155x_6 \geq 1.$$

Note that, in this case, the last inequality dominates the fourth one because every coefficient of the last one is smaller.

The extreme valid inequalities for one other problem have been computed. This problem has

$$U = \{(0)\},$$

$$W = \{(1,0),(1,1),(0,1),(-1,1),(-1,0),(-1,-1),(0,-1),(1,-1)\},$$

and $u_0 = (0.2, 0.1)$. This problem has not yet been attempted with SCRATCHPAD because of its size. Thus there is only one particular u_0 for which the answer is known.

There are seventeen faces and the values of $\mu(w)$, $w \in W$, are shown in Table 4. The first four faces are pictured, in their filled-in form, in Fig. 4. The contour lines of all seventeen are shown in Fig. 5. The filled-in versions of the last thirteen can be formed, but they become rather complicated.

Table 4

	$\mu(1,0)$	$\mu(1,1)$	$\mu(0,1)$	$\mu(-1,1)$	$\mu(-1,0)$	$\mu(-1,-1)$	$\mu(0,-1)$	$\mu(1,-1)$
1	10	0	1.111	2.222	1.111	0	10	20
2	5	5	0	1.25	1.25	1.25	0	5
3	3.333	6.667	3.333	0	1.429	2.857	1.429	0
4	0	10	10	10	0	1.111	1.111	1.111
5	2.8	7.2	4.4	1.6	1.2	0.8	3.6	6.4
6	4.706	5.294	0.5882	1.765	1.176	0.5882	5.294	10
7	4.848	5.152	0.303	1.515	1.212	0.9091	2.727	4.545
8	5.294	4.706	5.294	5.882	0.5882	1.176	0.5882	5.882
9	3.043	6.957	3.913	0.8696	1.304	1.739	0.4348	3.478
10	2.902	7.098	4.197	1.295	1.244	1.192	0.4663	3.368
11	3.478	6.522	3.043	3.913	0.8696	2.174	1.304	0.4348
12	2.881	7.119	4.239	1.358	1.235	1.111	1.111	1.111
13	3.704	6.296	2.593	1.358	1.235	1.111	1.111	1.111
14	3.704	6.296	6.708	7.119	0.4115	1.111	1.111	1.111
15	3.6	6.4	6.8	7.2	0.4	1.6	1.2	0.8
16	3.704	6.296	2.593	3.519	0.9259	1.111	1.111	1.111
17	4.186	5.814	6.279	6.744	0.4651	1.163	0.6977	2.558

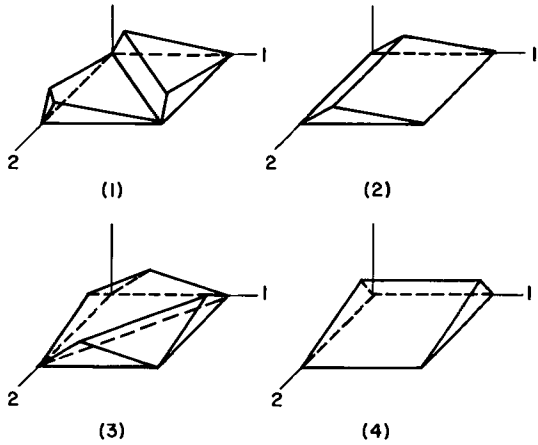

Fig. 4.

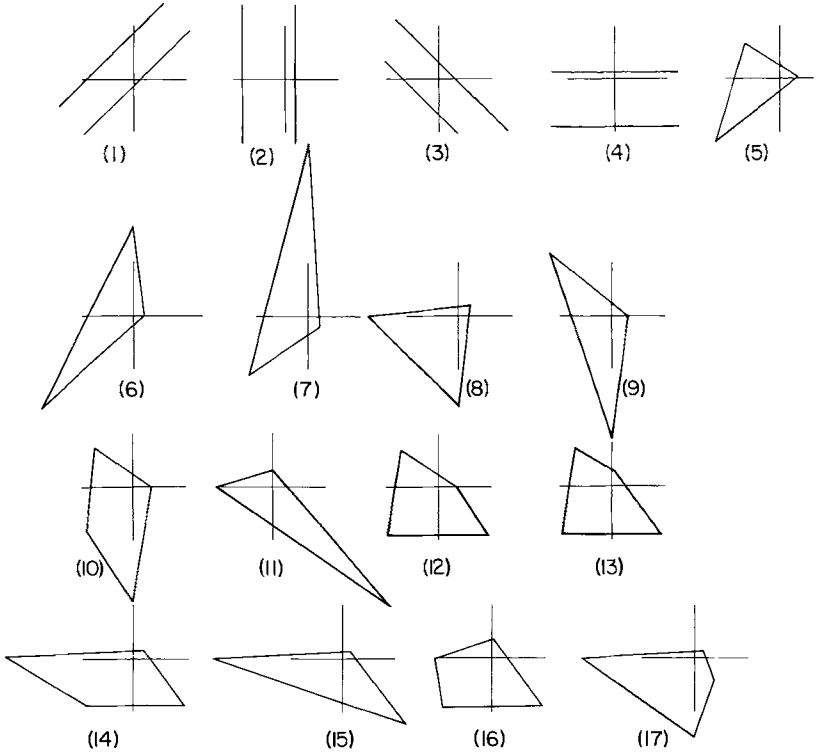

Fig. 5.

References

[1] E. Balas, "Intersection cuts--A new type of cutting planes for integer programming", *Operations Research* 19 (1971) 19–39.
[2] C. Burdet, "On the algebra and geometry of integer programming cuts", Management Sciences Report No. 291, Carnegie-Mellon University, Pittsburgh, Pa. (1972).
[3] G.B. Dantzig, *Linear programming and extensions* (Princeton University Press, Princeton, N.J., 1962).
[4] F. Glover, "Cut search methods in integer programming", *Mathematical Programming* 3 (1972) 86–100.
[5] R.E. Gomory, "Some polyhedra related to combinatorial problems", *Linear Algebra and Its Applications* 2 (1969) 451–558.
[6] R.E. Gomory and E.L. Johnson, "Some continuous functions related to corner polyhedra I", *Mathematical Programming* 3 (1972) 23–85.
[7] R.E. Gomory and E.L. Johnson, "Some continuous functions related to corner polyhedra II", *Mathematical Programming*, to appear.
[8] G.A. Gorry, W.D. Northup and J.F. Shapiro, "Computational experience with a group theoretic integer programming algorithm", *Mathematical Programming* 4 (1973) 171–192.
[9] J.H. Griesmer and R.D. Jenks, "The SCRATCHPAD System", Proceedings of *ON-LINE* 72. Brunel University, Oxbridge, Middlesex, England, September 4–7, 1972. (Also available as IBM Research Report RC 3925).
[10] T.S. Motzkin, H. Raiffa, G.L. Thompson and R.M. Thrall, "The double description method", in: H.W. Kuhn and A.W. Tucker, eds., *Contributions to the theory of games*, Vol. II, Annals of Mathematics Study No. 28 (Princeton University Press, Princeton, N.J., 1953) 51–73.
[11] R.T. Rockafellar, *Convex analysis* (Princeton University Press, Princeton, N.J., 1970).

Mathematical Programming Study 2 (1974) 180–197. North-Holland Publishing Company

EXPERIMENTS IN THE FORMULATION OF INTEGER PROGRAMMING PROBLEMS

H.P. WILLIAMS

University of Sussex, Brighton, England

Received 24 April 1973
Revised manuscript received 16 May 1974

Five practical problems are each formulated in two different ways as 0-1 integer programming models. All the models have been solved by the Branch and Bound method using a commercial package program. Full details are given of the manner of the different formulations and the computational ease of solving them. The purpose of this paper is to investigate the computational effects of different *formulations* on such problems. The problems considered are a market allocation problem, a combinatorial problem, two mining problems and a problem of logical design.

1. Introduction

Integer programming (IP) is computationally far more difficult than linear programming (LP). It has, however, become possible in the last few years to solve many practical integer programming problems using commercial packages. Some such developments are described by Benichou et al. [4]. Most commercial packages use a form of the Branch and Bound Algorithm as described by, for example, Land and Doig [8], Dakin [5] and Beale and Small [2]. Algorithms of this kind start by solving an integer programming problem as if it were a continuous LP problem. No variables are constrained to take integer values and the *continuous optimum* is obtained. A tree search is then carried out, systematically imposing bounds on integer variables which have not attained integer values. When all the integer variables have attained integer values, a *feasible integer solution* has been reached. If the tree search is continued to completion, the best such feasible integer solution is the *integer optimum*. The full details of the algorithms are described in the references above.

Five different optimization problems are described here. Each problem is formulated as an integer programming problem in two different ways. In most cases one formulation takes far less time to solve than the other

formulation. The superior formulation is always the one which is "tighter" in a continuous sense. Geometrically, the boundary of the feasible region has been moved to reduce its size but not to exclude any feasible integer points. The result is a problem which has less continuous solutions but the same number of integer solutions. It may therefore be expected that the continuous optimum be closer to the integer optimum in the "tighter" formulation.

There is little practical connection between the five problems. The choice of problems has been deliberately made in order that this shall be so. There is merit in comparing computational experience between quite different practical problems using the same algorithm. It is apparent that the computational ease of solving an integer programming problem is very dependent upon the problem's structure. An artificially constructed model very often has little structure. The choice of practical, rather than artificial, problems has therefore also been deliberate. All five problems are fairly small although far from trivial. They can all, however, be extended easily to similar, but much larger, problems.

Numerical data for Problems 1 and 3 can be obtained from the author on request.

2. Setting up the experiments

In all these problems, the ICL mathematical programming system XDLA [10] on a 1904A computer was used. All problems were run in a partition of 50 K words. In order to obtain as realistic a comparison as possible between formulations, penalty calculations were not used to control the tree search nor was any priority order used to control the choice of branching variable. In all cases, branching occurred on the integer variable with the *most significant fractional value* first in the direction of the nearest integer. Shorter runs could almost certainly have been obtained by a more sophisticated approach to the tree search. Using a knowledge of the structure of each problem by specifying a priority order for branching would almost certainly be valuable. Experience of doing this in combination with reformulation ideas similar to some of the ideas used here has been described by Beale and Tomlin [3]. Explicitly putting slack variables in certain rows and specifying them as integer variables as suggested by Mitra [12] would almost certainly have proved of value if such variables had been given priority in branching. As stated before, however, the purpose of the experiments was to compare *formulations* rather than algorithms.

Any approach that used an understanding of the structure of the model was avoided.

The solution times given for each formulation are Mill times for the total job. It is not, unfortunately, possible with the XDLA system to obtain Mill times for stages in the computation. Elapsed times have little meaning in a multi-programming environment. The Mill time for the computation up to the continuous optimum of each problem would be of interest. In most integer programming problems this time is very small in comparison with the time taken for the second phase involving the tree search.

3. Discussion of problems

Problem 1. *Allocating Market Share*

This problem involved a distribution company with two divisions A and B. The company was responsible for supplying its products to 23 retailers $M1$ to $M23$. It was desired that division A control 40% of the company's operations and division B the remaining 60%. The problem was therefore to allocate 40% of the retailers to A and 60% to B. Each retailer had a known market for each product. The allocation of retailers to divisions had, therefore, to be made in such a way that the 40/60 split applied over the total market for each product. Some retailers were considered to have better growth prospects than others. It was therefore desired to divide those with good prospects in the 40/60 ratio as well as those with lesser prospects. Each retailer also had a given number of delivery points. The division of retailers between A and B had to be such that the split in delivery points serviced by A and B was 40/60. To obtain a 40/60 split exactly over all these categories was almost certainly impossible. The condition was therefore relaxed that the split had to lie within the limits 35/65 to 45/55. It was, however, desirable to keep as near the 40/60 split as possible. The *objective* was to minimise the total of all the deviations as long as they lay within the specified $\pm$ 5%.

Formulation 1

The problem was formulated as an integer programming model where each of the 23 retailers was represented by an 0–1 variable X_i. If $X_i = 1$, retailer i was assigned to division A. Otherwise he was assigned to division B. Slack variables were introduced into each restraint to provide the required objective. For example Product 3 could only be supplied to retailers $M19$ to $M23$. The markets (in suitable units) of these retailers for

Product 3 were as below:

$M19$	$M20$	$M21$	$M22$	$M23$
6	15	15	25	39

It was therefore desired to split this total market of 100 in the appropriate ratio between A and B. This can be accomplished by the restraint

$$6X_{19} + 15X_{20} + 15X_{21} + 25X_{22} + 39X_{23} + y_1 - y_2 = 40. \quad (1)$$

y_1 and y_2 are slack variables which are introduced and each given a *cost* of 1 in the objective. They also have *upper bounds* of 5 to keep the split within the appropriate limits.

Following the suggestion of Mitra [12], y_1 and y_2 were also made integer variables. Other restraints were treated in a similar fashion. All the right-hand side coefficients were kept integer by, if necessary, taking the nearest integer to the appropriate value and suitably adjusting the bounds on the slack variables (such as y_1 and y_2). In this way it was possible to specify all variables in the problem as integer. The resultant problem contained 9 restraints and 40 integer variables (23 were 0–1).

This problem was "difficult" in that one integer solution need bear very little resemblance to another. The problem was constructed as a small version of a much larger problem which had given great difficulty on a much more powerful computer and produced no integer solution.

Formulation 2

This was conceptually almost the same as formulation 1 except that 3 of the restraints were replaced by "simpler" but "logically equivalent" restraints. Restraint (1) can be regarded as the two restraints

$$6X_{19} + 15X_{20} + 15X_{21} + 25X_{22} + 39X_{23} \leq 45, \quad (2)$$

$$6X_{19} + 15X_{20} + 15X_{21} + 25X_{22} + 39X_{23} \geq 35. \quad (3)$$

Since all the variables in these restraints are 0–1, they can be shown to be *logically equivalent* to the restraints

$$X_{19} + 2X_{20} + 2X_{21} + 3X_{22} + 4X_{23} \leq 5, \quad (4)$$

$$X_{19} + 2X_{20} + 2X_{21} + 3X_{22} + 5X_{23} \geq 5. \quad (5)$$

It is easy to verify that combinations of values 0 or 1 for the X_i satisfy (2) if and only if they satisfy (4). Similarly for (3) and (5). (4) and (5) are, however, "tighter" than (2) and (3).

The possibility of "simplifying" restraints in this way and a systematic means of doing it was suggested by Hammer [7]. It is applicable to any pure 0–1 restraint even if some coefficients are negative (in this case, X_i is replaced by $1 - X_i$). In this particular example, 3 restraints such as (1) were simpliefied in this manner yielding 6 new restraints. The original 3 restraints (now redundant) were left in the model to simplify calculation of the objective.

Clearly a certain amount of calculation is involved in simplifying the restraints. This was done by a combination of hand and computer calculation (it is itself an LP problem). The times involved in the computation were negligible.

The results of the two formulations are given below.

Results of formulation 1

Continuous optimum in 25 iterations. Objective = 0.
First integer solution in 89 nodes. Objective = 51.
No better integer solutions were sought.
Total Mill time 147 sec.

Results of formulation 2

Continuous optimum in 42 iterations. Objective = 0.
First integer solution in 49 nodes. Objective = 28.
No better integer solutions were sought.
Total Mill time 76 sec.

Clearly the second formulation not only obtains an integer solution more quickly, but also obtains a better first integer solution. Neither formulation was run until the tree search was completed.

Problem 2. *Three-dimensional noughts and crosses*

This is a combinatorial problem. It was thought of by the author and he is not aware of it having been considered before.

Given a $3 \times 3 \times 3$ cube consisting of 27 cells, the problem is to fill each cell with either a black ball or a white ball so as to minimise the number of "straight lines" in the cube containing balls of identical colour. A straight line is a row of 3 cells including all possible diagonals. (There are 49 such lines in all). There is no restriction on the ratio of black to white balls.

27 0–1 variables X_i are used to indicate whether a cell should be filled

by a black ball ($X_i = 1$) or a white ball ($X_i = 0$). This problem is clearly characterised by the existence of a large number of feasible integer solutions (2^{27}). Obviously the symmetry of the problem could be exploited to greatly aid solution. This was not, however, done in either formulation as it was desired to gain an understanding of how to tackle such problems with a large number of feasible solutions even if they had no symmetry.

A knowledge of "good" solutions could have been used to restrict the tree search. This was not done as it was hoped to obtain an understanding of how to "formulate in the dark".

Formulation 1

The cells of the cube were numbered 1 to 27. For each of the 49 lines in the cube (for example cells 1, 2, 3) the following type of restraint was given:

$$X_1 + X_2 + X_3 + y \geq \tfrac{3}{2}. \tag{6}$$

A different continuous variable y was introduced for each line. In the objective row (to be minimised) the X variables are given coefficients equal to the number of lines (restraints) in which they occur. The y variables are given coefficients of 2. For any line (such as cells 1, 2, 3) the following possibilities occur:

All of $X_1, X_2, X_3 = 0$, $y = \frac{3}{2}$, contribution to objective = 3.
One of $X_1, X_2, X_3 = 1$, $y = \frac{1}{2}$, contribution to objective = 2.
Two of $X_1, X_2, X_3 = 1$, $y = 0$, contribution to objective = 2.
All of $X_1, X_2, X_3 = 1$, $y = 0$, contribution to objective = 3.

By minimising the objective function as few lines as possible will have all cells with balls of the same colour.

This formulation has 49 restraints and 76 variables (27 0–1 and 49 continuous).

Although y is a continuous variable, it will only take values $\frac{1}{2}$ or $\frac{3}{2}$. The problem could therefore be easily converted into a pure integer problem.

Formulation 2

The cells of the cube were numbered as before. For each of the 49 lines in the cube (for example cells 1, 2, 3) 2 restraints of the following type were given:

$$X_1 + X_2 + X_3 - \delta \leq 2, \tag{7}$$

$$X_1 + X_2 + X_3 + \delta \geq 1. \tag{8}$$

The δ variables (one for each of the 49 lines) are integer (0–1) variables. In the objective row (to be minimised) only the δ variables have a coefficient. This coefficient is 1. If for a particular line (such as cells 1, 2, 3) $\delta = 0$, then not all of the balls in that line can be of the same colour. By minimising the objective function, as few lines as possible will have all cells with balls of the same colour.

This formulation has 98 restraints and 76 variables (all 0–1).

Although the variables δ are 0–1 they would automatically take integer values in the optimal solution if they were specified to be continuous.

In order to compare the two formulations it is convenient to regard each restraint (6) in formulation 1 as contributing a constant amount 2 to the objective as well as a variable amount 0 or 1 depending on the values of the X_i's. If the y variable in formulation 1 is restricted to its discrete values 0, $\frac{1}{2}$, $\frac{3}{2}$ but the X_i's are allowed to be continuous, it is always possible to obtain at most as small a contribution to the objective from restraint (6) as from the restraints (7) and (8). For example, if $y = 0$ in restraint (6), the minimum contribution to the objective is $\frac{3}{2}$ $(= 2 - \frac{1}{2})$ if X_1, X_2, X_3 are allowed to be continuous. The case corresponding to this situation with restraints (7) and (8) is $\delta = 0$. Here the contribution to the objective is 0 which is greater than the effective contribution of $-\frac{1}{2}$ with restraint (6). Therefore in this sense the second formulation is stronger. Solutions produced from the second formulation in the course of optimization will tend to be closer to the integer solutions than those produced from the first formulation.

Result of formulation 1

Continuous optimum in 52 iterations. Objective = 75 · 5.

First integer solution in 22 nodes. Objective = 104 (6 lines of same colour).

Second integer solution in 26 nodes. Objective = 104 (6 lines of same colour).

Third integer solution in 34 nodes. Objective = 103 (5 lines of same colour).

No better integer solution was found after 102 nodes.

Total Mill time 250 sec.

Note. The objective function in this formulation is effectively a constant of 98 added to a variable amount which is comparable to the value of the objective function in the second formulation.

Results of formulation 2

Continuous optimum in 136 iterations. Objective = 0.

First integer solution in 18 nodes. Objective = 4 (4 lines of same colour).

This is the known optimal solution although the run was halted at this point and optimality was not proved.

Total Mill Time 128 Secs.

The second formulation is clearly better than the first.

Problem 3. *Mining for profit*

This is an open-cast mining problem which has been approached through both separable programming and graph theory. These approaches are described by Meyer [11] and Lerchs and Grossman [9].

The problem is to excavate within a permissable area in order to obtain valuable ore. Ore body and overburden are divided into blocks (usually 50 foot cubes). Each block has a certain net income obtained by subtracting the cost of extracting it from the revenue obtained after refining and selling the ore. For blocks near the surface (overburden) the net income is often negative but usually positive for richer ore deeper down.

There is an "angle of slip" at the edges of the excavation so that a block can only be excavated if the upturned cone of blocks above it are also removed. The blocks are as shown in Fig. 1 when seen in a vertical plane. It is therefore not possible to excavate block 1 unless both blocks 2 and 3 have already been removed.

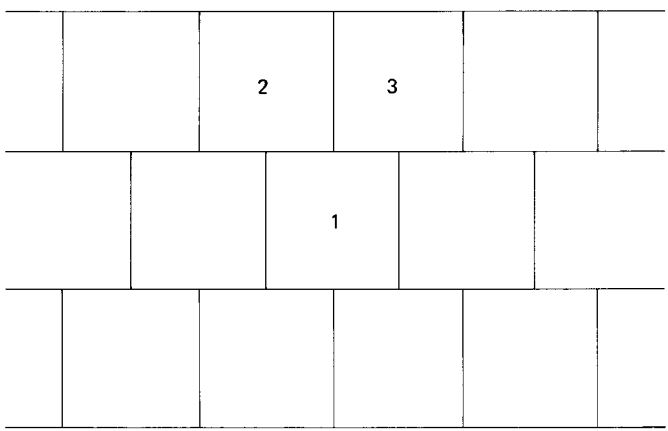

Fig. 1.

In this particular example, the area within which excavation could take place was a square 200 ft × 200 ft. Therefore the most that could possibly be excavated was an upturned pyramid of blocks. On the surface there were 16 blocks. At successively lower levels there were 9, 4 and 1 blocks. Each of the 30 blocks was given an estimated net income R_i (in some cases negative). An 0–1 variable X_i was assigned to each block. $X_i = 1$ indicated that block i should be excavated. Otherwise $X_i = 0$. The objective was to maximise

$$\sum_{i=1}^{30} R_i X_i. \tag{9}$$

Formulation 1

Each restraint has to specify the condition that a block can only be excavated if the 4 blocks directly above it are also excavated. (In three dimensions, Fig. 1 would show 4 blocks directly above block 1). Suppose block 1 has the four blocks numbered 2, 3, 4, 5 above it. Then the restraint can be written as

$$X_2 + X_3 + X_4 + X_5 - 4X_1 \geq 0. \tag{10}$$

Similar restraints are written for all the other blocks (apart from those on the surface).

This formulation gives a model with 14 restraints and 30 variables (all 0–1).

Formulation 2

Instead of a series of restraints of the type (10), each such restraint is replaced by the following 4 restraints.

$$X_2 - X_1 \geq 0, \tag{11}$$

$$X_3 - X_1 \geq 0, \tag{12}$$

$$X_4 - X_1 \geq 0, \tag{13}$$

$$X_5 - X_1 \geq 0. \tag{14}$$

This formulation gives a model with 56 restraints and 30 variables (all 0–1).

The second formulation has the property that each restraint now contains exactly one coefficient $+1$ and exactly one coefficient -1. The dual problem will therefore have exactly one $+1$ and exactly one -1 in

each column. This is a known sufficient condition for a problem to be expressible as a transportation problem. The matrix of both this problem and its dual are therefore unimodular. Hence the continuous optimum to formulation 2 will be also the integer optimum.

Results of formulation 1

Continuous optimum in 6 iterations. Objective = 21625.
First integer solution in 2 nodes. Objective = −2000.
Integer optimum in 7 nodes. Objective = 17500.
Optimality proved in 11 nodes.
Total Mill time 23 sec.

Results of formulation 2

Continuous optimum in 36 iterations. Objective = 17500 (= integer optimum).
Total Mill time 14 sec.

Obviously the second formulation is far superior to the first because of the unimodularity property. In an example of this size both formulations were easy to solve. For larger examples the avoidance of integer programming in the second formulation would be even more desirable.

It is interesting to note the exact equivalence of restraint (10) with restraints (11), (12), (13), and (14) in an integer sense but the greater "tightness" of the second set in a continuous sense.

Problem 4. *Mining for investment return*

This problem is the same as Problem 3, except that the objective is now to maximise return on investment rather than net income. Investment is taken to be the cost of extraction. The return on investment is the ratio

$$\frac{\text{total net income}}{\text{total investment}}.$$

If the cost of extracting block i is I_i, this objective is

$$\frac{\sum_{i=1}^{30} R_i X_i}{\sum_{i=1}^{30} I_i X_i}. \tag{15}$$

This hyperbolic objective function is converted into a linear function by the following steps:

(i) Introduce a continuous variable y into the problem to represent (15). The objective is then to maximise y.

(ii) Equating y to (15) and multiplying out gives the (non-linear) restraint

$$\sum_{i=1}^{30} I_i X_i y - \sum_{i=1}^{30} R_i X_i = 0. \tag{16}$$

(iii) Replace the expressions $X_i y$ (for each i) by Z_i. Z_i are now continuous variables.

(iv) It is now necessary to impose restraints that force Z_i to truly represent the value of $X_i y$. In this particular example it can be reasoned from the numerical data given that y will never exceed 10. The necessary restrictions are then imposed by the restraints:

$$Z_i - 10X_i \leq 0, \tag{17}$$

$$Z_i - y \leq 0, \tag{18}$$

$$-Z_i + y + 10X_i \leq 10 \tag{19}$$

for $i = 1, \ldots, 30$. It is also necessary to impose on the model a restraint which avoids getting an unbounded solution where the objective is 0/0 obtained by excavating nothing. This can be done by imposing the restraint

$$\sum_{i=1}^{30} X_i \geq 1. \tag{20}$$

Formulation 1

This contains all the new variables and new restraints described above together with the restraints of type (10) in Problem 3.

This formulation gives a model of 106 restraints and 61 variables (30 0–1, 31 continuous).

Formulation 2

This is the same as formulation 1, except that each of the restraints of type (10) is replaced by the 4 restraints of types (11), (12), (13) and (14) in Problem 3.

This formulation gives a model of 150 restraints and 61 variables (30 0–1, 31 continuous).

Results of formulation 1

Continuous optimum in 38 iterations. Objective = 9 . 668.
First integer solution in 24 nodes. Objective = 0.
Integer optimum (unproved) in 81 nodes. Objective = 0 . 333.
The search was abandoned after 101 nodes.
Total Mill time 215 sec.

Results of formulation 2

Continuous optimum in 86 iterations. Objective = 9 . 668.
First integer solution in 8 nodes. Objective = 0.
Integer optimum in 29 nodes. Objective = 0 . 333.
Optimality proved in 33 nodes.
Total Mill time 90 sec.

With this new objective, the second formulation is still "tighter" in a continuous sense than the first formulation although the continuous optima of both formulations are the same.

The property of the second formulation in Problem 3, that the continuous problem is so tight as to yield an integer continuous optimum no longer holds.

Problem 5. *Logical design*

This problem is concerned with building a logical system to respond in some prescribed logical fashion to possible inputs. Such a system has one output and a number of inputs. Signals which can occur at the inputs and outputs are all 2-valued (0 and 1). Given a logical function which the system is to perform (this is prescribed by a truth table), the problem is to construct a system out of NOR gates to perform the function.

The NOR gates act as "building blocks". A NOR gate has 2 inputs (A and B) and 1 output and performs in the manner indicated by the truth table (Table 1).

Table 1

Inputs		Output
A	*B*	
0	0	1
0	1	0
1	0	0
1	1	0

Table 2

Inputs		Output
A	*B*	
0	0	0
0	1	1
1	0	1
1	1	0

A "circuit" is connected up by outputs from some NOR gates being inputs to others. An optimization problem arises in that it is desirable to construct a circuit to perform a prescribed logical function using the minimum number of NOR gates. This problem is described by Williams [14] and a method of obtaining a good (though non-optimal) solution described. The possibility of using integer programming for this type of problem (together with many other problems of logical design) is discussed by Muroga [13].

Although the author doubts the feasibility of integer programming as a practical means of tackling this type of problem, he has experimented with two different formulations of a fairly simple example.

The problem considered here is to construct a circuit of NOR gates with 2 inputs to perform the logical function in Table 2.

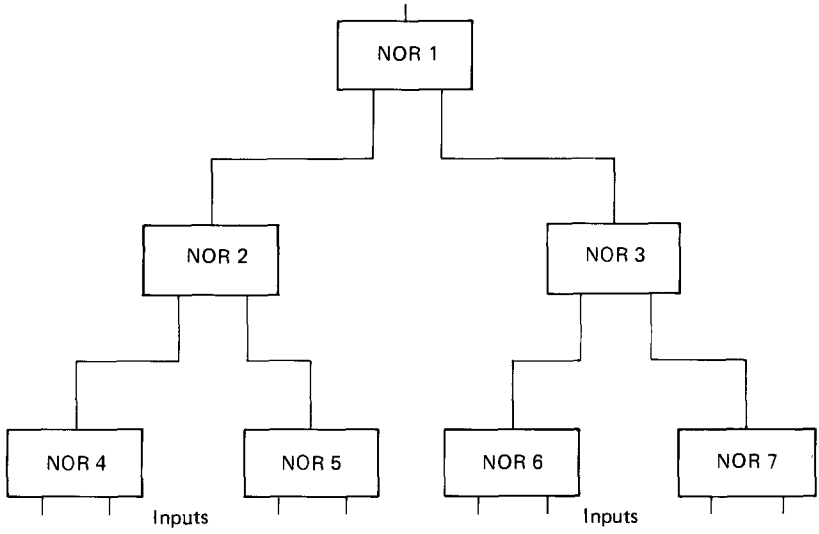

Fig. 2.

An external input signal (A or B) can be connected into inputs of more than one NOR gate.

"Fan-in" and "Fan-out", however, are not permitted. That is more than one output from a NOR gate cannot lead into one input. (Nor can more than one external input signal.) Nor can one output lead into more than one input. (Although an external input signal can.)

In order to simplify the formulation, the optimal circuit can be assumed to be a "subnet" of the maximum net shown in Fig. 2.

In both formulations, the following variables are introduced

(i) $S_i = 1$ if NOR gate i exists, $i = 1, 7$,
$= 0$ otherwise.

(ii) $t_{i1} = 1$ if external input A is an input to gate i,
$= 0$ otherwise.

$t_{i2} = 1$ if external input B is on input to gate i,
$= 0$ otherwise.

(iii) X_{ij} = output from gate i for the combination of external input signals specified in the j^{th} row of Table 2.

The objective function to be minimised is $\sum_{i=1}^{7} S_i$.

Formulation 1

The following restraints are imposed:

A NOR gate can only have an external input if it exists. These conditions are imposed by the restraints

$$-2S_i + t_{i1} + t_{i2} \leq 0, \qquad i = 1, \ldots, 7 \tag{21}$$

If a NOR gate has one (or two) external inputs leading into it, only one (or no) NOR gates can feed into it. These conditions are imposed by the restraints

$$S_j + S_k + t_{i1} + t_{i2} \leq 2, \qquad i = 1, 2, 3, \tag{22}$$

where j and k are the two NOR gates leading into i in Fig. 2.

The output signal from NOR gate i must be the correct logical function (NOR) of the input signals into gate i if gate i exists.

Let α_{1j} (a constant) be the value of the external input signal A in the j^{th} row of the Table 2. Similarly, α_{2j} corresponds to the external input signal B.

These restrictions can be written as

$$\alpha_{1l}t_{i1} + \alpha_{2l}t_{i2} + X_{jl} + X_{kl} + 2X_{il} \leq 2, \tag{23}$$

$$-S_i + \alpha_{1l}t_{i1} + \alpha_{2l}t_{i2} + X_{jl} + X_{kl} + X_{il} \geq 0, \tag{24}$$

$$i = 1, \ldots, 7, \qquad l = 1, 2, 3, 4,$$

where j and k are the two NOR gates leading into gate i in Fig. 2. The values X_{1l} ($l = 1, \ldots, 4$) are the outputs from the last NOR gate which are prescribed and can therefore be set to the constant values 0, 1, 1, 0.

If there is an output signal of 1 from a particular gate for any combination of the input signals, then that gate must exist. These restrictions can be imposed by the restraints

$$4S_i - X_{i1} - X_{i2} - X_{i3} - X_{i4} \geq 0, \qquad i = 1, \ldots, 7. \tag{25}$$

As before, X_{1l} are set at constant values.

This formulation has 73 restraints and 45 variables (all 0–1).

Formulation 2

In the formulation the restraints (21) are replaced by the two restraints

$$-S_i + t_{i1} \leq 0, \tag{26}$$

$$-S_i + t_{i2} \leq 0 \tag{27}$$

for $i = 1, \ldots, 7$. Restraints (23) are replaced by the series of restraints:

$$X_{jl} + X_{il} \leq 1, \tag{28}$$

$$X_{kl} + X_{il} \leq 1, \tag{29}$$

$$\alpha_{1l}t_{i1} + X_{il} \leq 1, \tag{30}$$

$$\alpha_{2l}t_{i2} + X_{il} \leq 1, \tag{31}$$

$$i = 1, \ldots, 7, \qquad l = 1, 2, 3, 4,$$

where j and k are the NOR gates leading into gate i in Fig. 2.

Since the α_{ij} are constants, some of the restraints (30) and (31) will be redundant for particular values of l and may be ignored.

Restraints of type (25) in the first formulation are replaced by the restraints

$$S_i - X_{il} \geq 0, \qquad i = 1, \ldots, 7, \quad l = 1, 2, 3, 4.$$

This formulation has 114 restraints and 45 variables (all 0–1).

Results of formulation 1

Continuous optimum in 14 iterations. Objective = 0.75.
Integer optimum in 15 nodes. Objective = 5.
Optimality proved in 45 nodes.
Total Mill time 120 sec.

Results of formulation 2

Continuous optimum in 14 iterations. Objective = 2.
Integer optimum in 5 nodes. Objective = 5.
Optimality proved in 17 nodes.
Total Mill time 80 sec.

In a similar fashion to the *Mining* problems, it proves worth expanding the logical restraints of types (21), (23) and (25) into a series of restraints. The resulting restraints are equivalent in an integer sense although "tighter" in a continuous sense.

The optimal solution to this problem is shown diagrammatically in Fig. 3.

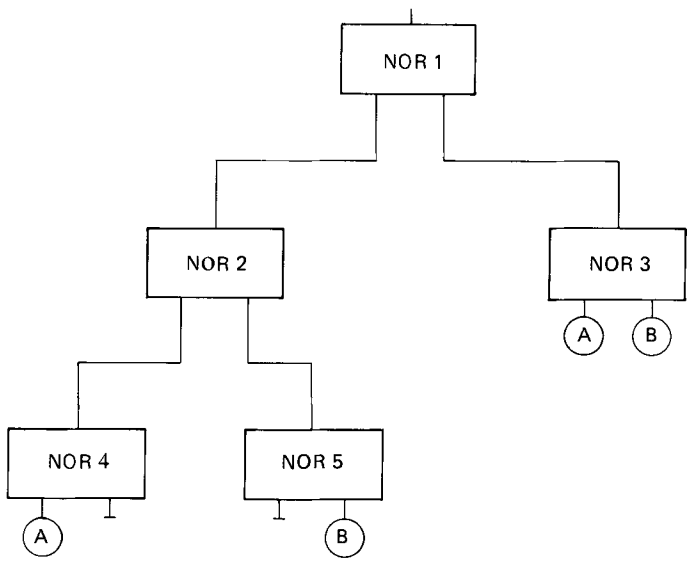

Fig. 3.

4. Conclusions

The following points are suggested as a result of these experiments.

(a) Ingenious formulations to make a model more compact (less restraints) are usually not as good as less sophisticated formulations.

(b) It is desirable to make an integer programming problem as tight as possible in a continuous sense. This may result in bringing the continuous optimum closer to the integer optimum, as in the *Mining for Profit* problem and the *Logical Design* problem. Even if the value of the objective in the continuous optimum is the same in the tighter formulation, the continuous optimal solution may be more realistic. This is the case in the *Market Share* problem. Even when the continuous optima of both formulation are the same, as in the *Mining for Investment Return* problem the tighter formulation may show beneficial effect later in the tree search.

(c) The number of iterations taken to reach the continuous optimum is usually greater in the tighter formulation. For all except the *Logical Design* problem, this is the case. The effect of this, however, is more than offset by the improvement in performance during the second (much more difficult) phase of the tree search.

It is worth remarking that the replacement of a single 0–1 restraint (such as (10) in Problem 3) by a number of restraints (such as (11), (12), (13) and (14)) is a special case of a procedure described by Hammer [6] for reducing pure 0–1 programming problems to generalised covering problems. An alternative way of doing this, based on geometrical considerations, has recently been described by Balas [1].

The value of doing this for more general types of restraints than [10] has not been investigated here and would generally lead to an enormous expansion in the size of the problem. Nor does it always lead to a tightening of restraints. In some cases the effect is to weaken restraints.

The replacements here of one restraint (such as (10)) by a series of restraints (such as (11), (12), (13) and (14)) can be deduced by simple common sense reasoning about the situation being modelled.

Acknowledgment

The author would like to thank Dr. Douglas Hawkins of the University of Witwatersrand, Johannesburg, for drawing his attention to Problem 3. He would also like to thank Professor E.M.L. Beale for a number of very helpful suggestions.

References

[1] E. Balas, "On the set covering problem", Paper presented at the Nato Advanced Study Institute, Figueira da Foz, Portugal (May 1972).
[2] E.M.L. Beale and R.E. Small, "Mixed integer programming by a branch and bound technique", in: W.A. Kalerick, ed., *Proceedings of the IFIP congress* (Spartan Press, East Lansing, Mi. 1965).
[3] E.M.L. Beale and J.A. Tomlin, "An integer programming approach to a class of combinatorial problems", *Mathematical Programming* 1 (1972) 339–344.
[4] M. Benichou, J.M. Gauthier, P. Girodet, G. Hentges, G. Ribiere and O. Vincent, "Experiments in mixed integer linear programming", paper presented at the 7th International Symposium on mathematical programming, The Hague, Holland, September 1970.
[5] R.J. Dakin, "A tree search algorithm for mixed integer linear programming problems", *The Computer Journal* 8 (1965) 250–255.
[6] P.L. Hammer, "On the use of Boolean functions in 0–1 programming", Paper presented at the 7th International Symposium on mathematical programming, The Hague, Holland, September 1970.
[7] P.L. Hammer, "Boolean methods for linear 0–1 programming", Paper presented at the Nato Advanced Study Institute, Figueira da Foz, Portugal (June, 1972).
[8] A.H. Land and A.G. Doig, "An automatic method of solving discrete programming problems", *Econometrica* 28 (1960) 497–520.
[9] H. Lerchs and I.F. Grossman, "Optimum design of open pit mines, *Transactions of the Canadian Institute of Mining* 118 (1965) 17.
[10] Linear Programming Mark 3, Manual for the ICL System XDLA for mathematical programming (International Computers Ltd., Putney, London, England, 1969).
[11] H.H. Meyer, "Applying linear programming to the design of ultimate pit limits", *Management Science* 16 (1969) B121–B135.
[12] G. Mitra, "Designing branch and bound algorithms for mathematical programming", Paper presented at the 7th International Symposium on mathematical programming, The Hague, Holland, September 1970.
[13] S. Muroga, *Threshold logic and its applications* (Wiley, New York, 1971).
[14] H.P. Williams, "The synthesis of logical nets consisting of NOR units", *The Computer Journal* 11 (2) (1968).