RUTCOR
RESEARCH
REPORT

Pivot, Cut, and Dive: A Heuristic for 0-1 Mixed Integer Programming

Mikhail Nediak[a] Jonathan Eckstein[b]

RRR 53-2001, October, 2001

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/~rrr

[a]RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854 USA. E-Mail: msnediak@rutcor.rutgers.edu

[b]Faculty of Management and RUTCOR, 640 Bartholomew Road, Rutgers University, Piscataway, NJ 08854 USA. E-mail: jeckstei@rutcor.rutgers.edu

Rutcor Research Report
RRR 53-2001, October, 2001

Pivot, Cut, and Dive: A Heuristic for 0-1 Mixed Integer Programming

Mikhail Nediak   Jonathan Eckstein

Abstract. We present a heuristic method for general 0-1 mixed integer programming, intended for eventual incorporation into parallel branch-and-bound methods for solving such problems exactly. The core of the heuristic is a rounding method based on simplex pivots, employing only gradient information, for a strictly concave, differentiable merit function measuring integer feasibility. When local minima of this merit function are not integer-feasible, several additional layers of the heuristic come into play. These successive layers include explicit probing of adjacent vertices, modification of the merit function, adjoining of "convexity" cuts to the formulation, and a diving procedure that attempts to fix multiple variables simultaneously. We present "stand-alone" computational results, running the heuristic by itself without an accompanying branch-and-bound optimization, on a variety of problems from the MIPLIB collection.

Acknowledgements: This work was supported in part by National Science Foundation Grant CCR-9902092. The work of first author was also supported in part by DIMACS. Finally, we would like to thank Robert Vanderbei for his help with modifications to the SIMPO solver.

1 Introduction

Many problems arising throughout finance, management, engineering and other fields can be naturally formulated as mixed-integer linear programs (MIPs). Although MIPs are theoretically difficult to solve, branch-and-bound methods have proven fairly successful at solving them in practice, especially when augmented by cutting planes and other sophisticated techniques.

Importantly, MIP branch-and-bound methods can be accelerated by the early detection of near-optimal solutions, via techniques such as reduced-cost fixing. Thus, it is desirable to have an efficient heuristic to provide such solutions.

In the last decade, there has also been significant progress in augmenting the power of branch-and-bound algorithms through parallel computing. In parallel branch-and-bound environments, good heuristics are even more important, since there are generally large amounts of unused processor resources at the early stages of a run. Finding good incumbent solutions early also has a special importance in parallel branch-and-bound because the discipline of the subproblem queue can rarely be exactly best-first. Thus, early detection of near-optimal solution is crucial in pruning non-critical parts of the search tree.

Of course, an efficient heuristic for MIP may also be useful in its own right, providing a solution in cases where proving optimality of an exact solution may be prohibitative or unnecessary.

Although numerous heuristics are available for solving special classes of mixed-integer linear programming, the literature on general-purpose MIP heuristics is limited. Hillier [17], Faaland and Hillier [10], and Ibaraki *et al.* [19] present rounding heuristics for MIPs whose LP relaxations' feasible regions have nonempty interior. The key idea of these heuristics is to construct a piecewise-linear path originating at the solution of the LP relaxation and leading into the interior of its feasible region. The methods then move parametrically along this path, strategically selecting points progressively farther inside the feasible region, and attempting to round each such point without losing feasibility. These operations are supplemented by local search procedures. We initially attempted to apply these same ideas to more general problems, but found the results disappointing.

A very general approach proposed by Glover and Laguna (see [14, 15] and a Chapter 6 in [16]) is based on the Glover's notion of cut search [12]. However, numerical results have not been reported in this work.

An extensively-tested heuristic for 0-1 MIP is the pivot-and-complement method of Balas and Martin [3], which is based on the optimal solution a 0-1 MIP being attained at a vertex where all binary variables are nonbasic. A disadvantage of this method is that it seems to require an availability of simplex tableau information.

A tabu-search-based method for 0-1 mixed integer programming by Glover and Løkketangen [20] uses local search over the pivot neighborhood of the current vertex. This work includes extensive computational results, but limited to relatively small multiconstraint knapsack problems.

Finally, there is some recent work by Balas *et al.* [2] which describes an "OCTANE" heuristic for *pure* 0-1 integer programming. It is based on enumerative intersection cuts (see [1, 7, 12]). The numerical results reported in this work are very encouraging.

Here, we present a heuristic algorithm for solution of mixed 0-1 integer programming problems of the form

$$\begin{aligned} \min \quad & \mathbf{c}^\top \mathbf{x} && (1)\\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b} && (2)\\ & \mathbf{l} \le \mathbf{x} \le \mathbf{u} && (3)\\ & x_i \in \{0,1\},\ \forall i \in I, && (4) \end{aligned}$$

where A is $m \times n$ matrix, $I \subseteq \{1, \ldots, n\}$ is a set of binary variables and $l_i = 0$, $u_i = 1$ for all $i \in I$. For future reference, we denote by P the set of all $\mathbf{x} \in \mathbb{R}^n$ satisfying (2) and (3).

Our heuristic is intended to be used in the context of a large-scale parallel branch-and-bound solver for MIP, such such as PICO, see [9]. However, in this initial report, we present only results for running the heuristic in a "stand-alone" mode on a simulated parallel system of modest size — 16 processors.

Our method attempts to round a fractional solution to LP relaxation (1)-(3) by a local minimization procedure for a concave merit function taking the value 0 at all integer-feasible points. This minimization is accomplished by simplex-like pivots starting from a solution to the LP relaxation. These pivots can rely only on local gradient information until they reach a local minimum, and can thus be implemented nearly as efficiently as traditional simplex pivots.

However, local minima of the merit function are frequently not integral, so we use supplementary "probing" operations to escape such fractional local minima. Probing involves explicitly evaluating adjacent vertices until an acceptable less fractional one is found, and thus bears a resemblance to the Glover-Løkketangen [20] method. Section 2 describes our rounding procedure, including both the basic pivoting techniques and probing.

If probing fails to find a less fractional adjacent vertex, we do not make a locally worsening move like that of a tabu search method, but instead adjoin a cutting plane to the formulation. These cutting planes arise naturally from the calculations of the failed probing step and are of the sort known as "convexity" or "intersection" cuts, first introduced in [22] in the context of concave minimization. After [22], cuts of this type were eventually applied in integer programming; see [1, 13, 21]. Section 3 describes our procedure for using convexity cuts.

If probing fails, and the resulting convexity cut appears excessively shallow, our heuristic applies a recursive, depth-first branching technique. In this "diving" method, we attempt to fix a group of variables simultaneously. If a satisfactory integer-feasible solution does not result, we backtrack and instead apply a complementary "integrality" or "vertex" cut that is deeper than the convexity cut. Our diving procedure is described in detail in Section 4.

Finally, we also simulate running the heuristic in parallel, with each process using a different form of the merit function. An alternative means of escaping fractional local minima is to modify form of the merit function dynamically within a run.

Our method is related to the pivot and complement and Glover-Løkketangen tabu search methods in that it is pivot-based. Furthermore, Glover and Løkketangen also use a merit function to measure integer infeasibility. However, their merit function is not necessarily concave, and their method does not use derivative information to evaluate pivots. Instead, it tests pivots explicitly by evaluating their resulting changes in the merit function and objective. This procedure is comparable to our probing operation, which we only invoke from fractional local minima.

We implemented the heuristic in C++, using a simplex optimizer based on the SIMPO code developed by Vanderbei for the linear programming textbook [24], supplemented by the EXPAND rule [11]. Section 5 gives numerical results on small to medium-size problems from the MIPLIB 3.0 collection. Section 5.4 also discussed future work we hope to conduct on the heuristic.

2 Rounding via Concave Pivots and Probing

The goal of the rounding part of our heuristic is, given a fractional solution to (1)-(3), modify it in such a way that it is integer-feasible while keeping the objective function value as low as possible.

We measure integer infeasibility by a merit function constructed as follows: consider a collection of continuously differentiable concave functions on $\phi_i(\cdot)$, $i \in I$, such that $\phi_i(0) = \phi_i(1) = 0$ and $\phi_i(x) > 0$ for all $x \in (0,1)$. The merit function is defined by

$$\psi(\mathbf{x}) = \sum_{i \in I} \phi_i(x_i). \tag{5}$$

Example 1. Given a parameter $\alpha \in (0,1)$, the following function satisfies the requirements on $\phi_i(\cdot)$:

$$\phi_\alpha(x) = 1 - \begin{cases} \left(\frac{x-\alpha}{\alpha}\right)^2, & x \le \alpha \\ \left(\frac{x-\alpha}{1-\alpha}\right)^2, & x \ge \alpha \end{cases} \tag{6}$$

Note that this function attains a maximum value of 1 at $x = \alpha$.

Rounding proceeds via simplex pivots starting with the basis corresponding to an optimal solution $\mathbf{x}^0$ of the relaxation (1)-(3). Let the basis matrix at a particular iteration of the procedure be denoted by B, the non-basic matrix by N, and corresponding basic feasible solution by $\mathbf{x}$. Let the reduced costs for any given cost vector $\mathbf{v}$ be denoted by $\mathbf{z}(\mathbf{v}) = S(\mathbf{v}_N - (B^{-1}N)^\top \mathbf{v}_B)$ (with elements $z_i(\mathbf{v})$), where $S = \operatorname{diag} \mathbf{s}$ is a diagonal matrix such that

$$s_i = \begin{cases} +1, & x_i = l_i \\ -1, & x_i = u_i \end{cases} .$$

Also, we define the sets

$$\begin{aligned} I_+(\mathbf{v}) &= \{i \in I : z_i(\mathbf{v}) > 0\} \\ I_-(\mathbf{v}) &= \{i \in I : z_i(\mathbf{v}) < 0\} \\ I_0(\mathbf{v}) &= \{i \in I : z_i(\mathbf{v}) = 0\} \end{aligned}$$

Finally, assume that at each iteration we compute the vector $\mathbf{d} = \nabla\psi(\mathbf{x})$.

We select pivots as follows: first, we look for pivots that decrease the merit function but do not increase the objective (rule 1 below). If no such pivot is possible, we try to pivot on a variable x_j that locally improves the merit function, that is, has $z_j(\mathbf{d}) < 0$, at the least possible cost in terms of the objective function (rule 2 below). Note that due to concavity of $\psi(\cdot)$, its value after pivoting will not be higher than predicted by the corresponding reduced cost $z_j(\mathbf{d})$. Thus, the ratio of reduced costs $z_j(\mathbf{d})/z_j(\mathbf{c})$ provides a lower bound on the relative change in the merit function to the change in the objective.

If no locally improving pivot exists, we are at a stationary point of $\psi(\cdot)$ on P. We then perform a "probing" operation in which we explicitly test possible pivots until a satisfactory improving pivot is found or the list of possible entering variables is exhausted (rule 3 below). Although it is possible that probing will examine all the neighboring vertices of the current iterate $\mathbf{x}$, we try to avoid this possibility by inspecting them in an order that gives priority to candidate entering variables x_i that have low values of $z_i(\mathbf{d})$ and $z_i(\mathbf{c})$; see (9) below. In inspecting a possible entering variable x_i, we compute the new iterate $\mathbf{x}'$ that would result. If $\psi(\mathbf{x}') \geq \psi(\mathbf{x})$, that is, $\mathbf{x}'$ is farther from integrality than $\mathbf{x}$, we abandon the pivot and move to the next pivot on the list; note that, due to the concavity of $\psi(\cdot)$, $\psi(\mathbf{x}') < \psi(\mathbf{x})$ is possible even though $z_i(\mathbf{d}) \geq 0$. If the pivot passes this test, we then consider its "objective sacrifice rate"

$$r(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{c}^\top\mathbf{x}' - \mathbf{c}^\top\mathbf{x}}{\psi(\mathbf{x}) - \psi(\mathbf{x}')}, \tag{7}$$

that is, the amount of objective value give up per unit improvement in the merit function. If this sacrifice rate is "acceptable" in relation to the prior history of the run — see (10) below — then $\mathbf{x}'$ it accepted as the next iterate and no further pivots are probed. Otherwise, probing continues. If it is necessary to examine every possible pivot, we select, from among all the pivots with $\psi(\mathbf{x}') < \psi(\mathbf{x})$, a pivot minimizing $r(\mathbf{x}, \mathbf{x}')$. If there was no pivot with $\psi(\mathbf{x}') < \psi(\mathbf{x})$, we declare the rounding process to have failed.

Formally, our rounding algorithm is as follows:

1. If $J = (I_-(\mathbf{d})) \cap (I_-(\mathbf{c}) \cup I_0(\mathbf{c})) \neq \emptyset$, then pivot on a variable

$$i \in \operatorname{Argmin}\{z_j(\mathbf{d}) : j \in J\}. \tag{8}$$

2. Otherwise, if $I_-(\mathbf{d}) \neq \emptyset$, then pivot on a variable

$$i \in \operatorname{Argmin}\left\{\frac{z_j(\mathbf{d})}{z_j(\mathbf{c})} : j \in I_-(\mathbf{d})\right\};$$

3. (Probing) Otherwise, $I_-(\mathbf{d}) = \emptyset$, and the current iterate $\mathbf{x}$ is a stationary point of $\psi(\cdot)$ with respect to P. We therefore commence the probing step. First, we order the indices i of the possible entering variables in order of increasing

$$(z_i(\mathbf{d}) + \theta)(z_i(\mathbf{c}) - \min_j z_j(\mathbf{c}) + \theta) \tag{9}$$

for some constant $\theta > 0$. In this order, we then evaluate each i as follows:

(a) Compute the new iterate $\mathbf{x}'$ that would result from pivoting x_i into the basis.

(b) If $\psi(\mathbf{x}') \geq \psi(\mathbf{x})$, move to the next pivot on the list.

(c) Otherwise, compute $r(\mathbf{x}, \mathbf{x}')$ as in (7). If

$$r(\mathbf{x}, \mathbf{x}') \leq \mu \frac{c^\top \mathbf{x} - c^\top \mathbf{x}^0}{\psi(\mathbf{x}^0) - \psi(\mathbf{x})}, \tag{10}$$

that is, it is no worse than μ times the rate of objective sacrifice sustained so far, then accept the new iterate $\mathbf{x}'$ and terminate the probing step. If the denominator of the right-hand side of (10) is zero, we take the ratio to be zero.

If the entire set of possible entering variables has been examined without one passing the test (10), we accept the one with the lowest $r(\mathbf{x}, \mathbf{x}')$; if no pivot with $\psi(\mathbf{x}') < \psi(\mathbf{x})$ was encountered, we declare the probing step to have failed.

We repeat the procedure above until an integer-feasible point is found or the probing step fails.

Remark 1. One can use steepest-edge selection instead of Danzig's rule in (8).

Remark 2. An alternative approach is to use a merit function that combines integrality and objective improvement at some fixed ratio. For a (sufficiently small) positive constant w, consider the modified merit function

$$\psi'(\mathbf{x}) = \psi(\mathbf{x}) + w\mathbf{c}^\top \mathbf{x} \tag{11}$$

and let $\mathbf{d}' = \nabla \psi'(\mathbf{x})$. Then one can simply select a pivot $i \in \text{Argmin}\{z_j(\mathbf{d}') : j \in I_-(\mathbf{d}')\}$ when $I_-(\mathbf{d}') \neq \emptyset$, and use a probing step otherwise. The natural ordering of probing pivots, in this case, is of increasing $z_j(\mathbf{d}')$. We stop probing if we encounter a pivot satisfying a historical average threshold on $\psi'(\mathbf{x}') - \psi'(\mathbf{x})$.

Remark 3. Note that probing may result in a large number of pivots being attempted and successively rejected. To avoid this phenomenon, we may instead modify the function $\psi(\cdot)$ in such a way that $\mathbf{x}$ is no longer a local minimum. In this case, however, special care must be taken to avoid cycling.

3 Rounding within a cutting plane method

If the rounding procedure terminates with a failed probing step, we invoke our cutting plane procedure, which we now describe.

Note that all integer feasible solutions to (1)-(4) belong to the the set $L = \{\mathbf{x} : \psi(\mathbf{x}) \leq 0\}$. Given a fractional vertex $\mathbf{x}$ of the polyhedron P, the intersection of the boundary of L with the edges of P extending from $\mathbf{x}$ will allow us to generate a cut. Cuts of this type are known in the literature as "convexity" or "intersection" cuts. They were introduced by Tuy [22, 23] in the context of concave minimization over a polyhedron, and applied to integer programming by Balas, Glover, Raghavachari and others (see e.g. [1, 13, 21]).

Denote the current vertex of P by

$$\bar{\mathbf{x}} = \begin{pmatrix} B^{-1}(\mathbf{b} - N\mathbf{g}) \\ \mathbf{g} \end{pmatrix},$$

where $g_i = l_i$ ($g_i = u_i$) for nonbasic variable at the lower (upper) bound. Edges of P extending from $\bar{\mathbf{x}}$ can be written as $\mathbf{x}^{(i)}(t) = \bar{\mathbf{x}} - ts_i\xi^{(i)}$ for $i \in J_B^c$ (the set of nonbasic indices), where

$$\xi^{(i)} = \begin{pmatrix} B^{-1}N\mathbf{e}^{(i)} \\ -\mathbf{e}^{(i)} \end{pmatrix}$$

and the s_i's are as defined above.

Note that the probing step must compute $\xi^{(i)}$ when it evaluates the possibility of pivoting on $i \in J_B^c$. Thus, if the probing step fails, the complete set of $\{\xi^{(i)} : i \in J_B^c\}$ is available without any further computation.

Observation 1. The polyhedron P is contained in $\bar{\mathbf{x}} + K$, where K is the cone spanned by the vectors $\xi^{(i)}$, $i \in J_B^c$.

Next, for all $i \in J_B^c$, we let t_i be the positive solution of the equation

$$\psi(\mathbf{x}^{(i)}(t_i)) = 0. \tag{12}$$

If this equation does not have a solution, we say that $t_i = +\infty$. We assume, for simplicity of notation, that all the t_i's are finite, since all equations corresponding to infinite t_i's can simply be dropped in the analysis below. Since the current vertex is not integer-feasible, we have $t_i > 0$. Consider a hyperplane passing through $\mathbf{x}^{(i)}(t_i)$, whose normal vector $\mathbf{f}$ is chosen so that

$$\mathbf{f}^\top \mathbf{x}^{(i)}(t_i) = \mathbf{f}^\top \bar{\mathbf{x}} + 1, \text{ for all } i \in J_B^c. \tag{13}$$

Since $\psi(\bar{\mathbf{x}}) > 0$ and $\psi(\cdot)$ is concave, the following holds:

Observation 2. For all points $\mathbf{x}$ in the intersection of $\bar{\mathbf{x}} + K$ with the open halfspace $\{\mathbf{x} : \mathbf{f}^\top\mathbf{x} < \mathbf{f}^\top\bar{\mathbf{x}} + 1\}$ we have $\psi(\mathbf{x}) > 0$.

Observations 1 and 2 immediately imply

Proposition 1. *The inequality*

$$\mathbf{f}^\top\mathbf{x} \geq \mathbf{f}^\top\bar{\mathbf{x}} + 1 \tag{14}$$

where $\mathbf{f}$ *satisfies (13), is a valid inequality for the MIP (1)-(4).*

While convexity cuts of the form (14) may not be very deep in general, they are guaranteed to cut off the current point and all of its neighboring vertices when probing failed to find an improving pivot.

Thus, we may repeat the following steps until an integer feasible solution is found or the problem becomes infeasible:

1. Perform rounding as described in Section 2. If an integer feasible solution results, stop; otherwise go to step 2.

2. Add a convexity cut using information from the last failed probing step. Regain primal feasibility (by backtracking pivots or reoptimizing the problem). If the problem is infeasible, terminate: no feasible solution exists. Otherwise, go to step 1.

Remark 4. Instead of adding a cutting plane in step 2 of the above procedure, one can change the merit function so that the current point is no longer a local minimum. Such modifications can be applied, for example, every fixed number of failed probing steps. The number of merit function changes should be limited, however, to avoid cycling among fractional vertices.

We present an explicit formula for the cut below. The system (13) is equivalent to

$$\mathbf{f}^\top(\bar{\mathbf{x}} - t_i s_i \xi_i) = \mathbf{f}^\top \bar{\mathbf{x}} + 1, \text{ for all } i \in J_B^c$$

which is, in turn, equivalent to

$$t_i s_i f_i - t_i s_i \mathbf{f}_B^\top B^{-1} N \mathbf{e}^{(i)} = 1, \text{ for all } i \in J_B^c.$$

Since this system is underdetermined, we can take, for example, $\mathbf{f}_B = \mathbf{0}$. Then

$$f_i = \frac{1}{t_i s_i},$$

which implies the following form of the cutting plane:

$$\sum_{i \in J_B^c} \frac{x_i}{t_i s_i} \geq \sum_{i \in J_B^c} \frac{g_i}{t_i s_i} + 1. \tag{15}$$

A similar expression for convexity cuts first appears in [22].

Proposition 2. *When the $\phi_i(\cdot)$'s are of the form (6), the equation (12) can be solved in $O(|I \cap J_B|^2)$ time.*

Proof. Given a basis B and the edge $\mathbf{x}^{(j)}(t) = \bar{\mathbf{x}} - t s_j \xi^{(j)}$,

$$\begin{aligned} \psi(\mathbf{x}^{(j)}(t)) &= \sum_{i \in I} \phi_i(\bar{x}_i - t s_j \xi_i^{(j)}) \\ &= \sum_{i \in I \cap J_B} \phi_i(\bar{x}_i - t s_j \xi_i^{(j)}) + \begin{cases} \phi_j(t s_j), & j \in I \\ 0, & j \notin I \end{cases}, \end{aligned}$$

as $\phi_i(\mathbf{x}^{(j)}(t)) = 0$ for $i \notin J_B \cup \{j\}$. Thus, when computing $\psi(\mathbf{x}^{(j)}(t))$, one needs to evaluate at most $|I \cap J_B| + 1$ of the ϕ_i's (in $O(1)$ time each).

Further, note that $\psi(\mathbf{x}^{(j)}(t))$ is piecewise quadratic. To solve (12), we first identify an interval $[t'_j, t''_j]$ containing t_j on which $\psi(\mathbf{x}^{(j)}(t))$ is a pure quadratic. The largest value of

$$0,\ \left\{ \frac{\bar{x}_i - \alpha_i}{s_j \xi_i^{(j)}} : i \in I \cap (J_B \cup j),\ \frac{\bar{x}_i - \alpha_i}{s_j \xi_i^{(j)}} > 0 \right\},\ +\infty$$

for which $\psi(\mathbf{x}^{(j)}(\cdot))$ is nonnegative will give us t'_j, and the smallest for which it is negative will give us t''_j. This calculation can be performed in at most $|I \cap (J_B \cup j)|$ evaluations of $\psi(\mathbf{x}^{(j)}(\cdot))$.

The values of $\psi(\mathbf{x}^{(j)}(\cdot))$ and its first two derivatives at any $t \in (t'_j, t''_j)$ give the parameters of a quadratic equation whose root in $[t'_j, t''_j)$ is the solution of (12). Note that evaluation of $\psi(\mathbf{x}^{(j)}(\cdot))$ and its first two derivatives can be accomplished in $O(|I \cap J_B|)$ time. Thus, the total complexity is $O(|I \cap J_B|^2)$. □

Remark 5. One can lower the complexity of finding the t_i's by using a sparse representation of the $\xi^{(i)}$. Let J_B^i be the set of nonzero components of $\xi^{(i)}$. Then, the complexity of finding t_i is $O((|I \cap J_B^i| + 1)^2)$ which, normally, does not exceed complexity of finding $\xi^{(i)}$ itself. Thus, it is not prohibitive to automatically find t_i for each $\xi^{(i)}$ examined while probing. In this case, we immediately obtain a convexity cut (15) at the end of every failed probing step.

4 Combining cutting planes with diving

It is difficult, in general, to guarantee finite convergence of concave minimization algorithms based on the cutting planes. The original concave minimization algorithm by Tuy, as presented in [22], was shown to be nonconvergent by Zwart [25]. Obtaining convergence results required modifications to the method; see for example [4, 23, 26]. Thus, the algorithm presented in Section 3 is unlikely to be finite. We can, however, add a conceptually simple feature that makes it finitely convergent. This same feature may in some cases facilitate finding feasible solutions more quickly.

Consider two disjoint subsets Q_0 and Q_1 of I. We can divide the solution of the problem (1)-(4) into the solution of two problems

$$\begin{array}{ll} \min & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} & \mathbf{x} \in P \\ & x_i = 0,\ \forall i \in Q_0 \\ & x_i = 1,\ \forall i \in Q_1 \\ & x_i \in \{0,1\},\ \forall i \in I \setminus (Q_0 \cup Q_1) \end{array} \tag{16}$$

and

$$\begin{array}{ll} \min & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} & \mathbf{x} \in P \\ & \sum_{i \in Q_0} x_i + \sum_{i \in Q_1} (1 - x_i) \geq 1 \\ & x_i \in \{0,1\},\ \forall i \in I \end{array} \tag{17}$$

Note that the first subproblem may be substantially easier than both the oringal problem and the second subproblem when the cardinality of $Q_0 \cup Q_1$ is comparable to that of I. We will generate and examine these subproblems systematically in a depth-first recursive search algorithm, with the child problem of the form (16) always being examined first, and the child of the form (17) being examined only when backtracking out of (16) without having found a feasible solution. By examining branches of the form (16) first, one hopes to quickly obtain feasible solutions to (1)-(4).

Classical branch-and-bound methods for integer programming (see [5]) have long used a initial depth-first search phase to quickly identify integer feasible solutions, and the technique is often referred to as *diving*. However, the branching rule historically tends to involve simple rounding up or down of a single integer variable.

Let $\bar{\mathbf{x}}$ be a current vertex of P and $\mathbf{a}^\top \mathbf{x} \geq \beta$ be a cut. We define the *depth* of the cut with respect to $\bar{\mathbf{x}}$ to be the distance between $\bar{\mathbf{x}}$ and the halfspace defined by the cut. An straightforward calculation shows this distance to be

$$d(\bar{\mathbf{x}}, \mathbf{a}, \beta) = \frac{(\beta - \mathbf{a}^\top \bar{\mathbf{x}})_+}{\|\mathbf{a}\|} \tag{18}$$

We will use cut depth as an approximate measure of quality.

We will refer to the cuts of the form

$$\sum_{i \in Q_0} x_i + \sum_{i \in Q_1} (1 - x_i) \geq 1 \tag{19}$$

used in (17) as *vertex* cuts. An advantage of vertex cuts over convexity cuts is that one can control their depth

$$d(\bar{\mathbf{x}}, Q_0, Q_1) = \frac{\left(1 - \sum_{i \in Q_0} \bar{x}_i - \sum_{i \in Q_1} (1 - \bar{x}_i)\right)_+}{(|Q_0| + |Q_1|)^{1/2}}$$

via the choice of Q_0 and Q_1. Our goal is to include as many indices in Q_0 and Q_1 as possible, while still making sure that the cut (19) is "sufficiently" deep. Starting with $Q_0 = Q_1 = \emptyset$, we process the elements $\bar{x}_i$ of $\bar{\mathbf{x}}$ in increasing order of $\min\{\bar{x}_i, 1 - \bar{x}_i\}$, as follows:

- If $\bar{x}_i < 1 - \bar{x}_i$, let $Q_0' = Q_0 \cup \{i\}$ and $Q_1' = Q_1$, otherwise let $Q_0' = Q_0$ and $Q_1' = Q_1 \cup \{i\}$.
- If $d(\bar{\mathbf{x}}, Q_0', Q_1') \geq d^*$ (for some fixed $d^* > 0$) set $Q_0 \leftarrow Q_0'$, $Q_1' \leftarrow Q_1$ and proceed to the next index, otherwise stop.

A natural value of d^* that guarantees at least one i in $Q_0 \cup Q_1$ with $\bar{x}_i \notin \{0, 1\}$ is $1/(2\sqrt{|I|})$.

We combine rounding, convexity cuts, and diving in the recursive procedure below. We start by invoking the procedure on arguments consisting of (1)-(4) and an optimal extreme point solution of its LP relaxation.

1. Perform rounding as in Section 2. If an integer feasible solution is found, stop and return it, otherwise continue with step 2.

2. Compute the depth d_{conv} of the convexity cut constructed in the last failed probing step. Also, construct sets Q_0 and Q_1 by the procedure outlined above, with a fixed $d^* \leq 1/(2\sqrt{|I|})$.

3. If $d_{\text{conv}} \geq d(\bar{\mathbf{x}}, Q_0, Q_1)$, add the convexity cut and go to step 6, otherwise go to step 4.

4. Construct a subproblem of the form (16) and determine whether its LP relaxation is feasible. If not, skip to step 5. Otherwise, recursively apply the procedure to this subproblem. If the procedure returns a feasible solution, return this solution. Otherwise, continue with step (5).

5. Add a vertex cut of the form (19). Continue with step 6.

6. Regain primal feasibility by backtracking pivots or reoptimizing the problem. If the problem becomes infeasible, return an error code indicating no feasible solution to the (sub)problem exists. Otherwise go to step 1.

It is now straightforward to prove convergence to some feasible solution, using techniques similar to standard cutting plane method convergence proofs for concave minimization (see, for example, [18]). It is necessary, however, to impose the assumption, standard in such proofs, that P is bounded.

Proposition 3. *Suppose P is bounded. Then the algorithm above terminates in a finite number of iterations either with an integer-feasible solution, or on detection of integer infeasibility.*

Proof. The proof is by induction on the number of integral variables $|I|$. When $I = \emptyset$, any vertex of P is vacuously integer-feasible.

Now suppose that we have proved finiteness for the numbers of integral variables less than $|I|$, but the algorithm does not terminate for some problem with $|I|$ integer variables. Whenever we generate a subproblem of the form (16), its number of integer variables is less than $|I|$, so the algorithm must terminate on the subproblem by the induction hypothesis. Thus, the only possibility is that the sequence of cuts added to the subproblem (17) is infinite. Consider the sequence of polytopes $\{P^k\}_{k=0}^{\infty}$ generated by the algorithm, where $P^0 = P$. Let $\bar{\mathbf{x}}^k$ be a vertex of P^k being cut off by each respective cut. Since the depth of all the cuts is at least d^*, the distance from $\bar{\mathbf{x}}^k$ to any point of P^{k+1} is at least d^*. Thus, $\|\bar{\mathbf{x}}^k - \bar{\mathbf{x}}^l\| \geq d^*$ for all $l > k$. Note, however, that the sequence $\{\bar{\mathbf{x}}^k\}_{k=0}^{\infty}$ belongs to the bounded, finite dimensional and, thus, compact polytope P. It follows, that we can extract a convergent subsequence $\{\bar{\mathbf{x}}^{k_i}\}_{i=0}^{\infty}$, for which we would have $\|\bar{\mathbf{x}}^{k_i} - \bar{\mathbf{x}}^{k_{i+1}}\| \to 0$ as $i \to \infty$, a contradiction. $\square$

Remark 6. Without sacrificing convergence, one can modify the decision rule between convexity and vertex cuts to $\gamma d_{\text{conv}} \geq d(\bar{\mathbf{x}}, Q_0, Q_1)$, where a positive constant γ specifies the level of preference given to convexity cuts.

Remark 7. In practice, vertex cuts may also be used to improve the stability of the method in cases where one may encounter numerical difficulties in the construction of convexity cuts.

Remark 8. One can also continue the algorithm after finding an initial integer-feasible solution, in the hope of attaining better solutions in subsequent iterations. When the only termination criterion is infeasibility of the branch (17), and P is bounded, the algorithm then becomes an exact cutting plane algorithm for mixed 0-1 integer programming.

5 Numerical experiments

5.1 Test set

We now present the results of numerical experiments with the method described above. The test problems were selected from the MIPLIB 3.0 collection [6]. To improve numerical stability, all the problems were scaled. We had to rule out problems containing general integer variables, leaving 49 MIPLIB problems. Of these, 7 problems were ruled out due to the limitations of the LP solver currently integrated into our method: `air04`, `air05`, `dano3mip`, `fast0507`, `nw04`, `rentacar` and `seymour`; on each of these, the LP solver would either encounter numerical difficulties or fail to terminate after 10,000 iterations. In future, we may be able to tackle such problems by upgrading our LP solver. We present our results for the remaining 42 problems below.

5.2 Experimental setup

We compared the two sets of pivot selection rules outlined in the Section 2: the standard or "ratio" rules, and the "sum" rules of Remark 2. For each method, we simulated a run on 16 "embarrassingly" parallel processors, with each processor using a different merit function. In reality, we ran the algorithm in a serial fashion starting from the optimal solution to LP relaxation, without using information obtained in the previous runs. Merit function parameters were selected as follows:

1. One "processor" with $\alpha_i = 0.5$ for all $i \in I$.
2. One "processor" with $\alpha_i = x_i^0 - \sigma c_i$, if $x_i^0 - \sigma c_i \in (0, 1)$, where σ is a sufficiently small positive number, and $\alpha_i = 0.5$ otherwise, for all $i \in I$. The hope here is that the initial point $\mathbf{x}^0$ will not be a local minimum of the merit function.
3. Fourteen "processors" with all α_i selected as a uniform random draw from $(0, 1)$ for all $i \in I$.

In cases 2 and 3, we changed the merit function, instead of adding a cut, at every fifth failed probing step; see Remark 4. New parameters were selected at random for case 3 and as $\alpha_i = \bar{x}_i - \sigma d_i$ for case 2.

Runs were terminated whenever any of the following conditions was met:

- A pivot limit of 10,000 was reached, counting 100% of the regular pivots and 5% of the probing pivots. Note that the probing pivots never require a basis refactorization.

- A probing step failed after a maximum of 50 cuts had already been added.
- An integer-feasible solution was found after a minimum of 500, or 5 times the number pivots performed in solving the LP relaxation, whichever was greater.
- A subsequent integer-feasible solution had not been found after the same number of pivots as in the previous condition; see Remark 8.
- Numerical difficulties were encountered due to pivot selection or feasibility tolerance selection problems [11].

We selected these termination criteria to simulate a situation when our heuristic is incorporated in a parallel branch-and-bound code and is invoked at the beginning of the run, while the branch-and-bound subproblem queue is not sufficiently populated to keep all processors busy. Thus, the goal is to obtain a reasonable quality feasible solution as quick as possible.

The method was implemented in C++ and run on a 500MHz Pentium Xeon system under RedHat Linux 7.1.

5.3 Experimental results

The method failed to find an integer-feasible solution on 10 out of the 42 test problems, `10teams`, `air03`, `enigma`, `harp2`, `misc03`, `misc07`, `mitre`, `p0548`, `p2756` and `swath`. On the remaining problems, the method found an optimal solution in 3 cases, an integer-feasible solution with a gap below 10% of the optimum in 18 cases, with a gap between 10% and 100% in 7 cases, and with a gap above 100% in 4 cases. The results for the successful runs with the ratio pivot selection rules are summarized in the Figure 1, which shows the best objective value obtained among the 16 virtual "processors". Detailed run information for both the ratio and sum rules can also be found in the Table 1.

As can be seen from the table, the ratio rule generally finds better solutions than the the sum method. For a few problems, such as `qiu`, the difference in solution quality is dramatic.

The only exceptions to the dominance of the ratio rules are the similar problems `mas74` and `mas76`. It is interesting to note that in both of these special cases, we would fail to find an integer-feasible solution by the sum method if we changed the convexity cut preference parameter γ to value 10 from its default value of 1. Thus, diving appears instrumental in finding feasible solutions for these two problems.

We note, however, that the sum method has the additional flexibility of the weighting parameter w. Increasing w for the problems where it is relatively easy to attain integer feasibility (such as `pk1`, `markshare1` and `markshare2`) will result in higher quality solutions. On the other hand, reducing w may facilitate finding integer-feasible solutions in some hard cases (for example, `danoint`).

Table 2 displays the average running time for both methods on the successfully solved problems. Run time seems to be highly dependent on the LP running time and does not differ from it too much in the absolute terms. The ratio and the sum methods finished in under 1 second on 66% and 59% of the problems, respectively. On the remaining problems,

Name	Method	Success	Best solution	Gap	Time (sec.)
CAP6000	ratio	12	-2449657	0.07%	92.174
CAP6000	sum	9	-2446147	0.21%	116.250
DANOINT	ratio	7	66.499999	1.26%	37.867
DANOINT	sum	1	83	26.39%	38.897
DCMULTI	ratio	16	188434.1	0.13%	0.393
DCMULTI	sum	16	188855.2	0.36%	0.508
EGOUT	ratio	16	612.62955	7.84%	0.354
EGOUT	sum	16	649.01969	14.24%	0.511
FIBER	ratio	16	1099992.05	170.98%	2.929
FIBER	sum	16	1170258.28	188.29%	1.582
FIXNET6	ratio	16	7135	79.14%	2.205
FIXNET6	sum	16	7557	89.73%	2.985
KHB05250	ratio	16	108737250	1.68%	0.129
KHB05250	sum	16	110702437	3.52%	0.136
L152LAV	ratio	13	4753.00007	0.06%	66.603
L152LAV	sum	12	4763.00003	0.27%	36.984
LSEU	ratio	16	1155	3.13%	0.112
LSEU	sum	16	1230	9.82%	0.099
MARKSHARE1	ratio	16	245	24400.00%	0.014
MARKSHARE1	sum	16	245	24400.00%	0.011
MARKSHARE2	ratio	16	188	18700.00%	0.019
MARKSHARE2	sum	16	188	18700.00%	0.025
MAS74	ratio	16	13861.468	17.46%	0.023
MAS74	sum	16	13279.7434	12.53%	0.306
MAS76	ratio	16	42526.0001	6.30%	0.014
MAS76	sum	16	41784.6211	4.45%	0.279
MISC06	ratio	16	12857.3864	0.05%	0.459
MISC06	sum	16	12870.1742	0.15%	1.315
MKC	ratio	16	-272.51	50.79%	15.133
MKC	sum	16	-272.51	50.79%	3.833
MOD008	ratio	16	308	0.33%	0.031
MOD008	sum	16	308	0.33%	0.068
MOD010	ratio	16	6555	0.11%	31.659
MOD010	sum	16	6665	1.79%	16.659
MOD011	ratio	16	-53641979	1.68%	77.969
MOD011	sum	16	-42783986.7	21.58%	146.125
MODGLOB	ratio	16	20786787	0.22%	0.158
MODGLOB	sum	16	20786787	0.22%	0.269
P0033	ratio	7	3345	8.29%	0.483
P0033	sum	6	3604	16.67%	0.556
P0201	ratio	13	8185	7.49%	3.399
P0201	sum	13	8675	13.92%	2.887
P0282	ratio	16	290859	12.56%	0.153
P0282	sum	11	306480	18.60%	2.669
PK1	ratio	16	37	236.36%	0.079
PK1	sum	16	37	236.36%	0.064
PP08A	ratio	16	9020	22.72%	0.180
PP08A	sum	16	9330	26.94%	0.186
PP08ACUTS	ratio	16	7860	6.94%	0.384
PP08ACUTS	sum	16	8910	21.22%	0.330
QIU	ratio	16	-115.248257	13.26%	20.263
QIU	sum	16	90.7886431	168.33%	20.013
RGN	ratio	16	82.1999992	0.00%	0.026
RGN	sum	16	82.1999992	0.00%	0.018
SET1CH	ratio	16	64801.25	18.82%	1.772
SET1CH	sum	16	67902.25	24.51%	1.938
STEIN27	ratio	16	18	0.00%	0.066
STEIN27	sum	16	18	0.00%	0.048
STEIN45	ratio	16	31	3.33%	0.549
STEIN45	sum	16	31	3.33%	0.581
VPM1	ratio	16	20	0.00%	0.129
VPM1	sum	16	20	0.00%	0.109
VPM2	ratio	16	15	9.09%	0.551
VPM2	sum	16	15.5	12.73%	0.361

Table 1: The number of successful runs, the gap of the best integer-feasible solution and the average run time for both methods

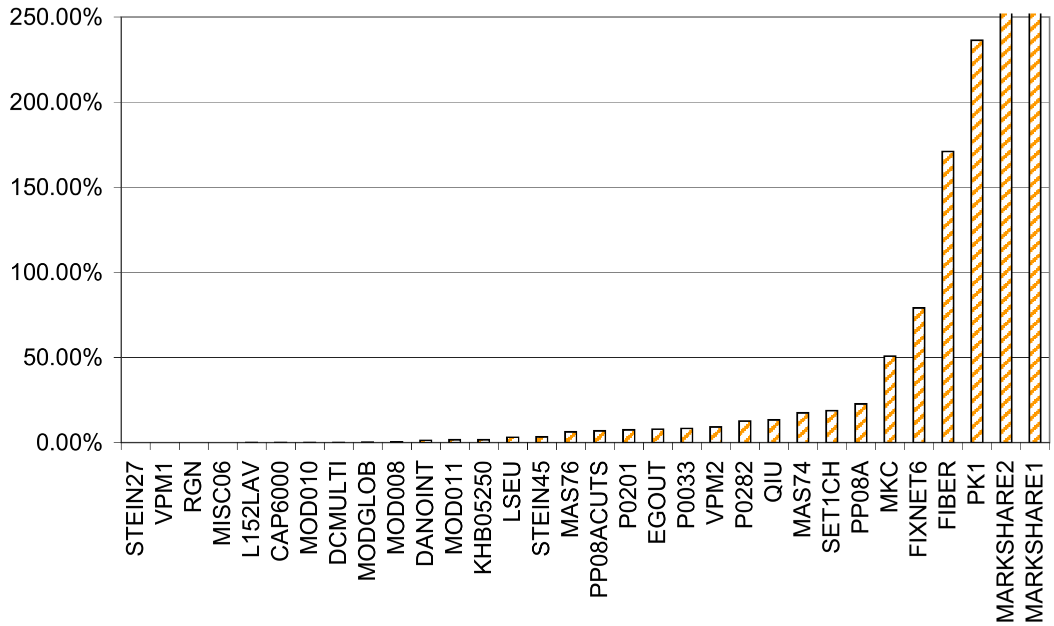

Figure 1: Percentage difference in objective value between the integer optimum and the heuristic solution obtained by the ratio pivot rules.

the highest ratio of average heuristic running time to LP running time was 24.1 and 17.4, respectively.

The run times for those problems where the method failed are, unfortunately, much worse, and are given in the Table 3. These longer run times can be explained by a very large number of probing pivots and the time per pivot increasing with the addition of large numbers of dense cuts.

We have several possible explanations for the failure cases. One is that some of these problems are highly degenerate and our LP solver does not handle degeneracy well, as with `10teams` and `harp2`. The use of an "industrial-strength" LP solver may alleviate this problem.

Many of the problems where we observed failure have large numbers of set partitioning constraints. Note that when these kinds of constraints are present, branching "up" in a standard branch-and-bound method normally rounds many variables to integrality. Thus, one can hope that incorporation of our heuristic throughout the search tree of an exact branch-and-bound algorithm will allow it to quickly round the remaining variables. Another indication that this may be the case is that, for these same problems, the OCTANE heuristic of [2] could find good feasible solutions only at relatively deep levels of the branch-and-bound tree.

Name	LP Time	Ratio Time	Sum Time
P0033	0.00	0.48	0.56
LSEU	0.01	0.11	0.10
MARKSHARE1	0.01	0.01	0.01
MARKSHARE2	0.01	0.02	0.03
MOD008	0.01	0.03	0.07
RGN	0.01	0.03	0.02
EGOUT	0.02	0.35	0.51
PK1	0.02	0.08	0.06
MAS74	0.03	0.02	0.31
MAS76	0.03	0.01	0.28
PP08ACUTS	0.03	0.38	0.33
STEIN27	0.04	0.07	0.05
P0282	0.08	0.15	2.67
VPM1	0.08	0.13	0.11
VPM2	0.08	0.55	0.36
KHB05250	0.10	0.13	0.14
PP08A	0.15	0.18	0.19
MODGLOB	0.16	0.16	0.27
P0201	0.23	3.40	2.89
STEIN45	0.23	0.55	0.58
FIXNET6	0.25	2.21	2.99
SET1CH	0.31	1.77	1.94
FIBER	0.37	2.93	1.58
DCMULTI	0.44	0.39	0.51
MKC	2.16	15.13	3.83
MISC06	2.17	0.46	1.32
L152LAV	2.89	66.60	36.98
CAP6000	3.83	92.17	116.25
MOD010	4.30	31.66	16.66
DANOINT	10.98	37.87	38.90
QIU	17.18	20.26	20.01
MOD011	50.82	77.97	146.13

Table 2: Average run time of both ratio and sum methods, in seconds, ordered by LP running time.

Name	LP Time	Ratio Time	Sum Time
10TEAMS	21.49	90.81	85.13
AIR03	33.86	1349.31	976.19
ENIGMA	0.01	4.81	4.44
HARP2	0.52	28.00	29.31
MISC03	0.13	48.00	65.88
MISC07	0.67	69.00	100.63
MITRE	7.22	56.69	40.25
P0548	0.11	12.56	14.44
P2756	0.11	19.19	12.38
SWATH	6.82	283.13	326.31

Table 3: Average run time for both methods (seconds) in cases of failure.

5.4 Conclusions and future research

Our method succeeded in finding an integer-feasible solution on 32 out of 42 test problems, with an objective gap below 10% in 21 cases, and optimality in 3 cases. The solution times are reasonable, and always within either one or two orders of magnitude of the LP solution time, depending on whether one counts the effect of the 16 simulated parallel processors. For many of the problems, results were obtained within one second.

We plan to investigate a number of topics in future research in this area. The first involves integration of the heuristic into a parallel branch-and-bound solver, both at the outset of the run and continuously throughout the search tree. We would also like to further study alternative stopping rules and consider adding communication between parallel runs.

Another issue is integrating the method into a general cutting plane framework. Other classes of cuts, which would be available in such an environment, could supplement the convexity and vertex cuts already used by the heuristic. Conversely, it is possible that the heuristic's convexity cuts might prove useful in proving solution optimality within a branch-and-cut search procedure.

For problems where the number of variables is very large in comparison to the number of rows, it may prove beneficial to modify the method to use some form of partial pricing or column generation.

We are currently re-implementing our method in a solver-independent manner, using an extension to the COIN solver interface; see [8]. This extension should facilitate the incorporation of the method into branch-and-bound and branch-and-cut codes, and will also allow it to use industrial-strength LP solvers such as CPLEX and OSL.

Finally, we would like to evaluate techniques for extending the heuristic to problems with general integer variables.

References

[1] Egon Balas. Intersection cuts — a new type of cutting planes for integer programming. *Operations Research*, 19:19–39, 1971.

[2] Egon Balas, Sebastián Ceria, Milind Dawande, Francois Margot, and Gábor Pataki. OCTANE: a new heuristic for pure 0-1 programs. *Operations Research*, 49(2):207–225, 2001.

[3] Egon Balas and Clarence H. Martin. Pivot-and-complement — a heuristic for 0-1 programming. *Management Science*, 26(1):86–96, 1980.

[4] S. Bali and Stephen E. Jacobsen. On the convergence of the modified Tui algorithm for minimizing a concave function on a bounded convex polyhedron. In *Optimization Techniques (Proc. 8th IFIP Conf., Würzburg, 1977), Part 2*, pages 59–66. Lecture Notes in Control and Information Science, Vol. 7. Springer, Berlin, 1978.

[5] Evelyn M. L. Beale. Branch and bound methods for mathematical programming systems. *Annals of Discrete Mathematics*, 5:201–219, 1979. Discrete optimization (Proc. Adv. Res. Inst. Discrete Optimization and Systems Appl., Banff, Alta., 1977), II.

[6] Robert E. Bixby, Sebastián Ceria, Cassandra M. McZeal, and Martin W. P. Savelsberg. An updated mixed integer programming library: MIPLIB 3.0. Technical Report 98-3, Department of Computational and Applied Mathematics, Rice University, 1998.

[7] Claude-Alain Burdet. Enumerative inequalities in integer programming. *Mathematical Programming*, 2(1):32–64, 1972.

[8] Common optimization interface for operations research. http://oss.software.ibm.com/developerworks/opensource/coin/.

[9] Jonathan Eckstein, Cynthia A. Phillips, and William E. Hart. PICO: An object-oriented framework for parallel branch and bound. In *Proceedings of the Workshop on Inherently Parallel Algorithms in Optimization and Feasibility and their Applications*, pages 219–265, Haifa, March 2000. Elsevier Scientific.

[10] Bruce H. Faaland and Frederick S. Hillier. Interior path methods for heuristic integer programming procedures. *Operations Research*, 27(6):1069–1087, 1979.

[11] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. A practical anti-cycling procedure for linearly constrained optimization. *Mathematical Programming*, 45(3):437–474, 1989.

[12] Fred Glover. Cut search methods in integer programming. *Mathematical Programming*, 3:86–100, 1972.

[13] Fred Glover. Convexity cuts and cut search. *Operations Research*, 21:123–134, 1973. Mathematical programming and its applications.

[14] Fred Glover and Manuel Laguna. General purpose heuristics for integer programming — part I. *Journal of Heuristics*, 2:343–358, 1997.

[15] Fred Glover and Manuel Laguna. General purpose heuristics for integer programming — part II. *Journal of Heuristics*, 3:161–179, 1997.

[16] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publisher, Boston, Dordrecht, London, 1997.

[17] Frederick S. Hillier. Efficient heuristic procedures for integer linear programming with an interior. *Operations Research*, 17(4):600–637, 1969.

[18] Reiner Horst, Panos M. Pardalos, and Nguyen V. Thoai. *Introduction to global optimization*. Kluwer Academic Publishers, Dordrecht, 1995.

[19] Toshihide Ibaraki, Tateaki Ohashi, and Hisashi Mine. A heuristic algorithm for mixed-integer programming problems. *Mathematical Programming Study*, 2:115–136, 1974.

[20] Arne Løkketangen and Fred Glover. Solving zero/one mixed integer programming problems using tabu search. *European Journal of Operational Research*, 106:624–658, 1998.

[21] M. Raghavachari. On connections between zero-one integer programming and concave programming under linear constraints. *Operations Research*, 17:680–684, 1969.

[22] Hoang Tuy. Concave programming under linear constraints. *Soviet Mathematics*, pages 1437–1440, 1964.

[23] Hoang Tuy. Normal conical algorithm for concave minimization over polytopes. *Mathematical Programming*, 51(2):229–245, 1991.

[24] Robert J. Vanderbei. *Linear programming*. Kluwer Academic Publishers, Boston, MA, second edition, 2001. Foundations and extensions.

[25] Philip B. Zwart. Nonlinear programming: counterexamples to two global optimization algorithms. *Operations Research*, 21(6):1260–1266, 1973.

[26] Philip B. Zwart. Global maximization of a convex function with linear inequality constraints. *Operations Research*, 22(3):602–609, 1974.