Solving Zero-One Mixed Integer Programming Problems Using Tabu Search

by

Arne Løkketangen [*]

Fred Glover [#]

20 April 1997

Abstract We describe a tabu search approach for solving general zero-one mixed integer programming problems that exploits the extreme point property of zero-one solutions. Specialized choice rules and aspiration criteria are identified for the problems, expressed as functions of integer infeasibility measures and objective function values. The first-level TS mechanisms are then extended with advanced level strategies and learning. We also look at probabilistic measures in this framework, and examine how the learning tool Target Analysis can be applied to identify better control structures and decision rules. Computational results are reported on a portfolio of multiconstraint knapsack problems.

Our approach is designed to solve thoroughly general 0/1 MIP problems and thus contains no problem domain specific knowledge, yet it obtains solutions for the multiconstraint knapsack problem whose quality rivals, and in some cases surpasses, the best solutions obtained by special purpose methods that have been created to exploit the special structure of these problems.

Keywords Tabu Search, Heuristics, Integer Programming

#
School of Business, CB419
University of Colorado
Boulder, CO 80309
USA

Email: Fred.Glover@colorado.edu

*
Molde College
Britvn. 2
6400 Molde
Norway

Email: Arne.Lokketangen@himolde.no

Introduction

Tabu search (TS) has been applied to solving a variety of zero-one mixed integer programming (MIP) problems with special structures, ranging from scheduling and routing to group technology and probabilistic logic. (See, for example, the volume *Annals of Operations Research* 41 (1993), and the application surveys in Glover and Laguna (1993), and Glover (1996)). Tabu Search has also been applied to solving general zero-one MIP problems by superimposing the TS framework on the "Pivot and Complement" heuristic by Balas and Martin (1980). (See Aboudi and Jörnsten, 1992; Løkketangen, Jörnsten and Storøy, 1993.)

In this paper we show how tabu search also gives a more direct approach for solving general zero-one MIP problems, relying on a standard bounded variable simplex method as a subroutine. Such a direct TS application permits the use of decision criteria that are natural extensions of earlier proposals for exploiting surrogate constraints, as well as others based on weighted evaluation functions. This paper first explores the basic, first-level tabu search mechanisms.

Building on this approach, we then explore the use of probabilistic measures instead of the usual deterministic ones for move selection and tabu tenure. We also explore the notion of abandoning tabu tenure in its deterministic sense almost completely (apart from a one-step move reversal), relying on probabilistic move selection mechanisms to guide our search. We also extend the first-level tabu search mechanisms to include strategic oscillation and diversification schemes, with the goal of driving the search process into other fruitful regions when the initial first-level search has been exhausted. Target analysis is incorporated as a learning approach to identify effective search parameters and choice rules. (For more detail, see Løkketangen and Glover (1995, 1996a). Also see Løkketangen and Woodruff (1996), where the methods described in this paper are successfully used as a sub-problem solver in a progressive hedging algorithm for multi-stage stochastic zero-one MIP problems.)

The general outline of the paper is as follows. In Section 1 we give basic formulations and background. Section 2 gives an overview of the method used to apply tabu search in a general mixed integer setting. Section 3 gives more details about the neighborhood and move evaluations. Memory structures used in our tabu search implementation are provided in Section 4, while Section 5 outlines the aspiration criteria employed. The probabilistic TS measures are described in Section 6, followed by Strategic Oscillation (SO) in Section 7. Section 8 outlines our diversification approach while Section 9 gives an overview of Target Analysis (TA) and how it is applied in this study. The test cases are described in Section 10, followed by computational results in Section 11. Finally, conclusions are presented in Section 12.

1. Formulation and Background

We represent the zero-one MIP problem in the form

Maximize $\quad z = \sum\left(c_j x_j : j \in N\right)$

Subject to

$$\sum\left(A_j x_j : j \in N\right) \le b$$

$$1 \ge x_j \ge 0 \text{ and } x_j \text{ integer} \qquad j \in I \subseteq N$$

$$U_j \ge x_j \ge 0 \qquad j \in C = N - I$$

The maximizing form is chosen because it provides a natural representation for our multiconstraint knapsack test cases. We refer to the objective function value z as the profit value for the MIP problem. The vectors A_j, $j \in N = \{1,...,n\}$ and b are column vectors of constants. The subsets I and C of N respectively constitute the index sets for the integer (zero-one) and continuous variables. We denote the vector consisting of both integer and continuous variables by the symbol x. Solutions (x vectors) that are feasible for the MIP problem will be called *MIP feasible*, and solutions that are feasible for the corresponding linear programming relaxation (dropping the integer requirement for the zero-one variables) will be *LP feasible.*

We allow upper bounds U_j for the continuous variables to be infinite (i.e., redundant) and stipulate that $U_j = 1$ for $j \in I$, which gives $x \le U = (U_1,...,U_n)$. By this convention, the preceding formulation suggests the use of the bounded variable simplex method for solving the LP relaxation, and in general as a vehicle for moving from one extreme point to another. Thus, feasible pivot moves leading to adjacent extreme points include those that change the value of a non basic variable from one bound (lower or upper) to the opposite bound, and it is unnecessary to refer to slack variables for the upper bound inequalities.

It is well known that an optimal solution for the zero-one MIP problem may be found at an extreme point of the LP feasible set, and special approaches integrating both cutting plane and search processes have been proposed to exploit this fact (Cabot and Hurter, 1968; Glover, 1968). We consider here a tabu search method for exploiting this extreme point property that incorporates more powerful search processes.

2. Overview of the Method

In broad outline, our MIP solution approach may be expressed in the following form. Let x^* denote the best MIP feasible solution found and let z^* denote its objective function value. (To begin, when x^* is unknown, z^* may be assigned a value of negative infinity.)

TS - MIP in Overview

Step 0. Begin by solving the LP relaxation of the zero-one MIP problem to obtain an optimal LP basic (extreme point) solution.

Step 1. From a current LP feasible basic solution, consider the feasible pivot moves that lead to adjacent basic feasible solutions.

(a) Isolate and examine a candidate subset of these feasible pivot moves.

(b) If a candidate move creates an MIP feasible solution x whose associated z value yields $z > z^*$, record x as the new x^* and update z^*.

Step 2. Select a pivot move that has the highest evaluation from those in the candidate set, applying tabu search rules to exclude or penalize moves based on their tabu status.

Step 3. Execute the selected pivot, updating the associated tabu search memory and guidance structures, and Return to Step 1.

Three elements are required to transform this overview procedure into a method that is fully explicit: (1) the candidate list strategy for screening moves to examine; (2)

the function for evaluating the moves; (3) the determination of rules (and associated memory structures) that define tabu status. We consider these elements in the following sections.

3. Notation and Fundamentals

3.1 Neighborhood Structure

Let $x(0)$ denote a current basic extreme point solution, let $\{x_j : j \in NB\}$ denote the current set of non basic variables and let $\{x_j : j \in B\}$ denote the current set of basic variables ($B = N - NB$). The extreme points adjacent to $x(0)$ have the form

$$x(h) = x(0) - D_h\theta_h \qquad \text{for } h \in NB$$

where D_h is a vector associated with the non basic variable x_h, and θ_h is the change in the value of x_h that moves the current solution from $x(0)$ to $x(h)$ along their connecting edge. The standard LP basis representation identifies the entries D_{hj} of D_h associated with the current basic variables x_j . The entries of D_h for current non basic variables are zero, except for x_h . We choose the sign convention for entries of D_h that yields a coefficient D_{hh} for x_h of -1 if x_h is currently at its lower bound, and of 1 if x_h is currently at its upper bound. Hence x_h respectively receives the value θ_h or $U_h - \theta_h$ at the extreme point $x(h)$. The value θ_h is always non negative, and is strictly positive except under degeneracy.

One useful view of the solution space is shown in figure 1, where the axes denote the total amount of *integer infeasibility* (see 3.3), and the associated *Objective Function Value*. The figure also shows typical relative positions of various important solution points, such as **LP OPT**, an optimal solution to the LP relaxation (and our starting point), **MIP OPT**, an optimal feasible solution (and the point we want to find), x*, the best feasible solution found at a given time in the search, and x(0), some intermediate solution point (the "current solution" and basis for our neighborhood).

3.2 Candidate List Strategy

The simplest candidate list strategy is to examine the full neighborhood of available moves, so that the candidate neighborhood, *NB**, will be equal to *NB*. This is appropriate for problems that do not involve a large number of variables (including slack variables).

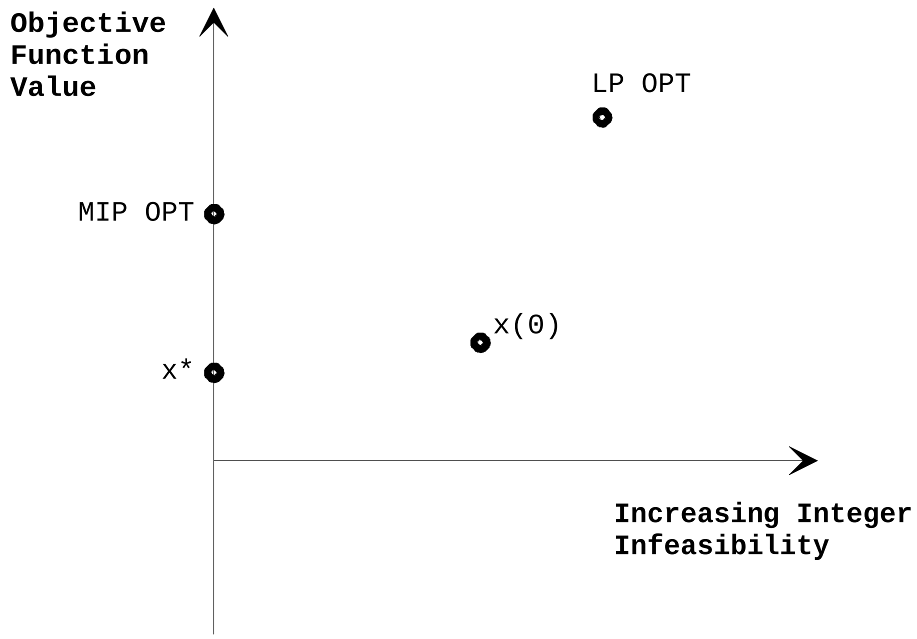

Figure 1. *One view of the solution space*

For problems of moderately large size, *NB* will be too large to permit all extreme points adjacent to *x*(0) to be evaluated conveniently, and a more refined candidate list strategy will typically be necessary to examine only a subset *NB** of *NB*. Characteristics of such strategies are described in Glover, Taillard and de Werra (1993) and in Glover (1995a).

3.3 Evaluation Rules

Given the determination of a current candidate subset *NB** of *NB*, we seek a best (non-tabu) element of *NB** to give the next extreme point. To guide us in choosing this best move, we use a combination of the change in objective function value resulting from applying a given move, and the corresponding change in integer infeasibility (see below). We look at two very different ways of combining these values, in one case incorporating a choice rule mechanism derived from surrogate constraint strategies and in the other case creating a weighted sum of the two measures.

Integer Infeasibility. We first require a measure of integer infeasibility, i.e., the degree to which an extreme point solution fails to satisfy the requirement that the variables x_j, $j \in I$ receive integer values.

As h ranges over $h \in NB^*$ and also $h = 0$, let $z(h)$ denote the profit of the corresponding solution $x(h)$, and let $x_j(h)$ denote the value assigned to x_j in this solution. Also let $near(x_j(h))$ denote the integer nearest to the value $x_j(h)$.

Integer Infeasibility Measure. To begin, we create a measure $u(h)$ of integer infeasibility for the solution $x(h)$ by the following simple rule. Define

$$u_j(h) = |x_j(h) - near(x_j(h))|^p \ , \ j \in I$$

In our experimentation we report the effects of letting the exponent p range from .5 to 2. Then $u(h)$ is defined by

$$u(h) = \sum (u_j(h) : \ j \in I \ )$$

Clearly $u(h) = 0$ if $x(h)$ is integer feasible, and $u(h) > 0$ otherwise. Restricting consideration to $h \in NB$, define

$$\Delta z(h) = z(h) - z(0)$$

$$\Delta u(h) = u(0) - u(h)$$

Then $\Delta z(h) > 0$ indicates that the profit $z(h)$ represents an improvement relative to the profit $z(0)$, and $\Delta u(h) > 0$ indicates that the integer infeasibility $u(h)$ represents an improvement relative to $u(0)$. (Hence $\Delta z(h) \leq 0$ for $h \in NB$ when $x(0)$ is an optimal LP solution.) We note it is not necessary to execute a pivot to identify $x(h)$ or the values $u(h)$ and $z(h)$, since only the vector D_h and the solution $x(0)$ is required to make

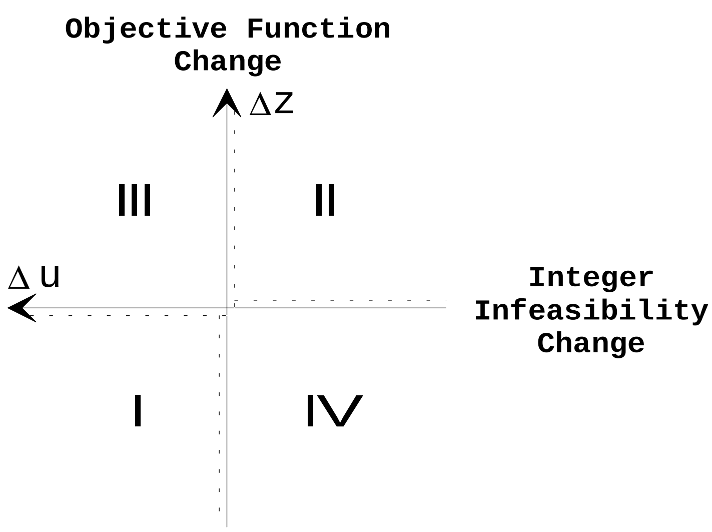

Figure 2. *Classification of move types*

this determination.

Classification of Move Types. There are four cases to consider in identifying a preferred extreme point $x(h)$, $h \in NB^*$, adjacent to $x(0)$. These four cases, or *Move Types*, generally classify the moves according to whether the changes in objective function value and integer infeasibility measure are preferable or not. We refer to the tabu status of extreme points with the understanding that this status is determined as indicated in Section 5.

We let H1, H2, H3, H4 denote the set of moves belonging to each move type, and let *h1*, *h2*, *h3*, *h4* be any move from within the corresponding group. Figure 2 shows how this relates to the view of the solution space shown in figure 1, where the horizontal and vertical axis in the present instance respectively identify changes in integer infeasibility and in objective function (or "profit") value. The dotted lines indicate the exclusion of the axis from the corresponding move type, so that the four regions are disjoint.

Move Type I (Decreasing integer infeasibility, decreasing profit). Let

$$H1 = \{h \in NB^* : \Delta z(h) < 0, \Delta u(h) > 0 \quad \text{and } x(h) \text{ is not tabu}\}$$

Move Type II (Increasing profit, increasing integer infeasibility). Let

$$H2 = \{h \in NB^* : \Delta z(h) > 0, \Delta u(h) < 0 \quad \text{and } x(h) \text{ is not tabu}\}$$

Move Type III (Nondecreasing profit and nonincreasing integer infeasibility). Let

$$H3 = \{h \in NB^* : \Delta z(h) \geq 0, \Delta u(h) \geq 0 \quad \text{and } x(h) \text{ is not tabu}\}$$

Move Type IV (Decreasing profit and nondecreasing integer infeasibility). Let

$$\begin{aligned} H4 = \{h \in NB^* : &\Delta z(h) < 0, \Delta u(h) < 0 \quad \text{or} \\ &\Delta z(h) = 0, \Delta u(h) < 0 \quad \text{or} \\ &\Delta z(h) < 0, \Delta u(h) = 0 \quad \text{and } x(h) \text{ is not tabu}\} \end{aligned}$$

Move Evaluation and Choice Rules. We employ four different ways to evaluate the moves, using both $\Delta z(h)$ and $\Delta u(h)$, and move type classifications:

1. Weighted sum
2. Ratio test
3. As 1, but sorted within each of the move type groups
4. Ratio test, but move type I before II

Note the importance in this context of checking candidate solutions to see if they qualify as a new best integer feasible solution, rather than checking only the solution chosen as the next extreme point, since the choice rule may not move to an integer feasible solution simply because it is available. The move evaluations are evaluated specifically as follows.

Weighted sum move evaluation. In this approach we form a weighted sum of $\Delta z(h)$ and $\Delta u(h)$, and choose the first non-tabu move with the highest move evaluation value E(h), defined by

$$E(h) = w_1 \Delta z(h) + w_2 \Delta u(h)$$

Good values for the weights are based on empirical determination.

Ratio test choice rules. To supplement the formal definition of this set of rules, an illustration of a typical move of each type is provided in figure 3.

Move Type I. Identify the preferred choice over H1 to be given by

$$h1 = \text{Argmax}\ (\Delta z(h) / \Delta u(h) : h \in \text{H1})$$

If more than one h qualifies for $h1$, choose one that maximizes $u(h)$. In other words, choose the move that minimizes the angle α depicted in figure 3, and choose the largest vector when vectors tie in producing this minimum angle.

Move Type II. The preferred choice over H2 is then given by

$$h2 = \text{Argmax}\ (\Delta u(h) / \Delta z(h) : h \in \text{H2})$$

If more than one h qualifies for $h2$, choose one that maximizes $z(h)$. In other words, choose the move that minimizes the angle β depicted in figure 3, and choose the largest vector when ties occur.

Move Type III. Define the preferred choice over H3 to be given by

$$h3 = \text{Argmax}\ (\Delta z(h)\ \Delta u(h) : h \in \text{H3})$$

If more than one h qualifies for $h3$, choose one that maximizes Min ($\Delta z(h)$, $\Delta u(h)$). As a special case, if $\Delta u(h) = 0$, then take Max(Max($\Delta z(h)$)), and similarly for $\Delta z(h) = 0$. This means that preference within the group is given to moves that give large improvements, but in a balanced way. The special case takes care of points that fall on the axis in figure 3.

Move Type IV. Define the preferred choice over H4 to be given by

$$h4 = \text{Argmin}\ (\Delta z(h)\ \Delta u(h) : h \in \text{H4})$$

If more than one h qualifies for *h4*, choose one that minimizes Max ($\Delta z(h)$, $\Delta u(h)$). This means that preference is given to moves that extend as little as possible in the unfavorable direction, but in a balanced way.

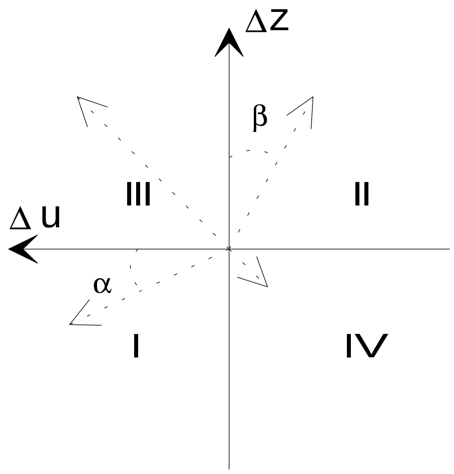

Figure 3. *Ranking of move types*

The cases with move types I and II embody a ratio test choice rule patterned after those used in surrogate constraint heuristics (Glover (1977), Løkketangen and Glover (1996b)), and the cases with move types III and IV represent natural variations of this rule.

It remains to decide which of the cases 1-4 gives the overall "best choice" for h when more than one of these cases occurs simultaneously. We denote this best h value by h^*. Case 3 is the most desirable case, provided $\Delta z(h)$ and $\Delta u(h)$ are both positive, and in this instance takes priority over all others, leading to the choice h^* = *h3*. At the opposite extreme, Case 4 is the least desirable and we choose h^* = *h4* only if no other cases exist. An intelligent candidate list strategy will undertake to enlarge the subset NB* of NB currently examined (up to a limit) when Case 4 is the only one available.

Choosing Between Cases 1 and 2. Deciding between Cases 1 and 2, and the borderline instances of Case 3 where profit and/or integer infeasibility stay unchanged, requires more subtle treatment. If Case 3 exists, we elect to set h^* = *h3* as long as at least one of $\Delta z(h)$ and $\Delta u(h)$ is positive. However, we handle the outcome $\Delta z(h) = \Delta u(h) = 0$ (which may occur under degeneracy, for example) by introducing a probability acceptance test, which chooses h^* = *h3* with a specified probability (such as 1/3 or 1/10). If the acceptance test fails, and if Case 1 or 2 exists, we choose h^* = *h1* or *h2*. Relying on these provisions (which are subject to special exceptions noted later), we need only determine how to assign priority between Cases 1 and 2 when both exist.

When both H1 and H2 are non empty, we seek a way to make the measures of Case 1 and Case 2 comparable in order to determine which of *h1* and *h2* is preferable to be selected as *h**. We do this by normalization as follows. Let H denote the union of H1 and H2.

Normalization 1. Define an (aggregate) integer infeasibility change $F(w,q)$, as a function of a multiplier w and an exponent q by $F(w,q) = w \sum(|\Delta u(h)|^q : h \in \mathrm{H})$. Then, treating an objective function change as a simple sum of component changes, we define the ratios R, R1(h) and R2(h) by

$$\mathrm{R} = \sum(|\Delta z(h)| : h \in \mathrm{H}) \,/\, F(w,q)$$

$$\mathrm{R1(h)} = (\Delta z(h) \,/\, \Delta u(h)) \,/\, \mathrm{R}$$

$$\mathrm{R2(h)} = (\Delta u(h) \,/\, \Delta z(h)) * \mathrm{R}$$

Then $h^* = \mathrm{Argmax}\,\{\mathrm{R1}(h1), \mathrm{R2}(h2)\}$.

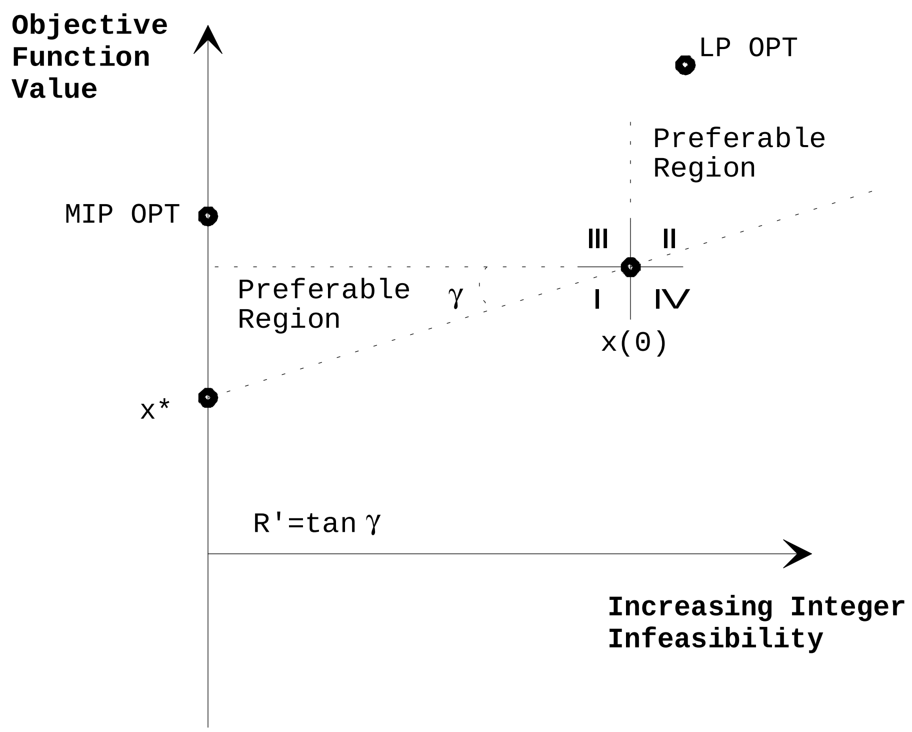

Figure 4. *Normalization 2*

With reference to figure 3, this means that we are comparing the angles α and β, again choosing the smallest. Values of the multiplier *w* other than 1 cause the angles α and β to shift, while values of the exponent *q* other than 1 causes these angles to curve, i.e. to vary at different distances from the origin. We will see that the determination of the integer infeasibility change *F*(*w*,*q*) as a function of *w* and *q* has an interesting and significant effect on the behavior of our choice rules. Note that the exponent *p* is part of the evaluation of Δ*u*(*h*), and values for this parameter other than 1 also makes the angles α and β curve. (A similar parametrization of the objective function change is possible, but was not found necessary to obtain high quality choices.)

Normalization 2. Let *z** denote the value of *z* for the best MIP feasible solution found, as earlier, or let *z** denote an aspired target for this value. If $z^* \geq z(0) + \varepsilon$, for some small value of *ε*, choose *h** = *h3*. (This rule may also be used to override the choice of *h** by Normalization 1.) But if $z^* < z(0) + \varepsilon$, let

$$R' = (z^* - z(0)) / u(0)$$

Then determine *h** as in Normalization 1, by replacing R with R´. A graphical interpretation of R´ is shown in figure 4.

Since Normalizations 1 and 2 may given different choices for *h** (from the two alternatives *h1* and *h2*), experimentation must be relied upon to determine the conditions under which one normalization is preferable to the other. However, we also examine other implications of these normalizations in the following.

Special Interpretation for Normalization 2. The second normalization, which replaces R by R´, yields a suggestive criterion to measure the merit of the choice given by R1(*h1*) and R2(*h2*) (where *h** = *h1* or *h2* according to which of these ratios is larger). In particular, when a succession of choices occurs with *h** = *h1* and R1(*h1*) > −1 at each step, a new MIP feasible solution must ultimately result that is better than the current best known. When a choice occurs with *h** = *h2* and R2(*h2*) > −1, the value of Δ*z*(*h*) / Δ*u*(*h*) required to yield R1(*h1*) > −1 on the next step becomes smaller, hence conceivably easier to obtain.

From these observations, when Max(R1(*h1*), R2(*h2*)) is somewhat less than −1, this can be a signal that the current choice is not very good, and hence a candidate list strategy should enlarge the set NB* of elements examined in the hope of doing better.

Also, when R1(h1) > −1, there may be merit in selecting $h^* = h1$ even if R2($h2$) is slightly larger than R1($h1$).

Weighted sum move evaluation, sorted by move type. This is a combination of the two approaches above. The moves are evaluated as a weighted sum, but first sorted according to move type, and then according to the move evaluation within each move type group. The ranking between the different move type groups is as for the ratio test: III, I & II, IV. (This means that moves belonging to move types I and II are sorted together.)

Ratio test, move type I before II. This test is intended to drive the search more strongly to achieve integer feasibility than the basic ratio test. In this, we always choose from H1 before H2. The rationale is that our two measures, $\Delta z(h)$ and $\Delta u(h)$, are not necessarily equivalent, or symmetric, and for some types of problems it is more important to focus on finding a feasible solution, than on finding high objective function values (at least until a feasible solution is found).

4. Tabu Status and Memory Structures

To establish tabu status, we use a straightforward approach that creates two tabu records, tabu_start(j) and tabu_frequency(j) for each variable x_j, $j \in N$. Tabu_start(j) is used to record recency information, and tabu_frequency(j) to record frequency related information (see Section 8). The record tabu_start(j) begins as a large negative number and then, whenever x_j becomes non basic, is assigned the value

tabu_start(j) = current_iteration.

By convention, we define an iteration to occur each time a new extreme point solution is visited (causing current_iteration to be incremented by 1), hence the first iteration occurs after obtaining an initial $x(0)$ (as by solving the original LP problem). The assignment tabu_start(j) = current_iteration is also made on any iteration where a non basic variable is changed from one of its bounds to the opposite bound since such a step likewise causes the variable to “become non basic”). Once this assignment is made, x_j is tabu and is prevented from changing its current non basic value, and hence prevented from being used to generate a new extreme point, for a chosen number of iterations we will denote by t. Thus x_j remains tabu as long as current_iteration does not grow larger than tabu_start(j) + t, and the tabu status of x_j can be checked by testing whether

$$\text{tabu\_start}(j) \geq \text{current\_iteration} - t$$

In a simple dynamic approach we choose t to vary, either systematically or randomly, within a small range about a preferred value.

5. Aspiration Criteria

As the tabu status given to moves in the current neighborhood can sometimes be too restrictive, in that otherwise good moves are forbidden, we employ aspiration criteria to override the tabu status for these moves. The simplest, and most frequently used, aspiration criterion is to accept a tabu move if it leads to a new incumbent, i.e., to a new solution better than the current best.

We have tried three different aspiration criteria for our moves, detailed below. Their relative merits are reported in Section 11.

5.1 Aspiration by Integer Infeasibility Levels

Our first proposed aspiration criterion allows tabu status to be overridden by making reference to measures of integer infeasibility, using the initial definition of integer infeasibility of Section 3.3. Let x' denote the x vector for the initial LP solution and let u' denote its integer infeasibility measure (hence $u' = u(0)$ when $x' = x(0)$). We assume $u' > 0$, or else the LP solution solves the zero-one MIP problem.

We create an aspiration vector ASPIRE(L) for integer levels L = 0 to LMAX (e.g., choosing LMAX = 1000), where each level L corresponds to a degree of integer infeasibility. (The level L = 0 corresponds to no infeasibility.) The idea of such a vector is to let ASPIRE(L) equal the z value for the best solution found that has an infeasibility level of L. Then if a currently considered solution x also has an infeasibility level of L, and if its profit z is greater than ASPIRE(L), we deem this solution admissible to visit even if it is tabu.

In order to translate the normal infeasibility measure u into different levels of integer infeasibility measured by L, we let the infeasibility u' of the initial LP solution correspond to a relatively high chosen level L´, e.g., L´ = .8 LMAX, thus creating a conversion factor F = L´/u'. Then for any solution x subsequently generated, with an infeasibility measure of u, we identify the corresponding integer infeasibility level for this solution to be given by L = ⌈*u*F⌉, where the notation ⌈v⌉ indicates the smallest integer ≥ *v*. For the special case where ⌈*u*F⌉ > LMAX, set L = LMAX.

The aspiration value ASPIRE(L) is maintained as the *z* value for the best solution found, with an infeasibility level of L, as follows.

1. Initially, set ASPIRE(L) = –BIG for L = 0 to LMAX.
2. Upon obtaining the optimum LP solution, set ASPIRE(L´) = *z*´, and identify the conversion factor F = L´/*u*´.
3. If a candidate solution *x*(*h*) is tabu, identify its infeasibility level L = Min (⌈F(*u*(*h*)⌉, LMAX). Then the aspiration criterion for *x*(*h*) is satisfied, and *x* is treated as admissible (not tabu) if *z*(*h*) > ASPIRE(L).
4. Let *x*(0) denote the current solution visited, and identify L = Min (⌈F(*u*(0)⌉, LMAX). If *z*(0) > ASPIRE(L) set ASPIRE(L) = *z*(0).

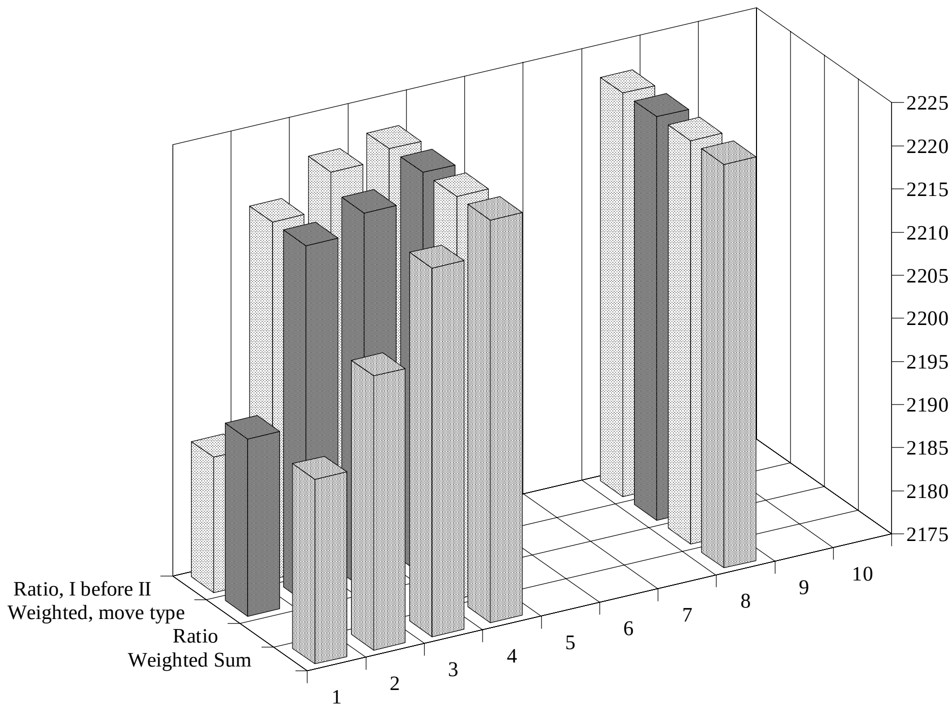

Figure 5. *Integer Infeasibility Aspiration Levels for the Fleischer Problem*

In the special case L = 0, recall that we may identify a new best solution by the choice rules of the procedure without actually moving to this solution. Hence ASPIRE(0), which refers only to solutions actually visited, may be smaller than the value z^* for the best MIP feasible solution found. As an example, referring to one of the problems in our test set (the Fleischer problem) the aspiration level values obtained for the different move evaluation functions are shown in figure 5, with 10 aspiration levels. Also note that the ratio test evaluation function for this case failed to find a feasible solution, while the optimum value was found by the other three approaches.

5.2 Aspiration by Objective Function Value Levels

An alternate way to exploit the idea of aspiration by levels, as outlined in Section 5.1, is to interchange the notions of integer feasibility and objective function value, and to introduce an aspiration by **objective function value levels**. The necessary mechanisms to achieve this are analogous to the ones used for aspiration by integer infeasibility levels. We use objective function value bounds of LP* and 0, for our multidimensional knapsack test cases. A difference between this approach and the

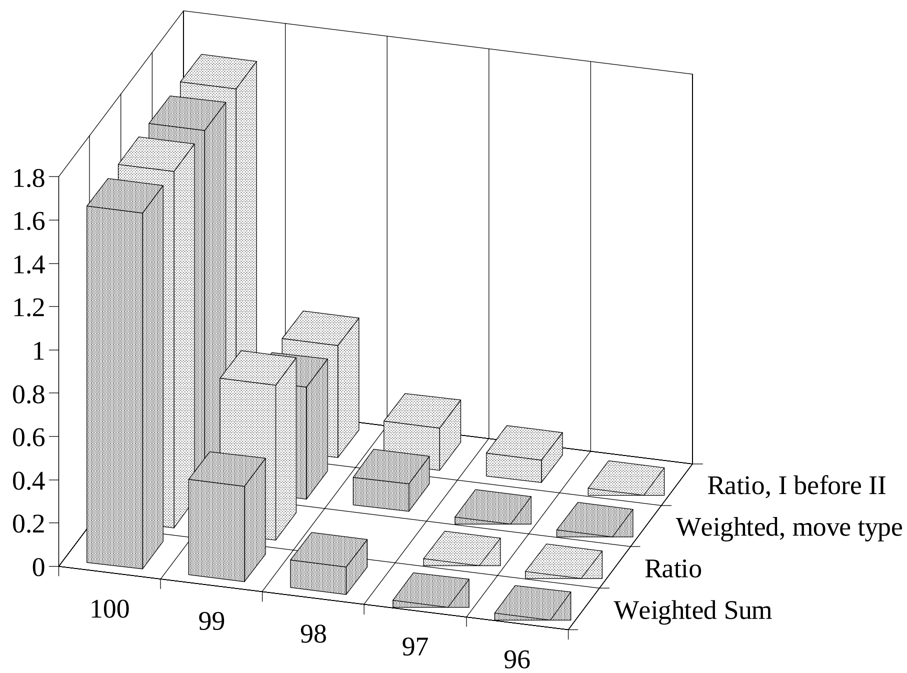

Figure 6. *Objective Function Value Aspiration Levels for the Fleischer Problem*

preceding is that more levels are usually needed, in order to have enough levels near the optimum value. In addition, there is no need to aspire for objective function values smaller than that of the incumbent. Figure 6 shows the aspiration level values obtained for the Fleischer problem for the different move evaluation functions, with 1000 aspiration levels. Only the top 5 are shown, since the rest are never reached.

5.3 Aspiration by new best detected

As mentioned in Section 2, a new best integer feasible solution may be encountered during a move evaluation, without subsequently selecting the move leading to it. This is due to the process for evaluating and ranking the moves. It can sometimes be advantageous to visit such a new best solution, so we pick the move leading to a new best solution as our choice for the current iteration, even though this move may have a lower rank than other acceptable moves. To avoid accepting moves that lead to a poor integer feasible solution, a threshold value that the objective function value has to surpass can be introduced, as derived from a trial run.

6. Probabilistic Tabu Search

As sometimes noted in the TS literature, “controlled randomization” (which uses the biases of probability) may be viewed as a substitute for memory—when we are ignorant of how memory should advantageously be used. But just as there are multiple kinds of memory that can supplement each other, probabilities may find their best uses in various supplementary roles. The ideas that are tested here originate in part from Glover (1989), and are adapted to our specific problem area.

For clarification, since more than one interpretation is possible, what is generally regarded as *Probabilistic TS* is usually applied to the move acceptance function, and is intended here to have the following design:

A. Create move evaluations (for a current candidate list of moves examined) that include reference to tabu status and other relevant biases from TS strategies—using penalties and inducements to modify an ordinary objective function evaluation.

B. Map these evaluations into positive weights, to obtain probabilities by dividing by the sum of weights. The highest evaluations receive weights that disproportionately favor their selection.

Among conjectures for why these probabilistic measures may work better than the deterministic analog, one may guess that move evaluations have a certain “noise level” that causes them to be imperfect—so that a “best evaluation” may not correspond to a “best move”. Yet the imperfection is not complete, or else there would be no need to consider evaluations at all (except perhaps from a thoroughly local standpoint — noting the use of memory takes the evaluations beyond such a local context). The issue then is to find a way to assign probabilities that somehow compensates for the noise level.

The application of Probabilistic TS as outlined above can be guaranteed to yield an optimal solution under certain easily controlled conditions, if allowed to run for an infinite number of iterations (see Glover 1989). This means that one may use Probabilistic TS as an escape hatch to hedge against persistent wrong moves in the absence of reliable knowledge to guide the search.

One may also view controlled randomization as a means for obtaining diversity without reliance on memory. In this respect it represents a gain in efficiency by avoiding the overhead otherwise incurred in the use of long term memory. However, it also implies a loss of efficiency as potentially unproductive wanderings and duplications may occur, that a more systematic approach would seek to eliminate.

It should be observed that there are significant differences between the ways probabilities are used in Probabilistic TS compared to the ways they are used in Simulated Annealing (SA). Notably, SA samples the neighborhood (either randomly or systematically), and accepts any improving move encountered, and non-improving moves are accepted with a decreasing probability as the search progresses. In contrast, PTS collects the evaluated moves in a candidate list (sorted by some measure of the goodness of the moves), and uses a biased probability to select from this list, with the first move on the list (i.e. the move that is considered best according to some criterion) having the greatest chance for being selected. This selection is done after the usual tabu restrictions and aspiration criteria have been applied. PTS is thus more aggressive, and gives more guidance to the search process. For a general description of SA, see Dowsland (1993), while Connolly (1994) describes the use of SA as a general search tool component for pure ILP problems.

Hart and Shogan (1987) describes the use of probabilistic measures for move selection in greedy heuristics. Here the move is selected with uniform probability either among the *n* best moves, or among the moves better than some threshold (compared to the best move), but only considering improving moves. We tried the ideas of Hart and Shogan in our PTS framework (thus accepting non-improving moves), both with normal tabu tenure and without (only with the rejection of the immediate move reversal), but with inferior results compared to the methods reported in Section 11.2, which gives a probabilistic bias towards the presumably best moves.

Probabilistic measures may also be applied to the tabu tenure, as outlined in Section 6.3, and in the following we will state explicitly which probabilistic measure we mean, when necessary.

6.1 Probabilistic move acceptance

To be able to apply the general probabilistic move acceptance approach as outlined in the previous section, we seek a move evaluation function that in some way reflects the *true merit* of all the moves in the candidate list.

However, in the present setting we do not have a comparable measure for all the different move types identified in Section 3. Also, the move evaluation within each move type is quite noisy with respect to the overall goal of maximizing the objective function value, as it also contains a component designed to reduce the amount of integer infeasibility, and the two move evaluation components are combined rather dramatically by multiplication or division. It is therefore difficult to assign a good comparable numerical measure which reflects the true merit of all the moves. However, as is evidenced by the computational results (see Section 11), there is good reason to believe that the *relative* ranking of the moves in the candidate list is quite good, as is the relative ranking of the move type groups, and that a move should be selected among the first few in the candidate list, if possible.

We therefore propose to introduce controlled randomization in the move selection process by exponentially decreasing the move acceptance probability when traversing the candidate list, thus relying solely on the individual move rankings, and not using the actual move evaluations.

6.2 Exponentially decreasing move acceptance

The basis for this approach is that we regard the relative ranking of the moves on the candidate list to be a good approximation to their *true merit*.

The general outline of the method is as follows. Let p be the probability threshold for acceptance of a move, and let r be a randomly generated number (both in the range 0 -1). Only the basic move selection core of the method is shown.

Step 1. Generate the candidate list in the usual way.

Step 2. Take the first (potentially best) move from the candidate list.

Step 3. Subject the move to the following sequence of tests:

- Accept if the aspiration criterion is satisfied.
- Reject if Tabu, and place the move at the end of the candidate list, removing its tabu status for the current iteration.
- Apply degenerate move rejection.
- Generate r. If $r > p$, reject.
  If $r \leq p$, accept move, exit.

Step 4. (In case of rejection.) Select the next move on the candidate list.
Go to Step 3.

If the last move on the list is examined and rejected, accept the first move on the list (or any move on the list at random).

The method is intriguing because of its simplicity, and can easily be implemented in most deterministic TS frameworks.

The value of p should not be too small, as we would usually like to select one of the top few moves. Testing will disclose good values for p, but as an example, consider $p = 1/3$. The probability of selecting each of the first d moves is then (disregarding aspiration criteria, tabu status, etc.):

$$1/3,\ 2/9,\ 4/27,\ 8/81,\ ...,\ 2^{d-1}/3^{d}\ .$$

The probability of not choosing one of the first d moves is $2^d/3^d$, so $p = 1/3$ gives a very high probability of picking one of the top moves: about .87 for picking one of the top 5, and about .98 for picking one of the top 10.

The effective value of *p* can also be viewed as a function of the quality of the move evaluation function. The better the move evaluation function, the higher the expected value of *p* for which the best solutions are obtained.

6.3 Probabilistic move acceptance and strategic oscillation

The probabilistic move acceptance scheme outlined above is easily combined with the strategic oscillation schemes outlined in Section 7. We illustrate this by outlining the version called *strategic oscillation by parametric evaluation*. In this approach, the relative attractiveness of type I and II moves are modified in a strategic way in order to alternate between giving higher emphasis to integer feasibility and giving higher emphasis to the objective function value. The ranking of the move type groups remains the same, i.e. III, I & II, IV.

The conjecture is that combining strategic oscillation with probabilistic move acceptance will have a symbiotic effect on the search efficiency, since the SO scheme gives an additional (alternating) emphasis to the move ranking, while the controlled randomization gives an extra degree of diversification.

6.4 Probabilistic tabu tenure

It is usually advised in the literature to use dynamic tabu lists, i.e. tabu lists that vary over time, usually within fixed limits. The actual timing of the change of the tabu list length, and the associated new length is often controlled randomly, although fixed patterns have also been applied quite successfully. In addition, variations in size and composition based on logical relationships have proved effective.

The purpose of having varying tabu list lengths, or equivalently, varying tabu tenure, is not simply to avoid cycling of the search, but also to introduce a form of vigor into the search that accommodates varying *widths* of local minima. (See Ryan, 1994).

A natural extension is to convert the tabu tenure for each variable into a probabilistic measure, causing the probability of retaining tabu status for a given tabu element to diminish over time from the initial tabu status assignment (Glover, 1989).

We assume that the expected probabilistic tabu tenure for a variable should have approximately the same value that would be suitable in the corresponding deterministic variant, so that the total amount of imposed "tabu influence" will be the same.

This probabilistic tabu tenure can be assigned in two ways, by assigning an individual tabu tenure when the element receives its tabu status, or by doing the actual checking for tabu status using a diminishing probabilistic measure. These two measures should give the same results.

We elected to implement the probabilistic tabu tenure in the checking phase (when traversing the candidate list to find a move), assigning tabu status to variable *j* with probability *p(j,t)*, where *t* is the time since the start of this variable's tabu tenure. The probability function *p(j,t)* was chosen to be linear in *t*, and decreasing from 1 to 0 over the time *T*. The time *T* was chosen to give the same overall expected tabu tenure as in the deterministic tabu tenure case. Let TL_D be this first-level deterministic tabu list length, and TL_P (=*T*) be the associated probabilistic one. Also note that the deterministic tabu list length was varied randomly between TL_D and $2*TL_D$. As can be seen from Figure 7, this gives a value for $TL_P = 3*TL_D$. (The amount of "tabu influence" for the two approaches is indicated by the areas of the dotted rectangle and the triangle.)

7. Strategic Oscillation

Strategic Oscillation (SO) is one of the basic diversification techniques for TS. The idea is to drive the search toward, away from, or through selected boundaries in an oscillating manner. This can be accomplished by altering the choice rules governing the move selection, or by altering the move evaluation function, e.g. by applying

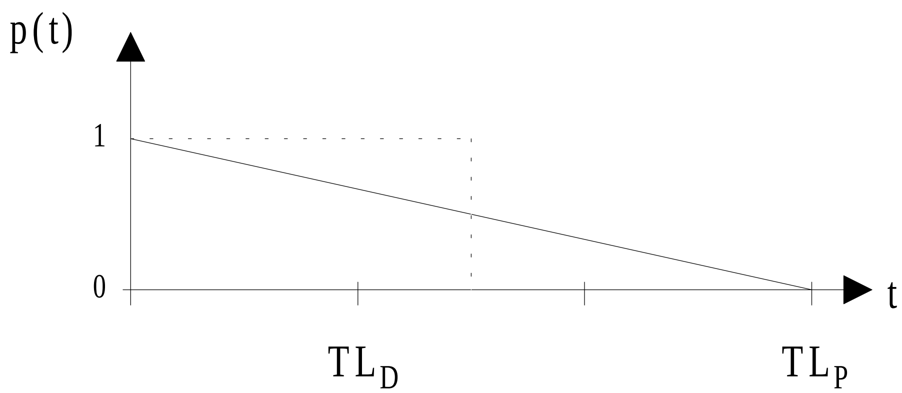

Fig 7. Probabilistic Tabu Tenure

appropriate incentives and penalties. The basic oscillating frequency of SO may be medium or long range depending on the specific problem, and the nature of oscillations.

The move evaluation function we use for our zero-one MIP problems is composed of two rather different measures, the change in objective function value, $\Delta z(h)$, and the change in the amount of integer infeasibility, $\Delta u(h)$. The relative emphasis between these two measures is fixed in the basic move evaluation, apart from the axis skew factor, w, which is used to alter the emphasis between $\Delta z(h)$ and $\Delta u(h)$ for type I and II moves, but which is fixed for the duration of the search.

Two SO schemes are proposed. One is by parameterizing the emphasis between $\Delta z(h)$ and $\Delta u(h)$ directly in the move evaluation function, and the other is by altering the choice rules for move selection. The value $u(h) = 0$ in both cases defines the boundary that the SO drives the search toward or away from. This is represented by the vertical axis in figure 1. The other turn-around point is generally defined either by the lapse of a certain amount of time (iterations), the absence of improving moves in that general direction, or both. Other criteria can also be applied. The SO in this setting is depicted in figure 8. Note that type III moves are always considered good, and type IV moves are always considered bad (in a relative sense).

7.1 Strategic Oscillation by Parametric Evaluation

Our original design has a fixed emphasis on the relative merit of the two components of the move evaluation function, possibly modified by w. Tabu search generally advocates placing a different emphasis on different variables, and varying this emphasis over time. This is accomplished in our setting by linking the component values through a parameter p in the following way. Define

$$\Delta(z(h):p) = p\Delta z(h)$$

$$\Delta(u(h):p) = (2-p)\Delta u(h)$$

By letting p vary between 0 and 2, one can obtain a range of possible emphases between the two move evaluation function components. As can also be seen, setting $p = 1$ reverts to the basic move evaluation scheme.

We can now define the following priority classes:

Integer Priority: Let $0 < p < 1$

Cost Priority: Let $1 < p < 2$

(Roughly) Balanced Priority: Let $p = 1$

Note that the parameter *w* in this setting can be viewed as a fixed bias for the SO. The best values for *p*, as well as the time spent in each phase, must be based on empirical tests.

Strategic oscillation based on parametric evaluation applies in this setting by invoking several patterns of oscillation. The general approach for creating such a pattern is as follows. For simplicity, we let $\Delta u(h)$ and $\Delta z(h)$ below represent the terms $\Delta(u(h):p)$ and $\Delta(z(h):p)$.

Parametric Strategic Oscillation

Step 0: Choose *p* according to one of the priority classes.

Step 1: Apply the normal choice rules as outlined in Section 1 for a selected number of iterations or until a specified condition is met.

Step 2: Identify a new priority class with associated choice of *p*, and return to step 1.

The actual value of *p* used within each priority class may also be allowed to vary,

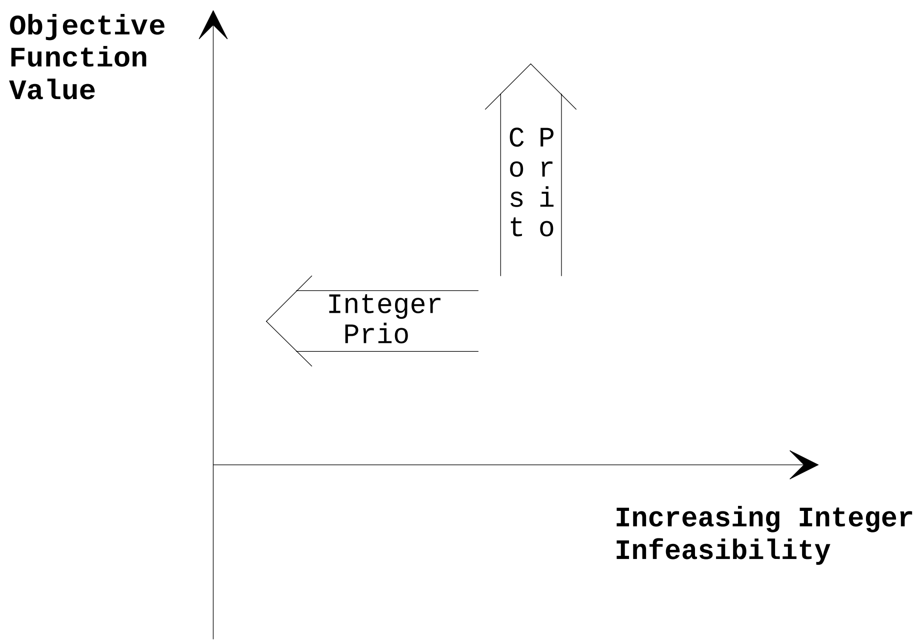

Fig 8. Strategic Oscillation

thus giving more or less emphasis to a specific move evaluation component over time. This can be done in a predetermined manner, or can be made to depend on the search progress.

The following is a special variant of the foregoing approach that is conjectured to be one of the better alternatives. The variant begins from the optimal LP solution. As earlier, z^* denotes a target for an optimal z value until an integer feasible solution is found, and then denotes the best z value found for such a solution. The value ε represents a small positive number.

Alternating Integer and Cost Priority

Step 1: *Integer Priority Phase*. Choose p by the Integer Priority criterion. Continue for a specified minimum number of iterations and then transfer to the Cost Priority Phase when one of the following occurs:

1a) $z(0) < z^* - \varepsilon$

1b) $\Delta u(h) < 0$ for all non tabu $h \in$ NB

Step 2: *Cost Priority Phase*. Choose p by the Cost Priority criterion. Continue for a specified minimum number of iterations, and then transfer to the Integer Priority Phase when both of the following occur:

2a) x(0) is not integer feasible

2b) $\Delta z(h) < 0$ for all non tabu $h \in$ NB

Step 3: Alternate between the two preceding phases until reaching a desired cutoff point.

In other words, if the search is in the integer priority phase, the cost priority is not changed as long as non-tabu moves exist of type I or III, *and* as long as the current objective function value is good enough. Similarly, if the search is in the cost priority phase, the integer priority is not changed as long as non-tabu moves exist of type I or II, *or* the current solution is integer feasible. Computational results for the *Alternating Integer and Cost Priority* scheme outlined above are reported in Section 11.

7.2 Strategic Oscillation by Altered Choice Rules

Instead of altering the move evaluation function to change the relative emphasis on the component parts of the move evaluation function, as outlined in section 7.1, the choice rules may be altered. One approach is as follows:

Choice Rule Modification

Step 1: *Integer Priority Phase*. Sort the candidate list in the order of move type III, I, II, IV. Change to the Cost Priority Phase after a specified number of iterations.

Step 2: *Cost Priority Phase*. Sort the candidate list in the order of move type III, II, I, IV. Change to the Integer Priority Phase after a specified number of iterations.

Step 3: Alternate between the two preceding phases until reaching a desired cutoff point.

Note that this approach has the same integer and cost priority phases as the parametric evaluation approach, but the grouping of moves is coarser, and the control scheme simpler. The method outlined above can thus be viewed as a more intense version of the *Alternating Integer and Cost Priority* scheme of Section 7.1, and comparative computational results are reported in Section 11.

An alternative to the fixed length approach outlined above, is to stay in each phase (only) as long as appropriate non-tabu moves for that phase are available. This means that the search stays in the integer priority phase as long as moves of type I or III are selected, and in the cost priority phase as long as moves of type II or III are selected.

8. Diversification Strategies

TS diversification strategies, as their name suggests, are designed to drive the search into new regions when the search fails to improve the search in the initial search area. Frequency based memory is normally used to guide the diversification process, and consists of gathering pertinent information about the search process so far, where the information gathered may, for example, be the number of times a specific element has been part of the solution. A simple form for diversification then penalizes the inclusion of frequently occurring elements in future solutions. More sophisticated diversification strategies can be defined, based on a major altering of the choice rules or by restarting the search process. The best strategies balance diversification against intensification, as by differentiating between frequency of occurrence in solutions of different quality. Elements that occur frequently in the highest quality solutions then are encouraged for selection, and the factor of encouragement may significantly reduce the diversification penalty or even turn it into a net inducement. (We explore a slightly different mechanism for balancing diversification and intensification, as subsequently noted.)

The first-level TS mechanisms described in Section 1 to 5 are clearly most successful when applied to the smaller test cases (see Section 11), but do not manage to diversify the search properly for the larger ones. This is a clear indication that some sort of diversification strategy should be included, which makes use of information from the current search history.

We tried out two different diversification schemes, one based on penalizing variables according to their accumulated time spent in the basis, and one that incorporated intensification concerns by identifying promising variables to include in the solution. The latter scheme is based on results from applications of Target Analysis (TA), and came about after preliminary testing showed that our first (pure) diversification scheme was less successful than anticipated. (See Section 9). It must also be stated that we have confined our diversification schemes to be based solely on extreme point pivoting, and have not considered diversification by generating new starting points (which could be obtained by modifying the objective function in an appropriate way and then re-solving the LP-relaxation), or by directly assigning 0 or 1 values to the integer variables (which also requires the LP-relaxation to be re-solved, possibly resulting in primal infeasible solutions).

8.1 Diversification by penalizing time spent in the basis

One simple diversification scheme is to penalize elements that have been active, or selected, for a disproportionate part of the time. For our approach we focus on the time spent in the basis by each variable.

To accomplish this we use the record *tabu_frequency*(*j*), introduced in Section 4, to measure the number of iterations that x_j has been basic. This record is used to create a diversification strategy, where x_j is penalized to induce it to become non basic, or to discourage it from becoming basic. Greater penalties are attached to variables that have been basic for a greater part of the time. To do this, we make additional use of the value of *tabu_start*(*j*), used for keeping track of basic tabu tenure, by setting this value equal to *current_iteration* not only when x_j becomes non basic but also when x_j

becomes basic. For those variables x_j that start basic in the first solution, set *tabu_start*(j) = 0 to indicate they start basic at iteration 0, and set *tabu_start*(j) equal to a large negative number at iteration 0 for all other x_j. We then update and maintain *tabu_frequency*(j) as follows

Step 1. To begin, set *tabu_frequency*(j) = 0 for all $j \in N$

Step 2. When x_j changes from being basic to non basic, set

tabu_frequency(j)= *tabu_frequency*(j) + *current_iteration* - *tabu_start*(j)

By these rules the value of *tabu_frequency*(j) is correct for every variable x_j that is non basic, and is given by the right hand side of the Step 2 update for every x_j that is basic. This results from the fact that Step 2 increases *tabu_frequency*(j) by the number of iterations x_j has been basic. This shows that in the worst case (for x_j basic) the correct frequency value can be determined by one addition and one subtraction, using the formula of Step 2 above (but without creating the update until x_j changes from being basic to non basic).

A pure diversification strategy using *tabu_frequency*(j) (without reference to intensification concerns) is initiated after an extended period when the integer feasible solution has not been improved. A non basic variable x_j is then penalized (to discourage it from being chosen to become basic) according to the value either of *tabu_frequency*(j) or *tabu_frequency*(j)/*current_iteration*. The division by *current_iteration* normalizes the frequencies to values less than 1, and hence requires a larger penalty.

A Diversification Penalty. To determine a simple diversification penalty, we compute a running value V, which starts equal to 0, and then at each iteration is set to

$V = V +$ (*current_iteration* - *tabu_start*(j))

Here x_j is the variable that becomes non basic at the current iteration. (If x_j already is non basic, and again becomes non basic by moving to its other bound in the same step, we treat it as becoming both basic and non basic at the same iteration. Hence the quantity *current_iteration* - *tabu_start*(j) is replaced by the value 0 in this case, both for updating V and *tabu_frequency*(j).)

To make use of V, keep a value $COUNT$, which starts equal to 0. Then, on the iteration that $tabu\_frequency(j)$ changes from 0 to a positive value (identifying x_j as the variable that becomes non basic) set

$$COUNT = COUNT + 1$$

Thus, $COUNT$ gives the number of variables that have positive $tabu\_frequency(j)$ values, and the value $V^* = V/COUNT$ gives the average $tabu\_frequency(j)$ value for these variables. For a pure diversification strategy we therefore can simply assign a large penalty to a given variable x_h under the following conditions.

Mild Diversification: $tabu\_frequency(h) \geq V^*$

Moderate Diversification: $tabu\_frequency(h) \geq V^*/2$

Strong Diversification: $tabu\_frequency(h) \geq V^*/8$.

Penalties assigned under the preceding conditions are used to discourage the choice of $h \in NB^*$ by the rules of Section 1 (first-level tabu search mechanisms) when x_h is under consideration for generating an extreme point $x(h)$ (i.e., h is being evaluated as a candidate to become h^*). These penalties should not be applied at each iteration, but only on those iterations when there exist no moves that can improve integer feasibility except those that are tabu. A variable that does not receive a large penalty by the foregoing criteria receives a zero penalty.

Another kind of pure diversification approach may be applied a small number of times during a solution run. At each iteration selected to initiate this approach, reset $tabu\_start(j)$ equal to a large negative number for all current non basic variables x_j (thus freeing all non basic variables from being tabu). Then set the tabu tenure t to twice its usual value, and apply the rules for Mild, Moderate or Strong diversification, as previously specified, throughout each of the next t iterations. After these t iterations are executed, again free all non basic variables from their tabu status and resume the usual tabu search approach.

Testing of the basic mechanisms of this approach disclosed that the penalizing of basic time did not result in the desired effect, even when the strong diversification was applied. The reason for this is probably that time spent in the basis by a variable is less important than the time it spends at the bounds, and the dynamics of how often and in what manner it changes its value. This is illustrated by the fact that the proposed mechanisms fail to capture the effect of variables that enter and leave the basis in the same iteration, going to opposite bounds.

8.2 Diversification by inclusion of promising variables

As previously noted, our testing showed that the foregoing diversification strategy failed to produce improved results. By applying target analysis (see Section 9), we came up with the following basic diversification scheme. Memory is introduced to record the number of times each variable has entered the basis from its upper or lower bound, registering this information in the tables *Fr1*(*j*) and *Fr0*(*j*) respectively. We also keep track of how many times each variable has been chosen for diversification in the table *DSEL*(*j*), and the number of successful attempts in the table *DSUCC*(*j*).

Diversification by Variable Inclusion

Step 0: *Ordinary TS phase*. When this phase either fails to produce improved solutions, or has spent a selected number of iterations, enter Step 1.

Step 1: *Variable identification*. Identify the variable that has tried to enter the basis from its lower bound the most times and from its upper bound the fewest times, subject to a penalty for being chosen many times. Do not reselect variables that have failed to be included in an earlier diversification phase.

Select $j = \max((Fr0(j) - Fr1(j))/DSEL(j))$, and update DSEL(j).

Step 2: *Variable inclusion phase*. Select the first move from the sorted candidate list that increases the integer value of the chosen variable *j*, disregarding tabu status. If no such moves exist, select the move yielding the smallest decrease in the value of the chosen variable *j*. Stay in this phase either until the chosen variable leaves the basis at its upper bound (Go to Step 3), or until a preselected number of iterations have elapsed (Go to Step 4).

Step 3: *Successful Inclusion*. When the designated variable has been successfully included in the current solution, give it extra tabu tenure, remove the tabu status from all the other variables, update DSUCC(j), and return to Step 0.

Step 4: *Unsuccessful Inclusion*. If variable inclusion fails, return to Step 0.

The extra tabu tenure introduced in Step 3 is required so that the solution may stabilize itself in the new search area. At the same time, freeing the other variables from their tabu tenure makes the initial search in this new area more aggressive, hence providing some balance due to intensification.

9. Target Analysis

Target analysis (TA) is a learning approach, designed to identify good search parameter values and effective search attributes. TA typically first launches an extensive search effort that concentrates on obtaining high quality solutions to a chosen sample of test cases. Each sample problem is then solved again, this time focusing on which choices should be made to guide the search most effectively to, or near, the previously discovered good solutions, using hindsight to select desired moves. Finally, the choices actually applied during the re-solving phase are analyzed in the context of available information and decision parameters, and integrated into new master decision rules. In other words, these rules are designed to use information ordinarily available during the search (including historical information of the type embodied in tabu search memory), to make decisions that are as close as possible to the decisions that hindsight would provide (where hindsight uses knowledge of elite solutions that would *not* ordinarily be available). By the proximate optimality principle, POP, these new rules are anticipated to give a good search strategy for the similar target test cases. (In repetitive problem solving environments, these rules include mappings that assign elite solutions from past efforts to become starting points for related current problems.)

Target analysis is more fully described in Glover and Greenberg (1989), Laguna (1990) and Glover (1995b). Some successful applications are in Glover and McMillan (1986), Laguna and Glover (1993), and Mulvey (1995).

We have applied TA in two different settings in our study. The first use was to identify the proper relationship between the various parameters in the ratio test move evaluation (see Section 3), while the second was to identify a better decision rule for diversification when our first approach failed to produce good results (see Section 8).

9.1 Target Analysis for the Ratio Test

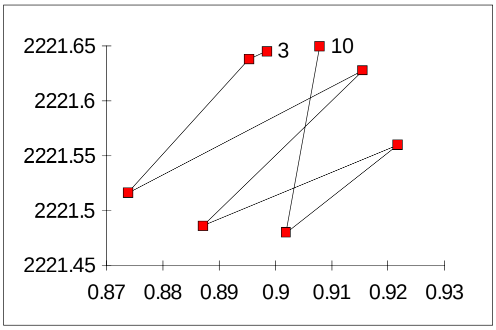

Figure 11. *Detail of figure 10*

Preliminary testing disclosed some problems for the ratio test (see Section 3) on some of the test problems (see Section 10). We will illustrate this by looking at the behavior of this heuristics on the Fleischer's problem (FLEI), where it failed to find an integer feasible solution. We have chosen this problem for illustration because it is fairly small, and previous heuristic approaches reported in the literature also have experienced difficulty in obtaining an optimal solution to this problem. This focus on the behavioral aspects of the search is also in the spirit of Hooker (1995). It is important to note that this particular application of TA does not focus on obtaining *good* solutions, but rather on finding a *feasible* solution, so a distance measure to the optimum is not required.

Recall that the ratio test depends on the integer infeasibility change measure, $F(w,q)$, as a function of multiplier w and exponent q, together with the exponent p in the definition of $u(h)$. The relevance of these parameters is usefully illustrated in

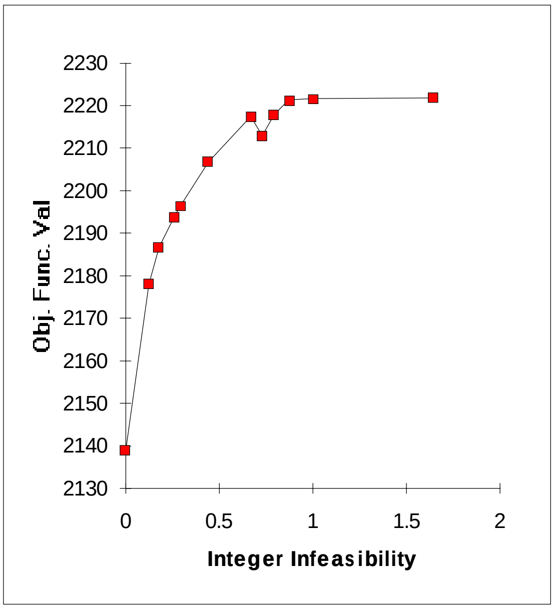

Figure 9. *FLEI for Weighted Sum*

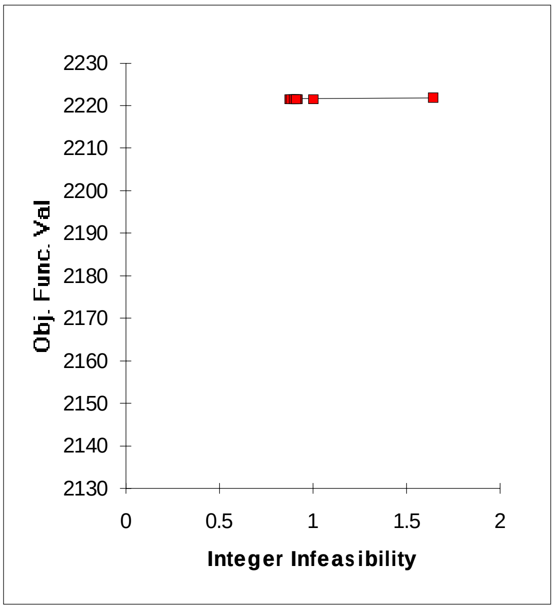

Figure 10. *FLEI for Ratio Test*

application to the FLEI problem. Specifically, we first note that taking $w = 1$ and $q = 1$ (together with $p = 1$ in the definition of $u(h)$), results in poor performance for the ratio test in this problem, in fact causing it to fail to locate a feasible solution. This is illustrated in figures 5, 9, 10 and 11. Figure 5 shows the various best values achieved for the different aspiration levels, for all four heuristics. As can be seen in figure 5, all the heuristics dive to the 4th aspiration level, but the ratio test heuristic stops there, while the other three continue toward the feasible region.

Figure 9 shows how the weighted sum heuristic focuses nicely on finding a feasible solution (which is also the optimum) in 11 iterations, while figure 10 shows the same type of plot for the ratio test heuristic. Iterations 3 to 10 are shown enlarged in figure 11. As can be seen, the search path consists of alternating steps consisting of type I and type II moves, which pull in opposite directions.

To compensate for an occasional tendency of the ratio test to wander too far away from the feasible region, we applied target analysis to try to find good combinations of search parameter values. Our focus in this experimentation was to determine an effective way to increase the push toward integer feasibility. We soon isolated a

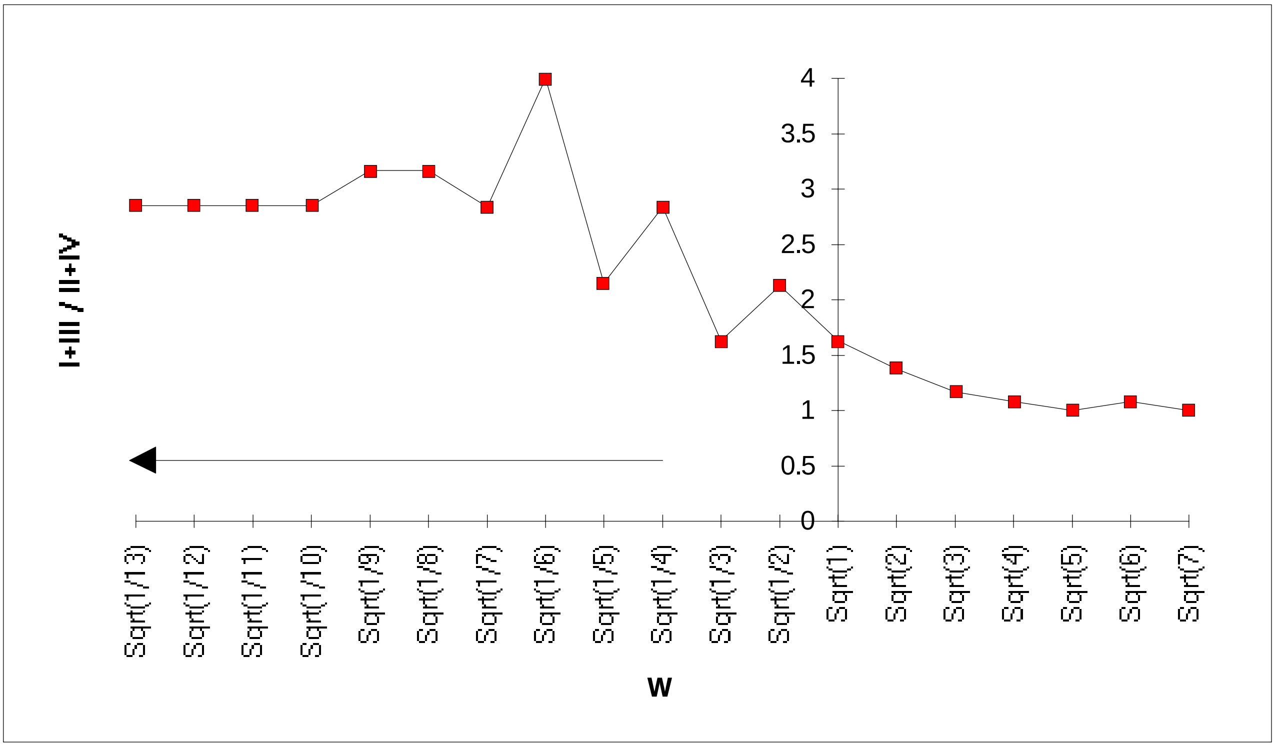

Figure 12. *Accepted Move Type Ratio vs. Axis Skew Factor*

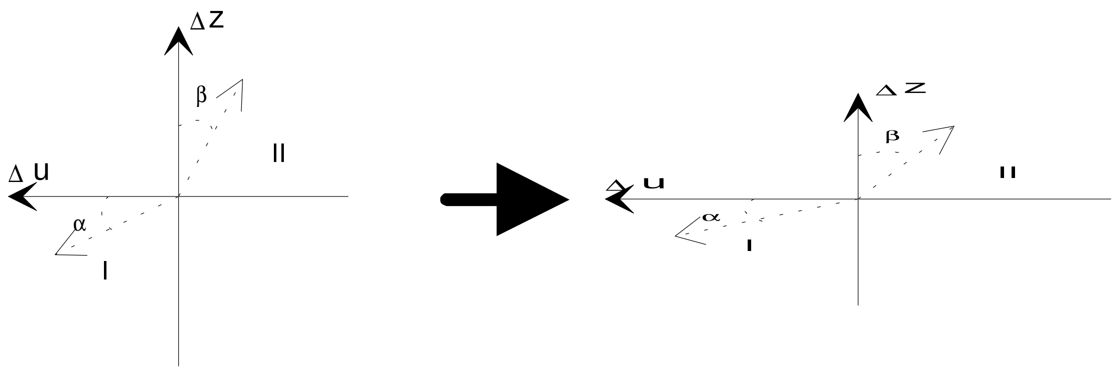

Figure 13. *Skewing of axis.*

strategy of doing this based on increasing the emphasis on type I moves, by altering the multiplier *w* (as defined for Normalization 1). Target analysis disclosed the existence of a good region for the expected ratio between the number of moves of types I + III versus those of types II + IV, in order to obtain good solutions fast. This is shown in figure 12 for the Fleischer problem. Feasible solutions are found for values of *w* of .5 or less, as indicated by the arrow. Altering *w* can also be thought of as skewing the axis in our search space, and is indicated schematically in figure 13. In order to force our heuristic to approach the desired ratio of selected moves, we skewed the axis in favor of integer feasibility, by shrinking the objective function value axis and expanding the integer infeasibility axis during move evaluation.

Infeasibility measure exponents *p* and *q*. It is reasonable to anticipate that ideal values for *w* will depend on *p* and *q*, and we conducted further tests to identify the nature of this dependency.

Figure 14 provides the insight into the pattern that emerges, by showing the maximum value of *w* for which feasibility was obtained for the Fleischer problem (with each test run for 50 iterations). This plot is based on maintaining either $p = 1$ or $q = 1$ and varying the other parameter, to identify $w = w(p)$ or $w = w(q)$ as a function of the parameter varied. As can be seen from the figure, the search is rather insensitive to the value of *q*, while the value of *p* is clearly correlated to the values of *w* that are necessary to obtain feasibility, where small values of *p* generally make feasibility easier to obtain. This same pattern emerged for the other test problems studied.

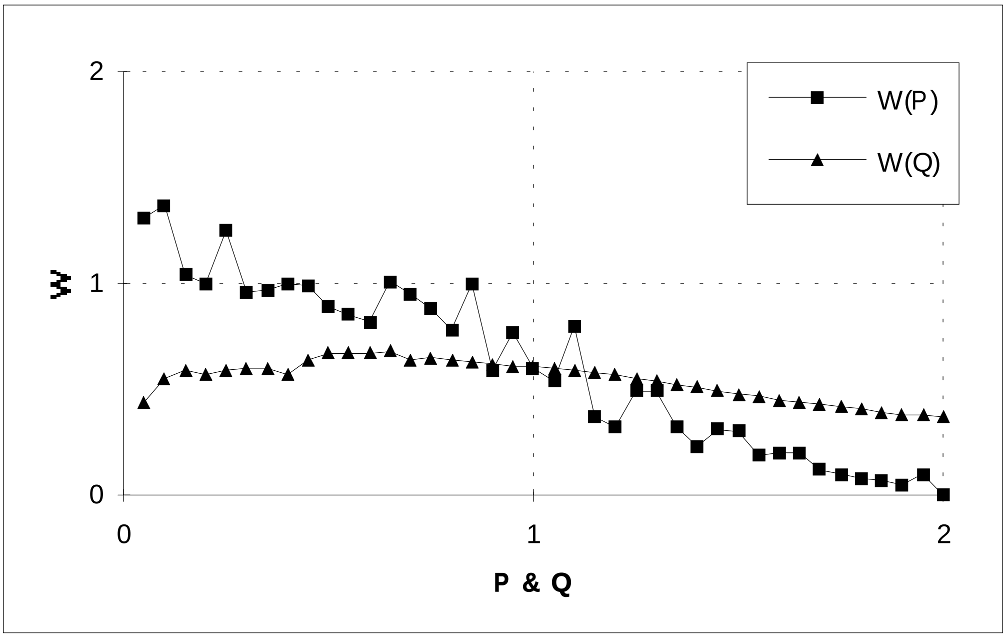

Figure 14. *Infeasibility measure exponents p & q vs. axis skew factor, w*

The reason for these results can be traced to the definition of Normalization 1. As we have noted, changing the value of *w* can be regarded as a linear skewing of the axis, with smaller values of *w* giving more emphasis on type I moves relative to type II moves. Altering *q* does not seem to have any great overall effect. The rationale for wanting to use a value of *p* greater than 1, the exponent in the definition of *u(h)*, is to put more emphasis on moves that have a large distance to cover to reach feasibility, with the intention of getting faster to a feasible solution. What really happens is that variables with a small degree of infeasibility are driven more strongly to achieve integer values than variables with a larger degree of integer infeasibility, as a result of this influence on *F(w,q)*, used in the value of R in Normalization 1. In particular, *F(w,q)* gets reduced exponentially when all the variables have small degrees of integer infeasibility. Since R is again used in the denominator for type I moves, and in the numerator for type II moves, undue emphasis is put on type II moves. This alteration of *p* can also be viewed as a nonlinear skewing of the axis, with more skewing for small changes in infeasibility, and with disproportionate emphasis on type II moves for values of *p* greater than 1.

The key observation is that it is not the ranking within the different move types that is important, but the relative ranking between type I and II moves. In addition, changes in *p* can shift moves to different categories, where high values of *p* assign more moves to categories II and IV, while low values of *p* assign more moves to category I and III.

Both lowering *w* and lowering *p* are effective in putting more emphasis on integer feasibility. However, altering *w* works in a linear way, while altering *p* works non linearly. We therefore give preference to altering *w*, if the search undergoes a period in which feasibility proves difficult to achieve (e.g. starting with *p*, *q* and *w* equal to 1).

9.2 Target Analysis for Diversification

The first phase of TA was easily done in our case, because optimal solutions for our test cases are known from the literature. To start off the TA second phase, we first undertook to define a measure of distance between different solutions, in particular between the identified optimal solution and any other solution, infeasible or not, encountered during the search. We elected to use a simple definition that equated distance with the number of variables that differed in value between the two solution states. Note that this is not necessarily equal to the number of pivots required to move between the two solutions. However, letting D_{xy} denote the distance between the two solutions *x* and *y*, and P_{xy} denote the actual minimum number of pivots needed to move between them, the following relation holds:

$$P_{xy} \leq D_{xy} \leq 2*P_{xy}$$

Information logged. In addition to the information described in Section 8, the following information was logged for each iteration. It is important in this phase to register many different aspects of the solution process, as one purpose of the TA process is to identify the pieces of information that will be required by the new guidance structures.

D_{xOPT}	- Difference between this solution and the optimum.
Feas	- Integer feasibility status
MoveType	- Selected move type
EnteringVar	- Entering variable
EntVarStatus	- Where *EnteringVar* enters from (upper or lower bound)
LeavingVar	- Leaving variable
LeaVarStatus	- Where *LeavingVar* goes to (upper or lower bound)

In addition, a running total of the following values for each integer variable was kept:

At1	- Number of iterations at 1
At0	- Number of iterations at 0
Basic	- Number of iterations spent basic
B(0)-B(9)	- Number of iterations that the variable acquired a fractional value, divided into 10 equal interval buckets.
Fr1	- Number of times entering the base from 1
Fr0	- Number of times entering the base from 0

Problem	M*N	$D_{LP^* - Opt}$	$D_{F\ F - Opt}$
PET5	10*28	5	4
PET7	5*50	9	6
PB4	2*29	6	13
PB6	30*40	11	7
PB7	30*37	7	5
WEISH07	5*40	4	3
WEISH08	5*40	5	4
WEISH16	5*60	3	3
WEISH18	5*70	5	3
WEISH19	5*70	3	3
WEISH22	5*80	4	5
WEISH25	5*80	4	5
WEISH26	5*90	4	3
WEISH27	5*90	3	5
WEISH29	5*90	2	2
WEISH30	5*90	3	4
SENTO1	30*60	10	6
SENTO2	30*60	7	5

Table I. Initial distances to the optimum solution.

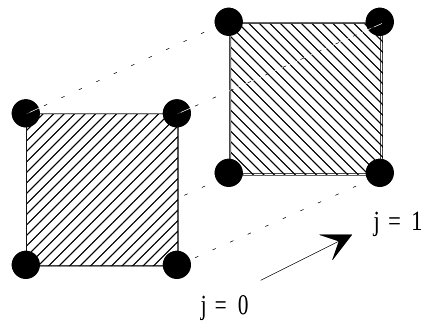

Figure 15. Diversification along variable j

Insights gleaned. Our application of TA within the diversification attempts outlined in Section 8, and also within the first-level tabu search approach of Section 1 to 5 led to the following insights.

Relative to our defined solution difference measure, our starting point, the LP relaxation solution, was usually fairly close to the targeted optimal solution. The first integer feasible solution found also turned out to be close to this optimal solution by this measure, indicating that this first feasible solution may be a good candidate for one of the points of a scatter search approach, and similarly for path relinking. (See Glover, 1995b, 1996.) The actual distances, as defined above, are shown in Table I, with the distance measures being denoted $D_{LP^* - Opt}$ and $D_{F F - Opt}$, and where the test cases are defined in Section 10. The distances for the test cases where the first-level TS found

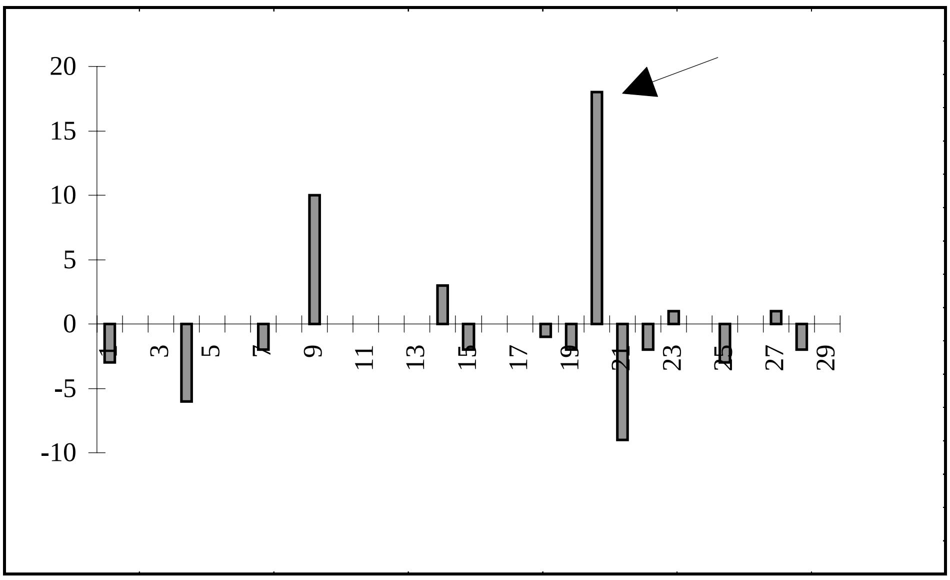

Figure 16. Fr0(j) - Fr1(j) for PB4

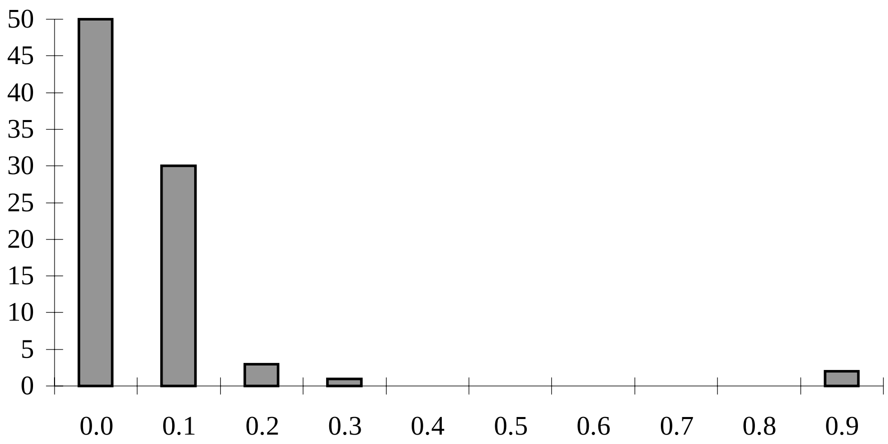

Figure 17. Basic values for selected variable

the optimum were similar.

The target analysis showed that the first-level TS was locked in a specific part of the search space that yielded similar distances, disclosing the existence of a small number of variables that should have been part of the solution that our first-level TS mechanisms failed to include. Moreover, each such variable was typically characterized by entering the basis from its lower bound, staying at a low value in the basis for a few iterations, and then leaving the basis, again at its lower bound. This prompted the diversification scheme described in Section 8.2. One might imagine the search going from the left part of figure 15 to the right, where variable j is induced to go from 0 to 1, and the shaded areas are where the search is active.

A typical example, taken from test case PB4 after 290 (=10*N) iterations, is shown in figures 16 and 17. Figure 16 shows the value of $Fr0(j)$ - $Fr1(j)$ for all j, identifying variable 20 (indicated by the arrow) as a good candidate for inclusion in the solution. Figure 17 shows the actual basic values of this variable, disclosing that it does not attain very large fractional values before being made non basic again. The few large fractional values stem from the initial descent phase after starting at the LP optimum, where variable 20 starts basic. The diversification scheme described in Section 8.2 does not use the information of the type contained in figure 17, but relies on the information from figure 16.

10. Test Cases

To test our heuristics, we employed a portfolio of 57 multi-constraint knapsack problems. These are the same as those used by Dammeyer and Voss (1993), Løkketangen, Jörnsten and Storøy (1993), Aboudi and Jörnsten (1992) and Drexl (1988), and were collected by Drexl from the literature. The problem sets that compose this portfolio are indicated below, where numbers attached to the abbreviated set names differentiate the elements of each set.

PET1 - PET7	Petersen (1967)
SENTO1 - SENTO2	Senyo-Toyoda (1968)
PB1 - PB7	Mod. of Senyo-Toyoda used by Freville Plateau (1987)
WEING1 - WEING8	Weingartner (1967)
WEISH1 - WEISH30	Shih (1979)
FLEI	Fleisher problem

11. Computational Results

The heuristics in this paper were implemented in FORTRAN, incorporating subroutines provided by ZOOM/XMP whenever possible. We note that this software is not nearly as efficient for solving LP problems as modern commercial software such as CPLEX and XPRESS-MP, but we used it to take advantage of the ability to obtain intimate access to key subroutines. The ZOOM/XMP software package was also used to solve the initial LP relaxation. (See Marsten 1989a, 1989b.)

11.1 Results for first-level TS mechanisms

We first tested the four first-level heuristics, based on the different move evaluation and choice rule schemes detailed in Sections 1 to 5:

1. Weighted sum
2. Ratio test
3. As 1, but with Weighted Sum within the move groups
4. Ratio test, I before II

Weights for the weighted sum heuristic. Figure 18 shows the ratio between the weights for which feasibility was obtained for heuristics 1 and 3 when applied to FLEI. The diagram shows that integer feasibility is successfully achieved by the mechanism of our choice rules, simply by placing sufficient emphasis on this goal within the framework of these rules.

Ratio test. See Section 8.1 (Target Analysis) for a brief discussion on finding good parameters for this test.

Aspiration. Contrary to our expectation, experimentation showed that aspiration by **integer infeasibility levels** did not help in the quest to produce better solutions. In fact, this type of aspiration had an adverse effect on solution quality and on the ability to find feasible solutions. The effect became more pronounced as the number of aspiration levels was increased. In retrospect, the reason for this behavior is probably that moves accepted by this criterion are of type II and III. As we have noted, type II moves pull the search away from the feasible region, and this tends to impair the success of the method.

By contrast, the use of aspiration by **objective function value levels** worked quite well. This type of aspiration tends to favor acceptance of type I and type III moves, which guide the search toward the feasible region. We use the best feasible solution value found so far as a threshold value, so that moves resulting in an objective function value lower than this best value are not screened by this aspiration criterion. (For moves that do not produce feasible solutions, this threshold is evidently weaker than the related aspiration test often applied in settings where all steps are feasible.) As indicated in section 5.3, this type of threshold proved to be quite advantageous.

Rejection of Degenerate Moves. By degenerate moves, we mean type III moves where both $\Delta z(h)$ and $\Delta u(h)$ are zero. These moves often occur in bunches, and our

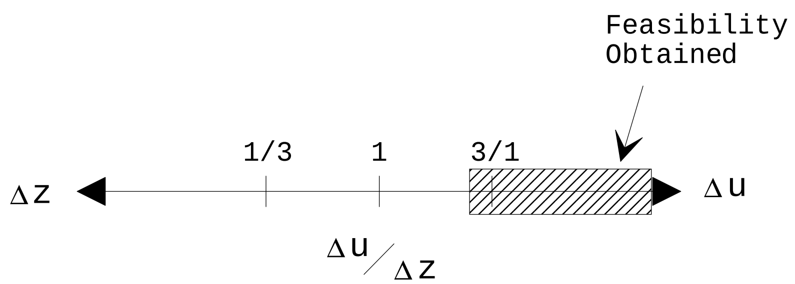

Figure 18. *Weights and Feasibility*

experience is that they should be accepted only rarely. The rejection of (most) degenerate moves does not in general lead to better solutions, but the number of iterations required is reduced, and the tabu list is not clogged with ineffectual moves.

Normalization 2 for the ratio test. Normalization 2 is not effective during the initial descent phase, as it needs a value for the best feasible solution found so far. Also, since we are exploring the full neighborhood for our candidate list, the suggestion of expanding the candidate list when obtaining low values for Normalization 2 is also not applicable. (For larger or computationally more expensive problems, of course, a more selective form of candidate list would be appropriate.)

When Max(R1(h1), R2(h2)) is somewhat less than −1, however, this can be used as an indicator that the method is confronted by an inferior search region, and a diversification of the search may be fruitful.

First-Level TS parameter values. All tests were done with a dynamic tabu list length size starting at 5 or $\lceil \text{SQRT}(N) \rceil$, whichever is larger, where *N* is the number of integer variables. The list size is then varied randomly between this number and 2 * $\lceil \text{SQRT}(N) \rceil$, every 10 iterations. Aspiration is by Objective Function Value Levels, and new best detected, with a threshold at 0.9 LP*. Degenerate moves are accepted with a probability of 0.1. The exponents *p* and *q* were both set to 1. The axis skew factor, *w*, was set to 1 for the ratio test. The value of w was halved and the method was restarted if a long run of at least 2*N* integer infeasible iterations occurred. Weights for the weighted sum were set at 3 to 1 in favor of the integer infeasibility measure. Up to 10*N* iterations were allowed per problem. We did not seek to tailor the tabulist lengths and the other search parameters for each of the component problem sets separately. We note, however, that each of these sets has its own features and customary practice in the literature has been to develop “fine tuned” parameters to exploit these differences. Since our method is a general purpose procedure, we felt it was more appropriate to use the same parameters for each test bed.

Results for First-Level TS. Table II shows the results obtained for the 57 test cases for the weighted sum and ratio test evaluation functions. We found the ratio test heuristic to be somewhat superior to the weighted sum heuristic. This is probably due to the inherent problem of scaling, or establishing compatibility, of the two rather diverse measures of integer infeasibility and the objective function value, which the ratio test conveniently gets around.

Both heuristics perform better on the smaller test cases than on the large ones. This is as should be expected, since none of the elements of diversification are built into this first-level testing.

Results for the two additional heuristics derived from the two main heuristics (see Section 3.3) gave results comparable to those is reported in table II.

p	10*N	20*N
1.0	90909	91935
0.9	90909	91935
0.8	90615	90909
0.7	91935	91935
0.6	90909	90909
0.5	89659	90909
0.4	90858	93118
0.35	92506	92506
0.3	90008	94965
0.25	90909	90909
0.2	87937	88724
0.1	90615	90615

Table III. Probabilistic TS for PB4

11.2 Results for Probabilistic TS

Probabilistic move acceptance. The first objective was to find a good value, or range of values, for *p*, the probability threshold for accepting a move, as outlined in Section 6. The results for varying *p* over the range from 0 to 1 in 0.1 intervals are shown in Table III. The chosen test case is PB4, for which the first-level TS fails to find the optimum. (The first-level test is represented in the table by *p* = 1.) We show the best achieved values for 10*N iterations, and 20*N iterations, since the probabilistic TS should need more time than the first-level TS, being less focused. The values for *p* of 0.25 and 0.35 were added to give extra resolution around the best area.

As can be seen from the table, good values for *p* for this problem lie in the range 0.3 to 0.4. Larger values for *p* lead to inferior solutions when compared to the deterministic variant. This is probably because too many potentially good moves are thrown away, while at the same time not obtaining enough diversification. At the other end of the scale, with values of *p* from 0.2 and down, too few good moves are chosen to focus the search properly. Figure 19 compares the search progress for the best (p = 0.3) probabilistic case for PB4 vs. the deterministic case. The plot shows the best feasible solutions visited in each implementation.

Problem	M*N	LP	IP	Weighted Sum	Ratio Test
PET1	10*6	4134.07	3800	**3800**	**3800**
PET2	10*10	92977.1	87061	**87061**	**87061**
PET3	10*15	4127.9	4015	4005	**4015**
PET4	10*20	6155.3	6120	6090	**6120**
PET5	10*28	12462.1	12400	12360	12380
PET6	5*39	10672.3	10618	**10618**	**10618**
PET7	5*50	16612	16537	16457	16470
PB1	4*27	3144.3	3090	3077	**3090**
PB2	4*34	3261.3	3186	3093	**3186***
PB3	2*19	32612.1	28642	**28642**	**28642**
PB4	2*29	99622.7	95168	**95168**	90909**
PB5	10*20	2221.3	2139	2102	**2139****
PB6	30*40	843.3	776	723	729**
PB7	30*37	1086.2	1035	**1035**	1033**
WEING1	2*28	142019	141278	140543	**141278**
WEING2	2*28	131637.5	130883	**130883**	**130883**
WEING3	2*29	99647.1	95677	**95677**	**95677**
WEING4	2*28	122485.3	119337	110667	**119337**
WEING5	2*28	100433.1	98796	98631	**98796**
WEING6	2*28	131335	130623	130233	**130623**
WEING7	2*105	1095741	1095445	**1095445**	**1095445**
WEING8	2*105	628773.7	624319	617715	**624319***
HP1	4*28	3472.3	3418	**3418**	**3418***
HP2	4*35	3261.8	3186	3159	**3186**
WEISH01	5*30	4632.3	4554	4525	**4554**
WEISH02	5*30	4592.7	4536	**4536**	**4536**
WEISH03	5*30	4177.8	4115	**4115**	**4115**
WEISH04	5*30	4611.01	4561	**4561**	**4561**
WEISH05	5*30	4530.8	4514	**4514**	**4514**
WEISH06	5*40	5585.2	5557	5494	**5557**
WEISH07	5*40	5601.9	5567	**5567**	5549**
WEISH08	5*40	5631.6	5605	5519	5603
WEISH09	5*40	5254.9	5246	**5246**	**5246**
WEISH10	5*50	6347.2	6339	6338	**6339**
WEISH11	5*50	5688.2	5643	**5643**	**5643**
WEISH12	5*50	6395.7	6339	**6339**	**6339**
WEISH13	5*50	6241.1	6159	**6159**	**6159**
WEISH14	5*60	7018.3	6954	**6954**	**6954**
WEISH15	5*60	7518.3	7486	**7486**	**7486**
WEISH16	5*60	7314.02	7289	7287	7287
WEISH17	5*60	8656.6	8633	8619	**8633**
WEISH18	5*70	9603.7	9580	**9580**	9565
WEISH19	5*70	7756.9	7698	7663	7667
WEISH20	5*70	9477.9	9450	**9450**	**9450**
WEISH21	5*70	9110.5	9074	9024	**9074**
WEISH22	5*80	9004.2	8947	8876	8868
WEISH23	5*80	8392.1	8344	8216	**8344**
WEISH24	5*80	10232.8	10220	10088	**10220**
WEISH25	5*80	9964.7	9939	9913	9928
WEISH26	5*90	9641.6	9584	9496	9532
WEISH27	5*90	9849.7	9819	9772	9718
WEISH28	5*90	9514.2	9492	**9492**	**9492**
WEISH29	5*90	9429.03	9410	9369	9309
WEISH30	5*90	11194.5	11191	11146	11146
FLEI	10*20	2221.8	2139	**2139**	**2139***
SENTO1	30*90	7839.3	7772	7719	7719*
SENTO2	30*60	8773.2	8722	8721	8702

Optimal solutions in bold

*   w reduced to 0.5

** w reduced to 0.25

Table II. Computational Results.

p	10*N	20*N
0.9	84730	84730
0.8	84730	84730
0.7	89659	89659
0.6	90909	90909
0.5	90909	90909
0.4	90909	90909
0.3	90615	90615
0.2	90909	90909
0.1	88030	90615

Table IV. No tabu tenure for PB4

Similar tests for other problems showed the same pattern, with good values for *p* being between 0.3 and 0.5, so an overall value of *p* = 0.4 was chosen for further testing.

One important finding from our preliminary testing was that low values of *p* make it more difficult to obtain integer feasibility. This is due to the fact that good moves are too often rejected that contribute to the reduction of integer infeasibility. Searches that experience this problem could either use a larger value for *p*, or a lower value for *w*.

It is also more important in a probabilistic TS approach based on ranking to have good aspiration criteria, since good solutions in the neighborhood are even less likely to be visited than in the deterministic case, and should be actively culled out.

Probabilistic move acceptance with no tabu memory. We also investigated the conjecture that the probabilistic move acceptance scheme should work without any tabu memory, solely relying on biased randomization as guidance. Table IV shows the results for PB4 for various values of *p* with a tabu list length of 1 (chosen to avoid direct move reversals). As can be seen, for values of *p* in the range 0.4 to 0.6, this scheme obtains the same results as the first-level tabu search approach, though inferior to the probabilistic tabu search approaches with memory. It is also evident that the best

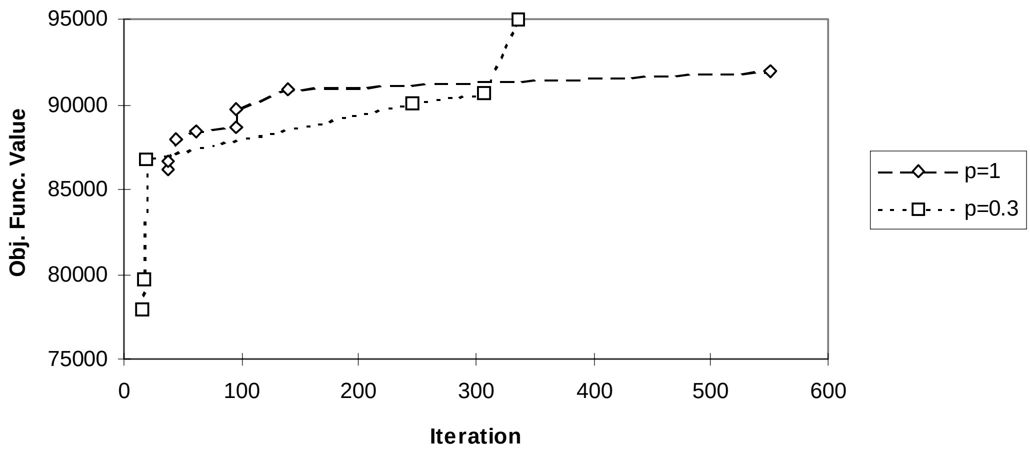

Fig 19. Search progress for PB4 with *p* = 0.3 and *p* = 1

relative results are obtained for few iterations, as the extra search effort expended when going from 10*N to 20*N iterations only yields marginally better outcomes.

This method is extremely simple and efficient to implement, and can be used in settings where one just wants to spend a small number of iterations, or where the tabu attributes might be difficult to identify or separate.

TL	10*N
4	89969
5	90909
6	90909
7	89784
8	91670
9	91000
10	90615

Table V. Probabilistic tabu list lengths for PB4.

We note that the tabu tenure of 1 for the "almost memoryless" case may be a stronger restriction in the present setting than in many others, since preventing the reversal of a single pivot step may cause the only paths for reaching the preceding solution to consist of multiple pivots, depending on the polyhedral structure of the problem. Consequently, a larger tenure may be appropriate to obtain similar results with probabilistic tabu search in other applications.

Probabilistic tabu tenure. As stated in Section 6, dynamic tabu lists have been used extensively in the literature. Usually the dynamic element has been achieved by varying the tabu list length between fixed or random bounds, in a fixed or random way. Our proposed scheme achieves the same effect, but more smoothly, making the dynamic tabu tenure span of each variable (or attribute) much larger. We conceive that this may in turn contribute more effectively to diversification and loop avoidance than the usual methods. The present method is also more convenient for assigning individualized probabilistic tabu tenures, as can be useful if the individual attributes can be identified according to the degree that they qualify as *consistent* or *strongly determined*. (Consistent attributes are those that receive their preferred values a larger fraction of the time, while strongly determined attributes are those that cause the solution quality or structure to be more greatly hurt by changing to other values. These notions depend on context, as in relation to solutions grouped by cluster analysis.)

Table V shows the results for various lengths of the first-level tabu list, suggesting that the relation between the first-level tabu tenure and probabilistic tabu tenure indeed is as anticipated: $TL_P = 3*TL_D$. The test case used is again PB4, which for the first-level TS uses a tabu list length of 6, obtaining a best value of 90909. Slightly better

Problem	**M*N**	**LP**	**IP**	**First-level TS**	**PMA**	**PMA0**	**PTT**	**PSO**
PET5	10*28	12462.1	12400	12370	12380	12390	12360	**12400**
PET7	5*50	16612	16537	16507	16468	16494	16468	16468
PB4**	2*29	99622.7	95168	90909	93118	90909	90909	**95168**
PB6**	30*40	843.3	776	723	765	729	745	765
PB7**	30*37	1086.2	1035	1033	**1035**	1011	1021	1011
WEISH07*	5*40	5601.9	5567	5525	5542	5541	5541	**5567**
WEISH08	5*40	5631.6	5605	5603	5603	**5605**	5603	5592
WEISH16	5*60	7314.02	7289	7287	7287	7288	7287	7287
WEISH18	5*70	9603.7	9580	9565	9565	9565	9565	**9580**
WEISH19	5*70	7756.9	7698	7674	**7698**	7674	7623	7661
WEISH22	5*80	9004.2	8947	8851	**8947**	8908	8856	**8947**
WEISH25	5*80	9964.7	9939	9923	9923	9923	9923	**9939**
WEISH26	5*90	9641.6	9584	9532	**9584**	**9584**	9532	9532
WEISH27	5*90	9849.7	9819	9811	**9819**	9811	9780	**9819**
WEISH29	5*90	9429.03	9410	**9410**	9354	9378	**9410**	**9410**
WEISH30	5*90	11194.5	11191	11160	**11191**	**11191**	11146	**11191**
SENTO1*	30*60	7839.3	7772	**7772**	7719	7728	7761	7761
SENTO2	30*60	8773.2	8722	8711	8701	**8722**	8704	8698

Table VI. General test results, 10*N iterations

results are obtained for some of the longer lists, but tests with other test cases supported the merit of using the first-level tabu length value.

Probabilistic TS and SO. We tried probabilistic move acceptance together with strategic oscillation as outlined in Section 6.3. The function of the SO is here to bias the ranking of the available moves of type I and II to favor either integer feasibility (move type III and I) or to favor objective function value (move type III and II).

Results for Probabilistic TS. We ran two series of tests for each test case, one running for 10*N iterations, to compare the results with those obtained for the first-level TS, and the other for 20*N iterations to check the assumption that probabilistic measures need more search time, as the search tends to be less focused. The results are reported in Table VI and VII respectively.

The tests for probabilistic move acceptance use $p = 0.4$, and are reported under the column PMA. The column PMA0 reports the probabilistic move acceptance without tabu memory, also with $p = 0.4$, while PTT reports probabilistic tabu tenure results using the same first-level tabu list length as for the first-level TS. PSO reports

Problem	M*N	LP	IP	First-level TS	PMA	PMA0	PTT	PSO
PET5	10*28	12462.1	12400	12370	**12400**	12390	12390	**12400**
PET7	5*50	16612	16537	16524	16508	16524	16468	16468
PB4**	2*29	99622.7	95168	90909	93118	90909	90909	**95168**
PB6**	30*40	843.3	776	723	765	729	745	765
PB7**	30*37	1086.2	1035	1033	**1035**	1011	1021	1011
WEISH07*	5*40	5601.9	5567	5525	**5567**	5541	5541	**5567**
WEISH08	5*40	5631.6	5605	5603	5603	**5605**	5603	5603
WEISH16	5*60	7314.02	7289	7287	7287	7288	7287	7288
WEISH18	5*70	9603.7	9580	9565	**9580**	9565	**9580**	**9580**
WEISH19	5*70	7756.9	7698	**7698**	**7698**	7674	7674	**7698**
WEISH22	5*80	9004.2	8947	8886	**8947**	8908	8886	**8947**
WEISH25	5*80	9964.7	9939	9928	9928	9923	9923	**9939**
WEISH26	5*90	9641.6	9584	9532	**9584**	**9584**	9542	9578
WEISH27	5*90	9849.7	9819	**9819**	**9819**	9811	9780	**9819**
WEISH29	5*90	9429.03	9410	**9410**	9354	**9410**	**9410**	**9410**
WEISH30	5*90	11194.5	11191	**11191**	**11191**	**11191**	11181	**11191**
SENTO1*	30*60	7839.3	7772	**7772**	7719	**7772**	7761	7761
SENTO2	30*60	8773.2	8722	8711	8701	**8722**	8704	**8722**

Table VII. General test results, 20*N iterations

probabilistic move acceptance together with strategic oscillation, with $k = 1.9$ and basic oscillation frequency set to 2*N/3 (See Section 11.5). The symbol '*' indicates that $w = 0.5$, while the symbol '**' indicates $w = 0.25$. Optimal values are indicated in bold. (The slight discrepancy between the results obtained here and those reported for the first-level TS are due to a change in the random number generator used.)

As can be seen from the tables, the use of probabilistic measures for the move selection function improves the general solution quality. The use of a probabilistic tabu tenure, by contrast, gave the same type of result as the first-level tabu search approach of our earlier study. This is to be expected, since the first-level tabu search approach also incorporates a measure of controlled randomization, which has been designed to appropriately reflect the amount of *tabu influence*.

The tables also disclose the expected outcome that solution quality is clearly correlated with the number of iterations. The tables do not reflect the observation, however, that the probabilistic move acceptance methods find good solutions very quickly compared to the first-level TS approach. This is especially true when no tabu memory is employed, but the method without memory has difficulty in improving on those early good solutions.

Since the "no memory" method actually employs a tabu tenure of 1, these outcomes suggest the relevance of using a progressively varying tenure for probabilistic tabu search, which begins very small to find high quality solutions rapidly, and then gradually enlarges. A periodic return to very small tenures, particularly at the conclusion of diversification steps, likewise would appear useful, based on our findings.

Combining probabilistic move acceptance with a simple strategic oscillation scheme (to alter the move evaluation function) works very well, though occasionally its outcomes are not quite as good as the best obtained by our other approaches.

We also checked our heuristics against some of the presumably easier cases solved by the first-level TS approach, and found results similar to those reported above.

11.3 Including Intensification by Locking Variables

Whenever a new incumbent z^* is found, we introduce an intensification component that seeks to lock variables at their upper or lower bounds. This locking derives from the TS notion of exploiting *influential* or *strongly determined* variables (Glover, 1977), which in its original form is designed to be carried out in phases, on an intermediate term basis, allowing variables to be unlocked over the longer term and replaced by alternative candidates for locking. In our application, we instead applied a simple version that locks a variable permanently if it satisfies a particular threshold rule. The rule compares the relative profit (or reduced cost) of each variable at LP OPT with the gap between the objective function value at LP OPT, z_{OPT}, and z^*. The reduced cost, d_j, for variable x_j, is calculated as follows:

$$d_j = c_j - \sum_{i=1,M} u_i * a_{ij}$$

where the u_i are the shadow prices. Then, if a variable x_j is at its upper bound, it can be locked at 1 if

$$d_j \geq z_{OPT} - z^*$$

On the other hand, if x_j is at its lower bound, it can be locked at 0 if

$$-d_j \geq z_{OPT} - z^*$$

The concept of locking variables may at first be imagined to be simply a time-saver. However, the effect is more profound. In essence, it serves to create a form of *combinatorial implosion*, by reducing the number of combinatorial possibilities in a manner exactly opposite to the increase of such possibilities created by the effect of combinatorial explosion. This means the search can focus on a subspace where the chances of finding an optimal solution (relative to that space) are greatly enhanced. In our case, the rules we applied for locking variables, combined with TS, had a significant effect on both the efficiency of the search, and on the quality of the solutions obtained. The first-level TS search mechanisms outlined in Sections 1 to 5, with the intensification mechanism of locking applied, actually found the optimum in 15 of the 18 remaining test cases, and ultimately locked up to 87% of the integer variables (79 out of 90 for WEISH29), usually in a relatively small number of pivots. For the remaining 3 test cases; PET7, PB4, and PB6, the gap was either too big to allow any locking at all for the values of z^* obtained, or too few variables got locked to have any effect. Anticipated reasons for these results are that the TS did not have to spend search effort trying variables that already had their proper values, and that the TS in itself is able to give high-quality solutions that are able to reduce the gap sufficiently to let locking have an effect.

The only alteration to the search parameters required to accompany the locking of variables, was to dynamically reduce the length of the tabu list to reflect the smaller number of variables, with the new value being

$$TLL_{new} = TLL_{org} * SQRT(N_{new} / N)$$

where N_{new} identifies the number of remaining integer variables.

It is also interesting to notice, with reference to Table I, that the problems for which locking had little or no effect were among those that have the largest distance between the LP OPT and the 0/1 MIP optimum solution.

Based on these findings, we chose to include the variable locking strategy in our general procedure for the remaining test problems. We note that it is entirely possible to extend the strategy to determine candidates for locking by making use of penalty calculations for basic variables as well as nonbasic variables, and by incorporating strategic thresholds other than determined by reference to z_{OPT} (particularly where the search suggests that a significant duality gap may occur). In the latter case, the candidates can appropriately be identified relative to reduced costs at extreme points

	1.2	1.5	1.9	1.95	*Average*
N/2	16499	16508	16510	16537	*16514*
2*N/3	16524	16519	16524	16499	*16517*
N	16472	16504	16499	16537	*16503*
3*N/2	16518	16507	16510	16499	*16509*
Average	*16503*	*16510*	*16511*	*16518*	

Table VIII. SO values for PET7

	1.2	1.5	1.9	1.95	*Average*
N/2	91935	91935	91670	94308	*92462*
2*N/3	90909	90909	94965	91721	*92126*
N	90909	90417	94308	90909	*91635*
3*N/2	90909	91935	95168	94308	*93080*
Average	*91166*	*91299*	*94028*	*92812*	

Table IX. SO values for PB4

other than an LP optimal extreme point, including for example the extreme points for a few of the best MIP feasible solutions found. Then, by estimating the maximum improvement in the current solution that might be contributed from the profitable nonbasic variables, a threshold for reduced costs of other nonbasic variables can be imposed. Since our goal at this stage was primarily to establish a "proof of concept" for applying the variable locking strategy, we restricted our implementation to the simple one that only makes reference to an LP optimal point.

11.4 Results for Strategic Oscillation

We tested SO both by parametric evaluation and by altered choice rules, as outlined in Sections 7.1 and 7.2, respectively. The tests were based on the first-level TS, and were run for 20*N iterations, with results shown in Table XI.

SO by parametric evaluation. For these tests we used the *Alternating Integer and Cost Priority* scheme. This version of SO is controlled by two parameters, the emphasis shift, p, and the Oscillation Frequency, f_{so}. To find out good values for the pair of parameters, we ran similar tests on PET7 and PB4. The results are shown in Tables II and III for selected values of p in the range 1.2 to 1.95 and f_{so} in the range N/2 to 2*N. The average of the best values found for the various parameters are also shown. These results indicate p = 1.9 and f_{so} = 2*N/3 as good general values to use in further testing.

SO by altered choice rules. This can be considered an intensified version of *SO by parametric evaluation*, in that the value of the parameter p can be considered to be given only the values of 0 and 2, driving the preferences to the extreme in both directions alternately. The only parameter we need to consider in this case is thus the SO frequency. Preliminary testing showed no clear pattern, but it seems advisable to

Div - Delay	PET7	PB4	PB6
N	16519	94461	723
1.5*N	16519	95168	776
2*N	16519	95168	723
2.5*N	16470	92414	776
3*N	16519	94308	776
4*N	16490	94308	723
5*N	16513	94308	723
6*N	16517	94308	723

Table X. Diversification

avoid values for f_{so} too close to N, and a value for f_{so} = 2*N/3 again gives a good result.

11.5 Results for Diversification

As stated in Section 8.1, penalizing time spent in the basis did not give the desired diversification effect. Diversification based on penalizing the move value is also awkward, as there is no comparable measure for the different move types, and such a scheme would have its primary effect only within each move type group.

The application of Target Analysis (see Section 9) to this facet of the problem indicated another avenue of approach, namely *diversification by intensification* (based on including promising variables), as outlined in Section 8.2. For this approach, good values for the following three parameters need to be determined:

Div-Delay - Diversification Delay. Iterations between diversification attempts.

D_{PT} - Diversification Pull Time. Max. no. of iterations to try to pull the selected variable through the basis and out at the desired bound.

D_{TT} - Extra Diversification Tabu Tenure. Extra number of iterations to keep the selected variable tabu for entering the basis again.

Preliminary testing disclosed that it always took less than N iterations to pull the selected variable through the basis and out at its upper bound. The value used for later tests was D_{PT} = N+M, with the purpose of aborting diversification attempts using excessively long time. Similarly, we found that a value of $D_{TT} \geq 2*N$, yielded the best results. Table X shows the results for varying Div-Delay over the range n to 6*N. As can be seen, the best values are obtained for Div-Delay in the range 1.5*N to 3*N.

Results for SO and Diversification. Table XI shows the overall computational results for the remaining test cases where the optimum was not found by the first-level TS of Section 1 or by the locking of variables as described in Section 11.3. We ran all the tests for 20*N iterations, as the proposed scheme needs more search time than the first-level TS. The test for *SO by parametric evaluation* is reported under the column *SO - param*, with a value of p = 1.9 and f_{so} = 2*N/3. The column *SO - choice* reports the results for *SO by altered choice rules*, also with f_{so} = 2*N/3, and DIV reports the results from the diversification tests, with Div-Delay = 1.5*N, D_{TT} = 2*N and D_{PT} = N. The symbol '**' indicates w = 0.25. Optimal values are indicated in bold, and italics indicates improvements over the first-level TS.

As is evident from Tables VIII and IX, better solution values can be obtained by tailoring the search parameters to each test case. We also checked our heuristics against some of the presumably easier cases solved by the first-level TS approach, and found results similar to those reported above.

12. Conclusions

The preceding strategic elements for solving zero-one mixed IP problems can be combined in a variety of ways. The fundamental theme of these observations is that extreme point solution processes have features that are highly exploitable by tabu search methods, and that this exploitation introduces novel considerations for balancing tradeoffs between different aspects of feasibility and optimality. We anticipate that these considerations, and the specific strategies derived from them, can be usefully adapted to create associated tabu search methods for solving other problems in which optimal solutions likewise occur at extreme points, as where the goal is to minimize a concave function over a convex polyhedral set.

Problem	M*N	LP	IP	First-level TS	SO - param	SO - choice	DIV
PET7	5*50	16612	16537	16524	16524	**16537**	16519
PB4**	2*29	99622.7	95168	90909	*94965*	*94308*	**95168**
PB6**	30*40	843.3	776	723	*765*	*765*	**776**

Table XI. Overall computational results

Our approaches gets results comparable to, and in some cases better than, solutions for the multiconstraint knapsack problem obtained by special purpose heuristics designed to take advantage of the special structure of these problems, even though our heuristics are designed to solve general zero-one mixed IP problems, and no problem specific knowledge is embedded in our system.

We have also shown how simple probabilistic measures can be used to improve a first-level TS significantly, enhancing diversifying aspects of the search. The use of probabilistic move acceptance is especially important if the move evaluation function is contaminated by noise or contains elements not directly related to the objective function value. We have also shown how a simple ranking scheme can give superior results when the move evaluation function is difficult to compare numerically for different parts of the search neighborhood. Our approach, which does not depend on any “deep formula,” is appealing in its simplicity, and is easily incorporated into other TS designs.

It is also noteworthy that we obtained relatively good – although not our best – results from the “degenerate” version of the probabilistic move acceptance method, where we abandoned tabu memory altogether (apart from the single iteration memory that prohibits an immediate move reversal). This approach appears useful when it is imperative to find good solutions quickly, or when tabu attributes might be difficult to identify or separate. These outcomes further suggests the value of combining the probabilistic design with a variable tabu tenure that periodically receives much smaller values than customarily envisioned to be relevant. Our findings indicate that an additional stipulation deserves to be added to the hypothesis that probability, appropriately used, may partly substitute for the role of memory. Apparently, the right use of probability can also enhance the effectiveness of memory.

Our results show how the relatively simple measures described for strategic oscillation and diversification can be used to improve a first-level TS significantly. The potent effect of the simple locking mechanism is also intriguing. This is a clear indication of the importance of being able to identify consistent or strongly determined variables. The outcome of locking can also be viewed as an indicator of the importance of the combinatorial implosion effect, when based on an intelligent use of information. The application of Target Analysis was instrumental in guiding us to identify such information (though we may not have found all forms of information that would be relevant), and also was useful for illuminating the appropriateness of strategic oscillation. The mechanisms outlined here are easily combined, as is seen in Section 6.3 where *SO by parametric evaluation* is successfully combined with *probabilistic move acceptance*.

Our proposed schemes are all restricted to work within the basic simplex extreme point pivoting framework. Further improvements are anticipated to be available by going beyond this restriction, e.g. by adaptive rules for directly assigning values to integer variables, and also by using more sophisticated candidate list strategies. Our ideas can be exploited in connection with interior point methods by making use of rounding and by incorporating an evolving network of reference points as in scatter search (Glover, 1977, 1996). Opportunities also exist to take advantage of cutting planes, to create a "search and cut" basis for our TS approach. (The TS memory structures suggest the use of strategies that impose potentially invalid cuts which may be discarded and "forgotten" as the search progresses.) We are currently exploring these avenues.

References

Aboudi R., and Jörnsten, K. (1994), “Tabu Search for general Zero-One Integer Programs using the Pivot and Complement Heuristic”, *ORSA Journal of Computing*, 6/1, 82–93.

Balas, E., and Martin, C. (1980), “Pivot and Complement - a Heuristic for 0-1 Programming”, *Management Science* 26/1, 86–96.

Cabot, A.V., and Hurter, A. P. (1968), “An Approach to Zero-One Integer Programming”, *Operations Research* 16, 1206–1211.

Connolly, D. (1992), “General Purpose Simulated Annealing”, *Journal of the Operational Research Society*, 43/5, 495-505.

Dammeyer F., and Voss, S. (1993), “Dynamic tabu list management using the reverse elimination method”, *Annals of Operations Research* 41, 31–46.

Dowsland, K. A. (1993). “Simulated Annealing”, in: C. Reeves, (ed.), *Modern Heuristics for Combinatorial Problems*, Blackwell Scientific Publishing

Drexl A. (1988), “A Simulated Annealing Approach to the Multiconstraint Zero-One Knapsack Problem”, *Computing* 40, 1–8.

Gent, I., and Walsh, T. (1994), “How Not To Do It”, Presented at the AAAI Workshop on Experimental Evaluation of Reasoning and Search Methods.

Glover, F. (1968), “A note on Linear Programming and Integer Infeasibility”, *Operations Research* 16, 1212–1216.

Glover, F. (1977), “Heuristics for Integer Programming Using Surrogate Constraints”, *Decision Sciences*, 8/1, 156–166.

Glover F. (1989). “Tabu Search - Part I”, *ORSA Journal of Computing*. 1/3, 190-206.

Glover, F. (1995a), “Tabu Thresholding: Improved Search by Nonmonotonic Trajectories”, *ORSA Journal on Computing*, 7/4, 426-442.

Glover, F. (1995b), “Tabu Search Fundamentals and Uses”, Graduate School of Business, University of Colorado (shortened version published in *Mathematical Programming: State of the Art*, Birge and Murty, eds., University of Michigan Press, 1994)

Glover, F. (1996), “Tabu search and Adaptive Memory Programming - Advances, Applications and Challenges”, *Interfaces in Computer Science and Operations Research*, Barr, Helgason and Kennington, eds., Kluwer Academic Publishers, 1–75.

Glover, F. and Greenberg, H. (1989). “New approaches for heuristic search: A bilateral linkage with artificial intelligence”. *European Journal of Operations Research 39*, 119–130.

Glover, F., and Laguna, M. (1993), “Tabu Search”, in: C. Reeves, (ed.), *Modern Heuristics for Combinatorial Problems*, Blackwell Scientific Publishing.

Glover, F. and McMillan, C. (1986). The General Employee Scheduling Problem: An integration of MS and AI. *Computers and Operations Research*, Vol 13. No. 5. pp 563-573.

Glover, F., Taillard, E., and de Werra, D. (1993), “A User’s Guide to Tabu Search”, *Annals of Operations Research*, 41, 3–28.

Hart, J. P., and Shogan, A. W. (1987), “Semi-Greedy Heuristics: An Empirical Study”, *Operations Research Letters*, 6/3, 107-114

Hooker, J. N. (1995), “Testing Heuristics: We Have It All Wrong”. *Journal of Heuristics*, 1/1, 33-42.

Laguna, M., (1990). “The application of heuristic methods and artificial intelligence to a class of production scheduling problems”, Ph. D. dissertation, University of Texas at Austin.

Laguna M., and Glover, F. (1993), “Integrating Target Analysis and Tabu Search for Improved Scheduling Systems”. *Expert Systems With Applications***,** 6, 287–297.

Løkketangen, A., and Glover, F. (1995), “Tabu Search for Zero/One Mixed Integer Programming with Advanced Level Strategies and Learning”. *International Journal of Operations and Quantitative Management*, 1/2, 89–108

Løkketangen, A. and Glover, F. (1996a), “Probabilistic Move Selection in Tabu Search for 0/1 Mixed Integer Programming Problems”, in: J. Kelly and I. Osman (eds.), *Meta-Heuristics: Theory and Applications*. Kluwer Scientific Publishers, 467–488.

Løkketangen, A. and Glover, F. (1996b), “Surrogate Constraint Analysis – New Heuristics and Learning Schemes for Satisfiability Problems”, Proceedings of the *DIMACS workshop on Satisfiability Problems: Theory and Applications*, D-Z. Du, J. Gu and P. Pardalos, eds.

Løkketangen, A., Jörnsten, K. and Storøy, S. (1994), "Tabu Search Within a Pivot and Complement Framework", *International Transactions of Operations Research*, 1/3, 305–316.

Løkketangen, A., and Woodruff, D. L. (1996), "Progressive Hedging and Tabu Search Applied To Mixed Integer (0,1) Multi-Stage Stochastic Programming", *Journal of Heuristics*, Vol 2, No 2, 111–128.

Marsten, R. E. (1989a), "XMP Technical Reference Manual", XMP Software.

Marsten, R. E. (1989b), "User's Manual for ZOOM/XMP", XMP Software.

Mulvey, J. (1995), "Generating Scenarios for the Towers Perrin Investment Systems", Technical Report SOR Report, Princeton University, Department of Civil Engineering and Operations Research, Princeton University, Princeton, NJ, to appear in *Interfaces*.

Petersen, C.C. (1967), "Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R&D Projects", *Management Science* 13, 736–750.

Ryan, J. (1995). "The Depth and Width of Local Minima in Discrete Solution Spaces", *Discrete Applied Mathematics*. 56/1, 75–82.

Senyu, S., and Toyoda, Y. (1968), "An Approach to Linear Programming with 0-1 variables", *Management Science* 15, 196–207.

Shih, W. (1979), "A Branch and Bound Method for the Multiconstraint Zero-One Knapsack Problem", *Journal of the Operational Research Society* 30, 369–378.

Weingartner H.M., and Ness, D. N. (1967), "Methods for the Solution of Multi-dimensional 0/1 Knapsack Problems", *Operations Research* 15, 83–103.