Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computers & Operations Research 33 (2006) 2449–2494

computers & operations research

www.elsevier.com/locate/cor

Parametric tabu-search for mixed integer programs

Fred Glover

Leeds School of Business, University of Colorado, Boulder, CO 80309-0419, USA

Available online 10 October 2005

Abstract

A parametric form of tabu-search is proposed for solving mixed integer programming (MIP) problems that creates and solves a series of linear programming (LP) problems embodying branching inequalities as weighted terms in the objective function. The approach extends and modifies a parametric branch and bound procedure of Glover [Parametic branch and bound. OMEGA, The International Journal of Management Science 1978;6:1–9], replacing its tree search memory by the adaptive memory framework of tabu-search and introducing new associated strategies that are more flexible than the mechanisms of branch and bound.

We also introduce new types of intensification and diversification procedures based on scatter search and frequency analysis, and also based on a variant of adaptive memory called *model embedded memory*. In one form this approach incorporates recency and frequency memory within the objective function structure of the parameterized LP formulations. In another form it generates valid inequalities from the optimized objective of the parameterized LP problem, and uses these as a supplement to traditional types of tabu memory. The integrated parametric approach affords a spectrum of variations that include useful simplifications for pure 0-1 MIP problems and problems with special structures, such as GUB constraints. The approach exploits both spatial and logical relationships, and is appropriate for discovering feasible solutions, as in the case of *satisfiability* problems. It is particularly designed to provide adaptive strategies for problems that are difficult for branch and bound and for branch and cut procedures, as opposed to problems these latter methods can handle effectively, and hence can be used to complement or supplement these more traditional types of methods.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Mixed integer programming; Tabu search; Metaheuristics; Adaptive memory

E-mail address: fred.glover@colorado.edu.

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.07.009

0. Introduction

This paper gives a general design for a parametric tabu-search method for mixed integer programming (MIP) problems. The approach embraces a collection of interlinking components that can be varied to provide a range of tradeoffs between strategic sophistication and ease of implementation. The method therefore constitutes a framework for comparative studies to determine the relative efficacy of different strategic options, each emerging from the same fundamental blueprint. To clarify the interplay of basic ideas, a series of numerical examples is provided to show their operation. In addition, several new types of strategies are proposed for uncovering feasible and high-quality solutions to MIP problems, making use of spatial and logical interdependencies.

1. Notation and problem formulation

We represent the MIP problem in the form

$$\begin{array}{lll} \text{(MIP)} & \text{minimize} & x_o = cx + dy \\ & \text{subject to} & (x, y) \in Z, \\ & & x \in X, \end{array}$$

where

$$Z = \{(x, y) : Ax + Dy \geqslant b, U \geqslant x \geqslant 0\},$$

$$X = \{x : U \geqslant x \geqslant 0 \text{ and } x \text{ integer}\}.$$

The vector U is permitted to have infinite components and in the case of *pure* integer programming problems the vector y of continuous variables can have 0 dimension. The foregoing slightly unorthodox representation, where the sets X and Z both impose the bounds $U \geqslant x \geqslant 0$, facilitates the description of the method subsequently introduced. We also assume the inequality constraints summarized by $Ax + Dy > b$ include an objective function constraint $cx + dy \leqslant x_o^* - \varepsilon$ (written in $\geqslant$ form), where x_o^* is the x_o value for the currently best-known solution to (MIP) and ε is a small positive number. We therefore assume the set Z is successively modified by updating the objective function constraint as new incumbent solutions are found. Our approach is compatible with a strategy of choosing ε to be somewhat larger than necessary, followed by reducing its value if no feasible solutions are generated within a chosen number of iterations. We also subsequently introduce modifications of the set X that incorporate additional restrictions on x that are contained in or implied by $(x, y) \in Z$.

The linear programming (LP) relaxation of (MIP), which contains only the restriction $(x, y) \in Z$, will be denoted by (LP). We say a vector (x, y) is LP feasible if it is feasible for (LP) but not necessarily for (MIP), hence if it satisfies $(x, y) \in Z$ but possibly not $x \in X$. We call x integer feasible if the components of x are integers, and hence an (MIP) feasible solution is one that is LP feasible and integer feasible.

2. Foundations of parametric tabu-search

Our solution approach consists of a parametric form of tabu-search utilizing moves based on the approach of parametric branch and bound [1]. The tabu-search framework amends parametric branch and bound by replacing its tree search memory structure with an adaptive memory structure that provides greater flexibility and facilitates the use of strategies outside the scope of tree search.

Let N^+ and N^- denote selected subsets of $N=\{1, 2, \dots, n\}$, the index set for x. Also, let $N'=N^+ \cup N^-$, and let x' denote an associated trial vector satisfying $x' \in X$. For convenience of terminology we understand the vector x' to be the partial vector consisting of the components x'_j for $j \in N'$, disregarding components for $j \in N - N'$. Parametric branch and bound uses a variant of L1 norm minimization to maintain LP feasibility while seeking to enforce the following conditions:

$$\text{(UP)} \quad x_j \geqslant x'_j \quad \text{for } j \in N^+ \text{ (provided } x'_j > 0\text{)},$$

$$\text{(DN)} \quad x_j \leqslant x'_j \quad \text{for } j \in N^- \text{ (provided } x'_j < U_j\text{)}.$$

Stipulating $x'_j > 0$ in (UP) and $x'_j < U_j$ in (DN) avoids consideration of redundant inequalities. The condition $x_j = x'_j$ is handled by allowing j to belong to both N^+ and N^-. A slight variation on the foregoing representation introduces two different instances of x'_j, one for N^+ and one for N^-, thereby permitting (UP) and (DN) to bracket x_j between two different values. However, by our present formulation, j can be an element of both N^+ and N^- only in the case where the preceding inequalities constrain x_j to the single value x'_j. By convention, throughout this paper we may speak of a variable belonging to a set as shorthand for saying that its index belongs to the set (as in referring to x_j as an element of some specified subset of N').

We refer to (UP) and (DN) as *goal conditions* and call x'_j the *goal value* in these conditions, because we do not seek to enforce (UP) and (DN) directly by imposing them as constraints in the manner of customary branch and bound procedures but rather indirectly by incorporating them into the objective function of the linear programming relaxation (LP).[1] This can be done by means of the following linear program, where M denotes a large positive number:

$$\begin{aligned} \text{(LP}'\text{)} \quad \text{minimize} \quad & u_0 = cx + dy + M(\Sigma(u_j : j \in N^-) + \Sigma(v_j : j \in N^+)) \\ \text{subject to} \quad & (x, y) \in Z, \\ & x_j = x'_j + u_j - v_j, \quad u_j, v_j \geqslant 0, \quad j \in N'. \end{aligned}$$

We say that (LP′) *targets* the inequalities of (UP) and (DN). Evidently, the inequalities will be satisfied in an optimal solution to (LP′) if and only if the problem (LP) has a feasible solution when expanded to include these inequalities.

The representation of (LP′) simplifies in the special cases, where $x'_j = U'_j$ for $j \in N^+$ and $x'_j = 0$ for $j \in N^-$ since we may then modify the objective by, respectively, replacing u_j by x_j, $j \in N^-$ and replacing v_j by $U_j - x_j$, $j \in N^+$, and do not need to introduce the constraints associated with u_j and v_j. Thus, letting N_U denote the subset of N^+ such that $x_j = U'_j$, and N_0 denote the subset of N^- such that $x_j = 0$, (LP′) can be written more precisely in the form

$$\begin{aligned} \text{(LP}'\text{)} \quad \text{minimize} \quad & u_0 = cx + dy + M(\Sigma(x_j : j \in N_0) + \Sigma(U_j - x_j : j \in N_U) \\ & \qquad + \Sigma(u_j : j \in N^- - N_0) + \Sigma(v_j : j \in N^+ - N_U)) \\ \text{subject to} \quad & (x, y) \in Z, \\ & x_j = x'_j + u_j - v_j, \quad u_j, v_j \geqslant 0, \quad j \in N' - N_0 - N_U. \end{aligned}$$

[1] It is possible as in parametric branch and bound to maintain a memory of additional goal conditions apart from those that define (LP′) which are currently superseded by those specified here.

The terms U_j in the objective can be removed by introducing a single constant term $c_o = M\Sigma(U_j : j \in N_U)$. Evidently, in the case of 0-1 MIP problems no u_j or v_j variables are required.

From an implementation standpoint, a two-phase approach can be used for optimizing (LP′) as an option to using the explicit form of the objective. That is, the objective of (LP′) can be equivalently handled by first creating a primary objective that sets $M = 1$ and disregards the $cx + dy$ component. Then, at optimality, all non-basic variables are held constant at their currently assigned (lower or upper) bounds, removing them from further consideration. The second phase then is implemented by optimizing $cx + dy$ over the residual system. Such an implementation is useful to reduce computer round-off error, and is relevant in special cases discussed later.

In fact, parametric branch and bound does not use a single "large *M*" value in the objective, but applies varying weights M_j^- and M_j^+ to different u_j and v_j (or x_j and $U_j - x_j$) variables, and successively adjusts these weights as the method progresses. The approach also introduces a modification of the primal simplex method that allows the problem to be solved without explicitly introducing the additional variables u_j and v_j. Our present adaptation of this framework likewise takes advantage of different weights M_j^- and M_j^+ (handled either directly or in a two-phase approach), but employs these weights in a different way as a means of incorporating the influence of tabu-search memory within the structure of (LP′). Our current focus will not be concerned with the specialized processes for avoiding the explicit introduction of the u_j and v_j variables, but we emphasize the value of being able to proceed from one instance of (LP′) to another by post-optimizing with the primal simplex method, a feature that yields a considerable saving in computational effort.[2] Subsequently, we also describe a special variant of our approach that permits its post-optimizing steps to be carried out by a dual LP method, thus permitting the use of the same types of LP subroutines customarily used in today's B&B approaches to MIP.

While parametric tabu-search makes reference to the same parameterized problem (LP′) as parametric branch and bound, it exploits the problem in a different way, by introducing sets of strategies to take advantage of the flexible forms of memory guidance available through tabu-search, in contrast to the more rigid memory structures of branch and bound. To apply parametric TS we begin from a given instance of (LP′) (that corresponds to the original LP relaxation of (MIP) if N' is empty), and denote its optimal solution by (x'', y''). We will often be interested only in the values of the integer variables in this solution, and in such cases we take the liberty of referring to x'' as the solution to (LP′), understanding y'' to be implicit. (The values of u_j and v_j in the solution are automatically determined by x'' and y'', and we consider them to be implicit as well.) If the solution is integer feasible (x'' is a vector of integers), then it qualifies as a new best solution since the objective function constraint is incorporated in the problem formulation. Consequently the value of x_o^* is updated each time an integer feasible solution is obtained.

The parametric TS method then proceeds by using information from the solution to (LP′), including the relationship between x'' and x' to determine a new x' vector and an associated new problem (LP′). Then the method repeats, continuing in this manner until reaching a chosen limit on the total number of iterations. As the method progresses, TS memory concerning attributes of previous x' vectors and their associated (LP′) problems is used to shape the decisions for constructing new problems.

[2] The ability to post-optimize with the primal method is evident in the case of 0-1 problems, since there are no changes of variables and the two instances of (LP′) differ only in their objective function coefficients. In the general case, a substitution and possible addition or deletion of variables suffices to go from one instance of (LP′) to another, and this can readily be done in a manner to keep the current LP basis primal feasible by selecting which of u_j and v_j should be made basic.

2.1. (LP$'$) *Transitions*

Transitioning from one instance of (LP$'$) to another is based on rules of the following general form. There are two kinds of transitions (T-UP) and (T-DN) to create the goal conditions (UP) and (DN), which consist of defining (or re-defining) the goal value x'_j to be an integer value just above or just below the LP solution value x''_j. We also utilize a third type of transition (T-FREE) that frees the variable x_j from its goal conditions (UP) and/or (DN).

We express these transitions as follows:

(T-UP) Set $x'_j := \lfloor x''_j \rfloor + 1$ and add j to N^+ (to target $x_j \geqslant x'_j$).
(T-DN) Set $x'_j := \lceil x''_j \rceil - 1$ and add j to N^- (to target $x_j \leqslant x'_j$).
(T-FREE) Remove j from N' (to release x_j from (UP) and (DN)).

The values $\lfloor x''_j \rfloor$ and $\lceil x''_j \rceil$ are the closest integers to x''_j, such that $\lfloor x''_j \rfloor \leqslant x'_j$ and $\lceil x''_j \rceil \geqslant x''_j$ (hence $\lfloor x''_j \rfloor + 1 = \lceil x''_j \rceil$ in (T-UP) and $\lceil x''_j \rceil - 1 = \lfloor x''_j \rfloor$ in (T-DN) unless x''_j is an integer). The operation of adding j to N^+ or N^- does not change the indicated set if it already contains j, although the associated goal condition will change as a result of re-defining the value of x'_j. The operation of removing j from N' implicitly removes the index from N^+ and/or N^-, and eliminates the associated goal condition(s).

The execution of these transitions for an appropriate set of variables rests on responding to two types of conditions, *goal infeasibility* and *integer infeasibility*. We examine each of these in turn.

2.2. Goal infeasibility

We say that an optimal solution $x = x''$ to (LP$'$) is *goal infeasible* if violates a current goal condition (UP) or (DN), hence if one of the following holds:

(V-UP) for some $j \in N^+$, $x''_j < x'_j$
(V-DN) for some $j \in N^-$, $x''_j > x'_j$.

Correspondingly, we call a variable x_j associated with a violation (V-UP) or (V-DN) a *goal infeasible variable*, and let $G = \{j \in N' : x_j \text{ is goal infeasible}\}$. We specify the *primary goal response* to such an infeasibility to consist of establishing a new goal in the opposite direction (changing the associated goal condition from UP to DN, or vice versa). Thus, for the two preceding types of violations, we obtain the respective (opposing) responses:

If (V-UP) holds, remove j from N^+ and execute (T-DN).
If (V-DN) holds, remove j from N^- and execute (T-UP).

Hence, in summary, these responses can be written in the form

Primary Goal Response for a selected subset G_P of G

(R-DN) If $x''_j < x'_j$ for $j \in N^+$, transfer j from N^+ to N^- and set $x'_j := \lfloor x''_j \rfloor + 1$.
(R-UP) If $x''_j < x'_j$ for $j \in N^+$, transfer j from N^- to N^+ and set $x'_j := \lceil x''_j \rceil - 1$.

In addition to the primary goal responses, we consider a *secondary goal response* that consists of freeing a goal infeasible variable. We call this response secondary (in the sense of subordinate) because we only execute it if one or more primary goal responses are also made, i.e., only if the selected subset G_{P} is non-empty.

Secondary Goal Response for a selected subset $\mathbf{G_S}$ of G

(R-FREE) If (V-UP) or (V-DN) holds, execute (T-FREE) (remove j from N').

We now consider how to evaluate the goal infeasible variables in order to choose which ones should make up the composition of the primary goal set $G_{\mathrm{P}} = \{j \in G : \text{ response } (R\text{-UP}) \text{ or } (R\text{-DN}) \text{ is made}\}$ and the secondary goal set $G_{\mathrm{S}} = \{j \in G : \text{ response } (R\text{-FREE}) \text{ is made}\}$.

2.2.1. Goal resistance

We create a measure based on the goal conditions in order to decide which responses from a collection of available responses (R-UP), (R-DN) and (R-FREE) are preferable. This measure, called the *goal resistance* $\mathrm{GR}_j(\mathrm{UP})$ or $\mathrm{GR}_j(\mathrm{DN})$ of variable x_j, $j \in G$, represents the amount by which the violation (V-UP) or (V-DN) resists the imposition of the corresponding goal condition (UP) or (DN), or more generally the amount (positive or negative) by which the countering response (R-DN) or (R-UP) causes the objective of (LP′) to improve. In particular, since (R-DN) and (R-UP), respectively, perform the transitions (T-DN) and (T-UP) that impel x_j to move counter to its goal condition, the resulting benefit to (LP′) is a measure of x_j's resistance to this goal condition. It is important to keep in mind that each goal infeasible variable has a unique response (R-DN) or (R-UP) assigned to it, according to the nature of its violation. Thus, we may speak of a single measure GR_j applicable to each variable, since we always know $\mathrm{GR}_j = \mathrm{GR}_j(\mathrm{UP})$ if the violation takes the form of (V-UP), and $\mathrm{GR}_j = \mathrm{GR}_j(\mathrm{DN})$ if the violation takes the form of (V-DN).

At the simplest level, GR_j can be the absolute value difference $|x''_j - x'_j|$ (identifying how distant the LP solution value x''_j is from its present goal value x'_j)[3] and at a higher level GR_j can be a post-optimization estimate of the (positive or negative) improvement in the objective of (LP′) caused by executing the response (R-DN) or (R-UP). Consequently, a larger goal resistance $\mathrm{GR}_j(\mathrm{UP})$ or $\mathrm{GR}_j(\mathrm{DN})$ indicates the relatively greater value of the countering response (R-DN) or (R-UP). It also suggests the value of the response (R-FREE), which involves a "partial counter move" that allows a subsequent determination of whether the full move should be made.

Our designation of (R-DN) and (R-UP) as primary responses and of (R-FREE) as a secondary response has the following significance: if two goal infeasible variables have been selected for consideration, with the purpose of executing one primary response and one secondary response, then the variable x_j with the larger goal resistance GR_j will be designated as the one to be handled by the primary response. More specifically, if we choose to place g_{P} goal infeasible variables in the primary goal set G_{P} (to be treated by the responses (R-DN) or (R-UP)) and to place g_{S} variables in the secondary goal set G_{S} (to be treated by the response (R-FREE)), then the g_{P} variables in G_{P} will be those having the largest goal resistance values over G and the g_{S} variables in G_{S} will have the next largest goal resistance values over G (i.e.,

[3] The value will be $\mathrm{GR}_j(\mathrm{UP}) = x'_j - x''_j > 0$ (the resistance to moving x_j up from x''_j to x'_j) or $\mathrm{GR}_j(\mathrm{DN}) = x''_j - x'_j > 0$ (the resistance to moving x_j down from x''_j to x'_j) corresponding, respectively, to the violations (V-UP) and (V-DN).

Table 1

1	$j=$	1	2	3	4
2	$x'_j=$	1	0	0	1
3	$x''_j=$	0.6	0.3	0.2	0.9
4	$\mathrm{GR}_j=$	0.4	0.3	0.2	0.1
	New $x'_j=$	0^*	1^*	#	1

$^*x_j$ changes its goal condition ($j \in G_\mathrm{P} = \{1, 2\}$).

$^\#x_j$ is freed from its goal condition ($j \in G_\mathrm{S} = \{3\}$).

the largest over $G - G_\mathrm{P}$). In the presence of goal infeasibility we always choose $g_\mathrm{P} \geqslant 1$, but may choose $g_\mathrm{S} = 0$. These policies are motivated by the observation that if the goal resistance values successfully reflect what they are intended to, then a change in a goal condition for a higher resistance value will have a greater impact than for a lower resistance value.[4]

2.2.2. *Illustrative examples*

In this and subsequent sections we provide a series of examples to illustrate the various components of our method. We assume for each that all integer variables are 0-1, and hence it suffices to indicate the goal value x'_j for x_j in order to know the associated goal condition, i.e., $x'_j = 0$ corresponds to the (DN) goal condition $x_j \leqslant 0$, and $x'_j = 1$ corresponds to the (UP) goal condition $x_j \geqslant 1$. We also use the simple measure of goal resistance given by $|x''_j - x'_j|$, indicating how far the LP solution value x''_j is from the goal value x'_j. In all cases, the variables will be indexed so that those with higher goal resistance values appear first.

Our first example, following, illustrates the allocation of variables to the sets G_P and G_S. Specific rules for choosing the values $g_\mathrm{P} = |G_\mathrm{P}|$ and $g_\mathrm{S} = |G_\mathrm{S}|$ are given in Appendix A.

Example A. Consider the situation depicted in Table 1 , involving four goal infeasible variables x_1, x_2, x_3 and x_4 (hence $G = \{1, 2, 3, 4\}$). The goal values x'_j on line 2 (corresponding, respectively, to the goal conditions $x_1 \geqslant 1$, $x_2 \leqslant 0$, $x_3 \leqslant 0$ and $x_4 \geqslant 1$) are combined with the solution values x''_j for (LP′) on line 3 to yield the goal resistance values $\mathrm{GR}_j = |x''_j - x'_j|$ on line 4. Thus, we note that $\mathrm{GR}_j = \mathrm{GR}_j(\mathrm{UP})$ for $j = 1$ and 4 (measuring the resistances to the (UP) conditions) and $\mathrm{GR}_j = \mathrm{GR}_j(\mathrm{DN})$ for $j = 2$ and 3 (measuring the resistances to the (DN) conditions). Line 3 also confirms that the x_j variables have been indexed in decreasing order of the GR_j values.

For this example we have chosen $g_\mathrm{P} = 2$ and $g_\mathrm{S} = 1$, allocating two variables to be in the primary goal set G_P and one variable to be in the secondary goal set G_S. Since the variables are indexed so that the largest goal resistance values come first, x_1 and x_2 are therefore selected to go in G_P and x_3 is selected to go in G_S (i.e., $G_\mathrm{P} = \{1, 2\}$ and $G_\mathrm{S} = \{3\}$). To determine the handling of the elements of the primary set G_P, we see that x_1 and x_2, respectively, involve the violations (V-UP) and (V-DN), and the appropriate responses are, respectively, (R-DN) and (R-UP). Thus the new goal conditions $x_1 \leqslant 0$ and $x_2 \geqslant 1$ are created to

[4] More timid (lower impact) strategies may invert portions of the ranking order. These provide options for diversification.

replace the current conditions $x_1 \geqslant 1$ and $x_2 \leqslant 0$. (For 0-1 variables the proper response for elements of the primary goal set G_P is to replace $x_j \leqslant 0$ by $x_j \geqslant 1$ and vice versa.)[5] For handling the secondary set G_S, the variable x_3 is simply freed from its goalcondition. No action is taken for the remaining variable x_4, which retains its current goal condition.

The values on the bottom line of the table identify the appropriate new x'_j goal values to replace the current x'_j values to create the next problem (LP$'$). The * symbol indicates that the variable changes its goal condition (hence the new x'_j value differs from the current x'_j value) and the # symbol indicates that the variable is freed.

It may be noted that the responses indicated on the bottom line of the table remain unchanged if we specify $x''_1 = 1$ instead of $x''_1 = 0.6$. In this case $\mathrm{GR}_1 = 1$, and x_1 continues to be the variable with the largest goal resistance, illustrating the situation where a goal violation occurs for a variable assigned an integer value by the LP solution. The example also shows that the use of the simple goal resistance measure given by $\mathrm{GR}_j = |x''_j - x'_j|$ does not necessarily lead to a new x'_j goal value that is the nearest integer neighbor of the LP solution value x''_j. In the present case, the only new goal value that is a nearest integer neighbor occurs for the variable x_4, whose goal value did not change. (When the goal value changes, it will be a nearest neighbor value only if the current x'_j is not the nearest integer neighbor of x''_j, which is exactly opposite to the outcome when the goal value does not change.)

As shown in Section 3.1, tabu restrictions for certain goal conditions can modify the determination of G_P and G_S, by causing some variables to be excluded from membership in these sets.

2.2.3. *Potential goal infeasibility*

We expand the use of goal resistance measures by observing that goal infeasibility can be caused not only by the existence of violated goal conditions, but also by the existence of other goal conditions whose influence operates less overtly than through the violations (V-UP) and (V-DN). Specifically, if the value of M applicable to some currently satisfied goal condition is slightly decreased, then this particular goal condition may become violated and one or more presently violated goal conditions may instead become satisfied. We say that a variable is *potentially goal infeasible* if a limited change in M (of a magnitude specified either implicitly or explicitly) will drive x_j goal infeasible. To emphasize the difference between goal infeasible variables and potentially goal infeasible variables, we refer to the former as *overtly* goal infeasible variables.

Potentially goal infeasible variables are handled by creating an extended measure of goal resistance, denoted $\mathrm{GR}^{\mathrm{o}}_j$, related to the decrease in M that would be required to make variable x_j goal infeasible. At the first level we set $\mathrm{GR}^{\mathrm{o}}_j = -\mathrm{RC}_j$, where RC_j is the LP reduced cost for the variable u_j or v_j (or x_j or $U_j - x_j$) associated with an optimal solution to (LP$'$). By such a measure, smaller reduced costs correspond to greater goal resistance. (At LP optimality, $\mathrm{RC}_j \geqslant 0$ for all $j \in N'$ and $\mathrm{RC}_j = 0$ holds for all x_j with overt goal resistance.)[6] A more advanced measure can be based on primal post-optimization to determine the change in goal infeasibility that occurs by changing the coefficient M for given goal conditions embodied in (LP$'$).

[5] For general integer variables the goal conditions $x_j \geqslant x'_j$ and $x_j \leqslant x'_j$ are, respectively, replaced by the conditions $x_j \leqslant x'_j - h$ and $x_j \geqslant x'_j + k$, where h and k are positive integers not necessarily equal to 1.

[6] A variable that is x_j non-basic at its upper bound U_j has a non-positive reduced cost, but by convention we may refer to the non-negative reduced cost for the complementary variable $s_j = U_j - x_j$, that is the slack variable relative to the upper bound.

Table 2

1	$j=$	1	2	3	4	5	6
2	$x'_j=$	1	0	0	1	0	1
3	$x''_j=$	0.6	0.3	0.2	0.9	0	1
4	$\mathrm{GR}_j=$	0.4	0.3	0.2	0.1		
5	$\mathrm{GR}^{\mathrm{o}}_j=$					−4	−25
	New $x'_j=$	0^*	1^*	1^*	#	#	1

$^*x_j$ changes its goal condition ($j \in G_{\mathrm{P}} = \{1, 2, 3\}$).
$^{\#}x_j$ is freed from its goal condition ($j \in G_{\mathrm{S}} = \{4, 5\}$).

The same primary and secondary goal responses for overtly goal infeasible variables apply to variables that are potentially goal infeasible, by replacing the conditions (V-UP) and (V-DN) that signal overt goal violations with alternative conditions (V-UP$^{\mathrm{o}}$) and (V-DN$^{\mathrm{o}}$) that signal potential violations. Let T^{o} denote a threshold value that determines when a goal resistance $\mathrm{GR}^{\mathrm{o}}_j$ is large enough to qualify x_j as potentially goal infeasible. Then we may identify (V-UP$^{\mathrm{o}}$) and (V-DN$^{\mathrm{o}}$) as follows:

(V-UP$^{\mathrm{o}}$) for some $j \in N^+$, $x''_j = x'_j$ and $\mathrm{GR}^{\mathrm{o}}_j \geqslant T^{\mathrm{o}}$
(V-DN$^{\mathrm{o}}$) for some $j \in N^-$, $x''_j = x'_j$ and $\mathrm{GR}^{\mathrm{o}}_j = T^{\mathrm{o}}$.

By convention, we consider a variable as relevant for inducing a potential goal violation only if it exactly satisfies its goal condition ($x''_j = x'_j$) as opposed to over-satisfying this condition ($x''_j > x'_j$ for $j \in N^+$ or $x''_j < x'_j$ for $j \in N^-$).

It is possible to avoid specifying an explicit value for T^{o}, since we may instead specify that a given number of variables with the largest $\mathrm{GR}^{\mathrm{o}}_j$ values will be admitted to the category of potentially goal infeasible variables, thereby implicitly establishing T^{o} as the smallest $\mathrm{GR}^{\mathrm{o}}_j$ value for the variables admitted (with an implied perturbation for eliminating tied values). Implicit and explicit uses of thresholds are discussed in Appendix B.

We establish a hierarchy where overtly goal infeasible variables always rank higher in importance than potentially goal infeasible variables. In any ordering of variables by the size of their resistance values, the goal resistance values GR_j for overtly goal infeasible variables are considered to be preemptively larger than the $\mathrm{GR}^{\mathrm{o}}_j$ values for potentially goal infeasible variables. Thus, it is understood that $\mathrm{GR}_p \gg \mathrm{GR}^{\mathrm{o}}_q$ for any two variables x_p and x_q that are overtly and potentially goal infeasible, respectively.

The inclusion of potentially goal infeasible variables is important for two reasons: first, as previously illustrated we may select multiple goal infeasible variables on a given iteration to treat by the responses (R-DN) and (R-UP), and it can be desirable to include some of the potentially goal infeasible variables among those chosen; second, the use of tabu-search criteria to exclude some variables from being selected may disallow the selection of some or all overtly goal infeasible variables, causing variables that are potentially goal infeasible to move higher in the ranks of those considered for selection.

Example B. Consider the situation illustrated in Table 2, which enlarges Example A to include two additional variables x_5 and x_6. In contrast to x_1, x_2, x_3 and x_4, the variables x_5 and x_6 satisfy their goal conditions exactly. We measure the goal resistance values of these additional variables by the first-level measure $\mathrm{GR}^{\mathrm{o}}_j = -\mathrm{RC}_j$. As seen from line 5 of the table, the indicated reduced costs RC_j are relatively small compared to a "big *M*" value. Hence, a relatively small decrease in *M* would induce these variables to become overtly goal infeasible by making their reduced costs negative. (We may imagine the presence of variables with larger RC_j values, which rank lower than those shown, but which are not included in the table.)

In this example, we have selected $g_{\mathrm{P}} = 3$ and $g_{\mathrm{S}} = 2$. By the convention that the GR_j values dominate the $\mathrm{GR}^{\mathrm{o}}_j$ values, it may be verified that the variables are indexed in descending order of their resistance measures. Therefore, we select the first three variables x_1, x_2 and x_3 to compose the primary goal set G_{P} and the next two variables x_4 and x_5 to compose the secondary goal set G_{S}. As indicated on the bottom row of the table, the variables x_1, x_2 and x_3 therefore change their goal conditions (replacing $x_1 \geqslant 1$, $x_2 \leqslant 0$ and $x_3 \leqslant 0$ by $x_1 = 0$, $x_2 \geqslant 1$ and $x_3 \geqslant 1$), while variables x_4 and x_5 are freed from their goal conditions. The variable x_6 is unaffected and therefore retains its current goal condition.

Rules for choosing g_{P} and g_{S} for the case of potential goal infeasibility are likewise discussed in Appendix A.

2.3. Integer infeasibility

Subordinate to goal infeasibility is the condition of *integer infeasibility*, which occurs if the solution $x = x''$ assigns a fractional (non-integer) value to some component of x_j of x. Let $F = \{j \in N : x_j = x''_j \text{ is fractional}\}$. We call a variable x_j an *unrestricted fractional* variable if x_j is fractional but not goal infeasible, hence such variables are represented by the index set $D = F - G$.[7] (Just as *G* represents the set of variables that fail to satisfy their Goal conditions, *D* is the set of variables, excluding those in *G*, that fail to satisfy their Discrete conditions. *D* is also called the set of *deviating* variables.) A variable x_j can be an unrestricted fractional variable even if it is subject to a goal condition ($j \in N'$), provided it "over-satisfies" its goal condition. An exception occurs in the case of a 0-1 variable x_j, which must be goal infeasible if it is fractional and $j \in N'$. For general MIP problems the option exists, as in parametric branch and bound, to free those variables that are not currently restrained by their goal conditions, thereby releasing them from their goal conditions and removing them from N'.

In contrast to the case for a goal infeasible variable, overt or potential, there is no specific primary response (R-DN) or (R-UP) for the case of integer infeasibility that pre-selects (T-DN) or (T-UP) as the transition to perform. Instead we may freely choose between these two transitions for variables in *D*. We now consider a basis for making such a choice.

2.3.1. Integer penalty and choice-preference measures

In order to select a preferred transition (T-UP) or (T-DN) for a unrestricted fractional variable, we create *integer penalty* measures $\mathrm{IP}_j(\mathrm{UP})$ and $\mathrm{IP}_j(\mathrm{DN})$ corresponding to each of these possible transitions, which (approximately) reflect the amount of deterioration in the objective of (LP′) that is induced by imposing the corresponding constraint $x_j \geqslant x'_j$ or $x_j \leqslant x'_j$, where x'_j is specified to be the integer value obtained by rounding x''_j up to $\lceil x''_j \rceil$ in the first case and rounding x''_j down to $\lfloor x''_j \rfloor$ in the second.

[7] The definition of *D* does not change if the set G^{o} of potentially goal infeasible variables is included in *G*, since G^{o} contains no fractional variables.

Let $f_j^+ = \lceil x_j'' \rceil - x_j''$ and $f_j^- = x_j'' - \lfloor x_j'' \rfloor$, identifying the positive fractions that represent the distances of x_j'' from its two integer neighbors (measuring distance in an L1 sense). Then at the simplest level, we can set $\mathrm{IP}_j(\mathrm{UP}) = f_j^+$ and $\mathrm{IP}_j(\mathrm{DN}) = f_j^-$, conceiving the distance values to be proxies for the difficulty of enforcing the associated goal conditions. At higher levels the values $\mathrm{IP}_j(\mathrm{UP})$ and $\mathrm{IP}_j(\mathrm{DN})$ can be based on traditional MIP penalty calculations that perform one or more dual simplex method post-optimizing pivots to impose $x_j \geqslant \lceil x_j'' \rceil$ or $x_j \leqslant \lfloor x_j'' \rfloor$. (Integer programming pseudo-penalties can likewise be used.) Then we choose the response that yields the smaller of these two penalties, thereby yielding $x_j' = \lfloor x_j'' \rfloor$ if $\mathrm{IP}_j(\mathrm{DN})$ is smaller and yielding $x_j' = \lceil x_j'' \rceil$ if $\mathrm{IP}_j(\mathrm{UP})$ is smaller. These two outcomes are exactly those given by the transitions (T-DN) and (T-UP) (since $\lfloor x_j'' \rfloor = \lceil x_j'' \rceil - 1$ and $\lceil x_j'' \rceil = \lfloor x_j'' \rfloor + 1$ when x_j'' is fractional).

Consequently we may write the associated responses in the form

(R-DN^D) If $\mathrm{IP}_j(\mathrm{DN}) \leqslant \mathrm{IP}_j(\mathrm{UP})$, $j \in D$, execute (T-DN).
(R-UP^D) If $\mathrm{IP}_j(\mathrm{UP}) < \mathrm{IP}_j(\mathrm{DN})$, $j \in D$, execute (T-UP).

To reflect the relative desirability of selecting the variable x_j and making the response (R-DN^D) or (R-UP^D) we create a *choice preference* measure CP_j. The preferred member of the two preceding responses, whose penalty value is $\mathrm{Min}(\mathrm{IP}_j(\mathrm{UP}), \mathrm{IP}_j(\mathrm{DN}))$, can be determined by a measure that is a monotonically increasing function of the sum $\mathrm{IP}_j(\mathrm{UP}) + \mathrm{IP}_j(\mathrm{DN})$ and the absolute difference $|\mathrm{IP}_j(\mathrm{UP}) - \mathrm{IP}_j(\mathrm{DN})|$. Thus, for example, we may choose this function as the product

$$\mathrm{CP}_j = (\mathrm{IP}_j(\mathrm{UP}) + \mathrm{IP}_j(\mathrm{DN}))(|\mathrm{IP}_j(\mathrm{UP}) - \mathrm{IP}_j(\mathrm{DN})| + w),$$

where w is a positive weight to give additional influence to the sum of the penalties (as is appropriate when their absolute difference is 0).

For the simple case where the penalties are given by $\mathrm{IP}_j(\mathrm{UP}) = f_j^+$ and $\mathrm{IP}_j(\mathrm{DN}) = f_j^-$, the sum of the penalties is always 1, and hence the choice preference value reduces to just

$$\mathrm{CP}_j = |f_j^+ - f_j^-|.$$

The responses (R-UP^D) and (R-DN^D) in this situation result in assigning $x_j' = [x_j'']$, where $[x_j'']$ denotes the nearest integer neighbor of x_j''.

In general, we stipulate that the variables with the largest CP_j values are preferred for making the response (R-UP^D) or (R-DN^D). Hence, just as we arrange the elements of the set G in descending order of the GR_j values, we arrange the elements of the set D in descending order of the CP_j values. By means of this ordering, the rule for choosing elements of D to handle by the responses (R-UP^D) and (R-DN^D) takes the same form as the rule for choosing elements of G to handle by the analogous responses (R-UP) and (R-DN). In contrast to the handling of G, however, we are not concerned with an option of freeing some of the variables, and hence instead of potentially extracting subsets corresponding to G_P and G_S of G, we only extract a single subset of D which we denote by D_o. By the notational conventions $d = |D|$ and $d_\mathrm{o} = |D_\mathrm{o}|$, the construction of D_o then results by choosing a number d_o satisfying $1 \leqslant d_\mathrm{o} \leqslant d$, and

Table 3

1	$j=$	1	2	3	4
2	$x''_j=$	0.2	0.7	0.4	0.65
3	$\text{IP}_j(\text{UP})=$	0.8	0.3	0.6	0.55
4	$\text{IP}_j(\text{DN})=$	0.2	0.7	0.4	0.65
5	$\text{CP}_j=$	0.6	0.4	0.2	0.1
	New $x'_j=$	0^*	1^*		

$^*x_j$ is made subject to a new goal condition ($j \in D_\text{o} = \{1, 2\}$).

specifying that D_o consists of the first d_o elements of D. The same process for determining g_P described in Appendix A can be used to select d_o.[8]

Under normal conditions where integer infeasibility is subordinate to goal infeasibility, we do not bother to identify D and generate D_o unless there are no goal violations, i.e., unless G is empty. Again, tabu criteria subsequently described may alter such conditions.

Example C. Assume the solution to (LP′) is integer infeasible, but not goal infeasible and the solution values $x_j = x''_j$ for $j \in D = \{1, 2, 3, 4\}$ are as shown on line 2 of Table 3, below. (Other variables may exist, which either do not violate their goal conditions or else do not belong to N'. For example, Table 3 could depict the situation that arises upon solving the original problem (LP), when N' is empty.)

As before, our illustration involves 0-1 variables, and hence x_1 to x_4 are not subject to goal conditions (or else, since they receive fractional values, they would violate these conditions). Consequently, in contrast to Examples A and B, there is no line in the table to give current goal values x'_j.

The integer penalty values $\text{IP}_j(\text{UP})$ and $\text{IP}_j(\text{DN})$ shown on lines 3 and 4 are given by $\text{IP}_j(\text{UP}) = f_j^+$ and $\text{IP}_j(\text{DN}) = f_j^-$. For 0-1 variables it may be noted that $f_j^- = x''_j$, and hence the $\text{IP}_j(\text{DN})$ values on line 4 duplicate the x''_j values on line 2. The choice preference values CP_j shown on line 5 are given by $\text{CP}_j = |f_j^+ - f_j^-|$. These values confirm that the elements of D are indexed so that larger CP_j values appear first.

For this example, we assume d_o is chosen to be 2, and consequently x_1 and x_2 are selected to be subject to new goal conditions (which may be the first goal conditions for these variables, if they have not been assigned such conditions previously and then freed by becoming members of G_S). The appropriate responses in these two cases are, respectively, (R-DN^D) and (R-UP^D) (which by the measures $\text{IP}_j(\text{UP}) = f_j^+$ and $\text{IP}_j(\text{DN}) = f_j^-$ yield the new goal value x'_j to be the integer closest to x''_j), and thus provide the entries shown on the bottom line of the table, corresponding to the goal conditions $x_1 \leqslant 0$ and $x_2 \geqslant 1$.

We point out that the CP_j values also give an alternative basis for generating the $\text{GR}_j(\text{UP})$ and $\text{GR}_j(\text{DN})$ goal resistance values. Since, only one of the two responses (R-UP) or (R-DN) is applicable to a goal

[8] The illustrated definition of CP_j should be amended for problems with special structure such as 0-1 MIP problems containing generalized upper bound (GUB) constraints, where disjoint sets of integer variables must sum to 1. In this case, some fractional-valued variable from a given GUB set must be chosen to receive a goal value of 1 and all others then automatically receive goal values of 0. Thus, the CP_j evaluation should be modified to replace the quantity $\text{IP}_j(\text{DN})$ by $\text{Max}(\text{IP}_h(\text{UP}) : h \in J - \{j\})$, where J is the GUB set that contains j. Rules based on modeling GUB sets as *special ordered sets* are also possible.

infeasible variable x_j, only one of the two $\mathrm{GR}_j(\mathrm{UP})$ and $\mathrm{GR}_j(\mathrm{DN})$ values is relevant to consider. The CP_j values identify the relative attractiveness of the preferred member of the (UP) and (DN) conditions, and hence a meaningful measure for the goal resistance value $\mathrm{GR}_j(\mathrm{UP})$ is to set $\mathrm{GR}_j(\mathrm{UP}) = \mathrm{CP}_j$ if the (UP) condition is preferred ($\mathrm{IP}_j(\mathrm{UP}) < \mathrm{IP}_j(\mathrm{DN})$) and to set $\mathrm{GR}_j(\mathrm{UP}) = -\mathrm{CP}_j$ if the (DN) condition is preferred ($\mathrm{IP}_j(\mathrm{DN}) \leqslant \mathrm{IP}_j(\mathrm{UP})$). Likewise, a meaningful measure for the goal resistance value $\mathrm{GR}_j(\mathrm{DN})$ is to set $\mathrm{GR}_j(\mathrm{DN}) = \mathrm{CP}_j$ if the (DN) condition is preferred and to set $\mathrm{GR}_j(\mathrm{DN}) = -\mathrm{CP}_j$ if the (UP) condition is preferred.

3. Tabu-search guidance

Various levels of tabu-search can be used to guide the foregoing processes. We begin by sketching the elements of a basic approach and illustrate its application.

3.1. Tabu conditions

At an initial rudimentary level, we attach a tabu restriction to an (R-DN) or (R-UP) response for a particular variable x_j, thereby forbidding the response from being executed, if the opposing response ((R-UP) or (R-DN), respectively) was executed for x_j within the most recent TabuTenure iterations. (That is, we forbid a move in a direction that is contrary to the direction of a move made within the selected span of TabuTenure iterations.) To simplify the discussion, we allow (R-DN) and (R-UP) to refer also to the responses (R-DN$^{\mathrm{o}}$) and (R-UP$^{\mathrm{o}}$). The value of TabuTenure varies according to the variable x_j concerned and the history of the search. We represent this value as $\mathrm{TabuTenure}_j(\mathrm{UP})$ and $\mathrm{TabuTenure}_j(\mathrm{DN})$ according to whether the tabu condition was launched by an (R-UP) or an (R-DN) response. When such a response is made we use $\mathrm{TabuTenure}_j(\mathrm{UP})$ or $\mathrm{TabuTenure}_j(\mathrm{DN})$ and knowledge of the current iteration, which we denote by Iter, to identify the iteration $\mathrm{TabuEnd}_j(\mathrm{UP})$ or $\mathrm{TabuEnd}_j(\mathrm{DN})$ that marks the end of x_j's tabu tenure. Specifically, when an (R-UP) response occurs, we set

$$\mathrm{TabuEnd}_j(\mathrm{DN}) = \mathrm{Iter} + \mathrm{TabuTenure}_j(\mathrm{DN})$$

to forbid the opposing (R-DN) response from being made for the period of $\mathrm{TabuTenure}_j(\mathrm{DN})$ iterations in the future. Similarly, when an (R-DN) response occurs, we set

$$\mathrm{TabuEnd}_j(\mathrm{UP}) = \mathrm{Iter} + \mathrm{TabuTenure}_j(\mathrm{UP})$$

to forbid the opposing (R-UP) response from being made for the period of $\mathrm{TabuTenure}_j(\mathrm{UP})$ iterations in the future.

By this means, an (R-DN) response is tabu for x_j as long as the (updated) current iteration satisfies

$$\mathrm{Iter} \leqslant \mathrm{TabuEnd}_j(\mathrm{DN})$$

and an (R-UP) response is tabu for x_j as long as the current iteration satisfies

$$\mathrm{Iter} \leqslant \mathrm{TabuEnd}_j(\mathrm{UP}).$$

Initially, before any responses have been made and before associated tabu conditions have been created, $\mathrm{TabuEnd}_j(\mathrm{UP})$ and $\mathrm{TabuEnd}_j(\mathrm{DN})$ are set equal to -1, causing this value to be smaller than every value of Iter and hence assuring that no tabu restrictions will be in effect.

We refer to the values $\text{TabuEnd}_j(\text{UP})$-Iter and $\text{TabuEnd}_j(\text{DN})$-Iter as *residual tabu tenures*. Hence, a response will be tabu as long as its residual tabu tenure is non-negative. (A negative residual tabu tenure accordingly indicates the response is free from a tabu restriction.) By convention, we refer to the residual tabu tenure of a variable x_j by taking it to be the residual tabu tenure of the response that is selected for this variable. We refer to the variable itself as being tabu when its associated response is tabu. (This reference is unambiguous since each goal infeasible and potentially goal infeasible variable has a single associated response.) Rules for generating the $\text{TabuTenure}_j(\text{DN})$ and $\text{TabuTenure}_j(\text{UP})$ values used to determine $\text{TabuEnd}_j(\text{UP})$ and $\text{TabuEnd}_j(\text{DN})$ are given in the next section.

In the application of the tabu tenures, a simple form of probabilistic tabu search can be used that replaces $\text{TabuTenure}_j(\text{DN})$ and $\text{TabuTenure}_j(\text{UP})$ in the formulas $\text{TabuEnd}_j(\text{DN}) = \text{Iter} + \text{TabuTenure}_j(\text{DN})$ and $\text{TabuEnd}_j(\text{DN}) = \text{Iter} + \text{TabuTenure}_j(\text{UP})$ by values that are randomly selected from an interval around the respective tabu tenure values. A fuller use of this type of randomizing effect occurs by making such a replacement each time the inequalities $\text{Iter} \leqslant \text{TabuEnd}_j(\text{DN})$ and $\text{Iter} \leqslant \text{TabuEnd}_j(\text{UP})$ are checked.

By design, tabu restrictions are prohibitions against returning to a state previously occupied. We only create these restrictions for states that seek to enforce a goal condition, hence that involve the responses (R-UP) and (R-DN) (understanding these to include reference to the responses (R-UP$^{\text{o}}$) and (R-DN$^{\text{o}}$)). Moreover, we only check tabu conditions when at least one variable is goal infeasible. In the case where no explicit goal infeasibility exists, and hence the only responses to consider are those applicable to unrestricted free variables, then no attention is paid to tabu restrictions. The situation where all goal conditions are satisfied (no goal infeasibility exists) may be viewed as meeting the requirements of a special type of aspiration criterion, which overrules all tabu conditions. We now examine the use of criteria that operate when goal infeasibility is present.

3.2. Aspiration criteria

As is customary in tabu-search, we allow a tabu response to be released from a tabu restriction if the response satisfies an auxiliary *aspiration criterion* that indicates the response has special merit or novelty (i.e., exhibits a feature not often encountered). A common instance of such a criterion, called *aspiration by objective*, permits the response to be made if it yields a better objective function evaluation than any response previously executed. In the present setting, we find it convenient to additionally consider an *aspiration by resistance*, based on the greatest resistance a particular response has generated in the past.

Specifically, let $\text{Aspire}_j(\text{DN})$ and $\text{Aspire}_j(\text{UP})$ denote the largest goal resistance values $\text{GR}_j(\text{DN})$ and $\text{GR}_j(\text{UP})$ that have occurred for x_j on any iteration, where x_j was selected to execute an (R-DN) or (R-UP) response, respectively. Then we disregard the tabu restriction for an (R-DN) response (identified by $\text{Iter} \leqslant \text{TabuEnd}_j(\text{DN})$) if

$$\text{GR}_j(\text{DN}) > \text{Aspire}_j(\text{UP})$$

and disregard tabu restriction for an (R-UP) response (identified by $\text{Iter} \leqslant \text{TabuEnd}_j(\text{UP})$) if

$$\text{GR}_j(\text{UP}) > \text{Aspire}_j(\text{DN}).$$

The rationale for these aspiration criteria is that a move can be allowed if its current resistance value, measured by $\text{GR}_j(\text{DN})$ or $\text{GR}_j(\text{UP})$, exceeds the greatest resistance value previously identified for moving in the opposite direction ($\text{Aspire}_j(\text{UP})$ or $\text{Aspire}_j(\text{DN})$, respectively). We initially set $\text{Aspire}_j(\text{UP})$ and $\text{Aspire}_j(\text{DN})$ to a large negative number, so that the first time a variable x_j is evaluated for a potential

Table 4

1	$J=$	1	2	3	4	5	6
2	$x''_j=$	1	0	0	1	0	1
3	$x'_j=$	0.6	0.3	0.2	0.9	0	1
4	$\mathrm{GR}_j=$	0.4	0.3	0.2	0.1		
5	$\mathrm{GR}^{\mathrm{o}}_j=$					−4	−25
6	Tabu status	T	T		T	T	
7	Aspire_j	0.5			0.3	0.1	0.4
8	$\mathrm{Aspire}^{\mathrm{o}}_j$		−44	−69			
	New $x'_j=$	1	1^*	1^*	1	0	#

$T=$ Tabu.
$^*x_j$ changes its goal condition ($j \in G_{\mathrm{P}} = \{2, 3\}$).
$^{\#}x_j$ is freed from its goal condition ($j \in G_{\mathrm{S}} = \{6\}$).

response (R-UP) or (R-DN), the response will automatically be allowed, and it will continue to be allowed until the opposing response is made, which establishes a resistance to be exceeded.

We call a response *admissible* if it is either not tabu or else satisfies the aspiration criterion, and call it *inadmissible* otherwise. If the unique available response for a goal infeasible variable is inadmissible, then the variable is not permitted to enter the sets G_{P} and G_{S}, even if this makes it impossible for one or both of these sets to attain its targeted size g_{P} or g_{S}. The only exception to this rule is that G_{P} is not permitted to be empty in the case of goal infeasibility. Hence in the extreme case where no variables would enter G_{P} the typical *aspiration by default* rule is invoked that allows G_{P} to contain a variable with a smallest residual tabu tenure. (Probabilistic variations of the aspiration by default rule can also be applied, by assigning larger probabilities to selecting variables with smaller residual tabu tenures.)

As observed earlier, $\mathrm{GR}_j = \mathrm{GR}_j(\mathrm{DN})$ or $\mathrm{GR}_j(\mathrm{UP})$ may be treated as a 2-element vector, with a dominant component for an overt goal infeasibility and a secondary $\mathrm{GR}^{\mathrm{o}}_j$ component for potential goal infeasibility. The Aspire_j values are treated in the same way, as 2-element vectors that include a secondary component $\mathrm{Aspire}^{\mathrm{o}}_j$ for potential goal infeasibility. Since overt and potential goal infeasibility for a given variable x_j never occur simultaneously, and since overt goal infeasibility is the dominant component, only a single component of the vector is relevant to consider—the overt component if it exists, and the potential component otherwise.

It is to be emphasized that $\mathrm{Aspire}_j(\mathrm{UP})$ and $\mathrm{Aspire}_j(\mathrm{DN})$ do not record the greatest values of $\mathrm{GR}_j(\mathrm{UP})$ and $\mathrm{GR}_j(\mathrm{DN})$ encountered over the history of the search, but only the greatest values that occurred in the instances where x_j was selected as a variable to be assigned a goal condition, and only in response to overt or potential goal infeasibility (i.e., not in response to integer infeasibility, which occurs only when x_j is an unrestricted fractional variable).

Example D. Consider once again the situation of responding to goal infeasibility depicted in Example B, which we expand to include reference to tabu restrictions and aspiration criteria. Responses that are tabu are indicated by the T symbol in line 6 of Table 4, below, which is an extended form of the previous Table 2. As shown, the goal responses for the variables x_1, x_2, x_4 and x_5 are currently tabu. The

Aspire_j (=Aspire_j(UP) and Aspire_j(DN)) values underlying the aspiration criteria are shown on lines 7 and 8, where line 7 identifies the component for overt goal infeasibility (simply labeled as Aspire_j) and line 8 identifies the component associated with potential goal infeasibility (labeled as $\text{Aspire}^{\text{o}}_j$). Since the dominant Aspire_j component is the only one that is relevant if it exists (and thereby "erases" any previous $\text{Aspire}^{\text{o}}_j$ component), only one entry, Aspire_j on line 7 or $\text{Aspire}^{\text{o}}_j$ on line 8, appears for each x_j.

Recall that for 0-1 variables, as considered here, a goal value of 1 ($x'_j = 1$ on line 2) corresponds to an (UP) condition and a goal value of 0 ($x'_j = 0$ on line 2) corresponds to a (DN) condition, respectively, giving $x'_j \geqslant 1$ and $x'_j \leqslant 0$. Thus, we know that $\text{GR}_j = \text{GR}_j$(UP) for $j = 1, 4$ and 6 and $\text{GR}_j = \text{GR}_j$(DN) for $j = 2$, 3 and 5, and by the reverse (UP)/(DN) association between GR_j and Aspire_j in the aspiration criteria, we also know that the applicable Aspire_j values are given by $\text{Aspire}_j = \text{Aspire}_j$(DN) for $j = 1$, 4 and 6 and $\text{Aspire}_j = \text{Aspire}_j$(UP) for $j = 2, 3$ and 5. (The same identities also hold for GR^{o}_j and $\text{Aspire}^{\text{o}}_j$.)

In the present instance we seek to respond to goal infeasible and potentially goal infeasible variables by constructing a primary goal set G_{P} of size $g_{\text{P}} = 2$ and a secondary goal set G_{S} of size $g_{\text{S}} = 1$. Consequently, we take G_{P} to consist of the first two (highest ranking) admissible variables-i.e., those whose responses are admissible. In particular, we look for the first two variables whose responses are not tabu or else that satisfy the applicable aspiration criterion $\text{GR}_j(\text{DN}) > \text{Aspire}_j(\text{UP})$ or $\text{GR}_j(\text{UP}) > \text{Aspire}_j(\text{DN})$. Similarly, we take G_{S} to consist of the next admissible variable after those assigned to G_{P}.

Proceeding through the variables in the order of their ranking, we see that x_1 is tabu. More precisely, x_1 currently violates its associated (UP) condition, and its associated counter response (R-DN) is tabu. Moreover, the GR_j value of .4 for x_1 does not exceed its Aspire_j value of .5. (The GR_1 entry represents GR_1(UP), the amount that x_1 resists the (UP) condition it currently is subject to, and the Aspire_1 entry represents Aspire_1(DN), the largest amount that x_1 previously resisted the (DN) condition that it now would undertake to enforce.) Consequently, the aspiration criterion is not satisfied for x_1 and the variable is not admissible to enter G_{P}.

Next we examine x_2, which is likewise tabu, and we see in this case that its aspiration criterion is satisfied, since the value GR_2=.3 dominates the associated value $\text{Aspire}^{\text{o}}_j = -44$. (There is no Aspire_2 value that offers a comparison on the same level as GR_2.) The tabu status for x_2 comes from a tabu restriction placed upon an (R-UP) response, since x_2 is currently subject to a (DN) condition. Consequently, GR_2 represents GR_2(DN) and $\text{Aspire}^{\text{o}}_2$ represents $\text{Aspire}^{\text{o}}_2$(UP). The latter condition indicates that the largest GR^{o}_2(UP) value previously applicable to x_2, for resisting an (UP) condition, was -44. That is, on an occasion when x_2 was potentially goal infeasible, satisfying the (UP) condition $x_2 = 1$, its resistance measure was derived from a reduced cost of 44. Moreover, x_2 was never overtly infeasible when it was subject to an (UP) condition, and then selected to receive an (R-DN) response, or else it would have an entry for GR_2. (Since x_2 currently is subject to a (DN) condition, it must have been selected for an (R-DN) response when it was potentially goal infeasible.) The fact that x_2 satisfies its aspiration criterion shows that it is admissible to be selected, and hence it is placed in G_{P}. If at this point no more variables were admissible to be selected, we would terminate with $G_{\text{P}} = \{2\}$ and G_{S} empty. (The fact that G_{P} is non-empty means that the construction is acceptable.)

The next variable x_3 is not tabu and hence also is admissible to enter G_{P}. The associated value $\text{Aspire}^{\text{o}}_3 = -69$ for x_3 indicates that the variable had previously satisfied its (UP) condition as a potentially goal infeasible variable, and was selected for an (R-DN) response (hence assigning x_3 the (DN) condition that currently applies to this variable). At this point, x_3 would have acquired a tabu restriction preventing an (R-UP) response. Since the variable is not currently tabu (i.e., its (R-UP) response is not tabu), we know that the previous tabu restriction for (R-UP) has expired.

With the assignment of both x_2 and x_3 to G_{P}, the primary goal set now receives its quota of $g_{\mathrm{P}} = 2$ variables, and now we look for a variable to be assigned to the secondary goal set G_{S}. The variable x_4 is next in line to examine, and is tabu. The value $\mathrm{GR}_4 = .1$ does not exceed the value $\mathrm{Aspire}_4 = .3$, and so the aspiration criterion is not satisfied. Thus, x_4 does not qualify to enter G_{S}.

Proceeding to x_5, we encounter a tabu variable whose value $\mathrm{GR}_5^{\mathrm{o}} = -4$ is dominated by the associated value $\mathrm{Aspire}_5 = .1$. Hence the aspiration criterion is not satisfied, and the variable is inadmissible to be assigned to G_{S}.

Finally, the variable x_6 is not currently tabu, and so it is added to G_{S} to complete the quota of $g_{\mathrm{S}} = 1$ variables for this set. The fact that x_6 has an associated value $\mathrm{Aspire}_6 = 0.4$ indicates that x_6 had once overtly violated a (DN) condition with a resistance measure of $\mathrm{GR}_6(\mathrm{DN}) = 0.4$, and had at that point been selected for an (R-UP) response (while at the same time placing a tabu restriction to forbid a subsequent (R-DN) response). The fact that x_6 is not currently subject to such a restriction discloses that its tabu tenure has expired.

3.3. Determining the values of TabuTenure$_j(UP)$ and TabuTenure$_j(DN)$

A variety of ways exist for creating tabu tenures, ranging from simple static tenures to more advanced dynamic and adaptive tenures, and can be used in the present setting. (For a discussion of alternatives, see for example, [2,3].) We suggest an approach for creating the values $\mathrm{TabuTenure}_j(\mathrm{UP})$ and $\mathrm{TabuTenure}_j(\mathrm{DN})$ that offers a further option.

For convenience we use a shorthand notation, where the symbol α refers to either the condition UP or DN and β refers to the opposite condition. Thus, for example, we note that an $(R\text{-}\alpha)$ response is made to a $(V\text{-}\beta)$ violation, for $\alpha = \mathrm{UP}$ and DN and $\beta = \mathrm{DN}$ and UP, respectively. Then our suggested procedure for creating the tabu tenure values can be expressed as follows.

Creating TabuTenure$_j(\alpha)$ for $\alpha = \mathrm{UP}$ and DN

(a) All $\mathrm{TabuTenure}_j(\alpha)$ values start at a selected minimum value MinTenure.
(b) Whenever x_j is subjected to an $(R\text{-}\alpha)$ response (because of a $(V\text{-}\beta)$ violation), increase $\mathrm{TabuTenure}_j(\beta)$ by a specified increment ΔTenure (setting $\mathrm{TabuTenure}_j(\beta) := \mathrm{TabuTenure}_j(\beta) + \Delta\mathrm{Tenure}$). An exception occurs if the response is tabu and executed because of satisfying the criterion of aspiration by resistance, in which case $\mathrm{TabuTenure}_j(\beta)$ is unchanged.
(c) After a selected number of iterations, and each time a new best solution is obtained, *clean the slate* by re-setting all tabu tenures again to MinTenure, and also re-setting all $\mathrm{Aspire}_j(\alpha)$ values (for $\alpha = \mathrm{UP}$ and DN) to a large negative value.

MinTenure and ΔTenure are parameters whose preferred values are to be determined by experimentation. (A reasonable starting point is to begin by exploring small values, as by selecting MinTenure in the range from 3 to 5 and ΔTenure in the range from 1 to 3. The maximum value attained by any tabu tenure should be limited by a pre-set upper bound.) We also note step (c) can optionally be executed by performing a *partial slate cleaning*, where the tabu tenures and the $\text{Aspire}_j(\alpha)$ values are simply reduced without re-setting them all the way back to their initial values. The degree of this reduction can be controlled by strategic oscillation (e.g., as a function of the smallest and average values previously assigned to these parameters following their initialization).[9]

3.4. A core version of parametric tabu-search

We summarize a core version of parametric TS that results from applying the basic ideas sketched in the preceding sections. The Core Method lies at the heart of the complete approach that incorporates associated intensification and diversification procedures.

In its initial execution, the Core Method begins by specifying (LP$'$) to be the original relaxation (LP) of (MIP), where no goal conditions exist and N' is empty. We assume the solution to (LP) is not (MIP) feasible, or else the problem (MIP) is already solved. The method then proceeds as follows.

Core Method

1. Solve (LP$'$) to obtain an optimal solution (x'', y''). If this solution is (MIP) feasible, update x_0^* and repeat this step to re-optimize (LP$'$). (If (LP$'$) has no feasible solution, the method stops and the best solution is optimal.) Terminate the solution process if this step has been executed a chosen number of times. Otherwise, continue to Step 2.
2. (a) If the solution to (LP$'$) is goal infeasible, create the sets G_P and G_S to consist of the g_P and g_S highest ranking goal infeasible (and potentially goal infeasible) admissible variables from G, defining admissibility in relation to the current tabu restrictions and aspiration criteria.
   (b) If the solution to (LP$'$) is not goal infeasible, but is integer infeasible, Create the set D_o to consist of the d_o highest ranking unrestricted fractional variables from D.[10]
3. According to the outcome of Step 2, generate the new goal conditions and identify the new problem (LP$'$). If the stipulations of Step 2(a) apply, update the associated tabu tenures and aspiration values. Then return to Step 1.

The repetition of Step 1 in the case where the current solution to (LP$'$) is (MIP) feasible invites the use of dual post-optimization to handle the updated objective function constraint $cx + dy \leqslant x_0^* - \varepsilon$. Alternatively, the objective function constraint may be temporarily maintained in the form $cx + dy \leqslant x_0^*$, which is satisfied by the current solution to (LP$'$). Although no variables are goal infeasible, the method then proceeds to Step 2(a) to create the sets G_P and G_S by reference to the potentially goal infeasible variables. Subsequently, the objective function constraint can be restored to the more restrictive form $cx + dy \leqslant x_0^* - \varepsilon$ when the solution to (LP$'$) in Step 1 is feasible for this constraint. The restoration will be possible immediately after the first re-execution of Step 1 if G_P is made empty and if a sufficient number of variables in G are is selected for inclusion G_S. This approach could naturally be executed under the policy where the slate-cleaning step of the preceding section is performed each time a new (MIP) feasible

[9] A referee has also suggested the interesting possibility of partially cleaning the slate of each individual variable x_j as soon as its TabuTenure reaches MaxTenure.

[10] Probabilistic TS rules can be employed to generate the composition of G_P, G_S and D_o as a function of the rankings or the underlying penalty measures.

solution is found. This option re-initializes the tabu tenures and aspiration values, essentially starting over from the solution to the original (LP) relaxation, whose basis can be saved for this purpose. The new determination of G_P and G_S can then be superimposed on this LP basis, by reference to the variables that were potentially goal infeasible upon finding the new best solution and postponing the introduction of the new goal conditions until recovering the solution to (LP). Another option is simply to post-optimize from the most recent solution to (LP′), using the objective function of minimizing $cx + dy$ (temporarily setting all penalties to 0), and then interrupting the execution as soon as $cx + dy$ becomes small enough to allow the constraint $cx + dy \leqslant x_o^* - \varepsilon$ to be instated. Then the new (LP′) objective can be introduced, allowing primal post-optimization to continue without having to revert all the way to the initial (LP) solution.

In the case of a pure IP problem where all variables are integer variables, the discovery of a new (MIP) feasible solution can also launch a supplementary improvement method, using a classical neighborhood that changes the values of selected variables by 1 at each iteration. If the supplementary method succeeds in finding an improved solution, yielding a further improved value for x_o^*, dual post-optimization can be avoided by same approaches previously noted. (Under the simple option of temporarily relaxing the objective function constraint, the form of the constraint can be expressed as $cx + dy \leqslant x_o''$, where x_o'' denotes the x_o value for the solution to (LP′); i.e., x_o'' gave the previous value of x_o^* before x_o^* was improved by the supplementary procedure.)

3.5. Summary of the complete parametric TS procedure

The complete parametric TS method contains the Core Method as a subroutine, and extends it by the inclusion of intensification and diversification strategies. We briefly sketch the form of the complete method, and then describe the additional supporting strategies in Section 4.

Parametric TS Method

0. Initialize the TabuTenure and Aspire arrays. Identify (LP′) as the original relaxation (LP) of (MIP), where no goal conditions exist and N' is empty.
1. Execute the Core Method.
2. (a) Terminate if the accumulated number of iterations from all executions of the Core Method exceeds a chosen limit, or if a criterion of progressive improvement is not met. Otherwise:
   (b) Apply an Intensification or Diversification Step to generate a new goal vector x' and associated sets N^+ and N^- to define a current problem (LP′). Then return to Step 1.

Step 2(b) is carried out by identifying a specific earlier instance of the problem (LP′) so that the current one can be solved by primal post-optimization, starting from the optimal LP basis to the earlier problem. The earlier (LP′) problem can be relaxed to be the original LP relaxation (LP), but other (LP′) problems can be used instead.

We subsequently identify a variation of Step 2(b) utilizing model embedded memory that first generates a problem (LP^o) whose form differs from (LP′), but whose solution can be treated exactly as if it were a solution to (LP′). Upon solving (LP^o) the method returns to Step 1 to execute the Core Method in the usual fashion.

We now examine the procedures that give substance to Step 2(b).

4. Intensification and diversification

The intensification and diversification methods treated in this section are divided into four main types: (1) basic approaches founded on exploiting strongly determined and consistent variables; (2) approaches derived from scatter search; (3) methods based on frequency analysis; and (4) procedures employing ideas of model embedded memory. These approaches cover a broad spectrum of strategic considerations, and invite empirical research to identify the ways they may best fit together in various settings.

4.1. Basic approaches

The simplest forms of intensification and diversification methods arise by manipulating the values g_P, g_S and g_D, choosing them to be small for intensification and large for diversification. Probabilistic choice can be incorporated as an added factor to increase diversification. Similarly, the tabu tenure values can be manipulated to favor intensification or diversification as in customary variants of tabu-search.

Such rudimentary considerations acquire greater scope by making use of basic types of frequency memory, particularly as a foundation for identifying and exploiting *strongly determined* and *consistent* variables. This calls for the creation of measures that differ from the measures used to identify resistance and integer infeasibility, and hence departs from evaluations that have familiar counterparts in the context of branch and bound.

Strongly determined and consistent variables are identified by reference to the phenomenon where some variables receive particular values with greater tenacity or frequency than other variables, over subsets of high quality solutions. The value a given variable is thus disposed to receive is called its *preferred* value. Saying that a variable receives its preferred value with greater tenacity means that an attempt to change the value of the variable causes a greater disruption in the objective function or in feasibility. (In this respect, classical branch and bound penalty measures, that identify the deterioration in x_o by performing one or more dual pivots upon imposing a particular branch, can be one means of pinpointing strongly determined variables, provided these measures are generated with respect to high-quality solutions.)

A key issue is to choose appropriate subsets of solutions over which tenacity and frequency are measured, by selecting subsets whose elements are related by exhibiting similar features in the case of intensification and dissimilar features in the case of diversification. This issue surfaces again in the context of strategies derived from scatter search, as discussed in the next sub-section. The link between these sub-sections results from the fact that the notions underlying scatter search and those underlying strongly determined and consistent variables come from the same source. Hence, the rules for isolating useful subsets of high quality solutions subsequently described in connection with scatter search can also be used within the present setting as a basis for identifying strongly determined and consistent variables.

By means of such rules, these variables can be exploited in the following manner drawing on the proposal of Glover [4].

Exploitation of strongly determined and consistent variables

0. During an initial set of solution passes, save a collection X^* of the highest quality solutions generated. Let NC denote a subset of N identifying variables that are constrained to receive specific values, and begin with NC empty.
1. Extract a subset S of high-quality solutions from X^*.

2. Identify a small number of variables x_j, $j \in N - \mathrm{NC}$, that have the highest evaluations for being strongly determined and consistent over S. Constrain these variables to their preferred values and add them (i.e., their indices) to NC.[11]
3. Perform an additional series of solution passes, allowing only the variables x_j, $j \in N - \mathrm{NC}$ to vary. Again save a collection X^* of the highest-quality solutions generated, including those solutions from the previous X^* in which the constrained variables x_j, $j \in \mathrm{NC}$ receive their associated preferred values. If the best solution found has not improved over a specified number of consecutive applications of this step, then stop. Otherwise, return to Step 1.

The rationale underlying the foregoing approach is that a process of progressively constraining strongly determined and consistent variables to their preferred values will cause new variables from those remaining to receive evaluations that qualify them more decisively as strongly determined or consistent, although quite possibly with different preferred values than those that might have potentially been associated with these variables on previous passes. After the first few executions of Step 1, the set S can be chosen to be the same as X^*, so that the identification of strongly determined and consistent variables is affected thereafter solely by the growing number of constrained variables in the set NC. The stopping rule in Step 3 can also terminate the procedure when NC contains a specified portion of the total number of variables in N. Then, after terminating, the approach can be iterated by returning to an earlier stage when S was a proper subset of X^*, and selecting a different subset S as a basis for the steps that follow. Tabu restrictions can be used to assure that the additions to the set NC do not duplicate those of previous passes, except where the preferred values of some of the variables added to NC differ from the values received earlier.

We assume an iterated version of the preceding approach is controlled so that only a few different subsets S are permitted to be derived from a given set X^*, over successive iterations where X^* is recovered at Step 1. Then, if S is made to be the same as X^* after only 1 or 2 successive visits to Step 1, the procedure will avoid examining a large number of different sets of constrained assignments.

In the original proposal for exploiting this framework, the solution passes referred to in Steps 0 and 3 were conceived to be an application of any method or methods the implementer might customarily use or devise for generating solutions. In the present setting, we refer to the approaches currently under examination. In particular, the determination of X^* and S can be made by equating these sets, respectively, with the set R and its associated subsets $R(k)$ described in the next sub-section.

4.2. Approaches derived from scatter search

Methods inspired by scatter search (e.g., [4,5]) create goal conditions for defining (LP′) by reference to points dispersed throughout various sub-regions, where these sub-regions are implicitly defined by subsets of previously generated solutions. The foundation for creating these subsets consists of maintaining a *reference set* $R = \{x^1, x^2, \ldots, x^b\}$ whose component solutions x^i are composed of the b best (MIP) solutions found,[12] hence each x^i, $i = 1, \ldots, b$ corresponds to a vector x'' from a solution (x'', y'') to a past problem (LP′). The value of b will usually be relatively small, e.g., in the range from 8 to 20, though the analysis can be carried out for both larger and smaller numbers of solutions.

[11] These variables may be constrained to lie in preferred ranges rather than be compelled to receive specific values.

[12] The definition of “best” used to select solutions in R can include criteria of diversity, giving rise to a form of R that is sometimes managed as two or three component sets.

It is to be emphasized that the solutions x^i recorded in the reference set need not be (MIP) feasible, but may be evaluated for inclusion in R by adopting a measure of quality that penalizes infeasibility. This can be important in applications where the primary goal is to identify a feasible (MIP) solution, as in settings where such solutions are difficult to find. Although, we may normally consider solutions that are (LP) feasible but fractional, in the present case we suppose each recorded x^i is integer feasible. To ensure this, if the associated solution (x'', y'') to (LP$'$) does not yield an integer feasible vector x'', we apply nearest neighbor or generalized rounding to x'' to achieve integer feasibility. In contrast to the types of populations maintained in customary evolutionary approaches such as genetic algorithms, R identifies a set in the strict meaning of the term, and hence no member of R duplicates any other. Since the composition of R varies over time, we also employ a larger historical set R^* composed of the union of previously generated reference sets R. For simplicity, R^* may be restricted by disregarding component solutions that were dropped before executing the first intensification or diversification strategy in Step 2(a) of the parametric TS method. R^* may also be further restricted, is space limitations are pressing, by considering only the composition of those sets R existing at the time an intensification or diversification step is executed, noting that some solutions that enter R may be displaced by others without having an opportunity to be examined in Step 2(a).

Accompanying R^* we also keep track of a set R' that consists of each goal solution x' that was used to create an element x of R^*. Duplicate records of solutions are not needed, hence we do not bother to add x' to R' if $x = x'$. The solutions $x^o \in R^* \cup R'$ are then checked against a new candidate solution x' that may be used to define a new (LP$'$). If $x' = x^o$ then the solution of (LP$'$) will launch a phase of the parametric TS method that re-produces solutions already found. Consequently, upon discovering that a new goal vector x' matches some $x^o \in R^* \cup R'$ we disregard this current x' and generate another in its place. To facilitate the checking for matches between pairs x^o and x', we store a hash value with each element of $R^* \cup R'$. Then it is only necessary to check for $x' = x^o$ if the hash value of x' is the same as that recorded for x^o.

Based on these conventions, we turn to the issue of generating a new candidate goal vector x' and an associated problem (LP$'$).

4.2.1. Intensification and diversification by generating x' from centers

While scatter search normally generates multiple linear combinations of the points that define a particular sub-region of R, we will primarily focus on generating centers of these sub-regions, in accordance with scatter search proposals that seek to economize the number of points examined. For this purpose, we implicitly identify a convex sub-region spanned by R by selecting a subset $R(k)$ consisting of k points drawn from R, which by re-indexing we denote by $R(k) = \{x^1, x^2, \ldots, x^k\}$. The value of k will typically be selected to be relatively small, e.g., between 2 and 9,[13] and we further suggest that in many cases, particularly for 0-1 integer variables, k should preferably be odd. The elements of $R(k)$ are then chosen one at a time from R, to produce successive sets $R(1), R(2), \ldots, R(k)$, thereby culminating in the creation of $R(k)$. We first describe the approach in overview, introducing relevant notation.

The point x^1 to compose $R(1)$ is selected as the best point in the set R, using the same evaluation criterion used to generate R itself. Having generated the set $R(i)$ for $k > i \geqslant 1$, the choice of x^{i+1} to create

[13] In many cases k can be restricted to be at most 5, and in any event need not exceed $(|R| + 1)/2$. However, larger values of k can be relevant for generating subsets $S = R(k)$ to be used for identifying strongly determined and consistent variables, as discussed earlier.

$R(i+1)$ is based on selecting the element of $R - R(i)$ that is closest to the center $x(i)$ of $R(i)$, defining $x(i) = (1/i)\Sigma(x^h : h \leqslant i\}$ (hence, $x(1) = x^1$). Let $D(x, x(i))$ denote the distance of a point x from $x(i)$. (The L1 norm $D(x, x(i)) = \Sigma|x_j - x_j(i)| : j \in J)$ is convenient for defining distance in this setting, though other norms can also be used.) Then, in particular, we choose

$$x^{i+1} = \text{argmin}(D(x, x(i)) : x \in R - R(i)).$$

The generation of $R(k)$ can be the foundation for both an intensification procedure and a diversification procedure, because the center $x(k)$ of $R(k)$ embodies a greater intensification influence by choosing smaller values of k and a greater diversification influence by choosing larger values of k. (Evidently, choosing $k = 1$ would trivially produce the greatest possible form of intensification, by simply returning the solution x^1 as the point for initiating subsequent search.) A more extreme form of diversification is produced (for $k > 1$) by replacing argmin by argmax in the preceding definition of x^{i+1}.

4.2.2. *Multiple executions and tabu guidance*

The procedure of generating $R(k)$ and launching a new search from its center $x(k)$ is applied multiple times, by performing multiple executions of Step 2(b) of the parametric TS method. Consequently, we construct more than one set of the form $R(k)$, generating a new set at each visit to Step 2(b). Variation is induced in the resulting sets by excluding some elements of R from belonging to a currently generated $R(k)$, making use of two tabu sets, denoted Tabu1 and Tabu2. The manner in which this is done is as follows.

Tabu1 is the union of all $R(k)$ sets previously generated. We use this set to compel the first element x^1 of the current $R(k)$ to come from R - Tabu1, thus assuring the current $R(k)$ cannot have a composition that duplicates any previous $R(k)$. Thus, prior to constructing the first instance of $R(k)$, we begin with Tabu1 $= \Phi$, and then as each new $R(k)$ is constructed we augment Tabu1 to incorporate its elements.

Tabu2 is constructed by keeping track of a value $n(x)$ for each $x \in R$ that identifies the number of times the solution x has appeared in the sets $R(k)$ generated on all previous executions of Step 2(b). (The number $n(x)$ may be accessed by an index assigned to x as a member of R.)

When a new solution x is added to R by displacing the current lowest quality solution, the value $n(x)$ for this solution begins at 0. Tabu2 is then composed by adding solution x to the set Tabu2 when $n(x)$ reaches 2. However, this condition is waived for the current best solution x^* in R, which is not restricted in the number of times it can be incorporated into the sets $R(k)$. However, if x^* becomes superseded as the best member by a new solution that is added to R, and if $n(x^*) \geqslant 2$, then x^* enters Tabu2.[14]

The rule for creating $R(k)$ then takes the following form:

Rule to generate R(k)

0. Begin with Tabu1 and Tabu2 inherited from generating previous sets $R(k)$, where Tabu1 and Tabu2 start empty before generating the first $R(k)$.
1. Choose x^1 to be the best element of R - Tabu1.
2. For $i = 1, \ldots, k-1$, let $x^{i+1} = \text{argmin}(D(x, x(i)) : x \in R - R(i) - \text{Tabu2})$.

[14] The limit of 2 for $n(x)$ is of course not mandatory, and an alternative is to allow the limit to vary according to the rank of x in R, as where $n(x) = 5$ (instead of infinity) for the best solution, $n(x) = 4$ for the second best solution, etc. These considerations are amplified in the next sub-section.

3. Set Tabu1 := Tabu1 $\cup R(k)$. For $i = 1, \ldots, k$ set $n(x^i) := n(x^i) + 1$, and if $n(x^i) \geqslant 2$, add x^i to Tabu2.

We observe that R can change in the interval between generating one set $R(k)$ and another, since each $R(k)$ generated can be used as the source of a goal solution x' (derived from $x(k)$) that launches an additional iteration of the parametric TS method. Consequently, the composition of Tabu2 will alter by removing reference to solutions that have been dropped from R and by keeping track of the value $n(x^i)$ for new solutions x^i that are added to R.

Upon generating $R(k)$ and identifying its center $x(k)$, we create the goal vector x' to define (LP′) by rounding the fractional components of $x(k)$ to integer values, hence setting[15]

$$x' = [x(k)].$$

The last step is to identify the sets N^+ and N^- to complete the definition of the problem (LP′). For this we may start from any solution (x'', y'') to a previous (LP′), and in particular we choose (x'', y'') to be the solution to the original LP relaxation (LP). However, the goal vector x' is determined as indicated above, rather than choosing x' to be the vector associated with the earlier problem (LP′).

Then we produce (LP′) by creating N^+ and N^- according to the following rule.

Constructing (LP′) from x′ and the solution x″ to a previous (LP′)

If $x''_j > x''_j$ or $x''_j = x'_j = 0$, add j to N^-.
If $x''_j < x'_j$ or $x''_j = x'_j = U_j$, add j to N^+.
If $0 < x''_j = x'_j < U_j$, add j to both N^+ and N^-.

The new problem (LP′) then becomes the one associated with Step 2(b) of the parametric TS method, and this problem may be solved upon returning to Step 1 by post-optimizing from the solution (x'', y'') to the selected previous problem (LP′). In this regard, different choices of (x'', y'') can change the trajectory of responding to the goal vector and goal conditions, and thus lead to possibly different candidate solutions to (MIP). As previously noted, the method can be used to emphasize either intensification or diversification, by choosing smaller or larger values of k, and achieving further emphasis on diversification by replacing argmin by argmax in the definition of x^{i+1} in Step 2 of the rule for generating $R(k)$.

4.2.3. Additional uses of scatter search

To rely more heavily on mechanisms derived from scatter search, as a means of generating a greater variety of combined solutions to yield goal vectors x', we may undertake to produce a greater variety of subsets $R(k)$ of R, thus giving rise to additional centers $x(k)$ and hence additional goal vectors x'. We indicate prescriptions for doing this that can also be used in other contexts where scatter search is employed. Our approach can of course be accompanied by generating additional combinations of the component solutions within a given set $R(k)$, rather than referring solely to the point $x(k)$. Additional

[15] As earlier, we use the square brackets to denote nearest neighbor rounding (i.e., $[x] = ([x_1], [x_2], \ldots, [x_{|J|}])$). The recommendation to choose k odd gives a natural tie breaking feature for setting $x' = [x(k)]$, since in this way, no component x'_j of x' can equal .5.

variation can result by allowing the number of different solutions contained within the subsets $R(k)$ to range over values that are not restricted to the intervals previously indicated.

The key idea is as follows. By reference to a chosen indexing of the solutions in R, we create an implicit matrix $\text{Match}(p, q)$ where $\text{Match}(p, q) = 1$ if the solutions x^p and x^q in R have previously been *matched*, i.e., have appeared together as elements of a previously generated subset $R(k)$, and where $\text{Match}(p, q) = 0$ otherwise. By convention we define $\text{Match}(p, p) = 1$. (This matrix representation is for descriptive convenience only, and can be replaced by the use of linked lists, as a means to take advantage of symmetry and sparsity.)

We then use $\text{Match}(p, q)$ to identify tabu conditions governing the choice of solutions x^p admitted to the subset $R(k)$ as follows. Define

$$\text{TabuMatch} = \{x^p \in R : \text{Match}(p, q) = 1 \text{ for all } x_q \in R\},$$

i.e., TabuMatch is the set of solutions in R that have been matched with every other solution in R over the collection of subsets $R(k)$ previously generated. As we undertake to generate a new $R(k)$, we exclude consideration of any solution x that belongs to TabuMatch.

Moreover, relative to a given set $R(i)$ being constructed in the process of generating a current $R(k)$ (by the construction sequence $R(1), \ldots, R(i), \ldots, R(k)$), and relative to any solution $x^p \in R - R(i)$ (in the role of a candidate to be added to $R(i)$ to produce $R(i + 1)$), define

$$n(i : p) = \Sigma(\text{Match}(p, q) : x^q \in R(i)),$$

i.e., $n(i : p)$ is the number of solutions $x^q \in R(i)$ that have been matched with x^p as a result of the situation where both x^p and x^q belonged to some previously generated set $R(k)$. In this case, we want to prevent x^p from being added to $R(i)$ if x^p matches with more than a specified fraction $f|R(i)|$ of the elements of $R(i)$, hence if $n(i : p) > f|R(i)|$ (where, for example, $f = .5$ or $.7$).[16] Thus, we define

$$\text{Tabu}(i) = \{x^p \in R - R(i) : n(i : p) > f|R(i)|\}$$

to identify the set of solutions that are excluded from being added to $R(i)$ to generate $R(i + 1)$.

Finally, we refine the earlier criterion for defining membership in the set Tabu2 by replacing $n(x^i) \geqslant 2$ with $n(x^i) \geqslant v(i)$, where the $v(i)$ values are integers, such that $v(p) \geqslant v(q)$ if x^p has a higher evaluation that x^q (by the evaluation criterion used to determine membership in R). It is reasonable to restrict all $v(i)$ values to be relatively small, e.g., not more than 5.

By these conventions we now consider an alternative to our earlier rule for generating the set $R(k)$. In addition to giving a role to the new sets TabuMatch and $\text{Tabu}(i)$, and to the values $v(i)$, we effectively replace Tabu1 by Tabu2 $\cup$ TabuMatch.

Alternative Rule to generate R(k):

0. Begin with TabuMatch and Tabu2 inherited from generating previous sets $R(k)$, where TabuMatch and Tabu2 start empty before generating the first $R(k)$. Likewise, $\text{Match}(p, q)$ used in updating TabuMatch and $\text{Tabu}(i)$ is inherited from generating previous sets $R(k)$, beginning with $\text{Match}(p, q) = 0$ for all pairs of solutions x^p and x^q in R.
1. Choose x^1 to be the best element of R - (Tabu2 $\cup$ TabuMatch).

[16] By our indexing convention that gives $R(i) = \{x^1, \ldots, x^i\}$, we note that $|R(i)| = i$.

2. For $i = 1, \ldots, k-1$, let

$$x^{i+1} = \text{argmin}(D(x, x(i)) : x \in R - R(i) - (\text{Tabu2} \cup \text{Tabu}(i)).$$

3. For $i = 1, \ldots, k$, set $n(x^i) := n(x^i) + 1$, and if $n(x^i) \geqslant v(i)$, add x^i to Tabu2.

As noted earlier R can change in the time interval between generating one set $R(k)$ and another, and consequently we understand that when a new element x^p of R replaces an old one, the values $\text{Match}(p, q)$ and $\text{Match}(q, p)$ become 0 for all x^q in R, $q \neq p$.

Although we have specified the foregoing rules as a way to flexibly generate a variety of differing subsets of a given set, we remark that these rules can also be adapted to provide tabu conditions for flexibly controlling the generation of solutions within tabu-search methods.

4.2.4. Diversification by complementation

A stronger emphasis on diversification can be achieved by creating new goal vectors x' by complementing chosen solutions $x \in R$. This approach can also be applied by first generating x' as in the Section 4.2.2 or 4.2.3, and then treating x' as the vector x to be complemented. We now indicate more precisely what we mean by the complement of a vector.

For the 0-1 case the complement of x is identified in the usual manner by setting $x'_j = 1 - x_j$ for $j \in N$. For the general integer variable case, we define the complement by once again making reference to an earlier problem (LP′), such as the initial relaxation (LP), and its solution (x'', y''). In the following we refer to an inequality $x''_j \geqslant x_j$ or $x''_j \leqslant x_j$ as *restrictive* if this inequality is either a goal condition used to define (LP′) or else x_j is 0 or U_j, respectively.

Generalized Complementation Rule (to Create a Complement x′ of x)

If $x''_j < x_j$, set $x''_j := \lfloor x''_j \rfloor$ and put $j \in N^-$
If $x''_j > x_j$, set $x'_j := \lceil x''_j \rceil$ and put $j \in N^+$
If $x''_j = x_j$ and $x''_j \leqslant x_j$ is restrictive, set $x'_j := x_j - 1$ and put $j \in N^-$
If $x''_j = x_j$ and $x''_j \geqslant x_j$ is restrictive, set $x'_j := x_j + 1$ and put $j \in N^+$
In all other cases, execute either of the two preceding assignments

The foregoing rule again defines a new (LP′) that becomes the basis for executing Step 2(b) of the parametric TS method.

4.2.5. Path relin king variant

The foregoing strategies based on scatter search also provide a foundation for related path relinking strategies. This can be done by first generating the solution $x' = [x(k)]$ from a center $x(k)$ of $R(k)$ as in Section 4.2.2 or 4.2.3, or by using complementation to generate x' as in Section 4.2.4. We may in fact choose x' to be any solution in the reference set R, and then identify the solution (x'', y'') as in the previous sections to be one whose vector x'' is another solution in R.

In contrast to the scatter search designs previously described, however, the path relinking approach does not create a fully determined set of goal conditions from the new goal vector x' as a means identifying a new problem (LP′) for executing Step 2(b) of the parametric TS method. Instead, x' is treated as a *guiding solution* for a process that begins from the solution (x'', y'') and then moves toward x' along a path created by a series of transitions that introduce only one or a very small number of new goal conditions at each step. We allow x' itself to change in the process, whenever the currently activated goal conditions are found to be unattainable. In this situation, we allow x' to be modified exactly as in the Core Method when goal infeasibility is detected. This approach departs from the customary form of path relinking, where a guiding solution is usually assumed to be feasible. In the present case, x' typically is not a component of a feasible (MIP) solution, and hence the guiding solution may appropriately become modified in the process of moving toward it. On the other hand, in the situation where x' is part of a feasible (MIP) solution, as by choosing it as an element of R rather than as a center or complemented solution, then the steps of moving toward x' may be discontinued after some portion of the total number of goal conditions are activated, permitting the method to transition directly from Step 2(b) back to the Core Method.

The fundamental procedure can be sketched as follows. The problem (LP$'$) identified below may implicitly carry forward an old set of goal conditions with it. Although these conditions define the form of (LP$'$), they are ignored for the purpose of identifying goal infeasibility. In other words, we allow for (LP$'$) to be defined by including reference to an "old goal vector" and associated goal inequalities, but such inequalities are disregarded in the step that checks for the presence of goal infeasibility. Old goal conditions are automatically replaced by new one as new x'_j values are *activated*, i.e., as these explicit new values are generated and their associated j indices are added to N' to define the updated form of (LP$'$).

Path Relinking Subroutine (Inserted in Step 2(b))

0. Start with the solution (x'', y'') to a previous problem (LP$'$), which may have an old associated N' and x', and designate the current N' to begin empty.
1. If $N' = N$, and hence all goal values x'_j have been activated, then return from Step 2(b) to Step 1 of the parametric TS method. Otherwise, execute (a) or (b) below, as appropriate.

   (a) If any assignment $x_j = x''_j$ from the current solution is goal infeasible, treat the variable exactly as in the Core Method to create a new goal value x'_j. (No variables will be goal infeasible on the initial iteration when the current N' is empty, although the residual part of an old N' that shares in defining (LP$'$) may not be empty.)

   (b) If no assignments are goal infeasible, but not all components x'_j of the current goal vector are activated (hence the index j does not belong to the current N'), then choose some such x'_j and apply the following response:
   If $x''_j > x'_j$ then activate x'_j and add j to N^-.
   If $x''_j < x'_j$ then activate x'_j and add j to N^+.
2. Solve the current (LP$'$) and return to Step 1 of the path relinking subroutine.

We observe that the foregoing method actually returns from Step 2(b) to Step 1 of the parametric TS method with (LP$'$) already solved, and hence it is not necessary to re-solve (LP$'$) at the point where this return step is executed. As previously noted, we allow the path relinking method to return to terminate even before $N' = N$, as when $|N'|$ reaches a certain proportion of $|N|$, in the situation where x' is taken directly from R and corresponds to a known (MIP) feasible solution. When x' instead comes from the generation of centers or complements as in Sections 4.2.2 to 4.2.4, then the old N' is empty and the foregoing method simplifies.

As a basis for the choice of x'_j in Step 1(b) above, we may choose the variable with the highest rank using the simple relative choice measure RC_j given by $\mathrm{RC}_j = |x''_j - x'_j|$. More than one of the highest ranking variables (with the largest RC_j values) can be selected in this step, but by the path relinking orientation we generally add only one new variable to N^- or N^+ at a time.

4.2.6. Solution intensified neighborhood spaces

We may consider the use of Solution Intensified Neighborhood (SIN) Spaces as a direct extension of the path relinking approach. Path relinking implicitly involves a specialized (constrained) search over an intensified neighborhood ND* of a given neighborhood ND. The neighborhood ND* is defined in relation to ND by starting from a specified parent solution (designated as the focal solution) and considering those moves that allow a given current solution to incorporate attributes of other parent solutions (designated as guiding solutions). Since the choice of the focal solution is made arbitrarily among the parents, and the relinking process typically allows a focal solution to exchange roles with any of the guiding solutions, the intensified neighborhood ND* in general is independent of any particular way of partitioning the parents into focal and guiding solutions.

The intensified neighborhood ND* is a bit more complex than suggested by the foregoing description, however, for two primary reasons: (i) the original neighborhood ND may consist of moves that do not always allow a current solution to directly incorporate attributes of other specified solutions, but may require intermediate moves in order to make such attributes susceptible to being incorporated (or may require intermediate moves that introduce a mix of such attributes and other attributes); (ii) aspiration criteria may be used that permit the selection of moves that are highly attractive even if they depart from the narrower objective of introducing attributes of one or more guiding solutions.

Nevertheless, we can consider an approach that is dedicated to a search over an intensified neighborhood ND*, without necessarily invoking the rules customarily adopted by path relinking, thus creating more latitude for exploring ND*, in line with the classical notion of an intensification approach. By its nature, path relinking traces out paths in ND* that tend to be relatively direct, avoiding trajectories that may be thought of as wildly zigzagging or tending to double back on themselves. Consequently, the manner in which path relinking searches ND* may cause it to bypass attractive solutions that might potentially be uncovered by an alternative form of search such as, for example, a standard form of tabu-search. Of course, the chance to find additional solutions of interest within ND* by such a search approach may not ultimately produce an advantage over path relinking, first because path relinking may tend to locate most of the solutions that are worth discovering and second because the method operates by initiating a further search of the larger neighborhood ND from chosen solutions encountered along its trajectory (or trajectories) through ND*. Yet, from a conceptual standpoint, the notion of flexibly searching various intensified neighborhoods ND*, which we give the label of "SIN Space Optimization," may be considered a natural extension of path relinking that opens up additional strategies worthy of investigation.

Within the framework that gives rise to these strategies, the following questions emerge as critical for identifying how to best exploit SIN spaces.

(1) What criteria should be applied for selecting the composition of a particular set of parents to define a SIN space? (And how many parents should be allowed to contribute to generating the space?)

(2) What neighborhoods should be employed to search the space? (A broad type of classification, for example, could be founded on a three-part division of neighborhoods into constructive, destructive and transition neighborhoods. More refined classifications can then result as variations within these, as augmented by approaches such as strategic oscillation. Finally, processes designed to exploit specific problem structure can be used as a source of additional forms of classification)
(3) Where should the search of a SIN space be launched? (What constitutes a good "starting point" for searching the space?)
(4) Where should the search exit the space? (What solutions encountered during the search should be subjected to a more thorough Improvement Method, carried out by reference to a neighborhood more encompassing than the one defined by ND*?)

We may call the approach for generating a subset of parent solutions from the Reference Set by the name "*Sub-Clustering*," and refer to each such subset of solutions produced as a *sub-cluster.* Sub-clustering therefore constitutes a process for identifying sub-clusters sets of restricted sizes whose component solutions are related as by a criterion for clustering. Although such sub-clusters can intersect, the collection of all those to be examined contains no duplications, following the usual approach used by scatter search and path relinking. The rules previously identified for generating the sets $R(k)$ constitute an example of one form of sub-clustering that can be used in the context of SIN space optimization, where each such $R(k)$ identifies an intensified neighborhood ND* for solving the problem (MIP).

More particularly, a collection of solutions such as embodied in a set $R(k)$ directly creates an associated neighborhood ND* for branching by considering the branches available to any current problem (LP′) with solution (x'', y'') to be those given by

$$x_j \geqslant \lfloor x''_j \rfloor + 1 \text{ if } x^i_j > x''_j \text{ for at least one solution } x^i \text{ in } R(k).$$
$$x_j \leqslant \lceil x''_j \rceil - 1 \text{ if } x^i_j < x''_j \text{ for at least one solution } x^i \text{ in } R(k).$$

The branching possibilities correspond exactly to those stipulated by (T-UP) and (T-DN) for creating a new (LP′) problem, though we note that those available depend entirely on the composition of $R(k)$. (We use the notation $R(k)$ here to encompass sets that may be generated by different rules than those specified above.) Consequently, in the setting where the original neighborhood ND consists of ordinary branching possibilities for (MIP), the determination of ND* is completely straightforward. Still more simply, the neighborhood ND* corresponds to replacing (MIP) by a more constrained problem (MIP(k)) in which each variable x_j is required to satisfy

$$\text{Min}(x^i_j : x^i \text{ in } R(k)) \leqslant x_j \leqslant \text{Max}(x^i_j : x^i \text{ in } R(k)).$$

Consequently, SIN space optimization can be carried in this setting by applying Parametric TS (or any other method) to solve collections of problems of the form (MIP(k)). These problems are effectively the same as those proposed to be solved in the treatment of strongly determined and consistent variables, as discussed Section 4.1. However, in the context of SIN spaces we have given more specific proposals for identifying the sets denoted here by $R(k)$, and replenish the reference set R (and hence provide new sets $R(k)$) from solutions derived in the course of examining the SIN spaces. We also emphasize the relevance of using aspiration criteria for allowing the search to go outside the bounds stipulated above for the x_j variables.

4.3. Tabu-search intensification by frequency analysis

We propose a form of intensification for parametric tabu-search that makes use of a frequency analysis applied to the same reference set R used in the procedures based on scatter search. We first describe the form of the analysis for 0-1 MIP problems and then indicate the form for more general MIP problems.

In the 0-1 setting, we consider a frequency matrix that contains two rows and two columns for each variable x_j, $j \in N$, representing the rows for a given variable x_j by

$$x_j(0) = (x_{jq}(0,0), x_{jq}(0,1) : q \in N) \text{ and } x_j(1) = (x_{jq}(1,0), x_{jq}(1,1) : q \in N).$$

The term $x_{jq}(\gamma, \delta)$ counts the number of solutions (from the b best) in which $x_j = \gamma$ and $x_q = \delta$, as γ and δ take the values 0 and 1. Thus, these values give the frequency that the assignments $x_q = 0$ and $x_q = 1$ occur in the b recorded solutions for each of the two cases where $x_j = 0$ and 1. (It is sufficient to simply count the number of times these assignments occur, since b is the same for each variable.)

By reference to this matrix and any assignment of values $x_j = x_j^{\text{o}}$, $j \in S$ to a subset of the variables identified by the index set S, we can create *Intensification Scores* $\text{InScore}_h(0)$ and $\text{InScore}_h(1)$ for assigning the values 0 and 1 to selected variables x_h, $h \in H$, where the set H consists of variables whose (UP) and (DN) responses we wish to evaluate. (H may or may not intersect with S, and we identify the composition of these sets more explicitly later on.) In particular, we define

$$\text{InScore}_h(\gamma) = \Sigma(x_{hj}(\gamma, x_j^{\text{o}}) : j \in S - \{h\}), \quad h \in H.$$

Hence, $\text{InScore}_h(0)$ is the sum of the frequency matrix entries for which $x_h = 0$ and $x_j = x_j^{\text{o}}$, and $\text{InScore}_h(1)$ is the sum of the entries for which $x_h = 1$ and $x_j = x_j^{\text{o}}$, thus counting the number of times the assignments of the form $x_j = x_j^{\text{o}}$, for $j \in S - \{h\}$ occur in the b best solutions in the two cases where $x_h = 0$ and $x_h = 1$. The frequency matrix of course need not be constructed explicitly, since we can always calculate the indicated quantities from their definitions, but the existence of the matrix permits these quantities to be generated more rapidly. A further refinement may be introduced by defining $x_{jq}(\gamma, \delta)$ to be a weighted count of the solutions, attaching larger weights to solutions of higher quality. The representation of $\text{InScore}_h(\gamma)$ does not change, but the component terms refer to weighted quantities.

We make use of these scores in an intensification strategy as follows. The subset of variables identified by S consists of those that are currently subject to goal conditions, and x_j^{o} identifies the targeted value for x_j previously denoted by x'_j. (We introduce an variant in the next sub-section that modifies this value, and therefore refer to the variable here by a notation that differentiates it from x'_j.) To evaluate the possible responses for x_h in the case of integer infeasibility, we have $H = D$, the set of unrestricted fractional variables. Instead of generating $\text{IP}_h(\text{UP})$ and $\text{IP}_h(\text{DN})$ values from penalty calculations from the current solution to (LP′), we make use of the historical information of the frequency matrix to set $\text{IP}_h(\gamma') = -\text{InScore}_h(\gamma)$, where $\gamma' = \text{UP}$ when $\gamma = 1$ and $\gamma' = \text{DN}$ when $\gamma = 0$. In other words, the greater the intensification score is for selecting $x_h = 1$ (or $x_h = 0$), the smaller will be the penalty $\text{IP}_h(\text{UP})$ for an (UP) goal condition (or $\text{IP}_h(\text{DN})$ for a (DN) goal condition), allowing penalties to take negative values. Then the ordering of the CP_h values over $h \in H$ ranks the variables according to the relative attractiveness of setting $x_h = 0$ versus $x_h = 1$, in terms of the negatives of the $\text{InScore}_h(\gamma)$ frequency measure rather than in terms of the customary penalty measure.

Similarly, to evaluate the response for x_h in the case of goal infeasibility, where $H = G$, we replace the goal resistance measures $\mathrm{GR}_h(\mathrm{UP})$ and $\mathrm{GR}_h(\mathrm{DN})$ by $-\mathrm{InScore}_h(1)$ and $-\mathrm{InScore}_h(0)$. Alternatively, we may use the CP_j or $-\mathrm{CP}_j$ measures derived from the intensification scores, by the rationale indicated at the end of Section 3.3. A simple example of this occurs by setting

$$\mathrm{GR}_h(DN) = \mathrm{InScore}_h(1) - \mathrm{InScore}_h(0) \quad \text{and} \quad \mathrm{GR}_j(\mathrm{UP}) = \mathrm{InScore}_h(0) - \mathrm{InScore}_h(1).$$

4.3.1. An iterative variant

An iterative approach allows the evaluation of responses for the variables to be taken to another level. The first step applies the evaluations previously indicated, taking x_j^{o} to be the currently targeted value x'_j for x_j, for each $j \in S$. Then each variable x_h, $h \in H$ is assigned a value x_h^{o} derived from this first evaluation. (This can change the value of x_h^{o} to differ from the value x'_h if x_h is among the variables that are subject to a goal condition.) In particular, the two responses $x_h = 1$ and 0 are evaluated for goal infeasible variables as well for unrestricted fractional variables, and the response that receives the higher evaluation is the one that identifies the new value x_h^{o}. If x_h is not subject to a goal condition, this value becomes the first x_h^{o} value for x_h. The set S is accordingly now enlarged to include all variables that are now associated with such a value.

Given the current S and the new x_h^{o} values we may now once again apply the intensification score formula to derive still another round of x_h^{o} values, and the process may thus repeat, either until the x_h^{o} values cease to change or until a limit on the number of iterations is reached. (Such a limit can be as small as 3 or 4.) This approach can also be used as a procedure for dynamically determining the number of variables to receive new goal conditions, by choosing a subset of variables whose assigned x_h^{o} values receive higher evaluations (and in the case of goal infeasible variables, correspond to the uniquely targeted new values for x'_j), with added consideration given to variables whose x_h^{o} values are more consistently assigned over successive iterations.[17]

4.3.2. A diversification counterpart

The intensification scores can be used in a diversification strategy by defining $\mathrm{IP}_h(\gamma') = \mathrm{InScore}_h(\gamma)$ and $\mathrm{GR}_h(\gamma') = \mathrm{InScore}_h(\gamma)$ for $\gamma' = \mathrm{UP}$ and DN and $\gamma = 1$ and 0, respectively. Then, by using the $\mathrm{InScore}_h(\gamma)$ values directly instead of their negatives, the lowest intensification scores for choosing $x_h = 1$ (or $x_h = 0$) give the lowest penalty and resistance measures for an (UP) (or (DN)) condition. This drives the procedure to select conditions for the variables that are least strongly correlated in past solutions. Consequently, for a diversification approach, the solutions used to generate the frequency analysis may be selected to be a collection of solutions whose quality is average or below average, rather than of highest quality. Also, among variables whose preferred responses have close to the same diversification evaluation, a preference should be given to the variable (and response) that receives the highest evaluation relative to the measures of earlier sections, independent of the $\mathrm{InScore}_h(\gamma)$ values. Finally, rather than submitting all variables in G or in D to an evaluation by a diversification criterion, it is appropriate to restrict such an evaluation a limited number of variables and to apply a customary evaluation (or an intensification evaluation) to the rest of the variables. Such a strategy reflects the fact that diversification procedures work best when they include a component that seeks to reinforce quality.

[17] A further refinement of this frequency analysis can make use of clustering and decomposition as illustrated in Chapter 10 of Glover and Laguna [2].

4.3.3. General integer variables

The application of the foregoing types of frequency analysis to MIP problems containing general integer variables (other than 0-1 variables) is straightforward. For any given variable x_h, $h \in H$, to be evaluated, there are just two adjacent integer values v and $v+1$ bracketing the current value of x_h that are relevant to become a possible new value for a goal condition, i.e., to become the next value denoted by x'_h. (A potentially goal infeasible variable may already receive one of these values in the current LP solution.) We then interpret the quantity $x_{jq}(\gamma, \delta)$ to be the count of the solutions in which $x_j \leqslant \gamma$ or $x_j \geqslant \gamma$ according to whether $\gamma = v$ or $v+1$, and similarly in which $x_q \leqslant \delta$ or $x_q \geqslant \delta$ according to whether $\delta = v$ or $v+1$.

To be precise, we specify the quantity $x_{hj}(\gamma, x_j^{\mathrm{o}})$ in the definition of InScore(γ) to be the count of solutions such that $x_h \leqslant \gamma$ or $x_h \geqslant \gamma$ and $x_h \leqslant x_j^{\mathrm{o}}$ or $x_h = x_j^{\mathrm{o}}$ according to whether $\gamma = v$ or $v+1$, and according to whether x_j^{o} is the targeted value for a (DN) or an (UP) condition. Then the previous observations relating to 0-1 variables apply as well to general integer variables. The $x_{hj}(\gamma, x_j^{\mathrm{o}})$ values can be generated as needed, sincere there is no merit in attempting to pre-compute such values in the general case.

4.4. Model embedded tabu memory

We propose a framework for implementing both simple and advanced TS strategies that departs from the customary approach of relying on externally imposed tabu restrictions and instead makes use of an internal *model embedded memory*. We accomplish this by taking advantage of the nature of the parametric process that defines (LP′).

4.4.1. Embedding tabu memory in the objective function

We first consider how recency-based TS memory can be embedded in the model via the objective function. As in parametric branch and bound, we replace M in the objective with varying weights M_j^- for the u_j variables and M_j^+ for the v_j variables. In the simplest case, we vary these weights as a function of the iteration when the associated goal conditions are created, choosing these weights to be larger for recent iterations than for earlier iterations. Then the bounding conditions (UP) and (DN) that these weights seek to impose are more strongly induced for recently targeted conditions than for those generated farther in the past.

An illustration of the process can be given by considering a rule that assigns each weight M_j^- and M_j^+ at iteration i a weight of M^i, where we let M^{o} denote a minimum value of M for iteration $i = 0$. (Iteration $i = 0$ coincides with the initial iteration when N' is empty.) Then we may specify that the value of M^i is given by

$$M^i = (1+r)^i M^{\mathrm{o}}.$$

The larger the value chosen for r, the smaller will be the relative emphasis placed on imposing targets that were introduced in the past.[18]

As new targets replace older ones, there will be an automatic "cleaning out" of earlier M_j^- and M_j^+ weights. However, some of these weights may remain, and we remove weights (along with their associated

[18] Experiments to determine values of r that work best may reasonably begin by examining values in the interval from 0.10 – 0.20.

u_j and v_j variables) that remain after a chosen lapse of time, just as we remove tabu restrictions after their tabu tenures expire. The approach can be reinforced by combining it with the imposition of tabu status as sketched in Section 3.1. This removal of old weights allows the remaining M^i values to be scaled back to prevent them from reaching a size that may cause overflow problems.

In addition, from an implementation standpoint, it is not necessary to choose M^{o} (and hence the other M^i values) to be large. In accordance with the observations of Section 2, it is possible to set $M^{\text{o}} = 1$ by using a two-phase approach for optimizing (LP′). It does not matter that the M_j^- and M_j^+ values are not all the same.

As a variation, rather than starting from M^{o} as a basis for generating the successive M^i values, a maximum value of M can be selected to be applicable to the current iteration, and the values applicable to preceding iterations can be determined by dividing by the quantity $(1 + r)^h$ for an appropriate value of h. This variation makes it possible to create more varied patterns of values for the weights, though it requires slightly more effort per iteration. To illustrate, the division by $(1 + r)$ can be implemented by selecting $r = .4$ for the two most recent iterations, followed by selecting $r = .2$ for the six next most recent iterations, and then selecting $r = .1$ for all iterations remaining until reaching the cut-off number of iterations where M is effectively set to 0. Alternatively, r can be continuously decreased over successive iterations until reaching a minimum value, or a completely independent pattern of M values applicable to successive previous iterations can be used. The determination of patterns of M values that work most effectively affords an area for experimentation.

Within this process, the weights M_j^- and M_j^+ may be normalized by dividing by a positive monotone increasing function of the (LP′) solution value x_j'', for the value of x_j'' at the time the weights were created. Such a normalization compensates for the fact that x_j will be induced to reach an integer value x_j' adjacent to x_j', and the relative change in x_j from x_j'' to x_j' will be smaller as x_j'' becomes larger. As shown later, there can be advantages to replacing these M_j^- and M_j^+ values by their closest integers, as a basis for exploiting another kind of model embedded tabu memory.

Finally, we may go beyond the simple illustrated approach for incorporating recency memory by permitting frequency memory likewise to be incorporated into the model. For example, at a rudimentary level we may bias the weights M_j^- and M_j^+ to be larger if x_j has changed its goal value more frequently, or if the current x_j' value has occurred in higher-quality solutions a larger number of times.

4.4.2. Embedding tabu memory in the problem inequalities

A form of model embedded memory can also be created by taking advantage of the fact that the parametric tabu method automatically gives rise to a valid inequality each time (LP′) is solved. Consider the portion of the objective u_{o} of (LP′) associated with M, i.e., the portion

$$M(\Sigma(x_j : j \in N_0) + \Sigma(U_j - x_j : j \in N_U) + \Sigma(u_j : j \in N^- - N_0) + \Sigma(v_j : j \in N^+ - N_U)).$$

When a single "big M" value is used, as in the preceding expression, then the solution to (LP′) minimizes the term

$$z_{\text{o}} = \Sigma(x_j : j \in N_0) + \Sigma(U_j - x_j : j \in N_U) + \Sigma(u_j : j \in N^- - N_0) + \Sigma(v_j : j \in N^+ - N_U)$$

to give a value $z_o = z''_o$ associated with a solution $x = x''$ to (LP′). We then may write $z_o \geqslant \lceil z''_o \rceil$, or

$$\begin{aligned}&\Sigma(x_j : j \in N_0) + \Sigma(U_j - x_j : j \in N_U) + \Sigma(u_j : j \in N^- - N_0)\\&\quad + \Sigma(v_j : j \in N^+ - N_U) \geqslant \lfloor z''_o \rfloor.\end{aligned} \tag{A}$$

The inequality (A) is clearly valid for all MIP feasible solutions, and if it is generated from a solution that is goal infeasible, then it excludes the solution $x = x''$ that produced this goal infeasibility. Consequently, it is relevant to consider a procedure that generates and maintains a collection of such inequalities from the solutions to successive (LP′) problems when goal infeasibility is encountered.

To do this, we apply a two-phase process to solve (LP′) as sketched in Section 2, applying the first phase to minimize z_o. The resulting solution gives the inequality (A) directly. Moreover, this inequality can be tightened by dropping each variable that has a reduced cost greater than 1 in the optimal Phase 1 solution (or a reduced cost greater than M in the optimal solution using the "big M" objective).[19]

In the case where the solution to (LP′) yields a feasible (MIP) solution, and hence z''_o is an integer, the resulting inequality (A) can be strengthened by replacing $\lceil z''_o \rceil$ with $\lceil z''_o \rceil + 1$. If all goal conditions are satisfied exactly ($x'_j = x''_j$) then the associated value of z''_o is 0. In the case where no new MIP feasible solution is found, but where all current goal conditions are satisfied, we exclude consideration of the associated inequality (A). Such an excluded inequality yields no useful information.

The foregoing accumulation of a collection of inequalities of the form of (A) encounters a serious limitation, however. Each such inequality involving a u_j or v_j variable must be accompanied by the associated equations $x_j = x'_j + u_j - v_j$, $j \in N' - N_0 - N_U$, and the changing identities of the u_j and v_j variables (as the values x'_j change from problem to problem) cause the number of variables embodied in the formulation to grow.

To avoid having to include a growing collection of u_j and v_j variables and their associated equations $x_j = x'_j + u_j - v_j$, we replace u_j by $x_j - x'_j$ and replace v_j by $x'_j - x_j$. Since $x'_j = 0$ for $j \in N_0$ and $x'_j = U_j$ for $j \in N_U$, the resulting inequality can be summarized by

$$\Sigma(x_j - x'_j : j \in N^-) + \Sigma(x'_j - x_j : j \in N^+) \geqslant \lceil z_o \rceil. \tag{B}$$

For the 0-1 problem, $N^- = \{j : x'_j = 0\}$ and $N^+ = \{j : x'_j = 1\}$, and the inequality (B) is the same as (A) and the sets N^- and N^+ are disjoint. In the case of integer variables that are not 0-1, if the same variable belongs to both N^- and N^+ then we only include a single instance of the variable—the instance associated with the variable u_j or v_j that is binding in the solution to (LP′). (We may assume that (B) is obtained from (A) after first dropping variables having sufficiently large reduced costs, as indicated previously.) While (B) is not the same as (A) when the sets N_0 and N_U do not compose all of N', (B) nevertheless implies (A) and we can make strategic use of the inequality in the context of tabu-search. Just as in the case of (A), when the solution to (LP′) gives a feasible MIP solution, we may tighten the inequality by replacing $\lceil z''_o \rceil$ with $\lceil z''_o \rceil + 1$, since this replacement is necessary and sufficient to remove the MIP solution from future consideration. Similarly, apart from this special case, we exclude consideration of inequalities (B) that otherwise result when no goal conditions are violated.

The usefulness of a collection of non-excluded inequalities (B) derives from the following fact. These inequalities will not permit a previously assigned set of goal conditions to be reassigned, provided the responses (R-UP) and (R-DN) are employed in the presence of goal infeasibility as previously specified

[19] These variables can take any if the forms x_j, $j \in N_0$ or $U_j - x_j$, $j \in N_U$ or u_j, $j \in N^- - N_0$ or $v_j : j \in N^+ - N_U$.

in the rules of the parametric TS method. Thus these inequalities may operate as supplementary forms of tabu restrictions that are adjoined to the problem constraints as the method progresses. In the same way that tabu restrictions are allowed to expire, we place a cut-off limit on the number of inequalities (B) incorporated within (LP′). Accompanying this, we adopt an aspiration criterion that assigns higher value to non-redundant inequalities than to currently redundant inequalities, and hence when the cut-off limit is reached we first drop the most redundant inequality and otherwise drop the oldest non-redundant inequality. Inequalities that become and remain redundant for a chosen duration can be dropped even if the cut-off limit is not presently attained. Should the collection of inequalities result in the situation where no LP feasible solution exists, then again the oldest non-redundant inequality is dropped, or else this may be used as an indication to terminate the search, which is a valid termination condition for 0-1 MIP problems.

The approach can be extended by maintaining a record of some set D of discarded inequalities, without explicitly adding them as constraints to (LP′). In this case, a single surrogate constraint inequality, created as a non-negative linear combination of the inequalities of D, is added to (LP′) in place of the members of D.

4.4.3. *Differential weights on the variables*

The same approach can be used to generate inequalities when different weights M_j^- and M_j^+ are used instead of a single weight M. These weights may be controlled to fall within a reasonable range, as in solving (LP′) with the two-phase approach. A natural way to do this is provided by the method of Section 4.3.1 that embeds TS memory within the objective function.

Then, letting $[M_j^-]$ and $[M_j^+]$ denote the integer values closest to M_j^- and M_j^+, we solve (LP′) with the parameterized Phase 1 objective given by[20]

$$\begin{aligned} z_\text{o} = {} & \Sigma[M_j^-](x_j : j \in N_0) + \Sigma[M_j^+](U_j - x_j : j \in N_U) + \Sigma[M_j^-](u_j : j \in N^- - N_0) \\ & + \Sigma[M_j^+](v_j : j \in N^+ - N_U). \end{aligned}$$

Defining z''_o as before (relative to the new representation of z_o), and replacing the u_j and v_j variables by $x_j - x'_j$ and by $x'_j - x_j$, we now obtain the inequality

$$\Sigma[M_j^-](x_j - x'_j : j \in N^-) + \Sigma[M_j^+](x'_j - x_j : j \in N^+) \geqslant \lceil z''_\text{o} \rceil. \tag{C}$$

In this case a variable $x_j - x'_j$, $j \in N^-$ or $x'_j - x_j$, $j \in N^+$ can be dropped from (C) if it has a reduced cost greater than $[M_j^-]$ or $[M_j^+]$, respectively, in an optimal LP solution to the problem of minimizing z_o. The resulting inequalities can be accumulated and adjoined in the same manner indicated for accumulating and adjoining inequalities of the simpler form (B). In the case where a feasible MIP solution is obtained, we can still make use of the earlier form of (B) and replace $\lceil z''_\text{o} \rceil$ by $\lceil z''_\text{o} \rceil + 1$.

4.4.4. *An intensification method using embedded tabu memory*

The set of goal conditions defining (LP′) is uniquely determined by the sets N^- and N^+ that compose N' and the associated vector x'. Upon identifying an optimal LP solution to (LP′) and the resulting value

[20] Fuller differentiation can be obtained by multiplying M_j^- and M_j^+ by a factor of 10 before identifying the values $[M_j^-]$ and $[M_j^+]$.

z''_o we have all the information required to generate the inequalities (B) and (C). We represent a collection of such inequalities, indexed over a set *P*, and associated with sets $N'(p)$ ($N^-(p)$ and $N^+(p)$) and goal solutions $x'(p)$, $p \in P$, by

$$\Sigma(\alpha_{pj} X_j : j \in N'(p)) \geqslant \alpha_{po}, \quad p \in P \tag{D}$$

The constant term α_{po} results by transferring the x'_j values in (B) and (C) to the right side of the inequality. All coefficients α_{po} and α_{pj}, $j \in N'(p)$, are integers, and may be both positive and negative.

For the purpose of the intensification strategy that makes use of (D), we base the inequalities indexed by *P* on a collection of high quality solutions, consisting of either MIP feasible solutions (where $\lceil z''_o \rceil$ has been replaced by $\lceil z''_o \rceil + 1$) or of goal infeasible solutions that are close to MIP feasible. In each case, the inequalities over *P* eliminate such solutions from the LP feasible set.

Our goal for exploiting (D) is to find a new solution that is close to those in the collection of high quality solutions that give rise to (D). We introduce slack variables s_p, $p \in P$, to permit the system (D) to be expressed equivalently as

$$\Sigma(\alpha_{pj} x_j : j \in N_p) - s_p = \alpha_{po}, \; s_p \geqslant 0, \quad p \in P \tag{E}$$

We then create an *Intensified LP Relaxation* (LP-Int) of (MIP) is then created by modifying the original relaxation (LP) to include the system (E) and replacing the objective function of (LP) by the new objective

$$\text{Minimize } s_o = \Sigma(w_p s_p : p \in P).$$

where the weights w_p for the variables s_p are selected to be positive integers.[21] We also stipulate that (LP-Int) includes the currently updated objective function constraint $cx + dy \leqslant x^*_o - \in$.

The Intensified LP Relaxation then serves as a starting point for launching a new phase of the parametric TS method. Specifically, (LP-Int) becomes the problem (LP^o) mentioned in Section 3.5 that is solved in place of problem (LP$'$) in Step 2(b) of the parametric TS method, thereby launching an Intensification Phase based on the new objective. In this case, no goal conditions are enforced and N' begins empty. However, in subsequent steps of the method, as such goal conditions are produced, the inequalities of (D) that underlie (E) can be carried forward and incorporated into (LP$'$), and updated exactly in the way previously specified for updating (B).[22]

The Intensified LP Relaxation can appropriately be varied by selecting *P* by reference to different subsets of the best solutions found. This can have an important influence, given that the best solutions may come from different regions. For the purpose of intensification it is more meaningful to choose subsets of solutions that come from a common region than from regions whose solutions bear little resemblance to each other. A useful way to capture a regional influence is to generate subsets of the best solutions by a clustering procedure, where the members of a common cluster are conceived to belong (by definition) to a common region. In fact, the rule for generating a collection of subsets $R(k)$ of the reference set *R* in Section 4.2.2 provides a means for producing such clusters.

[21] We can similarly create an objective to minimize the maximum deviation between the left- and right-hand sides of any inequality of (D).

[22] A separate collection of inequalities of the form of (B) or (C) can also be embedded in (LP-Int) and carried forward into (LP$'$). For example, the record of best solutions from which the collection (D) is derived may include members that are not all embodied in a collection (B) or (C) previously used to guide the method, and members of such a previous collection may also be carried forward.

Once a collection of relevant solution subsets has thus been used to generate associated problems (LP-Int) to launch one or more Intensification Phases, then the initiation of subsequent Intensification Phases must await the generation of additional high-quality solutions. To accentuate the influence of these additional solutions, larger weights w_p can be given to their associated s_p variables.[23]

4.4.5. A diversification analog of the system (E)

To create a diversification procedure for generating new starting solutions for the parametric TS method, we seek an objective function to replace that of (LP-Int) to drive the search to a vicinity close to a collection of solutions that complement the goal solutions underlying the system (E).

For any given set of goal conditions, determined by sets N^+ and N^- and goal values x'_j, $j \in N' = N^+ \cup N^-$, we define a complementary set of goal conditions and associated complementary goal values as follows:

Complementary Goal Rule

If $j \in N^+$, replace the (UP) condition $x_j \geqslant x'_j$ by $x_j \leqslant x'_j - 1$ and move j to N^-.

If $j \in N^-$, replace the (DN) condition $x_j \leqslant x'_j$ by $x_j \geqslant x'_j + 1$ and move j to N^+.

The operations above that move j from one set to another are to be interpreted so that the net effect is interchange the composition of the sets N^+ and N^-.

The objective function to replace that of (LP-Int) may now be determined as follows. Starting from an original system (D) (equivalently, (E)) defined relative to the goal values $x'_j(p)$, $p \in P$, we determine new complementary $x'_j(p)$ values and sets $N^+(p)$ and $N^-(p)$ for each $p \in P$ by the Complementary Goal Rule. For the case where (E) was derived from inequalities of the form of (B), and making reference to the complemented values and goal conditions, we define

$$s_p = \Sigma(x_j - x'_j(p) : j \in N^-(p)) + \Sigma(x'_j(p) - x_j : \ j \in N^+(p)).$$

The variable s_p is unrestricted in sign, although it will automatically be non-negative for 0-1 MIP problems. In the case where (E) is derived from inequalities of the form of (C), we re-express s_p by attaching coefficients to the terms of the summations, but retain the unique correspondence between these coefficients and the associated j indices. Hence, for example, a coefficient $[M_j^-(p)]$ attaches to j in the new set $N^+(p)$. Thus in both cases, for appropriately defined coefficients β_{pj} and constant term β_{po} we can write s_p in the form

$$s_p = \Sigma(\beta_{pj} x_j : j \in N) - \beta_{po}.$$

Relative to the foregoing modified definition of s_p we define a *Diversified LP Relaxation* (LP-Div) to result by minimizing s_o in the same form as before. Hence, we select positive integers w_p and replace the objective of (LP) by the objective

$$\text{Minimize}\, s_o = \Sigma(w_p s_p : p \in P).$$

[23] (E) can be used to generate still other inequalities. Specifically, if the optimal LP solution value s''_o for minimizing s_o is fractional, the inequality $s_o \geqslant \lceil s''_o \rceil$ may be incorporated into subsequent (LP′) problems.

In contrast to (LP-Int), however, we do not adjoin (E), since the s_p variables are no longer defined by reference to this system. In fact the s_p variables in the objective for (LP-Div) are unrestricted and hence can be removed from the problem by replacing each s_p in the objective by its defining equation. The resulting equivalent objective may be written as

$$\text{Minimize}\, s_{\text{o}} = \Sigma(\gamma_j x_j : j \in N) - \gamma_{\text{o}},$$

where $\gamma_j = \Sigma(w_p \beta_{pj} : p \in P)$ and $\gamma_{\text{o}} = \Sigma(w_p \beta_{p\text{o}} : p \in P)$.

By reference to this objective, we may execute *Diversification Phases* of the parametric TS method in a manner exactly analogous to the execution of Intensification Phases, treating (LP-Div) as the problem (LP^{o}) that is solved in Step 2(b) of the parametric TS method instead of (LP′).

5. A dual post-optimizing version of parametric TS

Current state-of-the-art software for solving MIP problems by branch and bound and by branch and cut is primarily adapted for using dual post-optimization. Consequently, it is useful to observe how the ideas underlying parametric tabu-search can be applied in conjunction with the dual simplex method rather than the primal method for carrying out post-optimizing steps.

To use dual post-optimization we do not embody goal conditions in the objective function, but rather impose such conditions as explicit branching constraints. However, these constraints must be managed to allow the identification of variables that are to be subjected to goal responses in this new setting.

To do this, we observe that goal infeasibility now corresponds to true LP infeasibility, where the current problem (LP′) based on the use of explicit branching constraints discloses the absence of a continuous feasible solution. The class of potentially goal infeasible variables becomes irrelevant and is eliminated. Infeasibility is recognized during the execution of the dual simplex method by encountering a pivot row, corresponding to a violated upper or lower bound for a current basic variable that contains no admissible pivot element. More precisely, we may write the pivot row equation in the form

$$z_i + \Sigma(a_{ij} x_j : j \in \text{NB}(x)) + \Sigma(d_{ij} y_j : j \in \text{NB}(y)) = b_i, \quad i \in B,$$

where NB(x) and NB(y) denote the index sets for the current non-basic x variables and y variables, B denotes the index set for the current basic variables, and the basic variable z_i can represent either an x variable or a y variable. The coefficients a_{ij}, d_{ij} and b_i are derived from the current basis representation, and do not correspond to coefficients of the original A and D matrices or b vector. Moreover, given that the solution process currently incorporates a set of branching inequalities of the form of (UP) and (DN) (i.e., $x_j \geqslant x'_j$ for $j \in N^+$ and $x_j \leqslant x'_j$ for $j \in N^-$), we will understand that the x_j variables in the pivot row equation can refer to slack variables s_j for such associated inequalities (i.e., $s_j = x_j - x'_j$ for $j \in N^+$ and $s_j = x'_j - x_j$ for $j \in N^-$). We suppose, as in the case of these slack variables, all non-basic variables in the pivot row equation refer to variables that are non-basic at their lower bounds, since we may complement variables that are non-basic at their upper bounds to assure this. Similarly we suppose that the bound violation for z_i refers to violating a lower bound of 0.

By these conventions, the coefficients of the pivot row equation signal the absence of a feasible LP solution by the following conditions:

$$b_i < 0, \quad a_{ij} \geqslant 0, \quad j \in \text{NB}(x), \quad d_{ij} \geqslant 0, \quad j \in \text{NB}(y).$$

We seek to identify a set of branching inequalities, and their associated variables, that have a role in producing the infeasibility identified by the pivot row equation. These inequalities and variables may be tagged by an index set G analogous to the one that identifies the set of goal infeasible variables in the case of the goal infeasibility for the parameterized form of (LP$'$).

If the variable z_i corresponds to a slack variable for a branching inequality, then the index i belongs to G. In addition, G includes the indices of all variables x_j, $j \in \mathrm{NB}(x)$, such that x_j corresponds to a slack variable s_j and such that $a_{ij} > 0$. (As before, for convenience we will refer to G as a set of x_j variables as well as a set of indices for these variables.)

By reference to this determination of G, we then seek as earlier to identify primary and secondary subsets G_P and G_S of G consisting of variables whose branching inequalities will be reversed and discarded, respectively. This can proceed exactly as in Sections 2.2.1 and 2.2.2, by creating measures GR_j that indicate the degree to which a branching inequality currently associated with a variable x_j is resisted by the solution to (LP$'$)- or, inversely, the degree to which a reversal of such a branching inequality would ameliorate the state of this solution to (LP$'$). This can be done by observing that reversing a branch inequality corresponds to replacing a current slack s_j by a new slack $t_j = -s_j - 1$, which replaces x'_j by $x'_j - 1$ or $x'_j + 1$ depending on whether $j \in N^+$ or $j \in N^-$. If this reversal is done for a variable x_j that represents a slack variable s_j in the pivot row equation, then by substitution the row equation changes its form relative to this variable (where x_j now represents t_j) by setting

$$b_i := b_i - a_{ij} \quad \text{and} \quad a_{ij} := -a_{ij}.$$

Consequently, the negative b_i value becomes more negative and the positive a_{ij} reverses its sign. This change in a_{ij} removes the immediate indication of infeasibility. However, the new x_j also receives a new coefficient c_j in the current objective function representation that is the negative of its previous coefficient. Consequently, to identify the full effect of allowing x_j to represent t_j, by reference to a dual feasible representation, unless $c_j = 0$ we must recapture the dual feasibility condition $c_j \geqslant 0$ by complementing x_j relative to its current upper bound. (This bound is $U_j - x'_j$ for $j \in N^+$ and x'_j for $j \in N^-$, relative to the new x'_j value and the new composition of N^+ and N^- created by the branch reversal.) Undertaking to trace the full effects of this change can be computationally onerous, and hence we may use the simpler measure GR_j of resistance by setting $\mathrm{GR}_j = |a_{ij}|$ (or the negative of this value). However, the preceding steps are required if the branch is actually selected to be reversed as a result of assigning x_j to the primary set G_P.

A slight complication results in the case where a slack for a branching inequality has a 0 upper bound, as results when the branch compels x_j to equal U_j or 0. In this situation, which occurs for branches involving a 0-1 variables, MIP codes will typically drop the 0-constrained slack variable as soon as it is non-basic, and hence such a variable may not be explicitly included in the current pivot row equation. The form of the variable in the pivot equation must then be recovered by an operation of the basis inverse in order to apply the preceding analysis. A variable s_j that has been dropped in this manner may have a negative a_{ij} coefficient when it is recovered. In this case complementing the variable has the effect of increasing b_i, though for simplicity we may still take the GR_j measure to be a function of the value $|a_{ij}|$.

Finally, if the basic variable z_i corresponds to a slack variable s_i, then replacing s_i by t_i results in a new z_i whose representation is given by setting

$$b_i := -b_i - 1 \quad \text{and} \quad a_{ij} := -a_{ij}, \quad j \in \mathrm{NB}(x), \;\; d_{ij} := -d_{ij}, \;\; j \in \mathrm{NB}(y).$$

If the resulting value of b_i is non-negative, then the infeasibility condition is removed. In any case the reversed signs of the coefficients in the pivot row equation permit feasibility to be re-established. Moreover, if $b_i < 0$ by this change, we may immediately evaluate the effect on x_o of a single dual pivot on the pivot row equation by simply computing ratios relative to the objective function, and this may be used as a basis for creating a measure GR_i applicable to reversing the branch for z_i.

Other methods can be used to determine GR_j values, referring to pseudo-penalties or making use of tabu-search memory—as, for example, by referring to prior evaluations associated with identifying strongly determined and consistent variables. Regardless of how the GR_j values are generated, however, all other components of the parametric TS method remain the same as previously described.

6. Conclusions

Parametric tabu-search for mixed integer programming (MIP) involves a collection of strategies that provide significantly greater flexibility than the branching strategies of branch and bound methods. The ability to rely on adaptive tabu memory in place of more rigid tree search memory opens up a wide domain of strategic variations embodying alternative forms of recency and frequency memory and various associated types of intensification and diversification procedures.

The application of tabu-search to the MIP setting affords a broad range of new possibilities for exploration. In terms of both implementation and testing, TS applications in MIP are at approximately the same stage of development as the primitive applications of branch and bound that first emerged four decades ago. Yet the potential strategic variation available for applying TS to MIP is substantially richer than available with branch and bound. A good deal of experimentation lies ahead to identify combinations of elements within this potential variation that yield the best results. The ideas and procedures described in this paper are offered in the hope that they may provide a useful blueprint to guide such exploration.

Acknowledgements

This research was partially supported by the Office of Naval Research Contract N00014-01-1-0917 in connection with the Hearin Center of Enterprise Science at the University of Mississippi.

Appendix A

A.1. Choosing the values $g_\mathrm{P} = |G_\mathrm{P}|$ and $g_\mathrm{S} = |G_\mathrm{S}|$.

Using Example A as a source of motivation, we examine the issue of choosing the values g_P and g_S that give the cardinalities of the primary and secondary goal sets G_P and G_S. A straightforward rule is to specify that these values are some fractions f_P and $f_\mathrm{S}(\leqslant 1)$ of the cardinality of the set G of goal infeasible variables, subject to requiring $g_\mathrm{P} \geqslant g_{\mathrm{Pmin}}$ and $g_\mathrm{S} \geqslant g_{\mathrm{Smin}}$, for minimum values g_{Pmin} and g_{Smin} selected for g_P and g_S which are desired to apply regardless of the size of G. (By the policy indicated for handling goal infeasibilities, g_{Pmin} must always be at least 1 if G is not empty, but g_{Smin} can be 0.) Thus the rules take the form

$$g_{\mathrm{P}} = \mathrm{Max}([f_{\mathrm{P}}|G|],\ g_{\mathrm{Pmin}}) \quad \text{and} \quad g_{\mathrm{S}} = \mathrm{Max}([f_{\mathrm{S}}||G|],\ g_{\mathrm{Smin}}),$$

where $[v]$ represents the nearest integer neighbor of v. If the resulting value of g_{P} exceeds $|G|$, then g_{P} will instead be given by $|G|$ (yielding $g_{\mathrm{S}} = 0$), and the value of g_{S} will be limited in any case to $|G| - g_{\mathrm{P}}$.

To illustrate, we may suppose the rules used in Example A were given by

$$g_{\mathrm{P}} = \mathrm{Max}([.3|G|],\ 2) \quad \text{and} \quad g_{\mathrm{S}} = \mathrm{Max}([.2||G|], 0).$$

Thus for $|G| = 4$ we obtain $g_{\mathrm{P}} = \mathrm{Max}([1.2],\ 2)$ and $g_{\mathrm{S}} = \mathrm{Max}([.8],\ 0)$, yielding $g_{\mathrm{P}} = 2$ and $g_{\mathrm{S}} = 1$.

An alternative embodiment of this approach that may be more convenient is to select a value g representing the sum of g_{P} and g_{S}, using a corresponding rule

$$g = \mathrm{Max}([f|G|],\ g_{\min}).$$

We then use the same formula for g_{p} as before, except that $|G|$ is replaced by g. The value g_{s} becomes simply the remaining part of g after reducing it by g_{P}, to give

$$g_{\mathrm{P}} = \mathrm{Max}([f_{\mathrm{P}}g],\ g_{\mathrm{Pmin}}) \quad \text{and} \quad g_{\mathrm{S}} = g - g_{\mathrm{P}}.$$

Illustrating again by reference to Example A, we may suppose the foregoing rules were given the form

$$g = \mathrm{Max}(.[8|G|],\ 2) \quad \text{and} \quad g_{\mathrm{P}} = \mathrm{Max}([.7g],\ 1).$$

Thus we obtain $g = \mathrm{Max}([3.2],\ 2) = 3$, $g_{\mathrm{P}} = \mathrm{Max}([2.1],\ 1) = 2$, and $g_{\mathrm{S}} = 3 - 2 = 1$.

In general, the parameters f, f_{P}, $g_{\min}$ and g_{Pmin} may be set by a schedule that first assigns them larger values and then decreases these values over time. This causes the values g, g_{P} and g_{S} correspondingly to diminish over time, thereby reducing the number of goal conditions changed on a given iteration. Such a strategy makes it possible to begin with a *diffuse search* that induces a relatively large number of changes on each iteration, and to proceed to a more *refined search* that focuses on making a smaller number of key changes at each step. The diffuse search has the function of quickly moving the search into a promising region and the refined search has the function of driving the exploration from there to reach a solution of somewhat higher quality. Because refined search is often essential for finding the best solutions, it can be appropriate in some settings to rely exclusively on this type of search, compelling the values g, g_{P} and g_{S} to be small, perhaps even at most 1, throughout all search phases.

Transitions between diffuse and refined levels can be controlled with a useful degree of flexibility by means of the TS strategic oscillation approach. In this case, smaller parameter values are restored periodically to larger values, according to a pattern influenced by transitions that produced past successes. The best solutions are re-inserted into the process at various stages. Strategic oscillation permits the search to operate with less reliance on other types of memory (typically permitting a smaller tabu tenure for short-term memory) and provides a useful structure for managing the interplay between intensification and diversification.

Finally, we observe that instead of choosing the elements of G_{P} and G_{S} sequentially from G, in the decreasing order of their goal resistance values, we can apply the type of decision rule employed in probabilistic tabu-search, as by assigning a probability P_j of selecting a given variable x_j in G that is proportional to a non-decreasing function of the evaluation GR_j. The approach then chooses elements of G_P by sampling g_P times without replacement from G according to the assigned probabilities, followed by choosing the elements of G_S in the same fashion. The introduction of tabu status modifies the probabilities by the mechanism of changing the resistance measures GR_j, as discussed in Section 3.2.

A.2. Choosing g_P and g_S for potential goal infeasibility

The foregoing ideas for choosing g_P and g_S can be applied directly to the situation that includes potentially goal infeasible variables. It suffices simply to enlarge G to include the set G^o, which we define to consist of those variables that exactly satisfy their goal conditions and whose goal resistances satisfy the threshold T^o, i.e.,

$$G^\mathrm{o} = \{j \in N' : \; x''_j = x'_j \quad \text{and} \quad \mathrm{GR}^\mathrm{o}_j = T^\mathrm{o}\}.$$

As in Example B of Section 2.2.3, if we use the negative reduced cost measure $\mathrm{GR}^\mathrm{o}_j = -\mathrm{RC}_j$, then we may choose T^o to be a relatively small fraction of $-M$ (e.g., $T^\mathrm{o} = -.1M$). With such a provision, the rules described above carry over unchanged.

In the case where G is relatively large without including G^o, it can be reasonable to forego enlarging G and to bypass the treatment of potentially goal infeasible variables. In general, we can place limits on the number of elements admitted to G^o independent of the determination given by T^o. Such considerations involving the use of thresholds are treated in [6].

We note that the starting point for the analysis, illustrated by the tables in Examples A and B, allows a variety of different initial choices of G, and of G_P and G_S within G, to be investigated. Thus, for example, in Example B the same initial information from Table 2 can be used to investigate the case of focusing on just the first three variables, with the intent of giving two of them a new goal condition and freeing the third (i.e., taking $g = 3$, $g_\mathrm{P} = 2$ and $g_\mathrm{S} = 1$). Thus, more than one scenario can be examined without requiring new information to be generated at the beginning. Only the new (LP′) problem based on the new x' vector changes. This new problem is more nearly the same as the starting (LP′) problem when G_P and G_S are small, hence entailing less computation when post-optimizing to solve the new (LP′) problem. The added advantage that results for “small G” scenarios gives a motivation for devoting more attention to stages of refined search versus stages of diffuse search involving changes in larger numbers of elements—a motivation that is reinforced by the effects of conditionality. That is, when the best choices for some goal conditions depend on the choices of others, a diffuse search that changes many conditions simultaneously is less likely to be able to uncover the best solutions.

Appendix B

B.1. Deferred decisions: A deferred determination of freed variables

The penalties for handling integer infeasibility can be used in an additional way, to support a decision process that determines the freed variables in a somewhat different manner than previously specified.

Instead of specifying a rule in advance for identifying the goal infeasible variables to be freed, such as selecting them as the g_S elements of the secondary goal set G_S after first identifying the g_P elements of the primary goal set G_{P}, a deferred determination of these variables can be made as follows. First, as in Section 2.2.2, we choose g variables from the set of variables in G to be allocated to the sets G_{P} and G_{S}, giving $g = g_{\mathrm{P}} + g_{\mathrm{S}}$. Call this set G_{PS}, since it represents the union of the sets that will ultimately be identified as G_{P} and G_{S} (hence $g = |G_{\mathrm{PS}}|$). Then we select a set G_{o} of *provisionally free* variables, consisting of some number g_{o} of these g variables having the highest goal resistance values, where $g_{\mathrm{o}} > g_{\mathrm{S}}$ (thus assuring that these variables will include a positive number of the variables to be assigned to the set G_{P}). The goal response (R-UP) or (R-DN) applicable to each provisionally free variable in G_{o} is noted, but not executed.

We assume that G includes the set G^{o} of potentially goal infeasible variables. For each x_j in this G, we differentiate the current goal value x'_j from the new one identified in the transition (T-UP) or (T-DN) by denoting the latter by x^{o}_j; i.e., $x_j \geqslant x^{\mathrm{o}}_j$ identifies the inequality targeted by (T-UP) and $x_j \leqslant x^{\mathrm{o}}_j$ identifies the inequality targeted by (T-DN). (For instance, Examples A and B identify the x^{o}_j values over $j \in G_{\mathrm{P}}$ by the numbers marked with an asterisk in the bottom row.)

The customary determination of the primary and secondary responses partitions G into the three sets G_{P}, G_{S} and $G - G_{\mathrm{PS}}$,[24] which respectively involve the three decisions of (1) changing x_j's current goal condition, (2) freeing x_j, and (3) leaving x_j's goal condition unchanged. We select any subset G^* of G containing elements that are subjected to at least two of these decisions, i.e., G^* has a non-empty intersection with at least two of the three sets G_{P}, G_{S} and $G - G_{\mathrm{PS}}$. Following earlier conventions, we treat G not simply as a set but as a vector whose elements are sequenced in descending order of the GR_j and $\mathrm{GR}^{\mathrm{o}}_j$ values (as illustrated in Example B). Then, we further stipulate that G^* is a sub-vector of G, i.e., G^* is composed of consecutive elements of G. (This condition can be superseded by tabu criteria and by implementing a TS diversification step.)

Then the deferred decision process operates as follows:

Deferred Decisions for Elements of G^*.

1. For the variables x_j in $G - G^*$, implement the decisions (1)–(3) as specified in the determination of the sets G_{P}, G_{S} and $G - G_{\mathrm{PS}}$.
2. Provisionally (temporarily) free the variables x_j in G^*, but keep a record of the goal condition $x_j \geqslant x^{\mathrm{o}}_j$ or $x_j \leqslant x^{\mathrm{o}}_j$ targeted for x_j by its membership in G, and also keep a record of the values $n(1)$, $n(2)$ and $n(3)$, identifying the number of variables in G^* that were prescribed to be treated by decisions (1)–(3) in the partition of G. (Thus, $n(1) = |G^* \cap G_{\mathrm{P}}|$, $n(2) = |G^* \cap G_{\mathrm{S}}|$, $n(3) = |G^* \cap G_{\mathrm{PS}}|$. At most one of these cardinalities can be 0.)
3. Solve the *provisional* (LP′) *problem* determined by steps 1 and 2.
4. By reference to the solution (x', y'') to the provisional (LP′) problem, identify the penalty $\mathrm{IP}^{\mathrm{o}}_j$ that results for each x_j in G^* by enforcing the goal condition recorded in step 2. ($If\, x_j = x''_j$ satisfies the goal condition, $\mathrm{IP}^{\mathrm{o}}_j = 0$. Otherwise, $\mathrm{IP}^{\mathrm{o}}_j$ is given by a standard penalty calculation for enforcing the violated goal condition.)
5. Order the elements of G^* in ascending order of the$\mathrm{IP}^{\mathrm{o}}_j$ values. ("Smaller is better.") The first $n(1)$ elements of G^* are assigned to G_{P} and handled by the goal condition saved in step 2, the next $n(2)$ elements are freed from their goal conditions (i.e., changed from provisionally free to *actually* free), and the final $n(3)$ elements are handled by the original goal condition that applied to them

[24] Of these sets, at least G_{P} is non-empty. Recall that G_{PS} is the union of G_{P} and G_{S}.

Table 5

1	$J =$	1	2	3	4	5	6
2	$x'_j =$	1	0	0	1	0	1
3	$x''_j =$	0.6	0.3	0.2	0.9	0	1
4	$\mathrm{GR}_j =$	0.4	0.3	0.2	0.1		
5	$\mathrm{GR}^o_j =$					−4	−25
	New $x'_j =$	0^*	1^*	1^*	#	#	1
	$x^o_j =$	0	1	1	0	1	0

Otherwise, x_j retains its goal condition (type (3) decision).
$^* = x_j$ changes its goal condition (type (1) decision).
$^\# = x_j$ is freed from its goal condition (type (2) decision).

before implementing this procedure. These give the final form of (LP′) that completes the deferred determination of the decisions applicable to the set G^*.

The penalty value IP^o_j of step 4, for the simple case where penalties are measured by the L1 distance measure is given by $\mathrm{IP}^o_j = |x^o_j - x''_j|$ when x''_j violates (or exactly satisfies) the goal condition associated with x^o_j. (This corresponds to the standard penalty measures illustrated earlier that yield $\mathrm{IP}_j(\mathrm{UP}) = f^+_j$ and $\mathrm{IP}_j(\mathrm{DN}) = f^-_j$.)

Example D. We illustrate the deferred decision approach by starting from the situation depicted in Example B. We create Table 5 below by adding an additional row to Table 2 to indicate the x^o_j values of the new goal conditions (corresponding to the new x'_j values, which for the 0-1 case are the complements of the original x'_j values).

We examine two cases, depicted in the two associated tables, Table 6—Case 1 and Table 6—Case 2 below. These two tables may be viewed as two extensions of Table 5, identifying the set of variables G^* by the @ symbol. Thus, the indicated x_j variables are those that are provisionally freed, to yield the new solution values x''_j for these variables shown in the respective tables. (That is, these values are those resulting from the solution of the provisional (LP′) problem. We do not bother to show the corresponding x''_j values for the other variables, because they do not enter into the current decision process.) To determine the penalties for deviating from the x^o_j values we use the L1 measure $\mathrm{IP}^o_j = |x^o_j - x''_j|$. The variables of G^* are then ranked in ascending order of the IP^o_j values to determine how to assign the appropriate decisions.

Table 6—Case 1 selects G^* to consist of the variables x_2, x_3 and x_4. From the preceding Table 6 we see that these variables were assigned two type (1) decisions (indicated by the *'s in the "New x'_j" row of Table 6) and one type (2) decision (indicated by the #'s in the "New x'_j" row of Table 6)—i.e., two of these three variables were assigned to G_P and one was assigned to G_S. Sequencing the elements of G^* in ascending order of the IP^o_j values gives the sequence x_2, x_4, x_3. Hence the two type (1) decisions, which are allocated first, go to x_2 and x_4, and the one type (2) decision goes to x_3, as indicated by the *'s and the # in the "New x'_j" row in Table 6—Case 1.

Table 6—Case 2 selects G^* to consist of the variables x_3, x_4, x_5 and x_6. From Table 6 we see that these variables were assigned one type (1) decision (indicated by *), two type (2) decisions (indicated by #) and

Table 6

$j =$	1	2	3	4	5	6
Case 1						
$x_j^o =$	0	1	1	0	1	0
$G^* =$		@	@	@		
$x_j'' =$		0.8	0.4	0.3		
$\mathrm{IP}_j^o =$		0.2	0.6	0.3		
New $x_j' =$		1*	#	0*		
Case 2						
$x_j^o =$	0	1	1	0	1	0
$G^* =$			@	@	@	@
$x_j'' =$			0.8	0	0	0.3
$\mathrm{IP}_j^o =$			0.2	0	1	0.3
New $x_j' =$			#	0*	0	#

Based on allocating two type (I) decisions (*) and one type (2) decision (#) to the variables of G^*. Based on the allocating one type (1) decisions (*), two type (2) decisions (#) and one type (3) decision (no * or # symbol) to the variables of G^*.

one type (3) decision (indicated by the absence of both * and #). Hence one of these four variables was assigned to G_P, two were assigned to G_S and one was assigned to $G - G_{PS}$. Sequencing the elements of G^* in ascending order of the IP_j^o values gives the order x_4, x_3, x_6, x_5. Hence the type (1) decision goes to x_4, the type (2) decisions go to x_3 and x_6 and the type (3) decision goes to x_5.

B.2. A conservative deferred decision strategy

A variant of the foregoing deferred decision approach does not automatically apply the decisions indicated for the x_j variables in G^*, but instead restricts attention to those x_j that were assigned the same decision in G^* that they were assigned in the original determination of G_P and G_S. These decisions determine the "true" G_P and G_S assignments, and all other goal conditions are left unchanged, as members of $G - G_{PS}$.

Both this approach and the ordinary deferred decision approach are likely to be more effective when the sizes of G_P and G_S are small, as where only one or two variables are assigned a new goal condition and a similar (or smaller) number are permitted to be freed. As in the case of Examples A and B, we note that the starting point for the current analysis, illustrated by Table 4 above, allows a variety of different initial choices of G_P and G_S to be investigated. Thus, for example, the same initial information can be used to investigate the case for selecting just the first two or three variables (or different subsequences of two or three variables) for analysis. Again, such investigation favors the use of "small G" scenarios that are examined in refined search stages.

References

[1] Glover F. A template for scatter search and path relinking. Mathematical Programming 1998;8:161–88.
[2] Glover F, Laguna M. Tabu search. Dordrecht: Kluwer Academic Publishers; 1997.

[3] Gendreau M. An introduction to tabu-search. In: Kochenberger G, Glover F, editors, Handbook of metaheuristics. Dordrecht: Kluwer Academic Publishers; 2003 [chapter 2].
[4] Glover F. Heuristics for integer programming using surrogate constraints. Decision Sciences 1977;8(1):156–66.
[5] Glover F, Laguna M, Marti R. Fundamentals of scatter search and path relinking. Control and Cybernetics 2000;29(3): 653–84.
[6] Tseng F, Glover F. Non-traditional sorting methods. Working paper, University of Colorado, Boulder, 2004.

Further Reading

[7] April J, Glover F, Kelly J, Laguna M. Practical introduction to simulation optimization. In: Chick S, Sanchez F, Ferrin D, Morrice D, editors. Proceedings of the 2003 winter simulation conference, 2003.
[8] Blum C, Roli A. Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Computing Surveys (CSUR) 2003;35(3):268–308.
[9] Crainic TG, Toulouse M. Parallel strategies for meta-heuristics. In: Kochenberger G, Glover F, editors. Handbook of metaheuristics. Dordrecht: Kluwer Academic Publishers; 2003 [chapter 17].
[10] Glover F. Parametric branch and bound. OMEGA. The International Journal of Management Science 1978;6:1–9.
[11] Glover F. Tabu search—Part II. ORSA Journal on Computing 1990;2(1):4–32.
[12] Glover F. Scatter search and star paths: beyond the genetic metaphor. OR Spectrum 1995;17:125–37.
[13] Harder R. Open source tabu-search (OTS), 2004. www.coin-or.org.
[14] Pedroso JP. Tabu search for mixed integer programming. Working paper, University of Porto, Portugal.