Metaheuristic Search with Inequalities and Target Objectives for Mixed Binary Optimization – Part II : Exploiting Reaction and Resistance

Fred Glover

OptTek Systems, Inc., USA

glover@opttek.com

Saïd Hanafi

Univ Lille Nord de France, F-59000 Lille, France

UVHC, LAMIH, F-59313 Valenciennes, France

Said.hanafi@univ-valenciennes.fr

2 March 2010

Abstract: Recent metaheuristics for mixed integer programming have included proposals for introducing inequalities and target objectives to guide the search. These guidance approaches are useful in intensification and diversification strategies related to fixing subsets of variables at particular values, and in strategies that use linear programming to generate trial solutions whose variables are induced to receive integer values. In Part I, we showed how to improve such approaches by new inequalities that dominate those previously proposed and by associated target objectives that underlie the creation of both inequalities and trial solutions. We also introduced procedures for generating target objectives and solutions by exploiting proximity in original space or projected space.

In this Part II of this study, we review the fundamental concepts underlying weighted pseudo cuts for generating guiding inequalities, including the use of target objective strategies. Building on these foundations, we develop more advanced approaches for generating the target objective based on exploiting the mutually reinforcing notions of reaction and resistance. We additionally show how to generate new inequalities by "mining" reference sets of elite solutions to extract characteristics these solutions exhibit in common. Model embedded memory, as proposed in parametric tabu search, is integrated to provide a range of recency and frequency memory structures for achieving goals associated with short term and long term solution strategies. Finally, we propose supplementary linear programming models that exploit the new inequalities for intensification and diversification, and introduce additional inequalities from sets of elite solutions that enlarge the scope of these models.

Key words: Zero-one Mixed Integer Programming, Adaptive Search, Valid Inequalities, Parametric Tabu Search

1. Introduction

We represent the zero-one mixed integer programming problem in the form

$$\text{(P)} \qquad \begin{cases} Minimize & z_0 = fx + gy \\ subject\ to & (x, y) \in Z = \{(x, y): Ax + Dy \geq b\} \\ & x\ integer \end{cases}$$

We assume that $Ax + Dy \geq b$ includes the inequalities $1 \geq x_j \geq 0$, $j \in N = \{1, \ldots, n\}$. The linear programming relaxation of P that results by dropping the integer requirement on x is denoted by LP. We further assume $Ax + Dy \geq b$ includes an objective function constraint $z_o \leq u_o$, where the bound u_o is manipulated as part of a search strategy for solving P, subject to maintaining $u_o < z_o^*$, where z_o^* is the z_o value for the currently best known solution z^* to P.

Recent adaptive memory and evolutionary metaheuristics for mixed integer programming have included proposals for introducing inequalities and target objectives to guide the search. These guidance approaches are useful in intensification and diversification strategies related to fixing subsets of variables at particular values, and in strategies that use linear programming to generate trial solutions whose variables are induced to receive integer values.

In this paper we make reference to two types of search strategies: those that fix subsets of variables to particular values within approaches for exploiting strongly determined and consistent variables, and those that make use of solution *targeting* procedures. Those targeting procedures solve a linear programming problem LP(x', c') where the objective vector c' depends on the target solution x'. LP(x', c') includes the constraints of LP (and additional bounding constraints) while replacing the objective function z_o by a linear function $v_o = c'x$. Given a *target solution* x', the objective vector c' consists of integer coefficients c'_j that seek to induce assignments $x_j = x'_j$ for different variables with varying degrees of emphasis. We adopt the convention that each instance of LP(x', c') implicitly includes the LP objective of minimizing the function $z_o = fx + gy$ as a secondary objective, dominated by the objective of minimizing $v_o = c'x$, so that the true objective function consists of minimizing $\omega_o = Mv_o + z_o$, where M is a large positive number.

A useful alternative to working with ω_o in the form specified is to solve LP(x', c') in two stages. The first stage minimizes $v_o = c'x$ to yield an optimal solution $x = x''$, and the second stage enforces $v_o = c'x''$ to solve the residual problem of minimizing $z_o = fx + gy$. An effective way to enforce $v_o = c'x''$ is to fix all non-basic variables having non-zero reduced costs to compel these variables to receive their optimal first stage values throughout the second stage. This can be implemented by masking the columns for these variables in the optimal first stage basis, and then to continue the second stage from this starting basis while ignoring the masked variables and their columns. The resulting residual problem for the second stage can be significantly smaller than the first stage problem, allowing the problem for the second stage to be solved efficiently.

A second convention involves an interpretation of the problem constraints. Selected instances of inequalities generated by approaches of the following sections will be understood to be included among the constraints $Ax + Dy \geq b$ of (LP). In our definition of LP(x', c') and other linear programs related to (LP), we take the liberty of representing the currently updated form of the constraints $Ax + Dy \geq b$ by the compact representation $x \in X = \{x: (x,y) \in Z\}$, recognizing that this involves a slight distortion in view of the fact that we implicitly minimize a function of y as well as x in these linear programs.[1]

In Part I (Glover and Hanafi (2010)), we proposed procedures for generating target objectives and solutions by exploiting proximity in the original space or projected space. To launch our investigation we first review weighted pseudo cuts for generating guiding inequalities for this problem and associated target objective strategies by exploiting proximity with embedded memory in Section 2. Section 3 indicates more advanced approaches for generating the target objective based on exploiting the mutually reinforcing notions of reaction and resistance. The term "reaction" refers to the change in the value of a variable as a result of creating a target objective and solving the resulting linear programming problem. We show how to generate additional inequalities by "mining" reference sets of elite solutions to extract characteristics these solutions exhibit in common. Section 4 describes models that use embedded memory, as proposed in parametric tabu search (Glover, 1978, (2006)), which offers a range of recency and frequency memory structures for achieving goals associated with short term and long term solution strategies. We examine ways this framework can be exploited in generating target objectives, employing both older adaptive memory ideas and newer ones proposed here for the first time. Section 5 focuses on supplementary linear programming models that exploit the new inequalities for intensification and diversification, and introduce additional inequalities from sets of elite solutions that enlarge the scope of these models. Concluding remarks are given in Section 6.

2. Target Objectives by Exploiting Proximity with Embedded Memory

To develop the basic ideas, let x' denote an arbitrary solution, and define the associated index sets

$N(x', v) = \{j \in N: x_j' = v\}$ for $v \in \{0, 1\}$,

$N(x') = \{j \in N: x_j' \in \{0, 1\}\} = \{j \in N: x_j(1 - x_j) = 0\} = N(x', 0) \cup N(x', 1)$

$N^*(x') = \{j \in N: x_j' \in]0, 1[\} = \{j \in N: x_j(1 - x_j) \neq 0\}$ (hence $N = N(x') \cup N^*(x')$)

$C(x') = \{ c \in IN^n_+ : c_j x'_j(1 - x'_j) = 0\}$.

Let x, x' be two arbitrary binary solutions and let c' be an integer vector in $C(x')$. Define

$$\delta(c', x', x) = \sum_{j \in N} c_j' x_j (1 - x_j') + c_j' x_j' (1 - x_j) \; . \qquad (1)$$

The following result is proved in Part I.

[1] In some problem settings, the inclusion of the secondary objective x_o in a primary objective $v_{oo} = Mv_o + x_o$ is unimportant, and in these cases our notation is accurate in referring to the explicit minimization of $v_o = c'x$.

Proposition 1. Let x' denote an arbitrary target solution with the associated vector $c' \in C(x')$. Let x'' denoted an optimal solution to the following LP problem

LP(x', c'): Minimize $\{\delta(c', x', x): x \in X\}$.

Then the inequality

$$\delta(c', x', x) \geq \max\{1, \lceil \delta(c', x', x'') \rceil^2\} \qquad (2)$$

eliminates all solutions in $F(x', c') = \{x \in [0,1]^n : c'_j(x_j - x_j') = 0 \text{ for } j \in N(x')\}$ as a feasible solution, but admits all other binary x vectors.

We observe that (2) is a valid inequality, i.*e*., it is satisfied by all binary vectors that are feasible for (P) (and more specifically by all such vectors that are feasible for LP(x', c')), with the exception of those ruled out by previous examination. We make use of solutions such as x' by assigning them the role of *target solutions and* by c' assigning them the role of *target objective vectors*.

Remark 1 : The special case of the inequality (2) where $c' = e$ has been used, for example, to produce 0-1 "short hot starts" for branch and bound by Spielberg and Guignard (2000) and Guignard and Spielberg (2003). This special inequality is called a *canonical cut* on the unit hypercube by Balas and Jeroslow (1972). The inequality (2) also generalizes the partial pseudo cuts used by Soyster et al. (1978), Hanafi and Wilbaut (2009) and Wilbaut and Hanafi (2009). The partial pseudo cuts are generated from a subset $J' \subseteq N(x')$, by using the partial distance

$$\delta(J', x', x) = \sum_{j \in J'} x_j (1 - x'_j) + x'_j (1 - x_j) \qquad (3)$$

The distance $\delta(J', x', x) = \delta(c', x', x)$ where $c'_j = 1$ if $j \in J'$ otherwise $c'_j = 0$.

In Part I, we identified a relatively simple approach for generating the vector c' of the target objective by exploiting proximity. The proximity procedure for generating target solutions x' and associated target objectives $\delta(c', x', x)$ begins by solving the initial problem (LP), and then solves a succession of problems LP(x', c') by progressively modifying x' and c'. Beginning from the linear programming solution x'' to (LP) (and subsequently to LP(x', c')), the new target solution x' is derived from x'' simply by setting $x_j' = \langle x_j'' \rangle$, $j \in N$, where $\langle v \rangle$ denotes the nearest integer neighbour of v. (The value $\langle .5 \rangle$ can be either 0 or 1, by employing an arbitrary tie-breaking rule.)

Proximity Procedure

[2] For any real number α, $\lceil \alpha \rceil$ and $\lfloor \alpha \rfloor$ respectively identify the least integer $\geq \alpha$ and the greatest integer $\leq \alpha$.

1. Solve (LP). (If the solution x'' to the first instance of (LP) is integer feasible, the method stops with an optimal solution for (P).)
2. Construct the target solution x' derived from x'' by setting $x'_j = \langle x''_j \rangle$, for $j \in N$. Apply the Rule for Generating c'_j, to each $j \in N$, to produce the vector c'.
3. Solve LP(x', c'), yielding the solution x''. Update the Problem Inequalities.
4. If x'' is integer feasible: update the best solution $(x^*,y^*) = (x'',y'')$, update $u_o < z_o^*$, and return to Step 1. Otherwise, return to Step 2.

The targeting of $x_j = x'_j$ for variables whose values x''_j already equal or almost equal x'_j does not exert a great deal of influence on the solution of the new (updated) LP(x', c'), in the sense that such a targeting does not drive this solution to differ substantially from the solution to the previous LP(x', c'). A more influential targeting occurs by emphasizing the variables x_j whose x''_j values are more "highly fractional," and hence which differ from their integer neighbours x'_j by a greater amount.

The following rule creates a target objective $\delta(c', x', x)$ based on this compromise criterion, arbitrarily choosing a range of 1 to *BaseCost* + 1 for the coefficient c'_j.

Proximity Rule for Generating c'_j:

Choose λ_0 from the range $.1 \leq \lambda_0 \leq .4$. For $j \in N$ do

$c'_j = 1 + BaseCost\ (1\text{-}2x'_j)(.5 - x''_j)/(.5 - \lambda_0)$ if $x''_j \notin]\ \lambda_0, 1 - \lambda_0]$

$c'_j = 1 + BaseCost\ (x'_j - x''_j)/\ \lambda_0$ otherwise

The values of c'_j coefficients produced by the preceding rule describe what may be called a *batwing* function. The following Figure 1 shows the shape of this function.

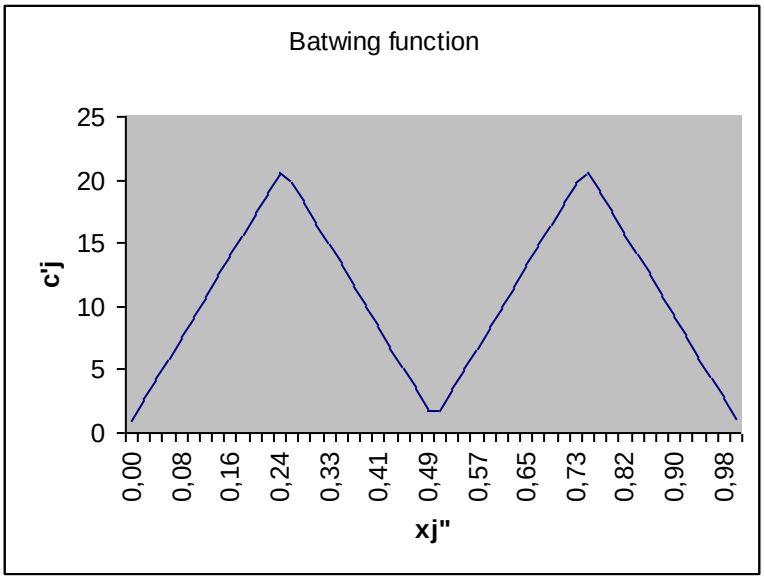

Figure 1 : Batwing function

We may modify the specification of the c'_j values by using model embedded memory, as proposed in parametric tabu search. For this, we replace the constant value *BaseCost* in the c'_j generation rules by a changing *BaseCost* value which is increased on each successive iteration, thus causing the new c'_j values to grow as the number of iterations increases. The influence of these values in driving variables to reach their targets will thus become successively greater, and targets that have been created more recently will be less likely to be violated than those created earlier. (The larger value of c'_j the more likely it will be that x_j will not resist its target value x_j' by becoming fractional.)

Consequently, as the values c'_j grow from one iteration to the next, the variables that were given new targets farther in the past will tend to be the ones that become resistors and candidates to receive new target values. As a result, the c'_j coefficients produced by progressively increasing *BaseCost* emulate a tabu search recency memory that seeks more strongly to prevent assignments from changing the more recently that they have been made.

The determination of the c'_j values can be accomplished by starting with *BaseCost* = 20 in Step 1 of the proximity procedure, and updating the value of *BaseCost* each time *iter* is incremented by 1 in Step 3 to give *BaseCost* := λ*BaseCost*, where the parameter λ is chosen from the interval $\lambda \in [1.1,1.3]$. (This value of λ can be made the same for all iterations, or can be selected randomly from such an interval at each iteration.)

To prevent the c'_j values from becoming excessively large, the current c'_j values can be reduced once *BaseCost* reaches a specified limit by the applying following rule.

1) Reset *BaseCost* = 20 and index the variables x_j, $j \in N(x')$ so that

$c'_1 \geq c'_2 \geq \ldots \geq c'_p$ where $p = |N(x')|$.

2) Define $\Delta_j = c'_j - c'_{j+1}$ for $j = 1, \ldots, p$ -1 and select $\lambda \in [1.1,1.3]$.

3) Set c'_p := *BaseCost* and $c'_j := \text{Min}(c'_{j+1} + \Delta_j, \lambda c'_{j+1})$ for $j = p$ -1, …, 1.

4) Finally, reset *BaseCost* := c'_1 (= Max(c_j', $j \in N(x')$).

Remark 2 : An equivalent implementation is obtained by replacing instruction 3 with

3-a) Set c'_p := *BaseCost* and $\Delta = c'_{p-1} - c'_p$;

3-b) for $j = p$ -1, …, 1 do{ $\Delta' = c'_{j-1} - c'_j$; $c'_j := \text{Min}(c'_{j+1} + \Delta, \lambda c'_{j+1})$; $\Delta = \Delta'$;}

The new c_j values produced by this rule will retain the same ordering as the original ones.

In a departure for diversification purposes, the foregoing rule can be changed by modifying the next to last step to become

Set $|c'_1|$:= *BaseCost* and $|c'_{j+1}| := \text{Min}(|c'_j| + \Delta_j, \lambda|c'_j|)$ for $j = 1, \ldots, p - 1$

and conclude by resetting *BaseCost* := $|c'_p|$.

3. Generating Target Objectives and Solutions by Exploiting Reaction and Resistance

In this section, we propose a more advanced approach for generating the vector c' of the target objective. This approach is based on exploiting the mutually reinforcing notions of *reaction* and *resistance*. The term "reaction" refers to the change in the value of a variable as a result of creating a target objective $\delta(c', x', x)$ and solving the resulting problem LP(x', c'). The term "resistance" refers to the degree to which a variable fails to react to a non-zero c_j coefficient by receiving a fractional value rather than being driven to 0 or 1.

Relative to a given vector x' and a target vector c' we consider the partition of N into the sets

$$N(c', x') = \{j \in N: c'_j \neq 0 \text{ and } x_j(1 - x_j) = 0\} \text{ and}$$

$$N^*(c', x') = \{j \in N: c'_j = 0 \text{ or } x_j(1 - x_j) \neq 0\}.$$

i.*e*. $N = N(c', x') \cup N^*(c', x')$. Note that the set $N^*(c', x')$ contains the indexes of variables that are not subject to a binding constraint or incorporated into a target affecting their values. The set $N(c', x')$ identifies the variables that have been assigned target values x_j'. (Equivalently, $N(c', x') = N - N^*(c', x')$.)

Remark 3 : $N^*(e, x') = N^*(x') = \{j \in N: 0 < x'_j < 1\}$ is the set of variables that receive fractional values in the solution x'. (Similarly, we note $N(e, x') = N(x')$.)

Corresponding to the partition of N into the sets $N^*(c', x')$ and $N(c', x')$, the set $N^*(x'')$ of fractional variables is partitioned into the sets

$$N^*(x'', c', x') = N^*(x'') \cap N(c', x') \text{ and}$$

$$N^{**}(x'', c', x') = N^*(x'') \cap N^*(c', x').$$

We identify two different sets of circumstances that are relevant to defining reaction, the first arising where none of the fractional variables x_j is assigned a target x'_j, hence $N^*(x'') = N^{**}(x'', c', x')$ (i.*e*. $N^*(x'') \subseteq N^*(c', x')$), and the second arising in the complementary case where at least one fractional variable is assigned a target, hence $N^*(x'', c', x') \neq \varnothing$. We start by examining the meaning of reaction in the somewhat simpler first case.

3.1. Reaction When No Fractional Variables Have Targets

Our initial goal is to create a measure of reaction for the situation where $N^*(x'') = N^{**}(x'', c', x')$, i.*e*., where all of the fractional variables are unassigned (hence, none of these variables have targets). In this context we define reaction to be measured by the change in the value x''_j of a fractional variable x_j relative to the value x^o_j received by x_j in an optimal solution x^o to (LP), as given by

$$\Delta_j = x^o_j - x''_j.$$

We observe there is some ambiguity in this Δ_j definition since (LP) changes as a result of introducing new inequalities and updating the value u_o of the inequality $z_o \leq u_o$. To remove this ambiguity, we understand the definition of Δ_j to refer to the solution x^o obtained by the most recent effort to solve (LP), though this (LP) may be to some extent out of date, since additional inequalities may have been introduced since it was solved. For reasons that will become clear in the context of resistance, we also allow the alternative of designating x^o to be the solution to the most recent problem LP(x', c') preceding the current one; i.*e.*, the problem solved before creating the latest target vector c'.

The reaction measure Δ_j is used to determine the new target objective by re-indexing the variables x_j, $j \in N^*(x'') = N^{**}(x'', c', x')$, so that the absolute values $|\Delta_j|$ are in descending order, thus yielding $|\Delta_1| \geq |\Delta_2| \geq \ldots$. We then identify the k-element subset $N(k) = \{1,2, \ldots, k\}$ of $N^*(x'')$ that references the k largest $|\Delta_j|$ values, where $k = \text{Min}(|N^*(x'')|, k_{max})$. We suggest the parameter k_{max} be chosen at most 5 and gradually decreased to 1 as the method progresses.

The c_j coefficients are then determined for the variables x_j, $j \in N(k)$, by the following rule. (The constant *BaseCost* is the same one used to generate c'_j values in the Proximity procedure, and $\langle v \rangle$ again denotes the nearest integer neighbor of v.)

$N^{**}(x'', c', x')$ Rule for Generating c'_j and x'_j, $j \in N(k)$ (for $N(k) \subset N^*(x'') = N^{**}(x'', c', x')$):

Set $c'_j = 1 + \langle BaseCost\ |\Delta_j| / |\Delta_1| \rangle$ and $x'_j = \text{sign}(\Delta_j)$.

When $\Delta_j = 0$, a tie-breaking rule can be used to determine which of the two options should apply, and in the special case where $\Delta_1 = 0$ (hence all $\Delta_j = 0$), the c'_j assignment is taken to be 1 for all $j \in N(k)$.

To determine a measure of reaction for the complementary case $N^*(x'', c', x') \neq \varnothing$, we first introduce the notion of resistance.

3.2. Resistance

A *resisting variable* (or *resistor*) x_j is one that is assigned a target value x'_j but fails to satisfy $x_j = x'_j$ in the solution x'' to LP(x', c'). Accordingly the index set for resisting variables may be represented by

$$NR = \{j \in N(c', x'):\ x''_j \neq x'_j\}$$

$$NR = \{j \in N:\ c'_j\,(x''_j - x'_j) \neq 0 \text{ and } x'_j\,(1 - x'_j) = 0\}$$

$$NR = \{j \in N:\ c'_j \delta(j, x', x'') \neq 0\}$$

If x_j'' is fractional and $j \in N(c', x')$ then clearly $j \in NR$ (i.*e.*, $N^*(x'', c', x') \subset NR$). Consequently, the situation $N^*(x'', c', x') \neq \varnothing$ previously identified as complementary to $N^*(x'') = N^{**}(x'', c', x')$ corresponds to the presence of at least one fractional resistor.

If a resistor x_j is not fractional, i.*e.*, if the value x''_j is the integer $1 - x'_j$, we say that x_j *blatantly resists* its targeted value x'_j. Blatant resistors x_j are automatically removed from *NR* and placed in the unassigned set $N^*(c', x')$ by setting $c'_j = 0$. (Alternatively, a blatant resistor may be placed in $N(x', 1 - x'_j)$ by setting $x'_j := 1 - x'_j$.)

After executing this operation, we are left with $NR = N^*(x'', c', x')$, and hence the condition $N^*(x'', c', x') \neq \varnothing$ (which complements the condition $N^*(x'') = N^{**}(x'', c', x')$) becomes equivalent to $NR \neq \varnothing$.

We use the quantity $\delta(j, x', x'')$ to define a *resistance measure* RM_j for each resisting variable x_j, $j \in NR$, that identifies how strongly x_j resists its targeted value x_j'. Two simple measures are given by

$$RM_j = \delta(j, x', x'') \text{ or } RM_j = c'_j\delta(j, x', x'') \qquad \text{for } j \in NR.$$

The resistance measure RM_j is used in two ways: (a) to select specific variables x_j that will receive new x_j' and c'_j values in creating the next target objective; (b) to determine the relative magnitudes of the resulting c'_j values. For this purpose, it is necessary to extend the notion of resistance by making reference to *potentially resisting* variables (or *potential resistors*) x_j, $j \in N(c', x') - NR$, i.*e.*, the variables that have been assigned target values x'_j and hence non-zero objective function coefficients c'_j, but which yield $x''_j = x'_j$ in the solution x'' to LP(x', c'). We identify a resistance measure RM_j^o for potential resistors by reference to their reduced cost values rc_j (as identified in Section 4):

$$RM_j^o = (2x'_j - 1)\, rc_j \quad \text{for } j \in N(c', x') - NR$$

We note that this definition implies $RM_j^o \leq 0$ for potentially resisting variables. (Otherwise, x_j would be a non-basic variable yielding $x''_j = 1$ in the case where $j \in N(x',0)$, or yielding $x''_j = 0$ in the case where $j \in N(x', 1)$, thus qualifying as a blatant resistor and hence implying $j \in NR$.) The closer that RM_j^o is to 0, the closer x_j is to qualifying to enter the basis and potentially to escape the influence of the coefficient c'_j that seeks to drive it to the value 0 or 1. Thus larger values of RM_j^o indicate greater potential resistance. Since the resistance measures RM_j are positive for resisting variables x_j, we see that there is an automatic ordering whereby $RM_p > RM_q^o$ for a resisting variable x_p and a potentially resisting variable x_q.

3.3. Combining Measures of Resistance and Reaction

The notion of reaction is relevant for variables x_j assigned target values x'_j ($j \in N(c', x')$) as well as for those not assigned such values ($j \in N^*(c', x')$). In the case of variables having explicit targets (hence that qualify either as resistors or potential resistors) we combine

measures of resistance and reaction to determine which of these variables should receive new targets x'_j and new coefficients c'_j.

Let x^o refer to the solution x'' to the instance of the problem LP(x', c') that was solved immediately before the current instance;[3] hence the difference between x^o_j and x''_j identifies the reaction of x_j to the most recent assignment of c'_j values. In particular, we define this reaction for resistors and potential resistors by

$$\delta_j = (1 - x'_j)(x''_j - x^o_j) + x'_j\,(x^o_j - x''_j) \;\; (j \in N(c', x'))$$

$$\delta_j = \delta(j, x', x'') - \delta(j, x', x^o)$$

If we use the measure of resistance $RM_j = \delta(j, x', x'')$, which identifies how far x_j lies from its target value, a positive δ_j implies that the resistance of x_j has increased as a result of the latest assignment of c'_j values, and a negative δ_j implies that the resistance of x_j has decreased as a result of this assignment. Just as the resistance measure RM_j is defined to be either $\delta(j, x', x'')$ or $c'_j\delta(j, x', x'')$, the corresponding reaction measure $R\delta_j$ can be defined by either

$$R\delta_j = \delta_j \text{ or } R\delta_j = \delta_j\, c'_j.$$

$$R\delta_j = \delta(j, x', x'') - \delta(j, x', x^o) \text{ or } R\delta_j = c'_j\delta(j, x', x'') - \delta(j, x', x^o).$$

Based on this we define a composite resistance-reaction measure RR_j for resisting variables as a convex combination of RM_j and $R\delta_j$; i.*e*., for a chosen value of $\lambda \in [0,1]$:

$$RR_j = \lambda RM_j + (1 - \lambda)R\delta_j, \;\; j \in NR.$$

$$RR_j = \lambda\delta(j, x', x'') + (1 - \lambda)(\delta(j, x', x'') - \delta(j, x', x^o)), \;\; j \in NR.$$

$$RR_j = \delta(j, x', x'') + (\lambda - 1)\delta(j, x', x^o), \;\; j \in NR.$$

Similarly, for implicitly resisting variables, we define a corresponding composite measure RR_j^o by

$$RR_j^o = \lambda RM_j^o + (1 - \lambda)R\delta_j, \;\; j \in N(c', x') - NR$$

In order to make the interpretation of λ more consistent, it is appropriate first to scale the values of RM_j, RM_j^o and $R\delta_j$. If v_j takes the role of each of these three values in turn, then v_j may be replaced by the scaled value $v_j := v_j/|Mean(v_j)|$ (bypassing the scaling in the situation where $|Mean(v_j)| = 0$).

To give an effective rule for determining RR_j and RR_j^o, a few simple tests can be performed to determine a working value for λ, as by limiting λ to a small number of default values (*e*.g., the three values 0, 1 and .5, or the five values that include .25 and .75).

3.4. Including Reference to a Tabu List

[3] This is the "alternative definition" of x^o indicated earlier.

A key feature in using both RR_j and RR_j^o to determine new target objectives is to make use of a simple tabu list T to avoid cycling and insure a useful degree of variation in the process. We specify in the next section a procedure for creating and updating T, which we treat both as an ordered list and as a set. (We sometimes speak of a variable x_j as belonging to T, with the evident interpretation that $j \in T$.) It suffices at present to stipulate that we always refer to non-tabu elements of $N(c', x')$, and hence we restrict attention to values RR_j and RR_j^o for which $j \in N(c', x') - T$. The rules for generating new target objectives make use of these values in the following manner.

Because RR_j and RR_j^o in general are not assured to be either positive or negative, we treat their ordering for the purpose of generating c'_j coefficients as a rank ordering. We want each RR_j value (for a resistor) to be assigned a higher rank than that assigned to any RR_j^o value (for a potential resistor). An easy way to do this is to define a value RR_j for each potential resistor given by

$$RR_j = RR^o_j - RR_1^o + 1 - Min(RR_j: j \in NR), \quad j \in N(c', x') - NR.$$

$$RR_j = RR^o_j - Max(RR^o_j: j \in N(c', x') - NR) - 1 + Min(RR_j: j \in NR), \quad j \in N(c', x') - NR.$$

The set of RR_j values over $j \in N(c', x')$ then satisfies the desired ordering for both resistors ($j \in NR$) and potential resistors ($j \in N(c', x') - NR$). (Recall that $NR = N^*(x'', c', x')$ by having previously disposed of blatant resistors.)

For the subset $N(k)$ of k non-tabu elements of $N(c', x')$ (hence of $N(c', x') - T$) that we seek to generate, the ordering over the subset $NR - T$ thus comes ahead of the ordering over the subset $(N(c', x') - NR) - T$, This allows both resistors and potential resistors to be included among those elements to be assigned new coefficients c'_j and new target values x'_j, where the new c'_j coefficients for resistors always have larger absolute values than the c'_j coefficients for potential resistors. If the set of non-tabu resistors $NR - T$ already contains at least k elements, then no potential resistors will be assigned new c'_j or x'_j values.

3.5. Overview of the Resistance & Reaction Procedure

The rule for generating the target objective $\delta(c', x', x)$ that lies at the heart of the Resistance & Reaction procedure is based on carrying out the following preliminary steps, where the value k_{max} is determined as previously indicated: (a) re-index the variables x_j, $j \in N(c', x') - T$, so that the values RR_j are in descending order, thus yielding $RR_1 \geq RR_2 \geq \ldots$; (b) identify the subset $N(k) = \{1,2, \ldots, k\}$ of NR that references the k largest RR_j values, where $k = Min(|N(c', x') - T|, k_{max})$; (*c*) create a rank ordering by letting R_p, $p = 1, \ldots, r$ denote the distinct values among the RR_j, $j \in N(k)$, where $R_1 > R_2 > \ldots > R_r$ ($r \geq 1$).

Then the rule to determine the c'_j and x'_j values for the variables x_j, $j \in N(k)$, is given as follows:

$N(c', x') - T$ Rule for Generating c'_j and x_j', $j \in N(k)$ (for $NR = N^*(x'', c', x') \neq \varnothing$):

If $RR_j = R_p$, set $c'_j = \langle 1 + \ BaseCost(r + 1 - p)/r\rangle$ and $x'_j = 1 - x'_j$.

We see that this rule assigns c'_j coefficients so that the c'_j values are the positive integers $\langle 1 + BaseCost(1/r)\rangle$, $\langle 1 + BaseCost(2/r)\rangle$,… , $\langle 1 + BaseCost(r/r)\rangle = 1 + BaseCost$.

We are now ready to specify the Resistance & Reaction procedure in overview, which incorporates its main elements except for the creation and updating of the tabu list T.

Resistance & Reaction Procedure in Overview

0. Initialize the current problem P to be the (MIP:0-1) problem.
1. Solve LP(P) yielding the optimal solution x^o. (Stop if the first instance of (LP) yields an integer feasible solution x^o, which therefore is optimal for (MIP:0-1).)
2. Construct the target solution x' derived from x^o by setting $x'_j = \langle x^o_j\rangle$, for $j \in N$. Apply the Rule of the Proximity Procedure for Generating c'_j (to each j in N) to produce the vector c'.
3. Solve LP(x', c'), yielding the solution x''. Set $x^{oo} = x''$. Update the current problem by adding the inequality derived from c' (i.*e*. $P = P \mid \delta(c', x', x) \geq \lceil v(\text{LP}(x', c'))\rceil + 1$).
4. Let $N^*(x'') = \{j \in N:\ 0 < x_j'' < 1\}$, $N^*(c', x') = \{j \in N: c'_j = 0 \text{ or } 0 < x_j' < 1\}$, $N(c', x') = N - N^*(c', x')$, $N^{**}(x'', c', x') = N^*(x'') \cap N^*(c', x')$ and $N^*(x'', c', x') = N^*(x'') \cap N(c', x')$, $NR = \{j \in N:\ c'_j \delta(j, x', x'') \neq 0\}$.
5. There exists at least one fractional variable ($N^*(x'') \neq \varnothing$). Remove blatant resistors x_j, if any exist, from NR and transfer them to $N^*(c', x')$ (or to $N(x', 1 - x'_j)$) so $NR = N^*(x'', c', x')$.
   (a) If $N^*(x'') = N^{**}(x'', c', x')$ (hence $NR = \varnothing$), apply the $N^{**}(x'', c', x')$ Rule for Generating c'_j and x'_j, $j \in N(k)$ to produce the new target objective $\delta(c', x', x)$ and associated target vector x' :

   Set $c'_j = 1 + \langle BaseCost|\Delta_j| / |\Delta_1|\rangle$ and $x'_j = \text{sign}(\Delta_j)$, where $\Delta_j = x^o_j - x''_j$.

   (b) If instead $NR \neq \varnothing$, then apply the $N(c', x') - T$ Rule for Generating c'_j and x'_j, $j \in N(k)$, to produce the new target objective $\delta(c', x', x)$ and associated target vector x':

   If $RR_j = R_p$, set $c'_j = \langle 1 + \ BaseCost(r + 1 - p)/r\rangle$ and $x'_j = 1 - x'_j$.

6. Set $x^{oo} = x''$ and solve LP(x', c'), yielding the solution x''. Update the current problem P by adding the inequality derived from c' (i.*e*. $P = P \mid \delta(c', x', x) \geq \lceil v(\text{LP}(x', c'))\rceil + 1$).
7. If x'' is integer feasible: update the best solution $(x^*, y^*) = (x'', y'')$, update the current problem P by adding $u_o < z_o^*$, (i.*e*. $P = P \mid u_o < z_o^*$) and return to Step 1. Otherwise, return to Step 4.

This design is completed by the processes described in the next section.

4. Creating and Managing the Tabu List T – Resistance & Reaction Procedure Completed

We propose an approach for creating the tabu list T that is relatively simple but offers useful features within the present context. As in a variety of constructions for handling a recency-based tabu memory, we update T by adding a new element j to the first position of the list when a variable x_j becomes tabu (as a result of assigning it a new target value x'_j and coefficient c'_j), and by dropping the "oldest" element that lies in the last position of T when its tabu status expires.

Our present construction employs a rule that may add and drop more than one element from T at the same time. The checking of tabu status is facilitated by using a vector $Tabu(j)$ that is updated by setting $Tabu(j) = true$ when j is added to T and by setting $Tabu(j) = false$ when j is dropped from T. (Tabu status is often monitored by using a vector $TabuEnd(j)$ that identifies the last iteration that element j qualifies as tabu, without bothering to explicitly store the list T, but the current method of creating and removing tabu status makes the indicated handling of T preferable.)

We first describe the method for the case where $k = 1$, i.*e*., only a single variable x_j is assigned a new target value (and thereby becomes tabu) on a given iteration. The modification for handling the case $k > 1$ is straightforward, as subsequently indicated. Two parameters T_{min} and T_{max} govern the generation of T, where $T_{max} > T_{min} \geq 1$. For simplicity we suggest the default values $T_{min} = 2$ and $T_{max} = n^{.6}$. (In general, appropriate values are anticipated to result by selecting T_{min} from the interval between 1 and 3 and T_{max} from the interval between $n^{.5}$ and $n^{.7}$.) The small value of T_{min} accords with an intensification focus, and larger values may be selected for diversification.

The target value x'_j and coefficient c'_j do not automatically change when j is dropped from T and x_j becomes non-tabu. Consequently, we employ one other parameter AssignSpan that limits the duration that x_j may be assigned the same x'_j and c'_j values, after which x_j is released from the restrictions induced by this assignment. To make use of *AssignSpan*, we keep track of when x_j most recently was added to T by setting $TabuAdd(j) = iter$, where *iter* denotes the current iteration value (in this case, the iteration when the addition occurred). Then, when $TabuAdd(j) + AssignSpan < iter$, x_j is released from the influence of x'_j and c'_j by removing j from the set $N(c', x')$ and adding it to the unassigned set $N^*(c', x')$. As long as x_j is actively being assigned new x'_j and c'_j values, $TabuAdd(j)$ is repeatedly being assigned new values of iter, and hence the transfer of j to $N^*(c', x')$ is postponed. We suggest a default value for *AssignSpan* between $1.5T_{max}$ and $3T_{max}$; *e*.g., $AssignSpan = 2T_{max}$.

To manage the updating of T itself, we maintain an array denoted $TabuRefresh(j)$ that is initialized by setting $TabuRefresh(j) = 0$ for all $j \in N$. Then on any iteration when j is added to T, $TabuRefresh(j)$ is checked to see if $TabuRefresh(j) < iter$ (which automatically holds the first time j is added to T). When the condition is satisfied, a *refreshing operation* is performed, after adding j to the front of T, that consists of two steps: (a) the list T is reduced in size to yield $|T| = T_{min}$ (more precisely, $|T| \leq T_{min}$) by dropping all but the T_{min} first elements of T; (b) $TabuRefresh(j)$ is updated by setting $TabuRefresh(j) = iter + v$, where v is a number randomly

chosen from the interval [*AssignSpan*, 2*AssignSpan*]. These operations assure that future steps of adding this particular element j to T will not again shrink T to contain T_{min} elements until *iter* reaches a value that exceeds *TabuRefresh*(j). Barring the occurrence of such a refreshing operation, T is allowed to grow without dropping any of its elements until it reaches a size of T_{max}. Once $|T| = T_{max}$, the oldest j is removed from the end of T each time a new element j is added to the front of T, and hence T is stabilized at the size T_{max} until a new refreshing operation occurs.

This approach for updating T is motivated by the following observation. The first time j is added to T (when *TabuRefresh*(j) = 0) T may acceptably be reduced in size to contain not just T_{min} elements, but in fact to contain only 1 element, and no matter what element is added on the next iteration the composition of $N(c', x')$ cannot duplicate any previous composition. Moreover, following such a step, the composition of $N(c', x')$ will likewise not be duplicated as long as T continues to grow without dropping any elements. Thus, by relying on intervening refreshing operations with *TabuRefresh*(j) = 0 and $T_{min} = 1$, we could conceivably allow T to grow even until reaching a size $T_{max} = n$. (Typically, a considerable number of iterations would pass before reaching such a state.) In general, however, by allowing T to reach a size $T_{max} = n$ the restrictiveness of preventing targets from being reassigned for T_{max} iterations would be too severe. Consequently we employ two mechanisms to avoid such an overly restrictive state: (i) choosing $T_{max} < n$ and (ii) performing a refreshing operation that allows each j to shrink T more than once (whenever *iter* grows to exceed the updated value of *TabuRefresh*(j)) The combination of these two mechanisms provides a flexible tabu list that is self-calibrating in the sense of automatically adjusting its size in response to varying patterns of assigning target values to elements.

The addition of multiple elements to the front of T follows essentially the same design, subject to the restriction of adding only up to T_{min} new indexes j of $N(k)$ to T on any iteration, should k be greater than T_{min}. We slightly extend the earlier suggestion $T_{min} = 2$ to propose $T_{min} = 3$ for $k_{max} \geq 3$.

Note that the organization of the method assures $T \subset N(c', x')$ and typically a good portion of $N(c', x')$ lies outside T. If exceptional circumstances result in $T = N(c', x')$, the method drops the last element of T so that $N(c', x')$ contains at least one non-tabu element.

Drawing on these observations, the detailed form of the Resistance & Reaction_Procedure that includes instructions for managing the tabu list is specified below, employing the stopping criterion indicated earlier of limiting the computation to a specified maximum number of iterations. (These iterations differ from those counted by *iter*, which is re-set to 0 each time a new solution is found and the method returns to solve the updated (LP),)

Complete Resistance & Reaction Procedure.

0. Choose the values T_{min}, T_{max} and *AssignSpan*.

1. Solve (LP). (Stop if the first instance of (LP) yields an integer feasible solution x'', which therefore is optimal for (MIP:0-1).) Set *TabuRefresh*(j) = 0 for all $j \in N$ and set *iter* = 0.

2 There exists at least one fractional variable ($N^*(x'') \neq \varnothing$). Remove each blatant resistor x_j, if any exists, from *NR* and transfer it to $N^*(c', x')$ (or to $N'(1 - x'_j)$), yielding $NR = N^*(x'', c', x')$. If j is transferred to $N^*(c', x')$ and $j \in T$, drop j from T. Also, if $T = N(c', x')$, then drop the last element from T.

   (a) If $N^*(x'') = N^{**}(x'', c', x')$ (hence $NR = \varnothing$), apply the $N^{**}(x'', c', x')$ Rule for Generating c'_j and x'_j, $j \in N(k)$.

   (b) If instead $NR \neq \varnothing$, then apply the $N(c', x') - T$ Rule for Generating c'_j and x'_j, $j \in N(k)$.

   (*c*) Set *iter* : = *iter* + 1. Using the indexing that produces $N(k)$ in (a) or (b), add the elements $j = 1, 2, \ldots min(T_{min}, k)$ to the front of T (so that $T = (1, 2, \ldots)$ after the addition). If *TabuRefresh*(j) < *iter* for any added element j, set *TabuRefresh*(j) = *iter* + v, for v randomly chosen between *AsignLength* and 2*AssignSpan* (for each such j) and then reduce T to at most T_{min} elements by dropping all elements in positions > T_{min}.

3. Solve LP(x', c'), yielding the solution x''. Update the Problem Inequalities.

4. If x'' is integer feasible: update the best solution $(x^*, y^*) = (x'', y'')$, update $u_o < z_o^*$, and return to Step 1. Otherwise, return to Step 2.

The inequalities introduced in Sections 3 and 4 provide a useful component of this method, but the method is organized to operate even in the absence of such inequalities. The intensification and diversification strategies proposed in Section 5 can be incorporated for solving more difficult problems.

5. Intensification and Diversification Based on Strategic Inequalities

More generally, for any positive integer k satisfying $n \geq k \geq 1$, the binary vectors x that lie at least a Hamming distance k from x' are precisely those that satisfy the inequality

$$\delta(e, x', x) \geq k \qquad (4)$$

The inequality (4) has been introduced within the context of adaptive memory search strategies (Glover, 2005) to compel new solutions x to be separated from a given solution x' by a desired distance. In particular, upon identifying a reference set $R = \{x^r, r \in R_I\}$, which consists of elite and diverse solutions generated during prior search, the approach consists of

launching a diversification strategy that requires new solutions x to satisfy the associated set of inequalities

$$\delta(e, x^r, x) \geq k^r, \qquad r \in R_I \qquad (5)$$

This system also gives a mechanism for implementing a proposal of Shylo (1999)[4] to separate new binary solutions by a minimum specified Hamming distance from a set of solutions previously encountered.

The inequalities of (5) constitute a form of *model embedded memory* for adaptive memory search methods where they are introduced for two purposes: (a) to generate new starting solutions and (b) to restrict a search process to visiting solutions that remain at specified distances from previous solutions. A diversification phase that employs the strategy (b) operates by eventually reducing the $e_o{}^r$ values to 1, in order to transition from diversification to intensification. One approach for doing this is to use tabu penalties to discourage moves that lead to solutions violating (5). We discuss another approach in the next section.

A more limiting variant of (5) arises in the context of exploiting strongly determined and consistent variables, and in associated adaptive memory *projection* strategies that iteratively select various subsets of variable to hold fixed at specific values, or to be constrained to lie within specific bounds (Glover, 2005). This variant occurs by identifying sub-sets J^{r1}, J^{r2}, … , of N for the solutions x^r to produce the inequalities

$$\delta(J^{rh}, x^r, x) \geq k^{rh}, \qquad r \in R_I, h = 1, 2 \ldots \qquad (6)$$

The inequalities of (6) are evidently more restrictive than those of (5), if the sum of the values k^{rh} over h is strictly greater than the values k^r (i.*e*., if $\sum_h k^{rh} > k^r$ for each r).

The inequalities (6) find application within two main contexts. The first occurs within a diversification segment of alternating intensification and diversification phases, where each intensification phase holds certain variables fixed and the ensuing diversification divides the index of variables N of each x^r into two sub-sets J^{r1} and J^{r2} that respectively contain the components of x^r held fixed (i.*e*. this arise by setting $k^{r1} = 0$) and the components permitted to be free during the preceding intensification phase. For example, the heuristic based on the LP-relaxation considers x^r which is an optimal solution of LP-relaxtion of a current problem and divides the set N into two sub-sets J^{r1} and J^{r2} that respectively contain the components of x^r held basis variables and the components no basis. Then a reduced problem is generated where the no basic variables are fixed to their values in x^r (i.*e*. this arise by setting $k^{r2} = 0$) and the components in J^{r1} permitted to be free. The heuristic solves approximatily or exactly the corresponding reduced problem (see Soyster et al. (1978) and Hanafi and Wilbaut (2009)). Another example is the Local Branching proposed by Fischetti and Lodi (2003), at each iteration r, the set J^{r1} corresponds to the set of variables over which branching has already been occured (those variables held fixed i.*e*. $k^{r1} = 0$) and J^{r2} contains the components of x^r permitted to be separated from a given solution x' by a desired distance k^{r2}. In Relaxation Induced Neighborhood Search for solving the MIP proposed by Danna et al. (2005), they use

[4] See also Pardalos and Shylo (2006) and Ursulenko (2006).

$J^{r1} = N((\bar{x} + x^*)/2)$ and $J^{r2} = N - J^{r1}$ where x^* is the current incumbent feasible solution and $\bar{x}$ is an optimal solution of the LP-relaxation.

The second area of application occurs in conjunction with frequency memory by choosing three sub-sets J^{r1}, J^{r2} and J^{r3} (for example) to consist of components of solution x^r that have received particular values with high, middle and low frequencies, relative to a specified set of previously visited solutions. (The same frequency vector, and hence the same way of sub-dividing the x^r vectors, may be relevant for all x^r solutions generated during a given phase of search.)[5]

Our following ideas can be implemented to enhance these adaptive memory projection strategies as well as the other strategies previously described.

5.1 An Intensification Procedure

Consider an indexed collection of inequalities of the form of (2) given by

$$\delta(c^p, x', x) \geq \lceil \delta(c^p, x', x'') \rceil, \quad p \in P \qquad (7)$$

We introduce an intensification procedure that makes use of (7) by basing the inequalities indexed by P on a collection of high quality binary target solutions x'. Such solutions can be obtained from past search history or from approaches for rounding an optimal solution to a linear programming relaxation (LP) of (P), using penalties to account for infeasibility in ranking the quality of such solutions. The solutions x' do not have to be feasible to be used as target solutions or to generate inequalities. In Section 6 we give specific approaches for creating such target solutions and the associated target objectives $\delta(c', x', x)$ that serve as a foundation for producing the underlying inequalities.

Our goal from an intensification perspective is to find a new solution that is close to those in the collection of high quality solutions that give rise to (7). We introduce slack variables s_p, $p \in P$, to permit the system (7) to be expressed equivalently as

$$\delta(c^p, x', x) - s_p = \lceil \delta(c^p, x', x'') \rceil, s_p \geq 0, \qquad p \in P \qquad (8)$$

Then, assuming the set X includes reference to the constraints (8), we create an *Intensified LP Relaxation*

$$\text{Minimize } (s_o = \textstyle\sum_{p \in P} w_p s_p : x \in X)$$

where the weights w_p for the variables s_p are selected to be positive integers.

An important variation is to seek a solution that minimizes the maximum deviation of x from solutions giving rise to (7). This can be accomplished by introducing the inequalities

$$s_o \geq \lceil \delta(c^p, x', x'') \rceil - \delta(c^p, x', x), \qquad p \in P \qquad (9)$$

[5] The formulas of Glover (2005) apply more generally to arbitrary integer solution vectors.

Assuming these inequalities are likewise incorporated into X,[6] the Min(Max) goal is achieved by solving the problem

$$\text{Minimize } (s_o: x \in X)$$

An optimal solution to either of these two indicated objectives can then be used as a starting point for an intensified solution pass, performing all-at-once or successive rounding to replace its fractional components by integers.[7]

5.2 A Diversification Analog

To create a diversification procedure for generating new starting solutions, we seek an objective function to drive the search to lie as far as possible from solutions in the region defined by (7). For this purpose we introduce the variables s_p as in (8), but utilize a maximization objective rather than a minimization objective to produce the problem

$$\text{Maximize } (s_o = \sum_{p \in P} w_p s_p : x \in X)$$

The weights w_p are once again chosen to be positive.

A principal alternative in this case consists of maximizing the minimum deviation of x from solutions giving rise to (7). For this, we additionally include the inequalities

$$s_o \leq \lceil \delta(c^p, x', x'') \rceil - \delta(c^p, x', x), \qquad p \in P \qquad (10)$$

giving rise to the problem

$$\text{Maximize } (s_o: x \in X).$$

The variable s_o introduced in (10) differs from its counterpart in (9). In the case where the degree of diversification provided by this approach is excessive, by driving solutions too far away from solutions expected to be good, control can be exerted through bounding X with other constraints, and in particular by manipulating the bound z_o identified in Section 1.

6. Conclusions

Branch-and-bound (*B*&*B*) and branch-and-cut (*B*&*C*) methods have long been considered the methods of choice for solving mixed integer programming problems. In recent years, efforts to create improved *B*&*B* and *B*&*C* solution approaches have intensified and have produced significant benefits, as evidenced by the existence of MIP procedures that are appreciably more effective than their predecessors. It remains true, however, that many MIP problems resist solution by the best current *B*&*B* and *B*&*C* methods. As a consequence, metaheuristic methods have attracted attention as possible alternatives or supplements to the more classical

[6] The inclusion of (8) and (9) is solely for the purpose of solving the associated linear programs, and these temporarily accessed constraints do not have to be incorporated among those defining *Z*.

[7] Successive rounding normally updates the LP solution after rounding each variable in order to determine the effects on other variables and thereby take advantage of modified rounding options.

approaches. Yet to date, the amount of effort devoted to developing good metaheuristics for MIP problems is almost negligible compared to the effort being devoted to developing refined versions of the classical methods.

The view adopted in Part I and II is that metaheuristic approaches can benefit from a change of perspective in order to perform at their best in the MIP setting. Drawing on lessons learned from applying classical methods, we anticipate that metaheuristics can likewise profit from generating inequalities to supplement their basic functions. However, we propose that these inequalities be used in ways not employed in classical MIP methods, and indicate two principal avenues for doing this: first by generating the inequalities in reference to strategically created target solutions and target objectives, as in Part I, and second by embedding these inequalities in special intensification and diversification processes, as described in this Part II. The use of such strategies raises the issue of how to compose the target solutions and objectives themselves. Classical MIP methods such as *B*&*B* and *B*&*C* again provide a clue to be heeded, by demonstrating that memory is relevant to effective solution procedures. However, we suggest that gains can be made by going beyond the rigidly structured memory employed in *B*&*B* and *B*&*C* procedures. Thus we make use of the type of adaptive memory framework introduced in tabu search, which offers a range of recency and frequency memory structures for achieving goals associated with short term and long term solution strategies.

References

Balas, *E*. and R. Jeroslow (1972) "Canonical Cuts on the Unit Hypercube," *SIAM Journal of Applied Mathematics, 23*, No. 1, pp. 60-69.

Blum, *C*. and A Roli (2003) "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison," to appear in the *ACM Journal*. Vol. 35 , Issue 3 pp. 268 – 308.

Crainic, *T*.G. and M. Toulouse (2003) "Parallel Strategies for Meta-Heuristics," Chapter 17 of *Handbook of Metaheuristics,* G. Kochenberger and *F*. Glover, eds., Kluwer Academic Publishers.

Dantzig, G. (1963) *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ.

Davoine, *T*., *P*. L. Hammer and *B*. Vizvári. (2003). "A Heuristic for Boolean optimization problems". *Journal of Heuristics* 9, pp 229-247.

Fischetti, M., *F*. Glover and A. Lodi (2005) "Feasibility Pump," *Mathematical Programming - Series A*, 104, pp. 91-104.

Fischetti, M., *F*. Glover, A. Lodi and M. Monaci (2006) "Feasibility Net," research study in process.

Glover, *F*. (1978) "Parametric Branch and Bound," *OMEGA, The International Journal of Management Science*, Vol. 6, No. 2, pp. 145-152.

Glover, *F*. (2005) "Adaptive Memory Projection Methods for Integer Programming," in *Metaheuristic Optimization Via Memory and Evolution: Tabu Search and Scatter Search*, eds. *C*. Rego and *B*. Alidaee, Kluwer Academic Publishers, pp. 425-440.

Glover, *F*. (2006) "Parametric Tabu Search for Mixed Integer Programs," *Computers and Operations Research*, Volume 33, Issue 9, pp. 2449-2494,

Glover, *F*. (2006a) "Satisfiability Data Mining for Binary Data Classification Problems," Research Report, University of Colorado, Boulder.

Glover, *F*. (2007) "Infeasible/Feasible Search Trajectories and Directional Rounding in Integer Programming," *Journal of Heuristics*, Kluwer Publishing (to appear).

Glover, *F*. and *H*. Greenberg (1989) "New Approaches for Heuristic Search: A Bilateral Linkage with Artificial Intelligence," *European Journal of Operational Research*, Vol. 39, No. 2, pp 119-130.

Glover, F. and S. Hanafi (2002), "Tabu Search and Finite Convergence", Special Issue on "Foundations of heuristics in Combinatorial Optimization", *Discrete Applied Mathematics* 119, p. 3-36.

F. Glover, S. Hanafi (2010). Metaheuristic Search with Inequalities and Target Objectives for Mixed Binary Optimization Part I: Exploiting Proximity. *International Journal of Applied Metaheuristic Computing*. Volume 1, No 1, pp. 1-15.Vol. 1, No. 1.

Glover, *F*. and M. Laguna (1997) *Tabu Search*. Kluwer Academic Publishers.

Glover, *F*. and *H*.D. Sherali (2003) "Foundation-Penalty Cuts for Mixed-Integer Programs," *Operations Research Letters*, 31, pp. 245-253.

Guignard, M. and *K*. Spielberg (2003) "Double Contraction, Double Probing, Short Starts and BB-Probing Cuts for Mixed (0,1) Programming," Wharton School Report.

Hanafi, S. and *C*. Wilbaut (2006) "Improved Convergent Heuristic for 0-1 Mixed Integer Programming," *Annals of Operations Research*, doi 10.1007/s10479-009-0546-z.

Hvattum L. M., A. Lokketangen and *F*. Glover (2004) "Adaptive Memory Search for Boolean Optimization Problems," *Discrete Applied Mathematics,* Vol. 142, pp. 99-109.

Nediak, M. and *J*. Eckstein (2007) "Pivot, Cut, and Dive: A Heuristic for Mixed 0-1 Integer Programming," *Journal of Heuristics*, Vol. 13, pp. 471-503, Kluwer Publishing.

Nowicki, *E*., and *C*. Smutnicki (1996) "A Fast Taboo Search Algorithm for the Job Shop Problem," *Management Science*, Vol. 42, No. 6, pp. 797--813.

Pardalos, *P*.S. and O. *V*. Shylo (2006) "An algorithm for Job Shop Scheduling based on Global Equilibrium Search Techniques," *Computational Management Science* (Published online), DOI: 10.1007/s10287-006-0023-*y*.

Patel *J*. and *J*.W. Chinneck (2006), "Active-Constraint Variable Ordering for Faster Feasibility of Mixed Integer Linear Programs," *Mathematical Programming* (to appear).

Pedroso, *J.P*. (2005) "Tabu search for mixed integer programming,".in *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*, eds. *C*. Rego and *B*. Alidaee, Kluwer Academic Publishers, Chapter 11..

Shylo, O.*V*. (1999) "A Global Equilibrium Search Method," (Russian) *Kybernetika I Systemniy Analys*, Vol 1, pp. 74-80.

SoysterA.L., *B* Lev and W. Slivka (1978) "Zero–one programming with many variables and few constraints," *European Journal of Operational Research*, Volume 2, Issue 3, pp. 195-201.

Spielberg, *K*. and M. Guignard (2000) "A Sequential (Quasi) Hot Start Method for BB (0,1) Mixed Integer Programming" *Mathematical Programming Symposium*, Atlanta.

Ursulenko, A. (2006) "Notes on the Global Equilibrium Search," working paper, Texas A & M University.

Wilbaut C. and Hanafi S. (2009) "New Convergent Heuristics for 0-1 Mixed Integer Programming," *European Journal of Operational Research*, 195, pp. 62-74.