J Heuristics (2007) 13: 505–541
DOI 10.1007/s10732-007-9029-z

Infeasible/feasible search trajectories and directional rounding in integer programming

Fred Glover

Received: 17 March 2005 / Revised: 6 June 2006 /
Accepted: 22 August 2006 / Published online: 10 May 2007

Abstract The notion that strategies in non-linear and combinatorial optimization can benefit by purposefully and systematically navigating between feasible and infeasible space has been around for many years, but still is sometimes dismissed as having little relevance for creating more effective methods. To reinforce the case on behalf of approaches that endorse infeasible/feasible search trajectories, it is possible to formulate simple theorems disclosing useful properties of such trajectories in the context of integer programming. These results motivate a closer examination of integer programming search processes based on *directional rounding* processes, including a special variant called *conditional directional rounding*. From these foundations a variety of new strategies emerge for exploiting connections between feasible and infeasible space.

Keywords Integer programming · Feasibility · Search trajectories · Adaptive memory

1 Introduction

A search strategy called strategic oscillation that shares its origins with tabu search (TS) consists in following an organized process of moving back and forth across a particular boundary or collection of boundaries commonly used to orient a search, such as boundaries demarcating stages of construction or neighborhoods or functional values. (See Glover 1977 and Glover and Laguna 1997, Chaps. 4 and 10.) One of the most common types of boundaries used in strategic oscillation is the boundary between feasible and infeasible regions, which is crossed in both directions as a means of exploiting different criteria for evaluating moves, according to which of

F. Glover (✉)
University of Colorado, Boulder, CO 80309-0419, USA
e-mail: fred.glover@colorado.edu

the regions is being visited and whether the search is approaching or receding from the boundary. The relevance of inducing boundary crossing has also recently been emphasized by Gendreau (2004).

In integer linear programming (ILP) there is a natural geometric motivation for an oscillation about the feasibility boundary. This can be summarized by three basic theorems about the ability to generate paths from a selected integer starting point to an optimal solution. The ideas underlying these results are highly intuitive, yet they pose provocative questions. More significantly, they motivate new strategies for ILP problems, embodying a synthesis of concepts from evolutionary and adaptive memory methods based on a process called directional rounding and, in particular, a variant called conditional directional rounding.

2 Basic results

We write the ILP problem in the form

$$\text{[ILP]} \quad \text{Maximize } x_o = cx: \quad Ax \leq b,\ x \geq 0 \text{ and integer.}$$

The associated linear programming relaxation that drops the integer requirement on x will be denoted by [LP].

Let $P(x^1, x^2)$ denote a directed path from an integer point x^1 to a second integer point x^2, generated by steps that change the value of just one variable at a time by $+1$ or -1. For convenience the path $P = P(x^1, x^2)$ will also be treated as a collection of the integer points that compose it. Attention is restricted to paths that are simple, i.e., that do not visit any point $x \in P$ more than once.

We define the length $L(P)$ of $P = P(x^1, x^2)$ to be the number of steps taken to reach x^2 from x^1, hence the number of integer vectors encountered on the path, excluding x^1. We identify the feasible region for the relaxed problem [LP] to be the set of points $F = \{x: Ax \leq b,\ x \geq 0\}$, and denote its boundary by $B(F)$. We also assume that F is bounded, i.e., $x \in F$ implies there is a finite U such that $U \geq x (\geq 0)$.

Denote the Euclidean (L_2 norm) distance between two points x and x', not necessarily integer, by $d(x, x')$, and define $D(x) = \text{Min}(d(x, x'): x' \in B(F))$, identifying the shortest Euclidean distance from x to $B(F)$. Finally, let $D(P) = \text{Max}(D(x), x \in P)$, thus identifying the greatest distance $D(x)$ of any integer point on P to the boundary of the feasible region.

As a point of reference, we let D^0 represent the Euclidean distance of the point $x = (0, 0, \ldots, 0)$ from the point $x = (1, 1, \ldots, 1)$, hence $D^0 = n^{.5}$ where n denotes the dimension of x. When we speak of rounding a solution x, we refer to a vector produced by rounding each fractional (non-integer) component x_j of x to one of its adjacent integer values. For the results below, we assume that [ILP] has at least one feasible integer solution. Justification of the theorems and an expanded discussion of their implications is provided in the Appendix.

Theorem 1 (Infeasible trajectory theorem) *Let x^0 be any non-integer point on $B(F)$ and let x^1 be an integer point in infeasible space obtained by some rounding of x^0. Then there exists a path $P = P(x^1, x^2)$ to an optimal solution x^2 of [ILP], where $D(P) \leq D^0$ and each $x \in P$ except for $x = x^2$ also lies in the infeasible space.*

The assumption that P is simple implies it does not cross over itself, but it can cross over the feasible boundary, jumping from one side of F to the other. A conspicuous and noteworthy feature of this theorem is the asymmetric roles it assigns to feasibility and infeasibility, as attested by the observation that there is no similar theorem about a path that lies within feasible space. Clearly, if the path starts from a feasible point, there may be no way to stay feasible at each step and generate a trajectory to an optimal solution.

The inequality $D(P) \leq D^0$ gives a relatively loose bound on the distance of P from the feasible boundary and tighter bounds under various conditions are clearly possible. Certainly there are ways to round x^0 to be closer to $B(F)$ than a distance of D^0, but it is less easy to identify what these may be under the assumption that x^1 is infeasible. More interestingly, the fact that the distance to $B(F)$ from the next-to-the-last point on P is at most 1, and the distance to $B(F)$ from the point immediately preceding is at most 2, and so on, invites speculation about conditions that may afford a stronger statement about the behavior of a "boundary hugging" path. It is also of interest to ask whether a path P can be generated whose distance from $B(F)$ follows some pattern of alternating monotonicity, although it is easy to demonstrate that the path cannot be assured to grow closer to $B(F)$ on every step.

The preceding theorem is not our ultimate concern, however. In the spirit of strategic oscillation, there is no reason to forego moving into the feasible space by waiting until the last moment to cross the feasibility boundary. In this regard, we can state a stronger result. Define $D^* = D^0/2$, identifying the Euclidean distance of the point $x = (.5, .5, \ldots, .5)$ from the point $x = (0, 0, \ldots, 0)$ (or equivalently from the point $x = (1, 1, \ldots, 1)$). Note that if x' is the integer neighbor of an arbitrary point x obtained by nearest integer rounding, then $d(x, x') \leq D^*$, and it follows that $D(x^1) \leq D^*$ if x^1 results by a nearest integer neighbor rounding of a point x^0 on the boundary of F.

Theorem 2 (Boundary hugging infeasible/feasible trajectory theorem) *Let x^0 be any non-integer point on $B(F)$ and let x^1 be an integer point in feasible or infeasible space obtained by any rounding of x^0 that yields $D(x^1) \leq D^*$. Then there exists a path $P = P(x^1, x^2)$ to an optimal solution x^2 of [ILP] such that $D(x) \leq D^*$ for each $x \in P$. In addition, such a path can be identified for which $L(P) \leq L(P')$ for every path P' generated by Theorem 1.*

The concluding remark of the preceding result says that the shortest path by Theorem 2 is shorter (or no longer) than the shortest path generated by Theorem 1. Again, a stronger result about the distance of points on P from $B(P)$ would be interesting to identify.

However, this theorem is still not as encompassing as might be desired. A better result for search purposes can evidently be stated as:

Theorem 3 (Relaxed infeasible/feasible trajectory theorem—boundary hugging from the infeasible side) *Let x^0 be any non-integer point on $B(F)$ and let x^1 be an integer point in feasible or infeasible space obtained by any rounding of x^0 that yields $D(x^1) \leq D^*$. Then there exists a path $P = P(x^1, x^2)$ to an optimal solution x^2 of*

[*ILP*] *such that* $D(x) \leq D^*$ *for each* $x \in P$ *that lies in the infeasible region. Moreover, such a path can be identified that satisfies* $L(P) \leq L(P'')$ *for every path* P'' *generated by Theorem* 2.

There are some subtleties about the trajectories identified in these theorems that deserve closer attention. For example, I suspect the paths for Theorems 2 and 3 can be structured so that they always cross from infeasible space to feasible space when such a move is available, while still maintaining their stated properties and without increasing the length of a shortest path. However, such a claim would require some added sophistication to establish. Another question would be whether such properties are shared by a path that always proceeds to a local optimum in the feasible space, before crossing into infeasible space. Most likely such a "piece-wise locally optimal" trajectory would at least be compatible with identifying a path of shortest length. A related question that springs to mind is whether the foregoing properties are compatible with the situation where P is directionally (component-wise) monotonic, i.e., where the value of any given variable is non-decreasing or non-increasing throughout successive points in the path.

Apart from a quest for such refinements, the preceding results provide an added rationale for a search process that does not hesitate to cross the boundary between feasible and infeasible space. However, it would be unsatisfying to stop here, without considering possible mechanisms for generating such search paths, to provide algorithms for locating feasible and optimal ILP solutions.

3 Algorithms

We confront the challenge of generating a "good" path P leading to an optimal solution by adopting a design that repeatedly passes in and out of feasible continuous space, and uses the geometry of this space to seek feasible integer solutions. Simultaneously, we use information derived from integer points that are not feasible to determine new ways to access the feasible region, creating an oscillation across the boundaries both of LP feasibility and integer feasibility. As integer feasible solutions are found, the search for optimality is additionally guided by imposing inequalities that constrain F more tightly.

3.1 Directional rounding

A natural way to generate trajectories that are influenced by the geometry of the feasible space is to employ the idea of *directional rounding* (Glover 1995; Glover and Laguna 1997; Lokketangen et al. 2000). We examine the fundamental ideas from a slightly different perspective here than adopted in other developments, as a basis for introducing additional strategies that have not previously been joined with these ideas.

3.1.1 Basic concepts

Directional rounding operates on two points, an *initiating point* x^0 and a *focal point* x', to generate a *resultant* integer point x''. We assume the inequalities defining

F imply integer bounds for each x_j given by $U_j \geq x_j \geq L_j$, where U_j can be ∞ if no finite bound is implied. (The lower bound L_j is included as being possibly different from 0, because we will later be concerned with augmenting the inequalities defining F to incorporate more restrictive bounds.) Relative to a selected focal point x', let v_j and $v_j + 1$ denote integer values that bracket the value of x'_j, taking $v_j = x'_j$ if x'_j is integer. Then we define $L'_j = \text{Max}(L_j, v_j)$ and $U'_j = \text{Min}(U_j, v_j + 1)$. (Hence L'_j and U'_j are the same as the integers v_j and $v_j + 1$ adjacent to x'_j unless x'_j violates one of its bounds.) Based on these definitions, the resultant integer point x'' created by directional rounding is given by $x''_j = L'_j$ if $x'_j \geq x^0_j$ and $x''_j = U'_j$ otherwise.

In the typical case, x^0 is taken to be an optimal extreme point solution to [LP], and the focal point x' is selected as a point on a (half) ray of the LP cone whose vertex is at x^0. Assuming that [LP] is solved by an extreme point method, the ray can be determined by taking a convex combination of the updated LP columns corresponding to the current non-basic variables. Denoting the resulting convex combination as the column vector D, the ray can be represented as the set of points $\{x = x^0 + Du\colon\ u \geq 0\}$, where u is a scalar. (A special case for generating D, where the LP basis representation of x^0 is primal degenerate, is discussed in Sect. 5.) We normally assume x^0 is not integer feasible, or else it would be optimal for [ILP] and there would be no need to search for an optimal solution.

Any selected focal point x' on the ray given by $\{x = x^0 + Du\colon\ u \geq 0\}$ is feasible relative to the inequalities of F that define x^0 as an extreme point solution. All points x' on the ray will produce resultant directionally rounded points x'' whose components x''_j successively increase or successively decrease (depending on j) as u increases, subject to bounding x''_j by U_j or L_j. Consequently, all possible integer solutions x'' produced by directional rounding relative to points on the ray can be identified, in sequence, starting from an integer point adjacent to x^0, by a simple series of calculations that give the breakpoint values of u at which the value of some L'_j or U'_j will change (Glover and Laguna 1997).

3.1.2 Edges and centers

A key motive for directional rounding is that every feasible integer point lies on some ray within the LP cone. In the 0–1 case, every focal point x' on such a ray will cause x^0 to directionally round to the same resultant point x'', which is the targeted integer feasible point (that is, the 0–1 point that the ray intersects). For general integer variables, every point on a segment of the ray will directionally round to the targeted point. Moreover, there is a collection of rays defining a sub-cone within the LP cone that will similarly contain segments having this property.

For these reasons, it is useful to consider edges that consist of the intersection of rays with the feasible LP region, i.e., line intervals of the form $[x^0, x^B]$, where x^B is the point on the ray that lies farthest from x^0 within the feasible region, hence lies on the boundary $B = B(F)$. If the feasible region is unbounded and the ray is infinite, we arbitrarily restrict x^B to lie at some finite distance from x^0. Every point on $[x^0, x^B]$ is LP feasible, and disregarding the unbounded case, all points outside the interval $[x^0, x^B]$ are infeasible. The point x^B is easy to identify, by a calculation that identifies the largest value of the scalar u such that the point $x = x^0 + Du$ satisfies the problem constraints.

Intuitively it seems reasonable that the center x^c of an edge $[x^0, x^B]$, which lies more fully "inside" F than other points on the edge, is a good candidate for a focal point x' that is likely to round to a feasible integer solution. Intuition can be misleading, of course, and in the case of 0–1 problems it is easy to see that all points on $[x^0, x^B]$ directionally round to the same 0–1 solution for $u > 0$, since by $x = x^0 + Du$ if a component D_j of D is positive x_j will directionally round to 1 and if it is negative x_j will round to 0. (A point can also directionally round not just to a single solution but to a set of solutions in the case where $D_j = 0$, since then $x_j = x_j^0$ and both directions of rounding are equally admissible. In such cases it is useful to let the values of D_j in other rays create a weighted vote for the sign of D_j in the vector D under consideration.) But centers retain their intuitive merit for more general integer variables, and for additional kinds of rounding, such as nearest neighbor rounding and conditional directional rounding as treated in Sect. 2.2. They also serve as a foundation for creating new edges to probe the search space.

3.1.3 Derived edges and centers

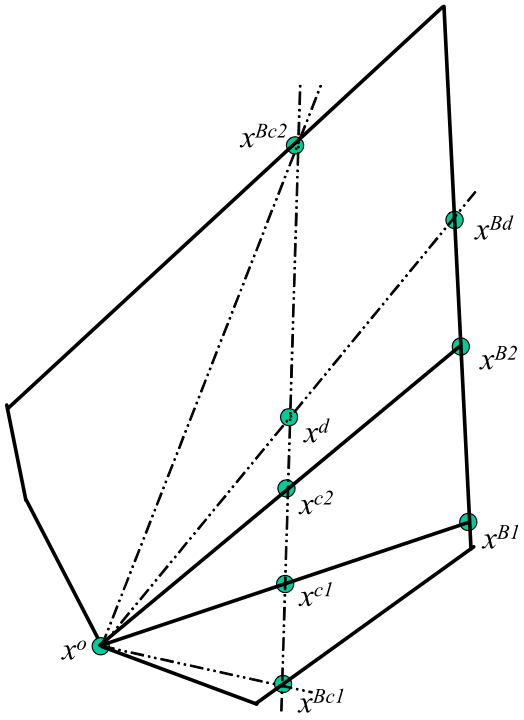
The idea of directional rounding originated as a way of exploiting the evolutionary *scatter search* approach, which comes from the same source as strategic oscillation. In its initial form, scatter search employs linear combinations of points to generate weighted centers of sub-regions, and compounds these by generating weighted centers of centers, and so forth, accompanied by a generalized rounding process to assign integer values to integer variables (Glover 1977). Directional rounding emerged to meet the goal of identifying a simple version of rounding that has useful properties.

From the scatter search perspective, it is natural to take advantage of centers of edges to create derived edges and centers. We illustrate such a process in the diagram of Fig. 1, which may be conceived to represent a 2-dimensional cross section of a feasible LP region in n-space. In this diagram, x^{B1} and x^{B2} are two boundary points generated by extensions of rays from x^0 to create edges $[x^0, x^{B1}]$ and $[x^0, x^{B2}]$. (The three points x^0, x^{B1} and x^{B2} therefore determine the two dimensions from which the cross-section shown in Fig. 1 is derived. Unlike x^0, the other corner points of this figure will not in general correspond to extreme points of F.) Upon identifying centers x^{c1} and x^{c2} of these two edges, a line $x = x^{c1} + (x^{c2} - x^{c1})u$ is constructed through these centers. A new derived edge $[x^{Bc1}, x^{Bc2}]$ is then obtained by minimizing and maximizing u, subject to keeping x within F, to identify the boundary points x^{Bc1} and x^{Bc2}. This derived edge is shown in Fig. 1 as the dotted line passing through the centers x^{c1} and x^{c2}.

Continuing the process at this second level, the next step is to locate the center x^d of this derived edge, and to create a ray $x = x^0 + (x^d - x^0)u$ from x^0 through x^d, thus identifying the new boundary point x^{Bd} (obtained by maximizing u subject to keeping x within F) and creating a second derived edge $[x^0, x^{Bd}]$, also shown as a dotted line.

Additional derived edges $[x^0, x^{Bc1}]$ and $[x^0, x^{Bc2}]$ that may be of interest are likewise shown in Fig. 1. The centers x^{c1}, x^{c2} and x^d created by this process are natural candidates for focal points for both directional and nearest neighbor rounding, and centers of other edges indicated may also be considered. Refinements of this type of construction are examined in Sects. 3 and 5.

Fig. 1



3.1.4 A first approach

We examine two main approaches for exploiting these ideas. The first approach, following, is a basis for creating a first integer point x^1 on a path P, or more generally, a set of such first points to be examined sequentially or in parallel. This procedure is then embedded in the second approach described subsequently.

Approach 1

1. Generate convex combinations of the non-basic columns associated with the solution of [LP], to produce a collection of D vectors and associated line segments $[x^0, x^B]$ to serve as the source of focal points x' for directional rounding. Incorporate centers and derived edges as an additional source of focal points.
2. Identify one or more candidate focal points from each chosen line segment, by selecting centers or by selecting among points sampled on the line segment.
3. Select a subset of focal points x' for directional rounding from the collection of candidate points identified in Step 2 and place the resultant points x'' in a set X''.

Within Step 2, the points examined on the line segment may be obtained by computing the breakpoint values of the implicit parameter u that yield distinct integer solutions, or more simply by sampling p points from the segment at intervals $1/p$, $2/p, 3/p, \ldots$, along its length. The choice criterion within Step 2 for selecting among focal points can be based on using an evaluation that favors points that are closer to

being integer, breaking ties in favor of larger x^0 values. An alternative is to combine Steps 2 and 3 by immediately identifying the x'' point associated with each x' examined in Step 2, and evaluating x'' based on an LP infeasibility measure and on its x_o value, to produce the set X''. As previously noted, nearest neighbor vectors may also be considered among the candidates for possible inclusion in X''.

Considerations for creating D When forming the vector D as a convex combination of updated non-basic columns derived from the current LP basis representation, it is useful to differentiate columns associated with non-basic slack variables for the inequalities defining F from the columns associated with non-basic variables that are components of x. The reason for this stems from the fact that non-basic x variables automatically receive integer-values as components of x^0, and a number of these may continue to receive these integer values in a feasible or optimal solution. Thus, in the interest of a parsimonious search process, we begin by focusing on variables x_j that receive fractional values by the assignment $x_j = x_j^0$. This is facilitated by starting with rays, and hence edges, defined relative to a D vector that is generated by assigning 0 weights to most of the non-basic columns associated with integer variables, allowing positive weights to be assigned only to columns associated with non-basic slack variables and to a relatively small number of columns associated with non-basic x variables, such as a subset of x variables having smallest reduced costs. Non-basic slack variables for inequalities of the form $x_j \leq U_j$ or $x_j \geq L_j$, for U_j and L_j integer, are treated in the same manner as x variables, since they directly yield integer values for the x variables when non-basic.

Such an approach has an additional advantage of reducing the amount of computation to generate D, particularly for problems involving large numbers of integer variables, many of which are likely to be non-basic. It is essential to allow for some non-basic x variables to be considered in generating D, as in the extreme case where $Ax \leq b$ embodies a collection of equality constraints for which no slack variables exist.

This approach evidently runs the risk of excluding consideration of non-basic x variables that will in fact receive positive values in an optimal solution. Refinements that combat such a potential deficiency are discussed in Sects. 4 and 5. The risk of excluding consideration of appropriate non-basic variables will also be offset by the second of the two main approaches, described next.

3.2 Conditional directional rounding

The idea of generalized rounding in scatter search includes reference to conditional (staged) procedures for applying a rounding process. In the present case we consider a conditional process within the setting of directional rounding, utilizing the all-at-once rounding process of Approach 1 as a means for generating integer trial solutions and also as a foundation for selecting variables, one at a time, to be subjected to directional rounding. Evidently, it is appropriate to take account of conditional relationships known from the problem structure even in the application of Approach 1, as in the presence of 0–1 multiple choice constraints where rounding a particular variable upward to 1 implies that all other variables in the same multiple choice set should be rounded down to 0. We assume that such structure-based forms of conditionality are

already embodied in the generation of points x'' as candidates to include in the set X'' in Approach 1.

From a broader perspective, however, we now consider a one-at-a-time process that generates more complete information about the consequences of rounding a selected variable in a chosen direction by the common approach of taking advantage of LP relaxations. Consequently, we examine a step-wise process where each iteration of generating the vector x'' induces the current rounding of a particular fractional variable x_j by means of a branching inequality that is adjoined to the [LP] problem. In this manner, implications of the selected rounding operation are generated by the solution of [LP], and are taken into account by re-optimizing the current LP problem before again invoking the directional rounding process of Approach 1. In turn, the new application of Approach 1 after each re-optimization (or after a chosen number of steps of re-optimization) gives a basis for creating additional all-at-once candidates x'' for X'', which are used in the choice rules for selecting the next variable to round by means of a branching inequality.

This conditional rounding approach gives an indirect way to modify the vectors D used in directional rounding, based on information from the resultant integer points x''. Such a mechanism also changes the point x^0 relative to which D is defined, as a result of the change in [LP] produced by the addition of branching inequalities.

3.2.1 Flexible branching

Rather than use branching inequalities in the manner commonly employed in branch and bound, we propose to manage them by the more flexible design employed in *tabu branching* (see, e.g., Glover and Laguna 1997, Chap. 6). To facilitate this process, we identify a simple scheme to track implications that result in compulsory branches at various stages of the search.

Branches that are freely chosen during the search will be called *independent* branches, and branches that are compulsory (such as those used to fix the values of variables in 0–1 problems) will be called *dependent* branches. We assume that compulsory branches identified by pre-processing are permanently included among the constraints defining F, and hence are eliminated from consideration as a source of dependent branches.

Branches generated in unbroken succession that are all independent or all dependent are treated as a *set* of branches, without any implied ordering of the elements of the set. The branch generation process is then viewed as composing an alternating sequence of these independent and dependent branch sets. Each independent and dependent set of branches generated in this sequence receives a *dependency marker*, starting by assigning a marker of 0 to members of the first set of independent branches. When a new set of dependent branches is generated, the dependency marker is incremented by 1, and all members of the dependent set therefore receive a marker whose value is 1 larger than received by members of the preceding independent set. No change is made to the marker when a new set of independent branches is produced, and hence members of this set receive the same dependency marker as the members of the preceding dependent set. All dependent branches having a dependency marker larger than that of a given independent branch are said to be *downstream* of the independent branch. Similarly, all independent branches having a marker smaller than that of a given dependent branch are said to be *upstream* of the dependent branch.

To make use of these conventions, we define a branch to be *binding* under the following conditions: (a) relative to an optimal (primal feasible) LP basis, the slack variable for the branch is non-basic with a positive reduced cost; (b) relative to a dual feasible LP basis that discloses the absence of a feasible solution, the slack variable is non-basic with a positive coefficient in an updated equation that signals infeasibility, or else is basic in this equation (and hence is assigned a negative value). A branch is defined to be *non-binding* if neither of these conditions holds. (Condition (b) will be used later on only to identify binding variables, and it is unnecessary to generate all equations identifying such variables.) Then the branching mechanism we employ is governed by two stipulations:

(S1) Whenever an independent branch is reversed (replaced by its complementary branch), all downstream dependent branches are dropped, thus removing their inequalities from the set defining the current instance of [LP].

(S2) Whenever an independent branch becomes non-binding in an optimal LP solution the branch is dropped, and all dependent branches downstream of this inequality are re-labeled with a dependency marker that is 1 greater than the marker of the independent branch most recently introduced.

The second of the foregoing stipulations places the re-labeled dependent branches in the same set as those dependent branches that are immediately identified after introducing the latest independent branch. The result may cause some independent sets to merge into the same set, and the re-labeling can maintain the property that all members of a common independent set are the same by the following simple device. Identify the marker value m of the independent branch that is reversed and then re-assign this value to the markers of all independent branches having markers greater than m. Upon doing this, the remainder of stipulation (S2) reduces to specifying that the markers of all dependent branches having marker values greater than m will be assigned a value equal to $m + 1$. On the other hand, the stipulations for processing the markers result in the same outcome if the rule for assigning markers is changed to specify that each branch, at the time of its execution, is assigned a marker that is 1 larger than the largest currently existing marker, and each re-labeled dependency marker similarly is assigned a new value 1 greater than the largest preceding marker. This does not create an implicit ordering among members of the same dependent or independent set, but the use of distinct marker values may be useful for other types of bookkeeping operations.

Taken together, stipulations (S1) and (S2) provide a means to allow dependency implications to be respected without having to rely on a tree search memory structure and to abide by the limitations such a structure imposes on the search. Stipulation (S2) is not essential for 0–1 problems, but it is useful for maintaining an emphasis on influential branches. In addition to these two primary stipulations, we identify two supplementary stipulations in Sect. 4.2 that yield interesting properties for the resulting branching options, including the property that a particular restricted application of the resulting rules will produce the same choices provided by branch-and-bound, hence producing a search structure that includes branch-and-bound as a special case.

To complete the preliminaries for describing Approach 2, let x^* denote the best integer feasible solution currently found and let $x^*_o = cx^*$ identify the associated objective function value, where $x^*_o = -\infty$ until a feasible integer solution is identified.

We assume the problem [LP] incorporates the inequality $x_o \geq x_o^* + \varepsilon$, for a small value of ε, to render an integer solution infeasible if it does not improve upon x^*.

Approach 2

1. Apply Approach 1 to produce a set X'' of integer trial points x''.
2. Relative to the solution x^0 of the current instance of [LP], let $J^0 = \{j\colon\ x_j^0$ is fractional$\}$, and for each $j \in J^0$, define $V(j) = \{v\colon\ x_j'' = v, x'' \in X''\}$, thus identifying the set of integer values v received by x_j over the solutions in X''. Then let $v^*(j) = \text{Max}(v\colon\ v \in V(j))$, and select $h = \arg\max(v^*(j)\colon\ j \in J^0)$, identifying the index h for the variable that yields the largest $v^*(j)$ value.
3. Impose an independent branching inequality for x_h as follows:

$$x_h \geq v^*(h) \quad \text{if } v^*(h) > x_h^0,$$
$$x_h \leq v^*(h) \quad \text{if } v^*(h) < x_h^0.$$

4. Re-optimize the resulting problem [LP] and impose any compulsory branches that may result (repeating the re-optimization as new compulsory branches are introduced).
5. If [LP] has no feasible solution, go to Step 6. If the LP solution x^0 is integer feasible (and hence qualifies as a new best solution x^*), update x_o^* to render this solution infeasible and likewise go to Step 6. Otherwise, drop any independent branches that become non-binding and return to Step 1 to generate new candidate integer vectors by Approach 1 relative to the current x^0 and its associated LP cone.
6. Select an independent branching inequality that is binding in the currently optimal LP basis or in the dual feasible basis that identifies infeasibility. Reverse the inequality (i.e., replacing $x_h \geq v^*(h)$ by $x_h \leq v^*(h) - 1$ or replacing $x_h \leq v^*(h)$ by $x_h \geq v^*(h) + 1$), and drop all downstream dependent branching inequalities. Then re-optimize the resulting [LP] problem, and return to Step 4.

The method may terminate either in Step 1 or in Step 4 after a chosen number of iterations. To control computational effort, it is generally prudent to limit the number of points included in the set X'' identified in Step 1.

Although we will not go into detail here about alternatives for using adaptive memory to guide the foregoing process, we observe that a simple procedure for using such memory is to modify the definition of $v^*(j)$ in Step 2 by reference to recency and frequency information to penalize or encourage choices in the usual manner of tabu search. Additional observations about the use of adaptive memory are made in Sect. 5.

4 Preliminary refinements

There are several refinements of Approaches 1 and 2 of the preceding section that provide opportunities to improve their performance. We identify a few preliminary refinements in this section before progressing to an "outside-in" method for searching the space that complements the directional orientation of Approach 2, Later we then examine additional, more advanced refinements.

4.1 Choice criteria for branching

(a) A simple alternative for implementing Step 2 of Approach 2 replaces the set $V(j)$ by creating two "counters" $N^+(j)$ and $N^-(j)$, respectively giving the number of solutions $x'' \in X''$ such that $x''_j > x^0_j$ and such that $x''_j < x^0_j$. Then select a variable x_h for branching by defining $h = \arg\max(|\,|N^+(j)| - |N^-(j)|\,|:\ j \in J^0)$. Finally, $v^*(h)$ is re-defined to be the "rounded up" value of x^0_h if $|N^+(h)| > |N^-(h)|$ and the "rounded down" value of x^0_h otherwise.

Auxiliary criteria for selecting the branching variable x_h in this step can be based on weighting the solutions $x'' \in X''$ according to an infeasibility measure, so that coordinates of solutions with a greater infeasibility are given a smaller count in defining the sets $N^+(j)$ and $N^-(j)$ than those of solutions with a smaller infeasibility measure. Adaptive memory guidance is implemented in this case by using recency and frequency memory to modify these counts.

(b) A somewhat different criterion for selecting the branching variable x_h can be based on incorporating additional information from Approach 1. To do this a memory is maintained that identifies the smallest positive values of u for which the edge extension $x = x^0 + Du$ assigns an integer value to each fractional-valued variable x_j, $j \in J^0$. Variables that receive integer values on the edge extension for smaller values of u may be tentatively viewed as those whose integer requirements are more easily satisfied.

By this interpretation, the choice of x_h may be biased in favor of variables that more often fall in the "easily satisfied" category on the assumption that accurate branching directions can be more readily identified for these variables, and once the associated branches are introduced, the branching process will cause other variables likewise to enter the "easily satisfied" category. Alternatively, it may be speculated that handling the "harder to satisfy" variables first may cause the problem to collapse and yield a residual problem that is easier to solve, in which case these variables should be given higher priority to be selected for generating branches. If the truth lies somewhere between these two extremes, a criterion aimed at choosing x_h to be in a category of "not too hard and not too easy to satisfy" may be preferable. Evidently, this is a situation where empirical research should provide useful insights.

(c) To accelerate the conditional rounding approach, it is possible to use a form of aggregate conditional rounding that rounds several variables simultaneously rather than one-at-a-time. If the information from the all-at-once directional rounding of Approach 1 discloses the existence of a subset of variables x_j, $j \in J$, such that x_j takes a value x''_j that is uniformly greater or uniformly smaller than x^0_j in a significant number of directionally rounded vectors x'', then a *brazen branching* ($|J|$-at-a-time rounding) step can be employed that imposes branches to establish the indicated rounding direction for each variable in the subset.

A more stringent application of such a rule would require that the vectors x'' yielding a uniform rounding direction for x_j are the same for all $j \in J$. Such a rule can also be employed with nearest neighbor rounding.

4.2 Supplementary stipulations to manage the branching operations

In addition to the stipulations (S1) and (S2) for managing branching operations, two others may be useful for structuring the search in certain applications:

(S3) when an independent branch is reversed, re-label it as a dependent branch and give it a dependency marker 1 larger than that of the largest current independent marker;
(S4) independent branches that are non-binding may be retained in Step 5 of Approach 2 (in contrast to the current rule that derives from (S2)), subject to retaining the rule of Step 6 that requires an independent branch to be binding in order to be chosen for reversal.

Although stipulation (S3) slightly decreases flexibility of the method, it provides a compensating ability to avoid certain types of unwanted repetitions without the need for a more complex memory structure. This stipulation does not require stipulation (S4) to accompany it, but (S4) does require the accompaniment of (S3). When (S3) and (S4) are employed together, the rules of Approach 2 include the rules of a backtracking branch-and-bound procedure as a restricted special instance.

The connection that causes these stipulations to provide a more general framework than ordinary branch-and-bound is not immediately evident, and requires some elucidation, because branch-and-bound methods for integer programming make no reference to selecting branches that are binding at those junctures where they return to a previous node of a search tree. However, the tree structure itself implies that such a selection rule is employed when branches are enforced by solving LP relaxations. When (S4) is added to (S3), Approach 2 permits non-binding branches to be retained on condition of obeying such a selection rule regarding binding branches, although without having to accept the restrictions of a tree structure.

To see the relevance of the rule requiring that reversed branches must be binding, other than the obvious motivation that such a rule limits attention to an influential subset of branches, consider the case where a variable x_j progressively receives tighter lower and upper bounds by branching, for example, $L_j^1 < L_j^2 < \cdots < L_j^r$, and $U_j^1 > U_j^2 > \cdots > U_j^s$ (subject to $U_j^s \geq L_j^r$), where the branches may have been executed by interleaving the successive L_j and U_j bounds in any manner. Then it is evident that the branch for one of the two final bounds L_j^r or U_j^s must be reversed before reversing earlier branches. It is also clear that this relationship is connected to the fact that these latter branches render their predecessors non-binding. (Additional subtleties are introduced by the role of dependent branches, and the need to integrate (S3) with (S4).)

More generally, apart from the need to structure a search to heed such relationships when currently non-binding branches are retained, each time a fractional variable is chosen as the source of a branch in ordinary branch-and-bound, the branch will become binding in the resulting re-optimized LP solution. (To be strictly accurate, this statement must allow for primal degenerate cases where the reduced cost for the branching slack variable may be 0 instead of strictly positive. In this sense our definition of "binding" that focuses on the non-degenerate case is more limiting than necessary, though it favors reversing branches that have a greater impact.) By the same token, when the tree search approach requires the associated node to be re-visited prior to its predecessors, it is assured that the branch chosen to be reversed is binding. This applies even to a non-backtracking form of tree search, such as one based on the best bound rule, since the recovery of an earlier node excludes subsequent branches from consideration, and hence the last active branch that leads to this

node is the one subject to change, and this branch is binding relative to the conditions inherited by this node.

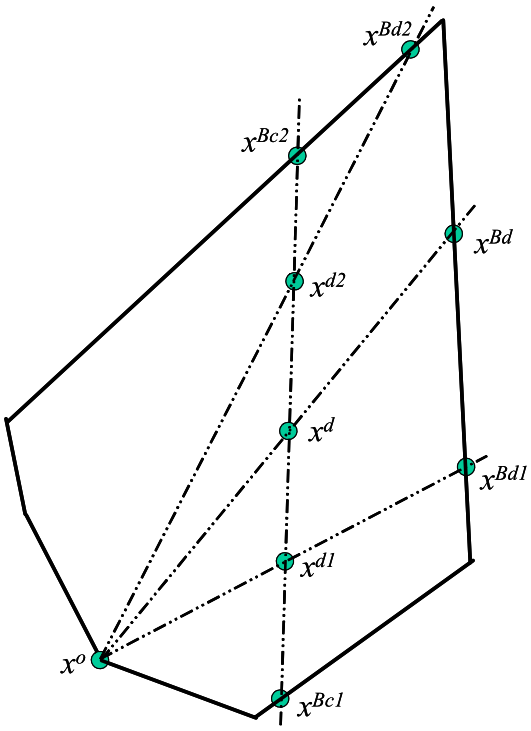
Among non-backtracking tree searches, we may note that a branch-and-bound procedure based on a best bound rule partially resembles a classical intensification approach in tabu search, which returns to promising regions to search them more thoroughly. However, in the present setting, as in customary applications of tabu search, such an intensification approach would retain a record only of highest quality solutions and the branch structures that engendered them, rather than keeping track of all potentially unexamined alternatives, in analogy to the best bound rule. The fathoming of LP optima based on an x_o bound occurs in both processes, but an intensification strategy within Approach 2 would reasonably keep track only of solutions that assign integer values to a significant number of the integer variables. Disregarding the proximity to integer feasibility by a best bound rule, as such a rule is applied in branch-and-bound, can generate distorted evaluations of solution quality at various nodes of the search tree. The structure of Approach 2, and of tabu branching generally, makes it possible to maintain the search in regions that are substantially closer to integer feasibility than achieved at most nodes of a branch-and-bound tree, while providing access to non-tree moves that leapfrog over intermediate tree search steps, corresponding to a process that jumps from a leaf node or near-leaf node of one tree to a similar node in a tree of a different structure. (These connections and the rationale behind them are discussed in Glover and Laguna (1997).) In the present case, the incorporation of the stipulations (S3) and (S4) makes it possible to adopt a tree search organization as closely as desired, while still offering a framework that embraces more flexible choices.

4.3 Further exploitation of derived edges and centers

The directional rounding framework makes it possible to get additional mileage out of the edges $[x^0, x^B]$ generated. The process of identifying centers and derived edges illustrated earlier in Fig. 1 can also be used to modify edges previously generated, to replace them by others that may be suitable for creating points that are more dispersed throughout the space. This edge modification and replacement approach can be illustrated by taking the diagram of Fig. 1 as a starting point. Once the initial derived edge $[x^{Bc1}, x^{Bc2}]$ is identified, together with its center x^d, we may replace the other two points x^{c1} and x^{c2} on this edge by new points that are more dispersed in relation to x^d. As shown in Fig. 2, below, this can be done by identifying the centers x^{d1} and x^{d2} of the two half-edges $[x^d, x^{Bc1}]$ and $[x^d, x^{Bc2}]$. Then the candidates for focal points can be taken as x^d, x^{d1} and x^{d2} rather than x^d, x^{c1} and x^{c2}, or as the union of these possibilities.

In the general case, derived centers can be generated at various levels from convex combinations of midpoints of derived edges, or of x^0 and the boundary points constituting the endpoints of these edges. Convex combinations that weight points according to their distance from x^0 and according to their objective function values give a strategic bias to a search for feasibility and optimality. The accumulation of new centers should normally be limited to points that are separated by some minimum distance from current members of the collection.

Fig. 2



A key question will be whether the creation of derived edges and centers is more effective than undertaking to create a larger number of D vectors initially, and focusing just on the points lying on the edges associated with these vectors. The answer will depend on the ingenuity of the strategies that are applied in each case, but the use of derived edges and centers affords useful intuition to guide the creation of such strategies.

4.4 Mixed directional rounding

Given that nearest neighbor rounding has an element of intuitive support, it appears relevant to consider a form of rounding that combines features of nearest neighbor rounding with those of directional rounding. The following approach, which we call *mixed directional rounding*, has this character. For a given focal point x' on a ray from x^0, the basic idea is to perform nearest neighbor rounding on all components x'_j of x' that are relatively close to the associated values x^0_j of x^0, and to perform directional rounding relative to remaining components of x'. The rationale is that components x'_j that move farther away from the associated x^0_j values are more likely to give a valid indication that the rounding should indeed be made in their direction. Thus, we select a threshold T and specify that x'_j should be rounded to its nearest neighbor if $|x'_j - x^0_j| < T$, and that x'_j should be directionally rounded (relative to x^0_j) otherwise.

The determination of T, and the distance from x^0 that the focal point x' is selected to lie, should be interrelated. In particular, in one variant we propose that x' be located

far enough along the ray from x^0 so that a chosen fraction f of the fractional variables in the solution $x = x^0$ have attained an integer value, or have gone beyond an integer value, in the resulting vector. (Examples that come to mind are $f = .2, .3, .5$ or .7.) But another variant derives from the intuitive argument that nearest neighbor rounding should in most cases be done in relation to components of x^0 rather than components of x', which will occur if x' is taken close enough to x^0 on the ray. These two different versions can be reconciled by a merged rule that first identifies which components x'_j of x' will constitute the chosen fraction that are directionally rounded, and then selects x' to be close to x^0, so that a component x'_j chosen to be directionally rounded will be transformed into a nearest neighbor of x^0_j in a directional sense, while each remaining components will be transformed into a nearest neighbor of x^0_j without regard for direction. A simpler rule that adheres to the use of T as initially specified would likewise choose x' close to x^0, but would select T based on the fraction of variables that are directionally rounded, rather than pre-specifying the identity of variables to be directionally rounded based on the somewhat different criterion that these variables will attain or pass beyond an integer value if x' were to be located farther out along the ray.

4.5 Diversification by reference sets

A useful way to take further advantage of the connections to scatter search is to make use of a *reference set* R consisting of points that are preferred candidates to launch new searches. For increased generality, we consider the optimization problem

$$P(x): \quad \text{Maximize } f(x) \quad \text{subject to } x \in X.$$

The objective function $f(x)$ may be nonlinear and X may be defined by feasibility requirements that derive from nonlinear constraints and compel only some, but not necessarily all, of the components of x to be integer. We continue to suppose that F is a convex feasible region capable of being expressed by linear programming constraints, and will assume that $x \in F$ is a relaxation of the requirement $x \in X$ (hence $X \subseteq F$). We desire to generate the reference set R by populating it with chosen elements of F, or more precisely, with chosen elements of a subset F' of F, whose members $x' \in F'$ have been generated to provide candidate points to initiate future phases of search. Since the requirement $x \in X$ may be somewhat stringent, as where it may be difficult to find integer feasible solutions to discrete problems, we allow x' to be evaluated by reference to a penalty function $f^*(x)$ that is the same as $f(x)$ for $x \in X$, but that yields $f^*(x) < f(x)$ for solutions $x \notin X$. In addition, we allow for the possibility that not all elements x' have been fully evaluated as potential solutions to $P(x)$ (by computing $f^*(x')$ and implicitly testing whether $x' \in X$), since this may involve time consuming operations in the general case. However, it can be important to restrict F' to points exhibiting a certain minimum quality level, and hence we assume that points that have not been subjected to a full evaluation have nevertheless been evaluated by a fast screening criterion to provide an approximate determination of their quality. Thus, in particular, in the integer programming context points chosen for inclusion in F' may come from derived edges and centers as discussed in Sect. 2 and (3.3), and may be supplemented by sampling processes in order to assure that F'

covers an appropriately dispersed portion of F. Subsequent sections give additional ways to generate points that usefully qualify for inclusion in F'.

As a foundation for periodically generating a reference set R that constitutes a diverse collection of points drawn from F', we begin with a starting set of points $R = R_o$, and then select successive points of F' so that each new one to be added to R is chosen to maximize its minimum distance from all points belonging to the current set R.[1] The process is initiated by selecting R_o to be a small set of points that have been fully evaluated in past searches and that we desire to be included within R as a foundation for launching new searches. (Scatter search applications verify the merit of including one or a few of the best points previously found when generating reference sets for diversification.) The rule for building R also makes reference to an *exclusion set* E_o consisting of points that we desire to permanently exclude from membership in R, and more generally, whose proximate neighbors we also seek to exclude.

Diversification rule to build a new reference set R

1. Begin with $R = R_o$, and set $F' := F' - R_o$.
2. Select $x' \in F'$ by the rule: $x' = \arg\max(\min(d(x', x):\ x \in R \cup E_o))$, and let $R := R \cup \{x'\}$, $F' := F' - \{x'\}$.
3. Repeat Step 2 until R contains a specified number (or composition) of points.

As evidenced in Step 2, each new point x' is chosen to maximize its minimum distance not only from points previously included in R, but also from points belonging to the exclusion set E_o. The distance measure $d(x', x)$ used in this step is not necessarily Euclidean (in contrast to the stipulation in Sect. 1), and we emphasize the importance of scaling or otherwise weighting the variables so that distances along particular dimensions do not inappropriately dominate or skew the measure.

Determination of E_o The issue of determining the exclusion set E_o rests on identifying subsets of points generated and evaluated during the search history that we loosely categorize as follows: G = a set of "good" points (e.g., local optima, or local optima satisfying a certain quality threshold), B = a set of "bad" points, M = a set of "mediocre" points. These sets need not constitute all points of their respective categories encountered during previous search, but rather may constitute representative sets, as by taking centers of clusters, or using other forms of representation to keep the sizes G, B and M within reasonable limits. (The set G will typically be defined to admit only a small number of points in any case, but the number of "bad" and "mediocre" points in F' may be much larger.)

Thus, in brief, we may identify evident alternatives for defining E_o to consist of (1) $E_o = B$; (2) $E_o = B \cup G$; (3) $E_o = B \cup G \cup M$; and so forth. Some diversification approaches in the literature effectively operate by re-starting the search at points distant from elements of G, analogous to selecting $E_o = G$,[2] but presumably if G is included in E_o then it is appropriate to include at least B or M in addition.

[1] The original version of this rule, as proposed in Glover and Laguna (1993), contains additional components to break ties among elements that lie the same max(min) distance from points in R.

[2] These re-starting approaches use a different organization than considered here and do not make use of the max (min) rule.

A diversification strategy that makes use of R may be augmented by an approach that continues to avoid regions containing points of B (and even M) after R has been generated to launch a new search. A *tabu reference set TR* can be used to enforce this continued avoidance in either of two straightforward ways: (i) using a coded hash vector, so that any solution that is a candidate to be visited can be translated into this code, and then avoided if the code matches the code of a solution in *TR*; (ii) defining a region that surrounds each point in *TR*, and then designating these regions tabu. (Approaches based on defining "tabu regions" are discussed in Glover (1994).)

Parallelization Finally, we observe that R (and *TR*) have natural uses within parallelization methods. As already noted, the processes of Approaches 1 and 2 in Sect. 2 can be carried out in parallel. In addition, the generation of the reference set R can serve as a centralized operation for periodically assembling information from subordinate methods, and using this information to compose the current sets F', R_o and E_o (as by updating information previously used to compose these sets).

Parallel exploitation of R

1. Assign each element of the set R to a different search process, to be carried out in parallel.
2. Each search generates candidate points to compose the sets F', R_o and E_o, and after a selected number of iteration passes these points to a master process.
3. The master process produces the sets F', R_o and E_o, and generates a new R. Then Step 1 is repeated, until a global termination criterion (such as an iteration limit) is reached.

We additionally observe that the construction of R can be based on rules other than the max (min) rule. For example, drawing further on scatter search ideas, R can be populated by generating linear combinations of elements in F'. If F' is built from edge extensions to include points beyond those spanned by the best points previously encountered, then the scatter search procedure can restrict attention to using convex combinations (having the advantage of yielding new points that lie in feasible convex regions if the parents from F' lie is such regions).

5 A complementary (outside-in) strategy

We turn to considering a strategy that embodies a somewhat different perspective than that of previous sections. The process of conditional directional branching, as exploited in Approach 2 of Sect. 2.2, may be viewed as an alternating "inside-out/outside-in" procedure, where directional rounding by Approach 1 moves from x^0 within the feasible region to points x'' that may lie outside F, followed by branching that returns to generate a new x^0. However, each new wave progresses forward from a new x^0, which anchors the process on the inside-out portion of the strategy. We now focus on a complementary approach that draws upon ideas of parametric branch and bound and parametric tabu search (Glover, 1978, 2006), and of a related feasibility net approach (Fischetti et al. 2006).

In adaptation to the present setting, the approach may be sketched in overview as follows. Upon generating the set of points X'' in Step 1 of Approach 2, instead of immediately proceeding to Step 2 of Approach 2 to branch from the current LP optimal vertex, the method selects a point $x'' \in X''$ (which will be infeasible relative to the current F that includes the objective function constraint), and works its way from the infeasible region back to LP feasibility. The guiding criterion for thus reaching a point of F is determined not by original the objective function $x_o = cx$, but by *a targeting objective function* $u_o = dx$ that is oriented to make x'' a preferred solution. In particular, d is defined relative to x'' so that x'' would be optimal if it were LP feasible. The resulting optimal LP vertex, which we denote by $\tilde{x}$, is an LP proxy for x'', which is a close to x'' as possible according the measurement of distance provided by the objective function u_o. To achieve an appropriate proximity between the solutions, and to cause $\tilde{x}$ to yield useful information about how x'' should be modified, we additionally modify the feasible region F by introducing provisional bounds by reference to x''.

Finally, relative to the new point $\tilde{x}$, we perform directional rounding to identify additional points x'' as candidates to belong to X''. Edges and centers are generated exactly as specified earlier by reference to x^0, except that the search extends rays into F from different directions when the extreme points are generated from the u_o objective rather than the x_o objective. Additionally, post-optimality analysis of the LP solution $\tilde{x}$ discloses information about how x'' may be improved for inclusion in X''. The process may then return directly to Approach 2, to initiate a branching step as previously described, or may select another element x'' from X'' to re-iterate the outside-in approach.

5.1 Details of the outside-in procedure

Starting from a current LP vertex x^0 and a selected point $x'' \in X''$, we augment the constraints of F to include the following inequalities:

$$\begin{aligned} x_j &\le x''_j \quad \text{if } x^0_j < x''_j, \\ x_j &\ge x''_j \quad \text{if } x^0_j > x''_j. \end{aligned}$$

The tied case, for $x^0_j = x''_j$, is resolved by choosing $x_j \le x''_j$ or $x_j \ge x''_j$ according to whether x''_j is closer to U_j or 0, respectively. This case can also be resolved by reference to most frequently encountered values for x_j or by reference to penalty calculations for imposing $x_j \le x''_j$ or $x_j \ge x''_j$. Ties that remain can be broken arbitrarily.

This modification of F, which is redundant for a 0–1 variable x_j, assures that x^0 remains feasible relative to the new region defined by F, and hence that this region is non-empty. Let $N = \{1, \ldots, n\}$ denote the index set for x, and let $N(+)$ and $N(-)$ respectively be the subsets of N associated with the inequalities $x_j \le x''_j$ and $x_j \ge x''_j$. We choose the vector d, to create the targeting objective "Maximize $u_o = dx$," by stipulating that

$$\begin{aligned} d_j &> 0 \quad \text{if } j \in N(+), \\ d_j &< 0 \quad \text{if } j \in N(-). \end{aligned}$$

In the simplest case, we may choose each d_j to be 1 or -1, or we may bias its value to reflect an influence by the coefficient c_j (e.g., by adding or subtracting some positive quantity from c_j). We denote the LP problem based on d and the inequalities $x_j \leq x''_j$ and $x_j \geq x''_j$ by LP(d, x'').

The foregoing construction guarantees that the solution $\tilde{x}$ to LP(d, x'') would be the same as the target solution x'' if x'' were LP feasible. Since this is not the case, and hence $\tilde{x} \neq x''$, we adopt a strategy from parametric tabu search (Glover 1994) that identifies and exploits *resisting variables*, i.e., variables that fail to achieve the assignment $x_j = x''_j$. Resisting variables are candidates to be assigned values different from those in the solution x'', thus suggesting that a subset of these variables that exhibit the greatest resistance may preferably be rounded to values different from their x''_j targets.

Resistance may be measured in several ways. The simplest is to define the resistance of x_j as $|\tilde{x}_j - x''_j|$, the amount by which the assignment $x_j = \tilde{x}_j$ deviates from the targeted x''_j value. Generally better is to weight the values $|\tilde{x}_j - x''_j|$ according to scale factors, and especially according to the frequency that a variable resists a particular target value (at various levels of resistance) throughout a history of solving associated LP problems. This history can be usefully supplemented by identifying additional *local resistances* using the strategy of modifying the d vector while maintaining its stipulated sign conditions. This approach can then identify additional optimal solutions $\tilde{x}$ that resist the assignment $x = x''$, and the different degrees of resistance by various x_j variables in these solutions. Particularly relevant is the case where some variables cease to be resisting in some of these solutions.

Since the only change that occurs in LP(d, x'') in this approach is to modify its objective function, the solutions $\tilde{x}$ can be obtained by post-optimizing with the primal simplex method. In fact, post-optimality analysis can be used to identify exactly the extreme points adjacent to a given $\tilde{x}$ that will become optimal by admissible changes in d, and the resistances at these points can be determined at once without the effort of a full pivot step to reach such points. This provides a fast way to expand the information available about resistance frequencies, and may be accompanied by pivoting to one or more adjacent solutions to repeat the process.

Three special considerations should be noted regarding the handling of resistances in general. First, if the reduced cost for the slack variable of the inequality $x_j \leq x''_j$ or $x_j \geq x''_j$ in an optimal basis for LP(d, x'') is small (close to 0), then this also indicates that the solution resists assigning x_j the value x''_j; we call this *quasi-resistance*. This reference to reduced costs is motivated by the fact that ordinary resistance, where x_j differs from x''_j, occurs when the reduced cost is 0. Consequently, quasi-resistance should be taken into account in identifying variables that should be rounded differently, and in tracking the history of variables that resist particular bounds.

Second, a slack variable whose reduced cost is greater than or equal to $|d_j|$ identifies a situation where the associated inequality $x_j \leq x''_j$ or $x_j \geq x''_j$ is subordinate to the other targeting inequalities, in the sense that the solution to LP(d, x'') would assign $x_j = x''_j$ even if $d_j = 0$. In particular, if the reduced cost is greater than $|d_j|$ then x_j may be considered an *anti-resisting* variable, because it not only accepts its target value, but tries to go beyond this value in the direction away from x^0_j. This is an important consideration, because by permitting an anti-resisting variable to move

in the direction it prefers (by allowing x''_j to increase or decrease accordingly), other variables that are currently resisting may become non-resisting, or less strongly resisting. This phenomenon can occur, for example, where an attempt to round one variable up compels another to also move upward. If the latter is blocked by its upper bound the first variable may appear to resist its targeted value, when in fact it is the bound on the second variable that prevents the target from being achieved. In the case of 0–1 variables, the bound on the second variable can not be changed to allow it to move farther, but in such a situation the resistance of the first variable may be deceptive, in that some other resisting variable may be the key to allowing it to change. Strategies to uncover such dependencies among resisting and anti-resisting variables can be highly relevant to identifying better target solutions.

Third, it is appropriate to account for fact that, relative to the space of integer solutions, and in particular relative to the inequalities added to F that define $\mathrm{LP}(d, x'')$, the integer point that is most centrally located in the cone spanned by these inequalities is the *complement* of x'' whose coordinates are given by

$$x_j = x''_j - 1 \quad \text{if } j \in N(+),$$
$$x_j = x''_j + 1 \quad \text{if } j \in N(-).$$

In the case of 0–1 variables, this definition is equivalent to the one that defines the complement to be given by setting $x_j = 1 - x''_j$ for each $j \in N$.

Just as we are motivated to extend edges into central regions to increase the likelihood of finding feasible integer solutions by directional rounding, we are motivated to examine the indicated complement point in the region determined by the inequalities added to yield $\mathrm{LP}(d, x'')$. However, still more relevant is to examine a *semi-complement* of x'' whose coordinates differ from the preceding complement by retaining the integer value assignment of variables that receive the same value in both x^0 and x''; that is, the point that differs from the complement by stipulating that

$$x_j = x''_j \quad \text{if } x^0_j = x''_j.$$

Accordingly, within the context of the outside-in approach, when the problem $\mathrm{LP}(d, x'')$ is created, we also examine the semi-complement of x'' for feasibility, and we may likewise evaluate it as a candidate to be included within the set X''. Of course, semi-complements can be treated in this fashion independent of the outside-in approach. In Sect. 5 we will consider additional mechanisms for isolating points that are centrally located relative to useful criteria, and that deserve to be examined for feasibility and possible membership in X''.

As previously observed, upon solving the problem $\mathrm{LP}(d, x'')$, the outside-in approach may now continue by generating directionally rounded points, and associated edges and centers, relative to the solution $\tilde{x}$ for $\mathrm{LP}(d, x'')$ in the same manner that such elements are generated relative to x^0. That is, each new $\tilde{x}$ is treated "as if" it were x^0, to produce yet another $\tilde{x}$. We note that edges rooted at $\tilde{x}$ extend into the space F from a different direction than from the point(s) x^0, thereby allowing the space to be searched in a different manner than in the approach described in preceding sections. The generation of such edges results in the creation of new points x'' to be considered for inclusion in X'' (and as candidates for target points for the iterated

outside-in process), and after a chosen number of iterations, the method may be organized to return to the branching process of Approach 2. Alternatively, upon solving LP(d, x''), the current target solution x'' can be directly modified by accounting for resisting and anti-resisting variables, thereby producing a changed target solution that can be a basis for iterating the process. The latter approach more nearly resembles that of parametric tabu search, and can readily be controlled by TS memory.

5.2 Illustration of the outside-in procedure

A graphical illustration of the outside-in approach is given in Fig. 3. In this case, the problem is a 0–1 problem, whose relevant integer points are restricted to vertices of the unit cube: $(0, 0)$, $(1, 0)$, $(1, 1)$ and $(0, 1)$. Among these only the point $(0, 1)$ lies within the feasible region. The initial LP solution x^0 appears in the lower left corner. As is true of all 0–1 problems, each focal point on the ray leading from x^0 (shown as the dotted line meeting x^0 in Fig. 3) will round directionally to the same point, which in this case is the infeasible point $x'' = (1, 1)$, depicted as $x''(A)$ in the diagram. We

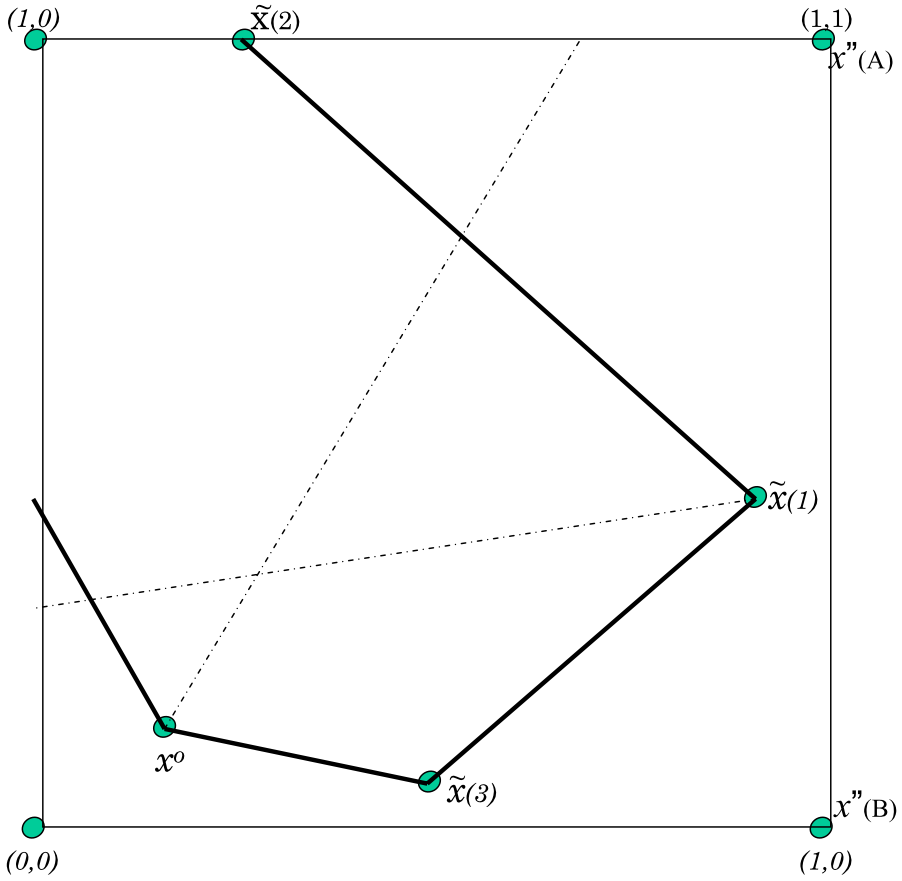


Fig. 3

do not bother to show the center of the edge produced by the dotted line, but note that it may be recorded for use as in previous examples.

The targeting problem LP(x'', d) associated with the point (1, 1), given that $x_j^0 < x_j''$ for $j = 1, 2$, is created by adding the inequalities $x_1 \leq 1$ and $x_2 \leq 1$ to the original LP problem. Such inequalities do not need to be explicitly introduced since they are already included in the 0–1 problem formulation. The semi-complement of (1, 1) (which in this case is simply the 0–1 complement) is (0, 0), and is infeasible. By the rules for generating the vector d for the problem LP(x'', d), the components d_1 and d_2 should be positive, and we can choose $d_1 = d_2 = 1$ to create the targeting objective "Maximize $u_o = x_1 + x_2$." Graphically, in two dimensions all acceptable objective functions that target a given x'' can be pictured as having slopes (when passing through x'') that range between the two coordinate lines defining x'', without extending into the interior of the unit cube. The objective function hyperplane when d_1 and d_2 both have an absolute value of 1 is the one whose slope is exactly in the middle of this range, creating 45° angles with the coordinate hyperplanes. (In general, an objective function optimized at any given vertex can be visualized as a hyperplane containing the vertex that does not cut into the cone that defines the vertex.) Consequently, the resulting optimal solution $\tilde{x}$ to the targeting problem appears as the rightmost point of F illustrated in the diagram, which is labeled $\tilde{x}(1)$.

Both x_1 and x_2 are resisting variables in the solution $x = \tilde{x}(1)$ since neither achieves its targeted value of 1. Using the simplest way to generate additional local resistances, which consists of examining adjacent extreme points that qualify as optimal for other d vectors defined by reference to the same target x'', we observe that the solution $x = \tilde{x}(2)$ also belongs to the set of optima that target $x''(A)$. In this case, x_1 resists its targeted value of 1 even more strongly, but x_2 no longer exhibits a resistance to its targeted value. If this limited preliminary history is used to modify x'' for the purpose of creating an alternative trial solution, the repeated resistance of x_1 to the value 1 leads to changing x_1'' from 1 to 0, which produces the feasible solution $x = (0, 1)$.

At this point, the method may proceed to Step 2 of Approach 2 to perform one or more iterations of its branching process, to resume the outside-in approach at some selected future iteration by intervening at the point where the method would pass from Step 1 to Step 2. We continue the illustration of the outside-in approach, however, by first examining the option of treating $\tilde{x}$ in the same way as x^0. Among the two $\tilde{x}$ solutions identified thus far, we choose $\tilde{x}(1)$ as a basis for illustration, and create a ray leading from it as shown by the dotted line meeting $\tilde{x}(1)$ in Fig. 3. Directional rounding by reference to focal points on this line yields the infeasible point $x'' = (0, 0)$, already encountered as the semi-complement of the previous $x'' = x''(A)$, but not yet used to generate a targeting problem LP(x'', d). The creation of such a targeting problem would lead to a solution $\tilde{x}$ that is the same as the original LP solution x^0, and the approach would now loop. This provides a natural juncture to proceed to the branching process at Step 2 of Approach 2, unless the outside-in procedure is guided by *TS* memory, in which case the approach can continue by altering components of subsequent directionally rounded solutions to avoid repetition.

The other option for continuing from the solution $\tilde{x} = \tilde{x}(1)$, is to establish a new targeting solution x'' based on the identity of the current resisting variables rather

than identifying the next targeting solution by extending a ray from $\tilde{x}$ and using directional rounding. To provide greater scope for the illustration, we do not use the information already generated about the relevance of setting x_1 to 0 to influence the determination of resistances. At the point $x = \tilde{x}(1)$, the variable x_2 has the greatest resistance (measured in terms of its distance from its target value of 1), and if we modify the targeted value for this variable, we get the point $x'' = (1, 0)$, denoted as $x''(B)$ in Fig. 3. This point is infeasible, but its semi-complement is the point $(0, 1)$, which again takes us to the unique feasible integer solution of this example.

To generate the next problem LP(d, x''), the vector d must be selected so that $d_1 > 0$ and $d_2 < 0$, and again we make the simplest choice by selecting $d_1 = 1$ and $d_2 = -1$. The solution to the resulting LP(d, x'') is exactly the same point $x = \tilde{x}(1)$ obtained as a solution when targeting $x''(A)$ instead of $x''(B)$. This outcome can be identified without having to perform any iterations to solve LP(d, x''). Specifically, the primal simplex method can be used to post-optimize, since the optimal solution to a given instance of LP(d, x'') is a feasible extreme point for the next instance. Consequently, the fact that $\tilde{x} = \tilde{x}(1)$ remains optimal for this new targeting problem will be determined instantly.

The next step of determining additional local resistances by the device of examining adjacent extreme points (while targeting the same solution $x''(B)$) leads to the solution $x = \tilde{x}(3)$ in Fig. 3. If the approach of identifying resisting variables is accompanied by the strategy of extending rays for directional rounding, the outcome of the latter procedure at this juncture would yield (almost certainly) the feasible point $x = (0, 1)$ once more.

Relative to the targeting solution $x''(B)$, the resisting variables in $\tilde{x}(1)$ are the same as those in $\tilde{x}(3)$. That is, in both solutions, x_1 resists its targeted value of 1, which is the same target it had relative to $x''(A)$, while x_2 resists its targeted value of 0, which differs from its target relative to $x''(A)$. This outcome expands the historical information about resistances, increasing the tally of x_1's resistance to the value 1, and hence reinforcing the presumed merit of investigating the assignment $x_1 = 0$. Also, given that x_2 has now on two occasions resisted the value 0, but has only once resisted the value 1 (and has once accepted that value), we may be motivated to examine the assignment $x_2 = 1$. This once more leads us to the desired solution $x = (0, 1)$.

We emphasize that in higher dimensions relative frequencies will normally require a larger set of trials before their differences become meaningful. Similarly, in general it should be emphasized that our illustration based on a two-dimensional example, while constructed to show some of the interesting situations that can arise in executing an outside-in procedure, must necessarily fall short in conveying the types of outcomes that can occur in higher dimensions.

One additional feature of the outside-in approach for the general case deserves to be mentioned. The variables x_j that are non-basic in the original LP basis yielding x^0, and which become basic at positive values in the various solutions $\tilde{x}$, identify variables whose columns D_j are good candidates for receiving positive weights to produce the vector D for directional rounding relative to x^0. Again, frequency measures are useful for selecting more promising candidates.

6 Additional refinements

We now consider additional strategies for identifying variables whose columns should be used to generate D, and elaborate other elements that can be exploited throughout the execution of Approach 2 and the complementary outside-in process. These include special ways to take advantage of memory, and advanced procedures for generating edges and centers, to be exploited by the processes of the preceding sections.

6.1 Marrying the outside-in approach with adjacent extreme point search

Within the context of the outside-in approach, we have already noted the potential value of examining adjacent extreme points, or a path of successive adjacent extreme points, to identify local resistances relative to a given targeting solution. By extension, we also observed that the history of resistances can be relevant to identifying values to be assigned to the variables. In this vein, the outside-in procedure and its resistance information are well-suited to be combined with a memory-based adjacent extreme point search procedure for 0–1 problems (see, e.g., Lokketangen and Glover 1995; Eckstein and Nediak 2001). The information concerning resistances can particularly be used to modify the choice rules of the adjacent extreme point search. The structure of the feasible region in Fig. 3 hints at the usefulness of such a modification. An adjacent extreme point method that starts at the initial LP point x^0 might already find itself at a local optimum, relative to reasonable rules for evaluating improvement. Drawing upon the outside-in approach as an avenue for continuing, and also as a means of identifying resistances, could add flexibility to an adjacent extreme point process and provide information for improving its evaluations of points to be visited next. At the same time, in reverse, the exploitation of periodic adjacent extreme point trajectories could prove useful for augmenting the outside-in procedure.

6.2 Frequency-based conditional memory and provisional inequalities

Frequency-based memory as used in tabu search is relevant not only to resistance measures, but is also appropriate for use in connection with the conditional directed rounding process in Approach 2. Within this setting, frequency-based memory can be given a useful conditional form. As previously observed, the use of branching operations in conditional directional rounding provides a way to combat the limitation of moving variables simultaneously in certain directions. The potential defect in all-at-once rounding is that the movement of a variable in a particular direction may exert an influence through the problem constraints that should drive another variable in a direction contrary to the prescription of an all-at-once rounding process.

Maintaining a frequency memory of *conditional changes* can provide a supplement to the branching approach of conditional directed rounding, by offering a way to analyze and anticipate consequences of certain rounding options before making recourse to the more computationally intensive effort of introducing branching operations. This *frequency-based conditional memory* may be structured as follows.

Let Change(j) denote a frequency memory for the integer variable x_j that counts the number of occurrences of primal feasible LP solutions (new x^0 solutions) generated by branching in which x_j achieved an integer value that differed from the value

assigned to x_j in the preceding LP solution. (In the setting of neighborhood search, Change(j) may be viewed as counting the number of times a variable or a solution attribute changed to receive an admissible value as such values are defined relative to moves performed in the neighborhood space.) Accompanying this, two frequency records Same(j, k) and Differ(j, k) are maintained that give the number of times among those in which x_j changed to receive an integer value (causing Change(j) to be incremented) that the variable x_k changed in the same direction as x_j and in opposite direction from x_j, respectively. It is not necessary that x_k receive an integer value by its change. These frequency records can then be a basis for a fast conditional rounding operation, where a high conditional frequency compels corresponding changes to be made in the rounding process.

More complete and more useful information results by keeping expanded frequency records Up(j) and Down(j), identifying how many times x_j increased or decreased to reach an integer value, accompanied by records UpUp(j, k), UpDown(j, k), DownUp(j, k), DownDown(j, k), identifying the number of times when both x_j and x_k increased together, when x_j increased but x_k decreased, and so forth, in each case limiting consideration to the case where x_j attained an integer value. (If attention is further restricted to the case where x_k receives an integer value, then DownUp(j, k) can be omitted, since it would be the same as UpDown(k, j).)

Variations on these records can be used to create different types of analysis of conditional assignments. For example, a useful alternative is to treat values of variables *as if* they are integer when they lie within a chosen distance of being integer. In this case, the direction of movement that resulted in these values may be considered less important than residing at these values, i.e., within a specified proximity of being integer. Records produced in this fashion that disclose highly consistent relationships can form the basis for aggregate (brazen branching) rules as discussed in (1c).

In the setting of 0–1 problems, the arrays could use the names "One" and "Zero" in place of "Up" and "Down" (i.e., OneOne(j, k), OneZero(j, k), etc.), to identify when x_j and x_k have received various combinations of assignments in the same solutions.

This type of memory can be used to generate *provisional inequalities* to help guide the search process, according to relationships that are strongly compatible with the frequencies recorded in the arrays. For example, if One(j) = OneZero(j, k) for some set of variables x_k, $k \in K$, then one may adjoin the provisional inequality $x_j + \sum(x_k\colon\ k \in K) \leq 1$. After using this inequality for some period to focus the search, then one may instead adjoin the complementary inequality $x_j + \sum(x_k\colon\ k \in K) \geq 2$ to seek different solutions. Similarly, if there is a set K such that Zero(j) = ZeroZero(j, k) for $k \in K$, then one may adjoin the provisional inequality $x_j \geq \sum(x_k\colon\ k \in K)$, followed after a period by introducing instead the complementary inequality $\sum(x_k\colon\ k \in K) - x_j \geq 1$. Relationships suggesting the merit of a variety of other types of provisional inequalities, such as $\sum(x_k\colon\ k \in K) \geq p$ or $\sum(x_k\colon\ k \in K) \leq q$, can also readily be identified. It is not necessary that the conditional frequency memory support such inequalities as invariably as suggested by the preceding examples, since the purpose of provisional inequalities is to take advantage of tendencies rather than invariant conditions to constrain the search in various ways for chosen periods of time. The introduction of such inequalities and their complements can be managed by a TS adaptive memory process, as in the case of managing the branching inequalities within Approach 2.

Frequency-based conditional memory can also be used to identify coordinated movements of variables, where variables in certain subsets move in consistent directions relative to each other. Again, attention need not be restricted to invariant behavior, but may be permitted to embrace patterns that occur with a specified frequency. The outcomes of such an analysis can be used not only for creating provisional inequalities, but also for creating a supplementary directional rounding process within Approach 1, where additional candidates for X'' are produced by modifying the rounding process to reflect the coordinated movements.

For problems too large to permit all variables to be tracked by frequency-based conditional memory in the manner indicated, attention can be focused on a subset of critical variables. The records that underlie this memory require an initial period of computational effort to build, but then can be used without resorting to further updating except at selected intervals. Also, as in TS intensification processes, it may be more important to maintain such memory relative to subsets of good solutions, by evaluations that permit infeasible (although nearly feasible) solutions to be admitted to such subsets, than to maintain the memory relative to a somewhat larger collection of solutions that include less attractive members.

6.3 Model embedded memory from inequalities

Inequalities can be added to F not only in a provisional fashion as discussed in Sect. 5.2, but also in an irrevocable (globally applicable) fashion based on identifying implications directly generated by solving LP problems during the branching process. The role of these latter inequalities is to eliminate portions of the feasible region previously examined and determined to be unproductive, and may be viewed as providing a form of model embedded memory to supplement the use of other forms of adaptive memory.

These inequalities can be produced as follows. When a current set of branches results in an infeasible [LP] problem, or yields a feasible integer solution that is to be excluded from future consideration, the first step is to remove all dependent branches, together with all independent branches that are currently non-binding, and then identify the remaining set of independent branches, which we represent in the form $x_j \geq v_j$, $j \in J^+$ and $x_j \leq v_j$, $j \in J^-$. This remaining collection of branches is then rendered infeasible for the problem [LP] by adding to F the inequality

$$\sum((v_j - x_j):\ j \in J^+) + \sum((x_j - v_j):\ j \in J^-) \geq 1. \qquad \text{(A)}$$

The inequality (A) can also be used to exploit dependency relationships. If any member of a collection of dependent branches is complemented, it will similarly result in a system that violates feasibility or a bound on x_o. Consequently, we represent a dependent branch in the form $x_j \leq v_j - 1$ or $x_j \geq v_j + 1$, causing its complementary branch to acquire the form $x_j \geq v_j$ or $x_j \leq v_j$, respectively. This complement of the dependent branch may then be included with its upstream independent branches to yield an inequality that has precisely the form of (A). All members of a set of dependent branches can be incorporated simultaneously into (A) in this manner, but the outcome is stronger if each dependent branch appears by itself in a separate instance of (A).

Variants of this inequality, without making reference to independent and dependent branches in the manner employed here, have been previously proposed for producing 0–1 "short hot starts" for branch and bound (Spielberg and Guignard 2000; Guignard and Spielberg 2003) and for selecting subsets of variables to hold fixed in adaptive memory projection methods (Glover 2005).

6.4 Handling primal degeneracy

A special case for generating D as a convex combination of non-basic columns of the LP basis representation of x^0 occurs where the basis representation is primal degenerate, as a result of the fact that x^0 is the intersection of more than the minimum number of hyperplanes needed to define an extreme point of F. In this situation some of the non-basic variables will typically not be associated with extreme rays of the LP cone, but rather will correspond to rays that extend into infeasible space for positive values of their associated non-basic variables.

To determine all of the true extreme rays of the LP cone requires the execution of degenerate pivots to identify alternative basis representations of x^0. A simpler alternative is to choose non-basic columns to generate D by first selecting columns whose rays do not begin by extending into infeasible space, or that create infeasibility only for a few constraints, offsetting these latter choices by selecting other columns that are feasible for these constraints. Computationally, this can be done by noting that a ray that extends into feasible space is one whose column has non-negative coefficients in the positions where the LP representation is primal degenerate (i.e., the "updated column of constants" has a 0 coefficient). Likewise, a negative coefficient in one of these positions identifies a constraint that is violated when the ray is extended. The vector D can then can be generated in two steps, first by creating a vector D^1 as a convex combination of the columns for rays that extend into feasible space. The second step then creates a convex combination of D^1 with the other columns, subject to giving a sufficient weight to D^1 to yield a non-negative coefficient for each primal degenerate position in the final D vector.

In situations where it proves difficult to produce D vectors that exhibit the properties indicated, it is still possible to use a collection of D vectors generated without reference such refinements to provide line segments for directional rounding, and hence to provide points x'' to be checked for ILP feasibility and to underlie the choice rules embedded in Approach 2.

6.5 Identifying non-basic columns to receive positive weights

A particularly important issue is that of selecting an appropriate set of non-basic columns to receive positive weights in convex combinations that define D vectors. The significance of this issue derives from the fact that there may be many x variables that are non-basic, yet only a small subset of them should be positive in an optimal solution. Yet if columns for other non-basic variables are used to generate D, these columns will create rays that move in the wrong direction from x^0, and in particular will produce directionally rounded solutions that assign positive values to each of the associated "incorrect" x variables. A limited number of errors may not create an adverse effect, because in Approach 2 we are interested in the implied movement

only of the fractional variables, which are basic, and the directionally rounded solutions may still give useful clues about the directions that the fractional basic variables should move. This holds particularly for the type of choice rule used in Step 2 of the method (and in the variant described in (1) above) that selects a fractional variable for branching that exhibits the greatest consistency in the direction it moves.

Consequently, we propose to step outside the directional rounding process to gather additional relevant information by creating a feedback loop from the branching operations to identify a promising set of non-basic columns for generating D. This is another natural application for frequency-based memory, by examining the effect of tentatively introducing several different branching operations at each step, and keeping track of the non-basic variables that are most often driven positive because of these operations. The "higher frequency positive value" non-basic variables become candidates for determining the collection that should play a role in generating D vectors. A limit may be set on the number of pivots allowed for dual re-optimization when tracing the consequences of a branching step, in the case where a new primal feasible solution is not obtained at once.

If both members of a pair of complementary branches drive a non-basic variable to be positive, then that is a compelling endorsement for giving the variable a role in creating D vectors. In general, branches can be rated according to a selected criterion of desirability, and this rating can in turn provide a bias to the frequency measure used to evaluate the resulting non-basic variables that become positive. After a chosen number of iterations of Approach 2, the approach can then be re-started, this time choosing to generate D vectors based on the recorded frequency information obtained during the branching steps.

A closely related option is to perform a look-ahead analysis before implementing Step 1 of Approach 2. In this case, the branching operations are performed and evaluated by a criterion separate from the one that derives from relying on information from Approach 1. The resulting frequency records are used to generate D vectors at once, more strictly limiting the number of options examined and the number of iterations performed in the branching process. Approach 1 can also be used to aid in the evaluation of branches to carry out such a procedure, though at some risk of generating candidates for creating D vectors that confirm its original choices of such candidates. The look-ahead analysis can be re-iterated after various branches are formally selected and imposed within Approach 2, to aid the determination of D vectors not only relative to the original x^0 solution but also relative to new LP solutions produced by branching. In this case the number of branching operations examined and executed during look-ahead steps may be more tightly curtailed to further reduce overall computational effort.

This type of approach merges conveniently with the use of frequency-based conditional memory discussed in Sect. 6.2 above, by giving a platform for generating and exploiting such memory.

6.6 Advanced determination of derived centers and edges

A more advanced procedure can be used to generate centers of sub-regions, which avoids the potential inconvenience caused by primal degeneracy and also has other advantages. The basis of this approach is to use linear programming to identify a ray

from x^0 that is strategically constructed to pass through a central portion of the feasible region. We draw on the fact that x^0 will remain optimal for any objective function that assigns non-negative reduced costs to the current non-basic variables. A "balanced" objective function of this type results by giving each non-basic variable the same positive reduced cost, which we arbitrarily take to be 1. The balanced objective can then be interpreted as that of minimizing the sum of these non-basic variables. (Other positive reduced costs can be used to achieve various definitions of balance, as by choosing the costs based on scaling the current non-basic columns.)

Replacing the balanced objective by a "counter-balanced" objective, which consists maximizing rather than minimizing the sum of these non-basic variables (defined by reference to reduced costs of -1 instead of 1), we obtain a boundary point, or more precisely an extreme point, that we call a *projected opposite* of x^0. The projected opposite point, which we denote as the boundary point x^{Bo}, can be found by re-optimizing with the primal simplex method, starting directly from the basis that yields x^0 (or from any other primal feasible basis, upon updating the objective appropriately). Upon finding x^{Bo} we have an edge $[x^0, x^{Bo}]$ whose midpoint x^{co} is a candidate for being centrally located within F.

The foregoing process can be extended to generate additional rays to traverse a central part of the space, and hence to generate additional associated edges and centers. One way to do this is to keep a record, as the primal simplex method moves from x^0 to x^{Bo}, that identifies an extreme point solution $x^{\#}$ encountered approximately "half-way" through the trajectory, such as a solution for which roughly half of the reduced costs become non-negative. (Several plausible candidates for $x^{\#}$ can be saved and a best choice can be identified after reaching $x^{B\#}$.) The balanced objective function for $x^{\#}$ that consists of assigning values of 1 to each of its reduced costs can also be recorded. Upon reaching the projected opposite x^{Bo} for x^0, the balanced objective function for $x^{\#}$ is updated and replaced by its negative, yielding the corresponding counter-balanced objective for this point. The primal simplex method can then continue from x^{Bo} to find the new boundary point $x^{B\#}$ that represents the projected opposite of $x^{\#}$.

The centers x^{co} and $x^{c\#}$ of the edges $[x^0, x^{Bo}]$ and $[x^{\#}, x^{B\#}]$ can thus be identified, yielding candidates for focal points, and a new line passing through x^{co} and $x^{c\#}$ can be constructed to identify the intersecting points $x^{B\#o}$ and $x^{Bo\#}$ with the boundary of F, to produce a further edge $[x^{B\#o}, x^{Bo\#}]$ whose center likewise provides a candidate for a focal point.

Before describing how to turn this process into an iterated method for generating additional points and projected opposites, an illustration of the approach at the present stage is shown in Fig. 4. Here, starting from x^0 at the base of the diagram, a counter-balanced objective causes the primal simplex method to locate the projected opposite point x^{Bo} shown at the top of the diagram. We suppose that the route followed by the simplex algorithm to locate x^{Bo} travels through extreme points on the left side of the diagram, which will enable the method to identify and record a "half-way" point such as the point $x^{\#}$ indicated in the diagram. Upon reaching x^{Bo}, the appropriate counter-balanced objective for $x^{\#}$ is activated, and the method proceeds from x^{Bo} to locate the projected opposite $x^{B\#}$ of $x^{\#}$, shown at the right of the diagram. The edges $[x^0, x^{Bo}]$ and $[x^{\#}, x^{B\#}]$ are generated, as shown by the crossing bold lines within the feasible region F, together with their centers x^{co} and $x^{c\#}$. Joining these

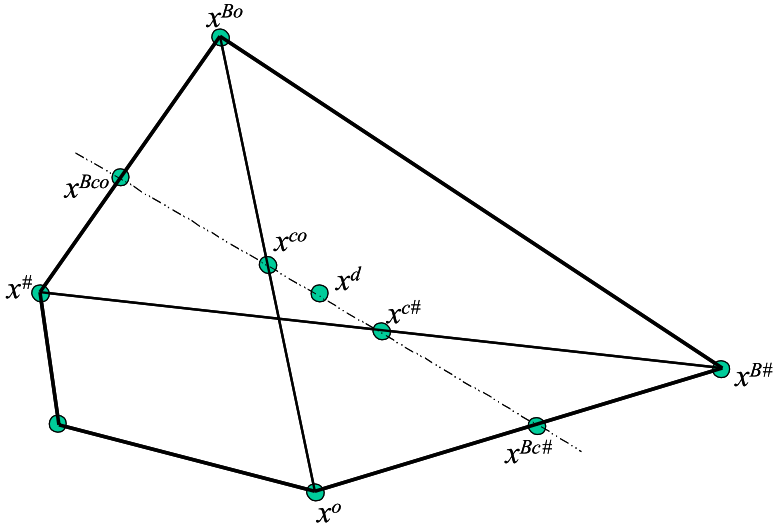


Fig. 4

last two points by a line gives rise to the edge $[x^{Bco}, x^{Bc\#}]$, depicted by the dotted line within F, whose center is denoted x^d. The center points x^{co}, $x^{c\#}$ and x^d can be included among candidates for focal points for directional rounding.

Unlike the situation illustrated earlier in Figs. 1 and 2, the situation portrayed in Fig. 4 will not normally involve a 2-dimensional cross section of n-space, and hence the 2-dimensional representation is somewhat deceptive. In general, assuming the extreme points of F are not co-planar, the edges $[x^0, x^{Bo}]$ and $[x^{\#}, x^{B\#}]$ will typically not lie in the same plane but will rather describe a 3-dimensional space, and hence the edge $[x^{Bco}, x^{Bc\#}]$ will extend through this space as well. However, even a 2-dimensional representation can disclose the fact that more than one extreme point can map into the same projected opposite, as in Fig. 4 where both of the two left-most extreme points in the diagram map into the same point $x^{B\#}$ on the right. This condition also implies that the mapping is not symmetric; i.e., the projected opposite of a projected opposite may not be the original point.

The process illustrated in Fig. 4 can be extended to generate additional points and their projected opposites in a similar manner. We mention two variants. In the first, the method records a "next" $x^{\#}$ extreme point, $x^{\#1}$, that is estimated to be "half-way" along the path from x^{Bo} to $x^{B\#}$ (assuming an intermediate point exists on this path). The balanced objective for $x^{\#1}$ can then be updated upon reaching the point $x^{B\#}$, and the process continues, similarly finding a point $x^{\#2}$ en route to finding the projected opposite $x^{B\#1}$ of $x^{\#1}$, and so forth. In the second variant, several $x^{\#}$ points can be identified at various intervals along the original path from x^0 to its projected opposite, and the projected opposites for each of these can be determined in succession after finding the projected opposite for x^0.

Once the edges $[x^{\#}, x^{B\#}]$ joining such points $x^{\#}$ and their projected opposites $x^{B\#}$ are generated, their midpoints $x^{c\#}$ together with the center x^{co} can be weighted in various combinations to produce derived centers of associated sub-regions, or these

midpoints can all be combined together to produce a candidate for a single global center. (Alternatively, convex combinations can be applied directly to the points x^0, x^{Bo} and the sets of points $x^{\#}$ and $x^{B\#}$.) The derived centers as well as the edge midpoints can be used to guide the creation of D vectors from the original x^0, noting that the D vector for a given point x' can be given simply by $D = x' - x^0$.

These D vectors based on derived centers can also be used to combine with D vectors that are instead generated by a simpler approach of forming convex combinations of LP non-basic columns associated with x^0. The process of deriving projected opposite points can of course be applied iteratively within Approach 2 by reference to new x^0 points obtained at selected stages of the branching processes.

6.7 Derived edges and centers from semi-counter-balanced objectives

The linear programming procedure of Sect. 6.6 for creating derived edges and centers can be modified to produce points that lie within special sub-regions of the feasible space, anticipating that it may be of value to give attention to certain regions that may not "centrally located." The modification consists of replacing the counter-balanced objective function by an alternative objective designed to keep a fairly significant portion of current non-basic x variables at a 0 value, following the motivation indicated in Sects. 2.1 and 5.3. (Recall that in all such processes, non-basic variables that assign bounds to x variables are to be treated in the same way as x variables.) The resulting objective, which we call a *semi-counter-balanced objective*, arises by giving the selected non-basic variables a reduced cost of 1, while giving remaining non-basic variables a reduced cost of -1 as in the ordinary counter-balanced objective. The point obtained by optimizing the semi-counter-balanced objective will be called a *projected semi-opposite* point.

A useful feature of creating a semi-counter-balanced objective is that the process of re-optimizing can yield useful information about the subset of x variables assigned a reduced cost of 1. Such a coefficient in the objective function exerts an influence on the associated variable that would keep the variable non-basic and hence equal to 0 if the problem constraints were sufficiently relaxed. Consequently, we term such a variable that instead receives a positive value in the resulting LP solution to be a *resisting variable*, since it moves in a direction counter to that encouraged by its positive reduced cost. By their "preference" for a positive value assignment, resisting variables provide attractive candidates to be included among the non-basic variables associated with x^0 whose columns define D. It may be noted that the influence of resisting variables will also automatically be reflected in creating convex combinations of the semi-opposite LP extreme point in which these variables receive positive values.

A semi-counter-balanced objective can be further modified to yield semi-opposite points that lie in regions more likely to be favored by the original objective function. Specifically, instead of composing the objective function coefficients to consist of 1 and -1, the positive coefficients can be biased according to the sizes of the reduced costs of the original objective function $x_o = cx$, and the negative coefficients of the semi-counter-balanced objective can be inversely biased according to the sizes of these reduced costs. The identification of mappings of the original reduced costs

that produce more effective form of a semi-counter-balanced objective provides an interesting area of empirical research.

In many settings, it can be desirable to create a variation of the foregoing approach that reduces the amount of computation involved. A straightforward way to do this is to employ a *truncated re-optimization*, in which a limit is imposed on the number of simplex pivots allowed for re-optimizing a semi-counter-balanced objective. Such an approach can take advantage of its reduced computational requirements by making use of different choices for creating a semi-counter-balanced objective relative to the same current LP optimum x^0. The outcome of subjecting each of the selected objectives to a certain limited number of re-optimization pivots will produce a collection of "moderately nearby" proxies for the semi-opposite points associated with x^0. Since each proxy constitutes an extreme point x^{Bo} it provides an edge $[x^0, x^{Bo}]$ and a center x^{co} that can be treated in the manner previously indicated. The selected variation in the structure of these several semi-counter-balanced objective functions can replace the variation sought by tracking intermediate $x^{\#}$ solutions and re-optimizing to find their semi-opposite points.

Such a process provides usefully exploitable information in a manner similar to that provided by re-optimization without truncation. Non-basic variables that receive positive reduced costs in the semi-counter-balanced objective, and that emerge as more strongly resisting as a result of receiving larger positive values (or larger positive values in relation to their positive reduced cost coefficients) likewise are appropriate candidates for taking part in identifying desirable D vectors or for directly creating convex combinations to identify focal points as a foundation for directional rounding.

An interesting variant would be to apply a truncated version of an interior point method to generate semi-opposite points, since by such an approach these points would more nearly correspond to centers than extreme points. The advantages of such an approach are partially countered by the disadvantages of relinquishing access to the type of exploitable information described in the preceding paragraph, though perhaps another form of exploitable information might be produced in compensation.

One other way to reduce computation, which lacks some of the appealing attributes of the preceding uses of linear programming, but requires less computation overall is to generate a very small number of proxy semi-opposite points (perhaps only one or two), accompanied by extending the iteration limit for the truncated solution procedure, hence causing the proxies to be somewhat closer to the actual semi-opposite points. Then, a small number of derived centers produced by combining the midpoints x^{co} of resulting edges $[x^0, x^{Bo}]$ are generated, and for each of these a small number of mutually orthogonal rays are constructed. These rays are then extended to the boundary of F to generate new edges, whose midpoints can be used to continue the process.

The attractive feature of being computationally less expensive than using the simplex method to find new proxies for semi-opposite points may be offset, however, by the ability of linear programming to locate better points and to extract more useful information. If the constraints defining the feasible region include a number of equality constraints, for example, it may be difficult to find directions for the rays that permit non-degenerate extensions, whereas the use of linear programming avoids such potential difficulties.

6.8 Compounding and refining membership in the set X''

Further exploitation of the connection between infeasible and feasible space arises by taking advantage of derived edges and centers to reflect infeasible points of X'' back into the feasible region, and thereby produced additional derived edges and centers as a foundation for generating new candidates to enter X''.

Specifically, we may take any infeasible solution $x'' \in X''$ and join it by a line to a chosen center (such as one selected to lie closest to x''). The intersection of the line with the boundary of F identifies the endpoints of a derived edge in the usual fashion, whose elements can serve as focal points for directional rounding. In this case, x'' or a point half-way between x'' and x^0 can replace x^0 as the initiating point for the rounding. Clearly, more than one center can be chosen to be create a derived edge by reference to x'', either selecting centers by a similar or opposite criterion used to select the first center. Likewise, it is not necessary to restrict x'' to being infeasible in order to apply this process, although the approach is likely to be more relevant for cases where feasible solutions are harder to find.

When nearest neighbor rounding is added to directional rounding as a source of candidates for membership in X'', the following strategy may be employed. In addition to rounding from the center of the newly derived edge, the edge can be divided into intervals (e.g., into 4ths or 8ths) and the points located at these intervals can be subjected to nearest neighbor rounding. Alternatively, to save time, an interval point that is closest to being integer-valued can be selected, according to an L0, L1 or L2 distance measure, and its nearest integer neighbor can then be tested for feasibility.

6.9 Additional foundations for branching

A refinement that is very easy to summarize, though no less important for that fact, involves changing the nature of the branching operations used in conditional directional rounding. Rather than to branch on components of x, an approach called *straddle branching* makes it possible to identify new variables created as integer combinations of these components, which provide an ability to make branching steps that eliminate larger portions of the feasible region than branching on the x variables themselves (Glover and Laguna 1997). Variables created by such a procedure can also be embedded in the operation of directional rounding, to provide additional strategies for probing the search space. The ability to perform stronger branching steps in this manner may likewise generate supplementary information that may improve the effectiveness of other processes described above.

Finally, we observe that the procedures described here can be implemented in the context of mixed integer programming, where some components of x are permitted to have continuous values. In this setting, where only the integer components of x are modified by directional rounding, the focal points and centers of derived edges produced by various directional rounding strategies are all feasible in the continuous sense (with possible exceptions noted in Sect. 6.7 for strategies that do not produce projected opposite or semi-opposite points). Consequently, the resultant directionally rounded points created by the mixed integer counterpart of Approach 1 can become candidates for a *completion operation*, which holds the integer components constant and solves the resulting LP problem to seek better values for the continuous variables.

Preference may reasonably be given to selecting directionally rounded points for the completion operation that satisfy, or very nearly satisfy, the constraints defining F.

Within the branching operations of Approach 2, completion operations are not required, since the residual LP problems are automatically solved as the method progresses. Guidance criteria derived from applying Approach 1 continue to be relevant, however, and the mechanisms for using adaptive memory likewise carry forward unchanged.

7 Conclusions

The asymmetric geometries of feasible and infeasible space produce a fascinating web of interlinking relationships, and motivate an approach for solving integer programming problems that seeks to exploit such geometries by passing freely from one space to the other.

An opportunity to navigate between infeasible and feasible space in a flexible manner is made available through the medium of directional rounding, and particularly through conditional directional rounding, which takes advantage of branching operations carried out in conjunction with guidance from the probing operations of all-at-once rounding. The flexibility of this approach is amplified by embedding it within an adaptive memory framework, which affords broader options than branch-and-bound and yet can be organized to include ordinary branch-and-bound as a special case.

Additional processes for guiding the search are based on an associated outside-in procedure and the identification of derived edges and centers as a way to discover promising candidates for creating directionally rounded solutions. Adaptive memory likewise proves useful in this context by the introduction of frequency-based memory of resistances, and also by a conditional form of frequency-based memory that provides provisional inequalities for strategically constraining the search space. Inequalities can also be employed to take advantage of the connections between independent and dependent branches under explicit or implicit infeasibility conditions. Advanced strategies for exploiting directional rounding processes are provided by alternative uses of linear programming, employing counter-balanced and semi-counter-balanced objective functions to explore the space, giving rise to additional derived edges and centers as a source of focal points.

These procedures present a rich vein of strategic opportunities to be mined by empirical research, affording a chance to improve our insights about the interlinking geometries of feasible and infeasible space, and to find more effective ways to take advantage of them.

Appendix: Expanded discussion and proof of theorems

To justify the theorems of Sect. 1 we first observe that an optimal solution x^* exists for which a unit change in a single variable will render x^* infeasible. In particular, if c is not the 0 vector, then a $+1$ change in any x_j^* such that $c_j > 0$ and a -1 change in any x_j^* such that $c_j < 0$ will yield an infeasible solution, since the resulting solution yields an improved x_o value. If c is the 0 vector, then the bounded condition on F

allows c to be perturbed so that selected components of c are replaced by small ("epsilon value") nonzero coefficients while assuring that at least one optimal solution x^* remains optimal.

All three of the theorems can be established by reference to a special construction that consists of a minimal collection of unit hypercubes that cover F, which we denote by $C(F)$. It is easy to see that $C(F)$ is also a minimum cover and, if F in not degenerate (i.e., is n-dimensional), then $C(F)$ is unique. Moreover, we can observe: (1) every hypercube in $C(F)$ must contain a point of F; (2) the graph $G(F)$ consisting of the vertices and edges of the hypercubes composing $C(F)$ is connected; and (3) $C(F)$ cannot contain any "holes," which is to say that if x^1 and x^2 are points of F that lie in two different hypercubes of $C(F)$, then every point on the line segment joining x^1 and x^2 must also lie in some hypercube of $C(F)$. We also note that a hypercube C of $C(F)$ that contains a non-integer extreme point x^0 of F cannot lie entirely within F, i.e., at least one vertex of C must be infeasible, since otherwise x^0 would be spanned by vertices of C that lie within F, contrary to the definition of an extreme point. Finally, every point obtained by some rounding of a non-integer point within F must lie on a vertex of some hypercube within $C(F)$.

Let Z denote the set of all infeasible integer vectors, and consider by extension of $C(F)$ the minimal (infinite) collection of unit hypercubes $C(Z)$ such that the union of $C(F)$ and $C(Z)$ cover all of n-space. It is clear that the graph $G(Z)$ consisting of the vertices and edges $C(Z)$ is connected, and hence there is a path starting from any integer point in Z that lies entirely in Z and terminates in a point just one step (edge) away from an optimal x^*.

More restrictively, Theorem 1 starts from a point x^1 in Z (and hence $G(Z)$) obtained by rounding a point x^0 on the boundary $B(F)$ of F. By our previous observations, x^1 must lie on $G(F)$ and hence on the intersection of $G(F)$ and $G(Z)$. If we enlarge $G(F)$ to a graph $G^*(F)$ that includes vertices and edges of $G(Z)$ such that each new vertex lies within a distance D^0 from $B(F)$, we can skirt all points of $G(F)$ that may lie on F, and find a path satisfying the conditions of Theorem 1. In fact, parts of $C(F)$ and hence $G(F)$ are superfluous, since as noted earlier a path can follow edges that cut across portions of F in traversing from one infeasible point to another, and hence it is not necessary to include reference to hypercubes that contain the infeasible regions bypassed.

Apart from Theorem 1, there is no need to refer to $G^*(F)$, since the graph $G(F)$ clearly contains the paths described in Theorems 2 and 3. We complete our discussion of the theorems by justifying the assertion following Theorem 1 that the path P cannot move progressively closer to $B(F)$. Consider a feasible region that looks like a long thin needle, whose "point" is an LP vertex x^0 that lies close to $x = 0$, and whose "shaft" passes close to $x = e$ (the vector of all 1's), to eventually include a single integer solution $x = ke$ for some positive integer value of k. Then by starting P at the infeasible integer point $x = 0$, a shortest path to a feasible integer solution that stays close to $B(F)$ must necessarily grow progressively farther from $B(F)$ for $n/2$ successive steps, but then can grow progressively closer for another $n/2$ steps, to lie again within a unit distance of $B(F)$. (A slight qualification applies if n is odd.) Then the pattern repeats in successive waves until ultimately reaching the integer solution $x = ke$. Multiple shortest paths exist, some of them lying farther from the feasible region over parts of their trajectory.

It may additionally be observed that starting with an LP solution x^0 that has only a small number of fractional components does not allow the theorems of Sect. 1 to be strengthened by defining D or D^* relative to the number of fractional variables, or by stipulating the existence of a trajectory that approaches $B(F)$ in a more nearly monotonic manner. The preceding comments concerning a feasible region that constitutes a "long thin needle" apply even if a single component of x^0 is fractional.

We conclude by noting that feasible regions having such a character can be exploited by joining the ideas of this paper with a transformation of variables to create a *bounding form* structure (Glover and Laguna 1997).

References

Eckstein, J., Nediak, M.: Pivot, cut, and dive: a heuristic for mixed 0–1 integer programming. RUTCOR Research report RRR53-2001 (2001)

Fischetti, M., Glover, F., Lodi, A., Monaci, M.: Feasibility net. (2006, in preparation)

Gendreau, M.: On the importance of allowing infeasible moves in tabu search heuristics. In: INFORMS National Meeting, Denver, 24–27 October 2004

Glover, F.: Heuristics for integer programming using surrogate constraints. Decis. Sci. **8**(1), 156–166 (1977)

Glover, F.: Parametric branch and bound. Int. J. Manag. Sci. **6**, 1–9 (1978)

Glover, F.: Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). Discrete Appl. Math. **49**, 231–255 (1994)

Glover, F.: Scatter search and star-paths: beyond the genetic metaphor. OR Spectr. **17**, 125–137 (1995)

Glover, F.: Adaptive memory projection methods for integer programming. In: Rego, C., Alidaee, B. (eds.) Metaheuristic Optimization via Memory and Evolution, pp. 425–440. Kluwer Academic, Dordrecht (2005)

Glover, F.: Parametric tabu search for mixed integer programs. Comput. Oper. Res. **33**(9), 2449–2494 (2006)

Glover, F., Laguna, M.: Tabu search. In: Reeves, C. (ed.) Modern Heuristic Techniques for Combinatorial Problems, pp. 71–140. Blackwell Scientific, Oxford (1993),

Glover, F., Laguna, M.: Tabu Search. Kluwer Academic, Dordrecht (1997)

Guignard, M., Spielberg, K.: Double contraction, double probing, short starts and BB-probing cuts for mixed (0,1) programming. Wharton School report (2003)

Lokketangen, A., Glover, F.: Tabu search for zero-one mixed integer programming with advanced level strategies and learning. Int. J. Oper. Quant. Manag. **1**(2), 89–108 (1995)

Lokketangen, A., Woodruff, D.L., Glover, F.: Scatter search to generate diverse MIP solutions. In: M. Laguna, J.L. Gonzalez-Velarde (eds.) OR Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research, pp. 299–317 (2000)

Spielberg, K., Guignard, M.: A sequential (quasi) hot start method for BB (0, 1) mixed integer programming. In: Mathematical Programming Symposium, Atlanta, 2000