ELSEVIER Discrete Applied Mathematics 49 (1994) 231–255

DISCRETE APPLIED MATHEMATICS

Tabu search for nonlinear and parametric optimization (with links to genetic algorithms)*

Fred Glover**

US West Chair in Systems Science, Graduate School of Business, Campus Box 419, University of Colorado, Boulder, CO 80309-0419, USA

Received 16 April 1991; revised 27 November 1991

Abstract

In spite of the widespread importance of nonlinear and parametric optimization, many standard solution methods allow a large gap between local optimality and global optimality, inviting consideration of metaheuristics capable of reducing such gaps. We identify ways to apply the tabu search metaheuristic to nonlinear optimization problems from both continuous and discrete settings.

The step beyond strictly combinatorial settings enlarges the domain of problems to which tabu search is typically applied. We show how tabu search can be coupled with directional search and scatter search approaches to solve such problems. In addition, we generalize standard weighted combinations (as employed in scatter search) to include *structured weighted combinations* capable of satisfying specified feasibility conditions (e.g., mapping weighted combinations of scheduling, partitioning and covering solutions into new solutions of the same type). The outcome suggests ways to exploit potential links between scatter search and genetic algorithms, and also provides a basis for integrating genetic algorithms with tabu search.

1. Introduction

This paper identifies ways to apply tabu search to nonlinear optimization problems that characteristically have remained beyond its scope. Successful uses of tabu search to overcome local optimality in discrete nonlinear settings motivates consideration of how the method may similarly be applied in continuous nonlinear settings. Our proposals offer new strategies for certain combinatorial problems, such as those

* Preliminary version presented at the EPFL Seminar on OR and AI Search Methods for Optimization Problems, Lausanne, Switzerland, November 1990. Revised for the special issue of *Discrete Applied Mathematics* titled Viewpoints on Optimization.

** E-mail: glover-f@cubldr.colorado.edu

arising in production, manufacturing and resource planning, where frameworks have evolved for obtaining discrete solutions by reference to nonlinear continuous equivalents, or by reference to special parameterizations.

Parametric optimization problems, as referred to here, involve searching an auxiliary space in order to obtain vectors that map into solutions in a primary space. The auxiliary space may be a superset of the primary space, or may be defined over an entirely separate domain. The goal is to generate parameters or coefficient values that map into solutions in primary space by means of algorithmic processes that become well defined once these values are specified.

Among the more conspicuous parametric optimization problems are those arising from Lagrangian and surrogate constraint relaxations. But there are also a number of other types, whose susceptibility to treatment by nonlinear search strategies is sometimes overlooked. Examples of such "problems within problems", where continuous trial solutions map into solutions in a different space, are as follows.

Example 1.1. The auxiliary space is defined over vectors of objective function or constraint coefficients, which modify original problem coefficients for strategic purposes. Typical applications include project selection problems, capital budgeting problems and multiconstraint knapsack problems. Coefficient manipulations induce the generation of alternative solutions by standard heuristics, and the solutions are used to gauge the merit of the modified coefficients.

Example 1.2. Auxiliary spaces based on modified coefficients as in Example 1.1 are linked with structural modifications in order to adapt network models to problems with more complex constraints. In applications such as resource distribution and telecommunications, selected constraints are replaced by associated simpler restrictions that impose bounds on newly created variables, giving rise to a pure or generalized network problem. The bounds are systematically adjusted, and the solutions to the resulting network problems give trial solutions for the original (primary) problem.

Example 1.3. Weights attached to decision rules in job shop scheduling problems serve as parameters for defining the auxiliary space. These parameters are used either probabilistically deterministically to generate schedules for the primary job shop problem.

Example 1.4. An auxiliary space based on a more rigorous treatment of embedded network structures occurs in *layering strategies* for linear and integer programs. Linking constraints are replaced by parameterized bounds to create decompositions into collections of generalized network problems. Trial solutions establish a feedback mechanism to evaluate the parameterized bounds.

Example 1.5. Applications of *target analysis* create an auxiliary space where historical search information is parameterized to generate decision rules that "learn from experience". The quality of solutions derived from the decision rules measures the efficacy of the parameters. (A standard search process at one level thereby is used to create an improved search process at another.)

Applications involving such auxiliary spaces occur in [1–3, 5–8, 11, 13–15, 17]. In each of these instances, a successful problem solving effort depends on the search of a nonlinear continuous space, using *algorithmic mappings* to link the outcomes to trial solutions in another space.

To apply tabu search to nonlinear problems, arising both from such parameterized applications and from standard nonlinear models, we consider two main approaches:[1] (1) *directional search*, where a transition from one point to another occurs by reference to gradients, subgradients, feasible directions, and projections; (2) *scatter search*, where successive collections of points are generated from weighted combinations of reference points, creating search lines and associated trial points that provide new reference points.

The organization of the paper is as follows. Directional search strategies are treated in Section 2, and shown to be susceptible to guidance by tabu search using simple region and direction avoidance mechanisms as a basis for implementing tabu restrictions. Section 3 is devoted to scatter search, which is organized to permit direct application of the intensification and diversification themes of tabu search. This is accomplished by using tabu restrictions and aspiration criteria to determine admissible subsets of reference points from among the historical best. Section 4 introduces the concept of structured weighted combinations to provide mappings that satisfy feasibility requirements, where the vectors used as arguments in these mappings receive an influence reflecting their assigned weights. A special subclass of these mappings, called *adaptive structured combinations*, generates new vectors with the guidance of evaluation criteria to produce outcomes with superior objective function values, allowing weights to be subordinate. These forms of vector combinations extend the domain for applying scatter search, and, as noted in Section 5, also suggest the use of this approach as a supplement to genetic algorithms.

Each of the sections is largely self-contained and independent of the others. An introductory understanding of tabu search is helpful for Sections 2 and 3.

2. Directional search strategies

We begin by focusing on ways to treat directional search strategies with the short term component of tabu search. By the standard TS approach, a prospective move (transition between solutions) is rendered tabu and excluded from consideration if it reverses a recent previous move. To implement the process, memory is maintained of selected attributes of recent moves and their associated solutions, where a current move is classified tabu if it reinstates an attribute (or attribute set) that was changed by earlier moves. Alternately, tabu moves are discouraged from being selected by attaching penalties based on their tabu status. Probabilistic variants incorporate probabilities of selection that are a monotonic function of their penalized evaluation.

[1] Included among the more effective approaches to nonlinear optimization are pivoting processes based on generalizations of the simplex method [21]. Tabu search methods already developed for exploiting exchange moves are directly relevant to such procedures, and hence will not be examined here.

In more refined implementations the tabu status, or degree to which a move is classified tabu, depends on the quality of the move and the contribution of its component attributes to the quality of past moves. In turn, this status determines how long the move remains tabu (or the penalty attached to its selection). We focus on the simpler types of tabu search implementations in the following discussion, restricting attention to tabu conditions based on move reversals.

Tabu search strategies for preventing move reversals may be adapted to the nonlinear context as follows. Let x' and x'' denote a pair of successive solutions where x'' was generated from x' by a recent move. Following the approach commonly used in discrete settings, we seek to avoid a move reversal by prohibiting a current move that would create a solution x containing selected attributes of x'. For the present purpose, we will select problem variables as a basis for defining move attributes, specifically identifying a subset of variables that change values in going from one solution to another. Then a move from x' to x'' is classified tabu if the designated variables would return to the values they had in x'.

Applying this approach to spaces with continuous as well as discrete properties, tabu restrictions may be imposed to prevent moves that bring the values of selected variables "too close" to values they held previously. (For an integer programming problem, "too close" may mean "less than a unit away.")

Specifically, let t denote a selected number of iterations that tabu status is allowed to operate (i.e., t is a tabu age limit, which causes tabu restrictions "younger than t" to render a move tabu). Then two kinds of tabu restrictions for nonlinear problems are the following.

(R1) A move is tabu if it creates a solution x which lies closer than a specified *tabu distance* d to any solution visited during the preceding t iterations. (The distance d can be a function of x and also of t.)

(R2) A move is tabu if it employs a direction vector contained in a *tabu region* created on one of the preceding t iterations. (Such a region created by a move from x' to x'' can be defined, for example, to be a set of direction vectors that lie in a cone with vertex x'' and with center line passing through x', for a selected tabu angle, e.g., 30° or 45°.)

Focusing on components of vectors rather than on complete vectors provides a useful way to make (R1) and (R2) easier to implement. This may be accomplished by the following special variants of (R1).

(V1) Identify k components of x whose values change the most in the move from x' to x''. Create a new variable y which equals a weighted linear combination of the chosen components; e.g., choosing a weight for the component x_j equal to its change, $x_j'' - x_j'$, or selecting all weights to be 1 or -1, according to the direction of change. For a duration of t iterations, prevent this variable from receiving a value less than $\frac{1}{2}(y' + y'')$, where y' and y'', respectively, denote the values that result for y when x equals x' and x''. More generally, the variable y may be prevented from receiving a value less than $y' + w(y'' - y')$, or excluded from falling inside the line interval bounded by this value and $y' - w(y'' - y')$, where $1 \geqslant w > 0$.

If k is selected equal to n, the dimension of x, then precisely the components of x that change their values will receive nonzero weights in defining y. At the other extreme, choosing k equal to 1 corresponds to selecting a single component x_j of x, which offers the advantage of making it unnecessary to keep track of definitions of t additional variables. In this case an alternative form of elaboration is the following.

(V2) Choose two components of x whose values change the most from x' to x''. Impose a tabu restriction that prevents at least one of these two components from changing in the reverse direction by an amount greater than half of its current change. If one (or both) of these components is already subject to such a restriction, then impose the stronger restriction of (V1). (The remaining component can be paired with the former partner of the first component to create a weaker restriction, preventing at least one of the two from reversing direction.)

Implementation of the preceding rules requires consideration of three strategic elements. (1) The x values should be normalized so that a given change in one component is comparable to the same change in another. If x is a vector of weights for a Lagrangian or surrogate relaxation method, for example, then the problem constraints should be normalized to assure that the change in one constraint has the same relative effect as that of another (since the weight changes are typically derived as a function of changes in constraint violations). (2) The "largest change" choice rule should be amended in a diversification strategy designed for the long term to favor the choice of components not yet subject to tabu restrictions. (This can sometimes may be a self-regulating consequence of handling other considerations appropriately.) (3) Derivative factors can be relevant in choosing a component of x. Specifically, in the context of Lagrangian and surrogate relaxations, a choice based on the magnitude of a change in weight (dual variable) may alternately be replaced by one based on the magnitude of a change produced in a corresponding problem constraint, giving priority to constraints that undergo the transition from being violated to satisfied.

3. Scatter search

Nonlinear optimization applications are conveniently suited to the use of intensification and diversification strategies to obtain generally better solutions in the intermediate and long term. An illustration is provided by the application of the *scatter search* approach.

Scatter search [8] is designed to generate a systematically dispersed set of points from a chosen set of reference points. This is done by creating trial points which consist of *weighted combinations* of the reference points, and then generating new points on lines that join the reference points to the trial points.[2] Selected reference points are also paired with each other to define additional lines for generating new points.

In an iterative implementation, reference points are derived from a historical progression and used in turn to influence this progression. We do that here in a simple manner by selecting each new set of reference points from trial points and reference points previously created. Tabu search is then superimposed to control the choice of

[2] Recently Michalewicz, Vignaux, and Hobbs [18], have applied an instance of this idea to create a "nonstandard genetic algorithm" for nonlinear transportation problems.

outcomes in the customary fashion by introducing tabu restrictions and aspiration criteria to determine the admissible composition of reference points at each stage.

By narrowing the focus of the method to generating new points only from previous ones, this implementation of a scatter search and tabu search combination constitutes an intensification strategy. To be most effective, therefore, the approach should feed back into a means of creating new reference points at a global level (as from a separate ongoing tabu search approach at this level), to create a suitable balance with diversification. Such an integration with a higher-level process provides a natural basis for parallelization. In addition, the scatter search approach is highly susceptible to parallelization by itself, which may enhance the efficacy of an SS/TS blend.

3.1. The method in overview

We assume the existence of a mapping M that transforms x into a trial solution $M(x)$ for the specified problem to be solved. A criterion of merit, such as an objective function, is then applied to the solution $M(x)$ to give the evaluation of x. (For example, the mapping M can correspond to one of those implicit in the examples of Section 1, or may represent a post-processing operation such as "generalized rounding" to produce integer trial solutions from fractional vectors. We stress that for many settings it is important to use mappings that transform solutions into local optima, which then provide the x vectors that continue the process.)

The method relies on a number of parameters that define the sizes of particular sets and intervals for exploring search lines. Following the intensification theme, the values of these constants can be kept small (mostly between 2 and 5), which conveniently limits calibration possibilities. In addition, we propose specific values of the constants as "defaults", to remove the burden of selecting these values where the setting does not immediately prompt different choices.

Distance measures used by the method can be based on the L_1 (absolute value) norm for convenience. The concept of structured weighted combinations that operate in a nongeometric space, introduced later, invite other measures of distance.

Steps of the SS/TS approach
Initialization and classification. The method starts with a set S of initial trial points, from which all other sets derive. Proposals for generating initial points often are based on random sampling. We suggest alternatively that the generation of such points be based on a strategy for creating *diverse subsets*, as subsequently described.

Reference points, initially selected from the more attractive elements of S, are used to generate new elements that replace previous members of S. Elite elements (those of especially high quality) become part of a special historical collection that also contributes to the choice of current reference points. The following sets are referenced and updated at each iteration.

H = a set of elite *historical generators*, consisting of h best (highest evaluation) points generated historically throughout the search (e.g., $h = 20$). On the first iteration, H will consist of h best elements of S.

T = a set of *tabu generators*, composing a subset of H currently excluded from consideration. Initially T is empty.

$H^* =$ a set of *selected historical generators*, consisting of h^* best elements of $H - T$ (e.g., $h^* = 4$).

$S^* =$ a set of *selected current generators*, constituting of s^* best elements of S (e.g., $s^* = 5$).

On the first iteration, S^* and H^* overlap (H^* is a subset of S^*). Subsequently they overlap only as current elements of S have higher evaluations than some of those in H^* (when H changes to incorporate elements of S).

$R =$ a set of current reference points, defined to equal the union of H^* and S^*. The number r of elements in R, which equals $h^* + s^*$ when H^* and S^* are disjoint, is given a lower bound r^* which is slightly less than the sum, to apply when H^* and S^* overlap. The bound is achieved by adjusting s^* upward so that the number of elements in $S^* - H^*$ will be large enough to compensate for elements shared with H^*. (For example, s^* will be increased to r^* on the first iteration, when H^* lies wholly within S^*.)

Outline of an iteration. Scatter search is first applied to create centers of gravity for S^* and for each $s^* - 1$ element subset of S^*. These newly created trial points are paired with points of R to create search lines for generating additional trial points. Elements of S^* are also paired with each other to create new search lines and trial points in the same manner. Finally, *diverse subsets*, $D(S^*)$ and $D(S - S^*)$ of S^* and $S - S^*$ are identified by rules subsequently indicated, and used to create additional search lines by pairing their elements with those of H^* and R, respectively.

Evaluation and updates. Throughout an iteration, each trial point with an evaluation superior to the mean evaluation of the points in S^* initiates an intensified search on the line that produced it. At the same time, elements that are candidates to compose S^* for the next iteration are identified, together with candidates for inclusion in H. Finally, T is updated by reference to the tabu restrictions and aspiration criteria, and the method repeats.

To provide a complete description of the method, we now indicate the rules for generating the trial points and for creating the tabu restrictions defined over elements of H.

3.2. Generating trial points

Trial points are produced by the following steps.

Step 1. Denote the elements of S^* by $x(k)$, $k = 1, \dots, s^*$. To begin, generate the center of gravity of S^*

$$y(0) = \sum x(k)/s^*$$

and the centers of gravity of each $s^* - 1$ element subset of S^*

$$y(k) = y(0) + (y(0) - x(k))/(s^* - 1), \quad k = 1, \dots, s^*.$$

These points will provide the initial set of trial points.

Step 2. Let $x(s^* + 1), \dots, x(r)$ denote the reference points in R that are not in S^*, and let (x, y) successively refer to each of the r pairs $(x(k), y(k))$ for $k = 1, \dots, s^*$ and $(x(k), y(0))$ for $k = s^* + 1, \dots, r$ (matching each reference point in R with one of the trial points created in Step 1).

Consider the line through x and y given by the representation $z(w) = x + w(y - x)$, where w is a scalar weight. Subject to refinements indicated later, we begin by restricting attention to the points $z(\frac{1}{2})$, $z(\frac{1}{3})$, $z(\frac{2}{3})$, $z(-\frac{1}{3})$ and $z(\frac{4}{3})$. (The point $z(\frac{1}{2})$ is the midpoint of the line segment joining x and y, $z(\frac{1}{3})$ and $z(\frac{2}{3})$ are the interior points of this segment that divide its length into thirds, and $z(-\frac{1}{3})$ and $z(\frac{4}{3})$ are corresponding exterior points on the x side and y side of the segment.)

The computational effort of transforming such points into solutions by the mapping M will influence the number that will be examined as trial points. For example, when it is expensive to examine all points, preference may be given to examining points closer to x or y, depending on which of x and y has a higher evaluation. (Observations about alternative approaches for generating such points are provided subsequently.)

Step 3. Apply the rule of Step 2 to create trial points from the pairs (x, y), $x \neq y$, both of whose members are in S^*.

Step 4. Let $D(X)$ denote a (small) *diverse subset* of the set X, for $X = S^*$ and $X = S - S^*$, generated as follows. The first element is selected to maximize its distance from the center of gravity of X. The second element is selected to maximize its distance from the first element, and the third element is selected to maximize its minimum distance from the first and second elements. The rule of Step 2 is then applied to the pairs (x, y) in the two cross product sets $H^* \times D(S^*)$ and $R \times D(S - S^*)$, to generate search lines and trial points. (Diverse subsets containing more than three elements are considered later.)

Step 5 (intensification component). During the execution of Steps 2–4 identify the highest-evaluation trial point z on each search line (including trial points generated in Step 1 as candidates). If the evaluation of z is higher than the average of the evaluations over S^*, then intensify the search around z by generating two new trial points at the midpoints of the segments that join z to its nearest neighbors already examined. (If z is one of the exterior points $z(-\frac{1}{3})$ or $z(\frac{4}{3})$, add the point $z(-\frac{2}{3})$ or $z(\frac{5}{3})$, respectively, to serve as its "missing" nearest neighbor.) Then if a newly created point has an evaluation that exceeds the evaluation of z (by an appropriate threshold increment) the process is repeated, designating the best new point as z, and generating analogous "missing neighbors" as necessary. In the special case where z is the point $y(0)$ from Step 1, this step should be executed only for the line joining z to the best element of H^*. (For problems at risk of unbounded optimality, of safeguarding limit may be imposed on the number of improving iterations.)

Refinements of the foregoing procedure will be indicated after describing the rules for maintaining and updating T.

3.3. Tabu restrictions and aspiration criteria

The implementation of tabu restrictions and aspiration criteria to guide successive iterations of the scatter search process likewise relies on parameters which will be expressed as small constants. First, tabu restrictions are introduced by transferring elements of the historical set H to the set T, and thus temporarily excluding them from membership in H^*. Such a restriction is initiated as a result of belonging to H^* (i.e., as a result of using an element to generate trial points).

Two types of aspiration criteria are applied to postpone or mitigate this restriction, based on the use of a *small iteration parameter* i^* (e.g., $i^* = 3$). First, when an element x is incorporated into H^*, a "minimum level of exposure" is provided by allowing x to remain in H^* for i^* iterations. At this point x automatically becomes tabu and enters T unless it satisfies the second aspiration criterion. This criterion permits x to remain in H^* for up to i^* iterations after generating a trial point z which entered the set S^* (i.e., where x belonged to a pair (x, y) that produced a search line containing z). Once a limit of i^*h^* consecutive iterations is accumulated in H^*, then x enters T regardless of the operation of the second aspiration criterion.

Progressive tabu restrictions

The first time an element becomes tabu, its tabu status remains in force for the same duration (i^* iterations) that the element is initially guarded from becoming tabu. However, this tabu restriction is progressively increased by incrementing the tabu tenure of an element x each additional time that x enters T, i.e., increasing by 1 by the number of iterations it must remain tabu. The tabu tenure for x returns to a value of i^* whenever x contributes to generating a trial point that qualifies for entry into H. (This is a variant of the second aspiration criterion.) In addition, the maximum duration that x remains in T is limited to h/i^* iterations.

Diversification

The final component of the tabu search process assures that all elements of H eventually have an opportunity to belong to H^*. Let *count*(x) denote the number of consecutive iterations an element x of H does not belong to either H^* or T, and let *frequency*(x) be the total cumulative number of iterations (not necessarily consecutive) x has been a member of H^*. Then the stipulation that H^* consists of h^* best elements of $H - T$ is modified by interpreting "best" to include reference to the values of *count*(x) and *frequency*(x). Whenever the maximum *count*(x) value exceeds a threshold of i^*h^*, then each x yielding this maximum *count*(x) value receives first priority for (inclusion in H^*, breaking ties to favor the smallest accompanying value of *frequency*(x) (and automatically favoring higher-ranking elements of H subject to this).

3.4. *Refinements*

There are a number of possibilities for refining the method. Evidently, for example, the centers of gravity produced in Step 1 can be weighted, as by reference to evaluations of the elements in S^*. An important refinement is to create a "parallel scatter" approach by simultaneously generating several different solution streams, each with its own sets S and H. Periodically (e.g., as the new admissions to H become infrequent) the elements from the union of the different sets H may be reallocated to form new sets. This may be done randomly, or by applying a rule to generate diverse subsets, thereby providing a more selective foundation for generating new points. A parallel processing approach coordinated in this way also can result in smaller preferred sizes for the sets H than otherwise would be appropriate. Additional alternatives for refinement are as follows.

(1) The values of w used to generate trial points ($\frac{1}{2}, \frac{1}{3}, \frac{2}{3}$, etc.) are "default estimates" of appropriate values, and assume the behavior of the evaluation function in the region examined is unknown. If knowledge of this behavior exists, or can be inferred from the search then it should be used to select other values for w.

(2) During an intensified line search, a preferred strategy is to iteratively construct a quadratic or cubic to "fit" the evaluations for successive trial points, and then calculate the location of hypothetical maximum. (Variants of Golden Section and Fibonacci search may similarly be applied.) In conjunction with such an approach, a line on which to intensify the search may alternately be chosen according to the progression of evaluations of its trial points, rather than according to the quality of its "best" trial point. (If there is some uniformity in the parameters used to create a fit by polynomial interpolation, then the resulting function can be used without recalculation to give first estimates of appropriate w values in (1).) However, this issue is complicated slightly in the case where the mapping $M(x)$ is designed to create a locally optimal point which then replaces x. Depending on the effort to apply such an $M(x)$, the method may proceed for selected numbers of iterations without generating local optima, alternated with generating local optima from the solutions that result.

(3) The method may be improved by recognizing *approximate duplications*. When a point considered for inclusion in S^* or H^* has an evaluation close to that of another point in the set, then the distance between these two points may be examined. If this distance is sufficiently small the point with the lower evaluation may be discarded. (The degree of separation that characterizes "sufficiently small" depends on the problem context. The choice of this degree can be a heuristic device to prevent points from clustering too tightly.)

(4) When a new element x is added to H, x preferably may be paired with other elements of H, to assure that all representative search lines generated by members of H have been explored. (If H is not large, as in a parallel approach, then the total computation will not be greatly affected, particularly since the composition of H changes less frequently than that of S.)

(5) A simple form of diversification may be achieved by restricting the number of trial points from any single search line permitted to be included in S^* or H. (The goal is related to that of avoiding "approximate duplications", as proposed in (3), but requires less effort to check.) For example, a rule may be imposed to prevent more than two points from entering S^* or H from any given line. These points can also be required to be separated by a minimum distance, as in (3), or else only one of them may be selected.

(6) Finally, more ambitiously, the definition of a diverse subset may be usefully broadened. Given a parent set X and a seed point u (where u need not belong to X), a diverse subset $D(X, u)$ of X may be characterized as a subset generated by the following rules: (a) the first element is selected from X to maximize its distance from u, (b) each subsequent element is selected from X to maximize its minimum distance from all elements previously chosen (excluding u), (c) ties are broken by maximizing

the minimum distance to the last $k-1$ elements chosen, and then the last $k-2$ elements, etc., where k is the total number of elements currently selected to compose $D(X, u)$. Note that when a selected point x is first mapped into a local optimum, this new point immediately replaces x in defining the current composition of $D(X, u)$.

Such a definition can become computationally expensive to implement if $D(X, u)$ is allowed to grow beyond a modest size. In this case, a relaxation of the definition can be used to significantly reduce the computation: select a small value of v, and choose each successive point to maximize the minimum distance from the v elements most recently added to $D(X, u)$, breaking ties by maximizing the minimum distance from the first v (or $\frac{1}{2}v$ elements) elements added to $D(X, u)$. (Ties that remain may be broken by randomization.) This rule is similar to the use of a simple form of short-term memory in tabu search.

The notion of a diverse subset has a broader relevance to the SS/TS method. As noted earlier, the initial set S to start the process may be generated by reference to this concept. Thus, for example, once a seed point u is selected, remaining points may be chosen by the diversification approach, retaining for S a subset consisting of the more attractive elements found. Since this subset will establish different diversification criteria than the full set, the process may be repeated relative to the points retained to create a more appropriate set of diverse elements. In this second phase (and in the first, if enough information is available), new points that do not qualify as acceptably attractive may be immediately screened out.

Two variants of this approach are relevant to initiate a *parallel scatter* procedure. One is simply to generate several initial diverse subsets, each to provide a different starting S. The other is instead to make each element of the initial diverse subset a "first member" of a separate S. Then these single element sets can be expanded by a series of steps that augment each one again by a single element, where the criterion for selecting this element is first to maximize the minimum distance from centers of gravity of other sets, and subject to this to maximize the minimum distance from elements of the set under consideration. (Maximizing weighted sums of distances alternately can replace the max-min criterion.) Such an approach assumes that points representative of the universe to be considered are already specified, in order to give candidates to augment the sets under consideration. When this is not the case, candidates may be generated by conditional random sampling as the process evolves.

Once the parallel scatter approach is thus initiated, the process for generating diverse subsets also can be periodically applied in order to reallocate elements from the current H sets to form new H sets, as previously noted. Within the operation of the SS/TS method itself, related options are to select additional diverse subsets of the elements generated, including subsets of H and S^*, and to define H^* to be a diverse subset of $H - T$.

4. Structured weighted combinations

The application of tabu search in the scatter search framework can be adapted to handle discrete optimization problems directly, without the intermediary of a parameterized continuous auxiliary space. We show how to do this by replacing the customary operations used to define weighted linear combinations with *structured transformations* of vectors that preserve specified discrete relationships and associated feasibility conditions. Our approach is based on three properties of the reference vectors from which the weighted combinations are created.

Property 4.1 (*representation property*). Each vector represents a set of votes for particular decisions (e.g., the decision of assigning a specific value to a particular variable, or of assigning a specific facility to a particular location, or of establishing a precedence relationship between a particular pair of elements, etc.). Standard solution vectors for many problems can directly operate as such voting vectors, or can be expanded in a natural way to create such vectors. For example, a solution vector for a job shop scheduling problem can be interpreted as a set of 0–1 votes for predecessor decisions in scheduling specific jobs on particular machines, as illustrated later.

Property 4.2 (*trial solution property*). The set of votes prescribed by a vector translates into a trial solution to the problem of interest by a well-defined process. A set of "yes–no" votes for items to include in a knapsack, for example, can be translated into a trial solution according to a designated sequence for processing the votes (such as the standard sequence determined by benefit-to-weight ratios), until either the knapsack is full or all votes are considered. More general numerical votes for the same problem may additionally prescribe the sequence to be employed, as where knapsack items are rearranged so the votes occur in descending order. (Note that the vectors may not represent feasible solutions to the problems considered, or even represent solutions in a customary sense at all.)

Property 4.3 (*update property*). If a decision is made according to the votes of a given vector, a clearly defined rule exists to update all vectors for the residual problem so that Properties 4.1 and 4.2 continue to hold. (For example, upon assigning a specific value to a particular variable, all votes for assigning different values to this variable are cancelled, and the remaining updated votes of each vector retain the ability to be translated into a trial solution for the residual problem in which the assignment has been made.)

Drawing on the three preceding properties, we provide a constructive method to create structured weighted combinations of vectors. The properties imply that the decisions voted upon can be made in a sequential order, which is established either *a priori*, or as a function of the decision history. The sequential decision steps also may be initiated or linked by solving associated optimization problems. (For example, an assignment problem to initiate the decision steps for a traveling salesman tour can be created by combining the votes of permutation vectors with original distance data.) To exploit the ability to make decisions sequentially, we require a means of transforming the weights of the specified weighted combination into evaluations of different decision outcomes. After giving rules for doing this, we will give concrete examples to demonstrate how the approach can be applied.

4.1. Scores and evaluations

The basis for creating the desired transformation will be to progressively modify the impact of votes cast for current outcomes, thereby enabling a vector that was "shortchanged" in earlier decisions to have greater influence in the current decision. The goal is to make the total influence of each vector $x(k)$ over time proportional to the weight $w(k)$ assigned to that vector.

Let $v(k,j)$ denote the number of votes of vector $x(k)$ for outcome j in the current decision. For convenience, we suppose $v(k,j) \geqslant 0$ for all k and j. (If the votes are additionally normalized to sum to 1, then $v(k,j)$ represents the proportion of $x(k)$'s voting power that is assigned to outcome j.) By reference to outcomes selected, we translate these votes into a score that measures each vector's degree of contribution to all decisions made to the present stage. This score is a relative score, which is divided into a current and a past component.

We refer to the set of outcomes that is voted upon (i.e., considered for possible selection) during the current iteration as the current decision set. Assume that a specific outcome p is selected from this set to give the current decision (by a mechanism to be described). Then the current score for $x(k)$'s relative contribution is defined to be

$$\mathit{current\_score}(k) = v(k,p) - \mathit{Mean}(v(h,p)\colon h \neq k), \tag{D1}$$

where $\mathit{current\_score}(k) = 0$ if this value is negative. (Although the score depends on the selected outcome p, we suppress reference to p for notational convenience, and to avoid confusion with later subscripting.)

The motive for this definition can be understood from the situation where all votes are 0 and 1, which arises in a variety of applications. Then the relative contribution of $x(k)$ to outcome p is 0 if $x(k)$ does not vote for the outcome, and is 1 if $x(k)$ is the only vector to vote for the outcome. Otherwise, the relative contribution lies between these values. In the special case where all vectors vote for outcome p, the relative contribution of each is 0. Implicitly, the goal is that the weights associated with these vectors should characterize their relative influence (or contribution) over the decisions where they prescribe different results. The relevance of this objective will become clear in later examples. (Other objectives can be invoked to create different definitions of $\mathit{current\_score}(k)$.)

The cumulative historical score associated with $x(k)$ is defined by

$$\mathit{historical\_score}(k) = \textstyle\sum(\mathit{current\_score}(k)\colon \text{over prior decisions}). \tag{D2}$$

We seek to bring the *relative* (as opposed to absolute) historical contribution of each $x(k)$ in line with the weight $w(k)$. For this purpose we will assume the weights are all nonnegative and normalized to sum to 1. To put the historical contributions on the same footing, we define

$$\mathit{contribution}(k) = \mathit{historical\_score}(k) \Big/ \left(\textstyle\sum \mathit{historical\_score}(h)\colon h\right). \tag{D3}$$

Using this definition, the decision to select an outcome p for the current decision can be evaluated directly by reference to the relationship between the values of *contribution*(k) and $w(k)$ under the condition where outcome p is chosen. Denote the contribution value for this tentative future choice by *contribution*$(k\!:\!p)$. Then we may select a current outcome according to one of the following objectives, as q ranges over outcomes in the current decision set.

$$\operatorname*{Minimize}_{q} \textstyle\sum(|\mathit{contribution}(k\!:\!q) - w(k)|\!: k), \tag{OBJ 1}$$

$$\operatorname*{Minimize}_{q} \left(\operatorname*{Maximum}_{k} |\mathit{contribution}(k\!:\!p) - w(k)| \right), \tag{OBJ 2}$$

$$\operatorname*{Minimize}_{q} \textstyle\sum(\mathit{deficiency}(k\!:\!q)\!: k). \tag{OBJ 3}$$

In the latter minimization, *deficiency*$(k\!:\!q)$ is defined to be the larger of 0 and $w(k)$ − *contribution*$(k\!:\!q)$.

By eliminating all vectors $x(k)$ with $w(k) = 0$ from consideration, objectives based on ratios of contributions to weights can also be created, i.e., minimizing deviations of these ratios from the value 1. Also, the objectives can be modified by subtracting a monotonically increasing function of the number of votes for each outcome, to favor the choice of outcomes that receive higher votes. The appropriateness of specific instances of such objectives for given problem settings is a matter of empirical determination, constituting an avenue for useful applied research. Note that the preceding objectives do not entail solving an optimization problem, but simply involve comparing the quantities whose minimum is sought.

Using definitions of the form indicated, the sequential decision process for creating a structured weighted combination is straightforward. At each step, a preferred outcome is selected from the current decision set by one of the minimization objectives indicated. This outcome gives the current decision, which is implemented by the updating process to create a new residual problem. The selection and updating are then repeated until a complete trial solution is created.

We specifically emphasize the advantage of selecting vectors to create such weighted combinations that use elite local optima as points of departure. Aggressively generating local optima and maintaining a record of the elite members for intensification purposes, as advocated by tabu search, has recently been introduced as a strategy to modify the source of parents and yield improvements in genetic algorithms by Mühlenbein [19]. Following the scatter search approach, this can be augmented by maintaining separate historical sets H of such solutions, each producing its own parallel solution stream, where elements from the union of these sets are periodically reallocated by rules for generating diverse subsets.

4.2. Illustrative applications

We demonstrate how the preceding methodology can be used to generate weighted combinations of permutation vectors, as in traveling salesman and scheduling problems. For simplicity, all decision votes will be assumed 0 and 1, where each permutation vector $x(k)$ is processed to identify a single preferred outcome to which it assigns a vote of 1 at each stage. The current decision set will be restricted to consist of the currently preferred outcomes (those receiving nonzero votes) in determining the outcome selected. We base the weighted combinations in these examples on a single pair of reference points. The admissible weights may be viewed as creating "convex combinations" of the reference vectors (since they are nonnegative and are normalized to sum to 1).

Consider the following the permutations and their associated weights:

$$x(1) = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9 : w(1) = 2/3,$$

$$x(2) = 1\ 8\ 6\ 4\ 9\ 3\ 5\ 2\ 7 : w(2) = 1/3.$$

Example 4.4. We first assume the permutation vectors represent 0–1 votes for precedence relationships, where a vector assigns a vote of 1 for i to precede j if this precedence order occurs in the permutation, and otherwise assigns a vote of 0 to this order. We further suppose the vectors represent a cyclic permutation, as in a directed traveling salesman problem. Hence $x(1)$ and $x(2)$ are both "arbitrarily started" at 1 (and cycle to conclude at 1). Thus the new vector to be constructed will also start at 1. To give each vector a single preferred outcome, we begin by voting on the precedence order $(1, j)$, for j as determined from $x(1)$ and $x(2)$, hence $(1, 2)$ for $x(1)$ and $(1, 8)$ for $x(2)$. Thereafter, at each step, the preferred precedence (i, j) will be restricted so that i is always the element that was last (most recently) added to the new permutation being constructed, and j is the (currently remaining) successor of i in $x(1)$ or $x(2)$.

The updating process underlying Property 4.3 will be the standard update used to shrink a permutation vector on the basis of fixing the order of some of its components. In particular, each time an element i is assigned a position in the new permutation, and used to establish votes by $x(1)$ and $x(2)$ for the next precedence link (i, j) of the new permutation, then i is deleted from $x(1)$ and $x(2)$. This deletion yields an "inherited precedence order" for $x(1)$ and $x(2)$ that assigns new votes of 1 to pairs (i, j) where i did not immediately precede j initially, but does so after stipulated elements have been removed. (The votes for such inherited precedence orders evidently can be composed with greater refinement – e.g., by dividing the standard (unit) vote by the number of elements that originally separated i and j, or dividing by a factor based on comparing the distance from i to j with the distance from i to its original successor. The class of adaptive structured combinations described later provides additional alternatives.)

Following this scheme, the first decision has two outcomes, as identified by the votes of $x(1)$ and $x(2)$: initiating a sequence $(1, 2, \ldots)$ or initiating a sequence $(1, 8, \ldots)$. The first sequence creates a current score of 1 for $x(1)$ and 0 for $x(2)$, while the second creates a current score of 0 for $x(1)$ and 1 for $x(2)$. We evaluate these alternatives by computing their relative contributions, as defined earlier in (D3), and using the "minimum sum of absolute deviations" objective (OBJ 1). This gives $|1 - \frac{2}{3}| + |0 - \frac{1}{3}| = \frac{2}{3}$ for the first outcome and $|0 - \frac{2}{3}| + |1 - \frac{1}{3}| = \frac{4}{3}$ for the second. Choosing the minimum of these yields the sequence that begins as $(1, 2, \ldots)$.

The outcomes to be submitted to a vote on the next step are $(1, 2, 3, \ldots)$ and $(1, 2, 7, \ldots)$, since 3 follows 2 in $x(1)$ and 7 follows 2 in $x(2)$. The associated evaluations are $|\frac{1}{2} - \frac{2}{3}| + |0 - \frac{1}{3}| = \frac{2}{3}$ versus $|\frac{1}{2} - \frac{2}{3}| + |\frac{1}{2} - \frac{1}{3}| = \frac{1}{3}$, hence the next choice yields the sequence $(1, 2, 7, \ldots)$. Repeating the process generates the permutation indicated in Table 1.

The current scores identify which of $x(1)$ and $x(2)$ contributes the decision made at each step, except where both scores are 0, which indicates that both vote for the same decision. This shared vote occurs automatically at the beginning, and also occurs (in this particular example) at the end. The final decision to follow 5 with 6 agrees with the updated vote of $x(2)$ in its residual form, since element 6 is the first remaining successor of 5 at the current stage, considering the cyclic form of the permutations.

A streamlined rule

A useful relationship emerges from this example which can be shown to hold in any setting where $x(1)$ and $x(2)$ have exactly one vote of 1 at each step, and all other votes at that step are 0 (given our selected definition of current scores). Specifically, regardless of the structure of $x(1)$ and $x(2)$, the sequence of current and historical scores will be identical if all steps where both current scores equal 0 are discarded. For the weights $w(1) = \frac{2}{3}$ and $w(2) = \frac{1}{3}$, the progression of current scores shown in Table 1 for $x(1)$ and $x(2)$ completely characterizes the scoring sequence upon eliminating the "double 0" (or shared vote) cases. Hence $x(1)$ endlessly repeats the sequence $(0, 1, 1)$ after the second step. The corresponding score sequence for $x(2)$ complements that of $x(1)$, and hence endlessly repeats $(1, 1, 0)$.

This relationship makes it possible to determine in advance the relevant decision sequence for combining any two permutations, given their weights (and a rule for breaking tied evaluations). Such a property does not in general hold for other definitions of current scores. (For example, the definition that sets $current\_score(k) = v(k, p)$, although perhaps appropriate in some settings, can cause a more heavily weighted vector to determine the selected outcome of every contested vote, provided enough shared votes occur at steps where the less heavily weighted vector normally would dictate the selection.) We will use this knowledge to facilitate the treatment of the next examples.

Example 4.5. The permutation vectors for this example will be interpreted to represent 0-1 votes for selecting the first available element in $x(1)$ or $x(2)$ to become the next

Table 1

Permutation	Current scores		Historical scores	
	$x(1)$	$x(2)$	$x(1)$	$x(2)$
1	0	0	0	0
2	1	0	1	0
7	0	1	2	1
8	1	0	3	1
9	1	0	3	1
3	0	1	4	2
4	1	0	5	2
5	1	0	5	2
6	0	0	5	2

element of the permutation being constructed. Such an interpretation is relevant to classes of scheduling problems where the objective is to minimize lateness penalties for processing jobs to meet their due dates.

The updating procedure for $x(1)$ and $x(2)$ is again to delete elements as they become assigned to the new permutation, while constructing this permutation in sequential order. At each iteration, instead of examining ordered pairs to identify precedence orders, the first remaining element of each of $x(1)$ and $x(2)$ defines the two possible continuations of the new permutation and, hence, determines the current decision set. (Thus, $X(1)$ assigns a vote of 1 to its first remaining element and $x(2)$ likewise assigns a vote of 1 to its first remaining element.) The scores and relative contributions are calculated as before, selecting the preferred outcome by the "minimum sum of deviations" objective (Obj 1). Table 2 shows the permutation that results.

The scoring (and hence decision) sequence is identical to the one before, but the permutation generated from this sequence is different. This results from the different definition of the current decision set appropriate to this context. Also, in the present example, no situation occurs where both current scores are 0, other than at the start.

Example 4.6. We now consider a setting of the type encountered in facility location problems, and interpret $x(1)$ and $x(2)$ to identify 0–1 votes for assigning an element to an exact position, or as close to this position as possible. We employ a simple sequential decision process that visits the positions one at a time. As before, $x(1)$ and $x(2)$ are updated by deleting elements assigned to the new permutation, but in this context it is appropriate to maintain a record of the original position each element occupied. Then, in succession, for each position $j = 1, \ldots, n$ of the new permutation, $x(1)$ and $x(2)$ each casts a vote of 1 to fill this position with its currently remaining element whose original position was closest to j, breaking ties to choose an element whose original position (of $x(1)$ or $x(2)$, respectively) is already filled. Using the same scoring and evaluation process of the preceding examples, the new vector created by this rule appears in Table 3.

Table 2

Permutation	Current scores	
	$x(1)$	$x(2)$
1	0	0
2	1	0
8	0	1
3	1	0
4	1	0
6	0	1
5	1	0
7	1	0
9	0	1

Table 3

Permutation	Current scores	
	$x(1)$	$x(2)$
1	0	0
2	1	0
6	0	1
4	0	0
5	1	0
7	1	0
3	0	1
8	1	0
9	1	0

The sequence previously established for current scores reappears, broken here in the fourth position where both $x(1)$ and $x(2)$ vote for the same outcome, hence yielding a 0 score for each. The resulting permutation is again notably different than those preceding.

Other voting rules appropriate to this example may allow $x(1)$ and $x(2)$ to cast nonzero votes, not necessarily equal to 1, for more than one potential assignment at each stage. Likewise, instead of pre-selecting a sequence for visiting positions, the choice of the position to be filled can be a consequence of the voting and evaluation process. Such alternatives are carried a step further in the treatment of adaptive structured combinations in the next section.

4.3. *Adaptive structured combinations*

Instead of establishing weights in advance to determine the influence of each $x(k)$, we identify an alternative approach that allows this influence to be determined adaptively, based on the evaluation of outcomes relative to the problem of interest. (As before, we stress the value of an aggressive tabu search orientation that seeks to generate locally optimal $x(k)$ as points of departure.) This approach applies the ideas previously developed, but places additional reliance on the existence of a criterion for measuring the worth of outcomes voted upon by the $x(k)$ vectors. Specifically, we require that each outcome in the current decision set can be evaluated efficiently to determine an objective function value for a trial solution associated with this outcome (or an approximate measure of such a value).

The word "efficiently" although imprecise, is important in this statement. The trial solution referred to may or may not be the trial solution characterized in Property 4.2, e.g., it may be a trial solution to a problem relaxation that weakens or disregards selected constraints and, therefore, may violate feasibility requirements. Alternatively, as in job shop scheduling, it may be a look-ahead trial solution, where the objective function value does not apply to the complete problem of interest, but constitutes a measure attached to a local decision rule.

By this interpretation, it is evident that the indicated evaluation property (in common with Properties 4.1–4.3), is readily satisfied by many standard sequential procedures for generating solutions to optimization problems.

To take advantage of this property, we are interested in sequential procedures whose decision sets offer a relatively diverse range of alternatives (in contrast to the "two alternative" decision sets of the earlier examples) and whose updating processes can be characterized as *parsimonious*. That is, we seek updating procedures that minimally change the structure and information content of reference vectors, subject to preserving Properties 4.1–4.3.

The notion of parsimony has an important implication for the goal of creating a new vector that embodies a high-quality solution. A trial solution used for evaluation should not only encompass decisions previously made but should characteristically depend upon the votes for decisions remaining. (In this way, the goal of obtaining a good new solution is tempered by the goal of reflecting the influence of the reference vectors, under the assumption that their guidance may lead to discovering alternatives that could otherwise be missed.) The parsimony condition is a means for maintaining a continuity in the dependency on the reference vectors as successive update steps are performed. By keeping each updated $x(k)$ "close to" its previous form, the guidance prescribed by the original vector will be more faithfully replicated across successive updates.

The concept of parsimony derives from the notion of *strongly determined and consistent variables* [8], which underlies one of the basic forms of intensification proposed in tabu search.[3] In general, the goal is to reinforce the selections of elements that occur repeatedly (or instrumentally) in subsets of elite solutions. (Characterization of such subsets for establishing measures of repetition, hence "consistency", may appropriately be based on clustering strategies.)

Three types of implementations provide examples of how the vector updating process may be structured. First is to take the separate solutions prescribed by the $x(k)$ vectors and merge them (e.g., by a "modified sum" or "union" operation that maintains variables within their bounds), and then create a trial solution by progressively adjusting variables in the merged solution until feasibility conditions are recaptured. Adjustments that minimally change the merged solution values approximate the goal expressed in the parsimony condition. Such an approach is natural for treating multiconstraint knapsack and covering problems. For example, a merged solution can be created by assigning a value of 1 to each variable that takes this value in at least one of the solutions prescribed by the $x(k)$ vectors. Then a standard successive elimination rule (e.g., based on surrogate constraint ratios [8]) can be used to set a minimal number of variables to 0 to recapture feasibility (or in covering, to set a maximal number of variables to 0 while retaining feasibility). Using heuristic guidance to determine adjustments leads the method adaptively to produce a weighted combination that is likely to be superior to one derived from "voting on" these adjustments according to predetermined weights.

A second approach similarly begins by merging solutions, but replaces the adjustment sequence by a construction sequence, using the merged vector as a target. The goal is to progressively assign values to variables to come close to this target. Again, heuristic guidance may be used to determine the outcome of each choice step. For example, a higher form of merging and progressive targeting occurs by creating a goal programming augmentation of the original problem, with penalties attached to deviating from target values. Applying choice rules based on deviations from targets in the resulting solution, variables are progressively constrained to specific values, repeating the process until a trial solution is obtained. (Alternately, such approaches can operate with states of variables, rather than with variables, as by targeting elements to be basic and nonbasic in constructing an extreme point solution.)

[3] Computational evidence supporting the relevance of the parsimony condition is provided in a genetic algorithm context by the study of Whitley et al. [20] which disclosed that an edge-preserving crossover operator for traveling salesman problems leads to better performance. More recently, Mühlenbein [19] also has demonstrated the merit of such a strategy for traveling salesman problems, using what he calls a "maximal preservative crossover operator". Our present development allows the parsimony concept to be applied in more general problem settings, without restriction to the use of GA crossover operations.

A third approach, illustrated subsequently in fuller detail, does not merge the reference vectors, but progressively transforms each of them into the new vector being created. The vectors are made to embody the decisions prior to the current stage as a basis for generating the current trial solution, and when the construction is complete, all information in the new vector is encompassed within each $x(k)$ vector. The parsimony condition causes the new vector to incorporate the structures of the reference vectors more fully, as they converge to a common identity.

In each of the preceding approaches, the relative contributions of the $x(k)$ vectors do not conform with predetermined weights, but rather adapt to the goal of producing a new vector of high quality, within the boundaries accepted for remaining faithful to the guiding influence of the $x(k)$ vectors. The weights of the reference vectors, in effect, are determined after the fact.

Such a design is especially relevant to the scatter search philosophy. In particular, when structured combinations are employed, it appears appropriate to create at least one adaptively generated point for each search line.

Example implementation of adaptive structured combinations

We illustrate the preceding ideas in application to a symmetric traveling salesman problem (TSP). We will demonstrate the approach that progressively transforms each $x(k)$ into the new vector being constructed. For convenience, we will operate with only two reference (permutation) vectors, $x(1)$ and $x(2)$, as in earlier illustrations. Our goal will be to clarify the manner in which adaptive structuring can occur in a simple setting, rather than to identify the most effective version of the approach.

To build a TSP solution, or tour, we iteratively select an edge identified by a pair of adjacent elements (i, j) from $x(1)$ or $x(2)$, requiring that this edge be in the new vector, and also in the updated form of both $x(1)$ and $x(2)$. Once chosen, such an edge is designated *fixed* and other edges are designated *free*.

To maintain a broader range of alternatives than in preceding examples, we consider every free edge in both $x(1)$ and $x(2)$ to be a candidate for selection at the next step. The update procedure that assures $x(1)$ and $x(2)$ both contain all edges assigned to the new solution will automatically give the trial solution construction to evaluate potential decisions.

To reduce total computation, we divide the evaluation into two stages. The first stage picks the smallest length free edge from each of $x(1)$ and $x(2)$ as a candidate for the next stage of evaluation. The second stage evaluates these candidate edges as follows.

First suppose the edge to be evaluated comes from $x(1)$. If the edge is selected to become part of the new tour, the operation of fixing the edge in $x(1)$ will not cause a change in $x(1)$. Similarly, if the edge is already in $x(2)$, no change will result in this vector. If the edge is not in $x(2)$, it must be incorporated into $x(2)$. The way to do this is illustrated in Figs. 1–4.

The traveling salesman tour of Fig. 1 represents the permutation vector $x(2)$ which must incorporate the candidate edge from $x(1)$. The heavy lines indicate fixed edges and the dotted line identifies the candidate $x(1)$ edge, denoted as edge (i, j).

To incorporate edge (i, j) into the permutation vector for $x(2)$, one of the two edges meeting node i and one of the two edges meeting node j must be dropped, excluding

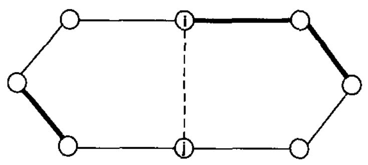

Fig. 1.

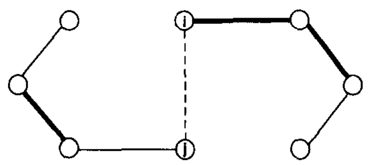

Fig. 2.

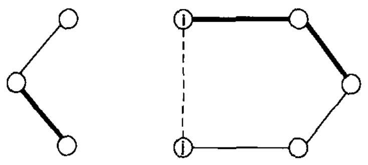

Fig. 3.

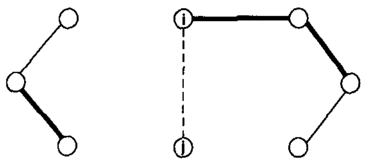

Fig. 4.

fixed edges. (At least one such free edge must meet each endpoint of edge (i, j), given that this edge is not yet fixed in $x(1)$.) There is only one option in Fig. 1 to drop a free edge meeting node i. Fig. 2 shows one of the two combinations that result by dropping edges meeting nodes i and j as indicated.

The graph of Fig. 2 that includes (i, j) is a connected chain that can be joined at its endpoints to create a new tour. This operation accords with the goal of creating a parsimonious update since it changes $x(2)$ by a minimal amount. The other possible combination for dropping two edges is shown in Fig. 3.

In this case the graph has two components, a chain and a cycle. (The chain could also consist of a single node.) A parsimonious update to change this structure into a tour results by dropping a single free edge from the cycle (at least one will always exist), and then joining the resulting new chain with the previously created chain (or node). To keep the structure "close to" the original structure, we may elect to drop a free edge that lies nearest to edge (i, j). This results in the outcome of Fig. 4.

The two chains of Fig. 4 finally may be connected (in either of two ways) to complete the update. Each of the resulting tours (including the tour obtained from Fig. 2) yields a trial solution, and the one with the best objective function value may be designated the preferred outcome. (The updates depicted in these diagrams constitute special cases of 2-exchange and 3-exchange operations, guided by the structure of the reference vectors.)

To complete the process initiated in the second stage, recall that edge (i, j) from $x(1)$ represents one of two candidate edges to be considered for possible selection. Hence a corresponding candidate from $x(2)$, say (h, k), will similarly be considered. The preferred updating outcome for (i, j) and the associated preferred outcome for (h, k) provide the trial solution evaluations for determining the candidate edge to incorporate next into the new tour.

Depending on the effort to be allotted to such a process, several edges from $x(1)$ and $x(2)$ may be selected in the first stage screening as candidates to be added to the new tour. (A screening process may also be applied to choose an edge to drop from a cycle, as illustrated in Fig. 3, rather than selecting an edge that lies "nearest" (i, j).) The choice rule of the first stage screening also can begin by selecting all edges common to $x(1)$ and $x(2)$, hence fixing the intersection of these tours before examining other edges.

Once all edges of both tours become fixed, the tours coincide with the new tour and the construction is complete. This same type of process also can be used to generate a structured *weighted* combination of $x(1)$ and $x(2)$, by the rules illustrated earlier, by allowing each step to select either a preferred $x(1)$ choice or a preferred $x(2)$ choice according to which of $x(1)$ and $x(2)$ wins the current voting competition.

A graph partitioning illustration

To show how the preceding ideas apply to problems of significantly different forms, we conclude with an example applied to graph partitioning. The objective for this problem is to divide a collection of $2n$ nodes into sets P and Q, containing n nodes each, to minimize the sum of weights of edges connecting these two sets (i.e., to minimize $\sum(w(p, q): p \in P,\ q \in Q)$, where $w(p, q)$ denotes the weight of edge (p, q)).

A reference vector $x(k)$ for this problem may be construed as a pair of sets $P(k)$, $Q(k)$ identifying a candidate partition. The goal of obtaining an adaptive structured combination will be illustrated by reference to two partitions $P(1)$, $Q(1)$ and $P(2)$, $Q(2)$. As in the approach previously illustrated, we will progressively transform these partitions until they become the same, coinciding in this case with a new partition P, Q we seek to create.

The process may be implemented straightforwardly as follows: maintaining P as the intersection of $P(1)$ and $P(2)$ and Q as the intersection of $Q(1)$ and $Q(2)$. At each iteration we identify one new element u to add to P and one new element v to add to Q, selecting u and v to be exchanged either between the sets $P(1)$ and $Q(1)$ or between the sets $P(2)$ and $Q(2)$, but not both. The two options are: (1) select u from $Q(1) - Q(2)$ and

v from $P(1) - P(2)$, adding u to $P(1)$ and v to $Q(1)$; (2) select u from $Q(2) - Q(1)$ and v from $P(2) - P(1)$, adding u to $P(2)$ and v to $Q(2)$. (Since $Q(1) - Q(2) = P(2) - P(1)$ and $Q(2) - Q(1) = P(1) - P(2)$, the first exchange does not alter $P(2)$ or $Q(2)$, and the second does not alter $P(1)$ or $Q(1)$.)

A preferred alternative from these possibilities can be selected by considering the change in the objective function evaluation for the pair of sets whose composition changes. (The effort of the selection process can be reduced by a first-stage screening to select a best element for a "half-exchange" from $P(k)$ to $Q(k)$ or from $Q(k)$ to $P(k)$, for $k = 1, 2$. Then each of these four half-exchanges is completed by identifying the best element to transfer to the alternate set.) We note that the same approach can produce weighted structured combinations by selecting the winner ($k = 1$ or 2) are each step according to the weighting and scoring criteria specified earlier, instead of by the objective function evaluation. The considerations of the next section include an accelerated procedure for graph partitioning.

Parallel processing and special case simplifications

Conspicuous opportunities for parallel processing arise in the creation of structured combinations. For example, a collection of new vectors (instead of only one) can be generated by selecting more than one outcome from the current decision set. Members of this collection in turn may be used to influence the voting process. Parallel implementation of a more global form results by operating with different sets of reference vectors simultaneously. As noted earlier, the trial solution evaluations used to generate adaptive structured combinations can also be used to narrow the decision sets for generating weighted structured combinations.

Issues of trade-offs are relevant in applying the illustrated rules. For example, in creating adaptive structured combinations the effort spent at each step to identify highest evaluation alternatives must be balanced against the option of generating more alternatives, to take effective advantage of opportunities made available by parallel processing. In either event, selected members of the new vectors produced may be subjected to post-processing by standard improvement processes such as embodied in tabu search.

The definitions (D1)–(D3) and their associated objectives for creating weighted structured combinations can be simplified to exploit similar trade-offs. For special circumstances where the goal may be to generate large numbers of vectors rapidly, rather than emphasize the relative influence of the reference vectors, two "relaxed versions" of the preceding ideas are as follows. First, replace the votes $v(k,j)$ for alternative outcomes j by weighted votes $v'(k,j) = w(k)v(k,j)$. Then select an outcome that receives a largest number of weighted votes. (This approach permits the use of negative weights.) Second, assuming the weighted votes are nonnegative, define $v'(j) = \sum(v'(k,j){:}\ k)$ and $v' = \sum(v'(j){:}\ j)$. Then assign a probability of $v'(j)/v'$ to selecting outcome j from the current decision set. These same ideas can be applied by replacing weighted votes with weighted contributions, as by reference to the definitions (D1)–(D3).

Variants that combine the themes of the preceding simple rules are also possible. For example, such a variant can be constructed for the earlier graph partitioning illustration as follows. First remove $P(1) \cap P(2)$ from $P(1)$ and $P(2)$ and put it in P, leaving k elements (say) in each of $P(1)$ and $P(2)$. Then, using the normalized weights $w(1)$ and $w(2)$ as proportions, select (either heuristically or randomly) $kw(1)$ elements from $P(1)$ and $kw(2)$ elements from $P(2)$ to fill the remainder of P, rounding fractions appropriately. Q then receives all elements not assigned to P. Analogous variants, directly based on Properties 4.1–4.3 (and on simple alterations of the definitions (D1)–(D3) and their associated evaluation objectives), can readily be tailored to specific contexts. (In fact, in the present example these definitions are not altered, but are simply implemented efficiently, using characterizations of votes and current decision sets appropriate to the graph partitioning context.)

The relative effectiveness of such indicated ways of creating structured combinations will depend on the application considered. It should be noted that most contexts susceptible to treatment by branch and bound can be encompassed within the proposed framework, because the sequential branching processes of branch and bound typically are included under those governed by Properties 4.1–4.3. In addition, these properties also hold in settings where branch and bound is inconvenient to apply or is not highly effective. (For example, they can be applied in local improvement processes where the number of alternatives to be considered by branch and bound is overwhelming [16].)

5. Connections with genetic algorithms

Analogies evidently can be drawn between the approaches of scatter search and genetic algorithms, in spite of their somewhat different origins and emphasis. Loosely summarized, scatter search unites preferred subsets of reference points to generate trial points by weighted linear combinations, and selects the best members to become the source of new reference points. Genetic algorithms unite alternative parents, paired pseudorandomly, to generate offspring by operations such as "crossover" and "mutation", and select best members to become new parents.

The proposed integration of tabu search and scatter search thus suggests analogous ways to integrate tabu search and genetic algorithms. A few modifications are required in the way GA's are implemented to give them a form more nearly resembling scatter search – as by creating and managing a set of elite historical generators, systematically re-using selected points as parents (reference points) at each iteration, and allowing combinations of parents to be created strategically instead of randomly. We have already cited several instances where genetic algorithms have begun to be modified to incorporate such elements, with advantageous results.

More broadly, the use of structured combinations makes it possible to combine component vectors in a way that is materially different from the result of crossover operations. Integrating such an approach with genetic algorithms may open the door to new types of search procedures. The fact that weighted and adaptive structured combinations can readily be created to exploit contexts where crossover has no evident meaning (or has difficulty ensuring feasibility), suggests that such integrated search procedures may have benefits in settings where genetic algorithms presently have limited application.

Acknowledgement

This research was supported in part by the Joint Air Force Office of Scientific Research and Office of Naval Research Contract No. F49620-90-C-0033 at the University of Colorado.

References

[1] W.B. Crowston, F. Glover, G.L. Thompson and J.D. Trawick, Probabilistic and parametric learning combinations of local job shop scheduling rules, ONR Research Memorandum No. 117, GSIA, Carnegie-Mellon University, Pittsburg, PA.
[2] M.L. Fisher, Optimal solution of scheduling problems using generalized Lagrangian multipliers: Part I, Oper. Res. 21 (1973) 1114–1127.
[3] M.L. Fisher, W.D. Northrup and J.F. Shapiro, Using duality to solve discrete optimization problems: Theory and computation experience, Math. Programming Stud. 3 (1975) 59–94.
[4] A. Freville and G. Plateau, An efficient preprocessing procedure for the solution of 0–1 multiknapsack problems to optimality, Math. Programming 31 (1991) 78–105.
[5] B. Gavish and H. Pirkul, Efficient algorithms for solving multiconstraint zero–one knapsack problems to optimality, Math. Programming 31 (1985) 78–105.
[6] A.M. Geoffrion, Lagrangian relaxation and its uses in integer programming, Math. Programming Stud. 2 (1974) 82–114.
[7] F. Glover, A multiphase-dual algorithm for the zero–one integer programming problem, Oper. Res. 13 (1965) 897–919.
[8] F. Glover, Heuristics for integer programming using surrogate constraints, Decision Sci. 8 (1977) 156–166.
[9] F. Glover, Future paths for integer programming and links to artificial intelligence, Comput. Oper. Res. 13 (1986) 533–549.
[10] F. Glover, Artificial intelligence, heuristic frameworks and tabu search, Managerial and Decision Economics 11 (1990) 365–375.
[11] F. Glover and D. Klingman, Layering strategies for creating exploitable structure in linear and integer programs, Math. Programming 40 (1988) 165–182.
[12] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning (Addison-Wesley, Reading, MA, 1988).
[13] H.J. Greenberg and J.S. Maybee, Computer-Assisted Analysis and Model Simplification (Academic Press, New York, 1980).
[14] H.J. Greenberg and W.P. Pierskalla, Surrogate mathematical programming, Oper. Res. 18 (1990) 924–939.
[15] A. Hertz and D. de Werra, The tabu search metaheuristic: how we used it, Ann. Math. Artificial Intelligence 1 (1991) 111–121.
[16] M. Laguna, W. Barnes and F. Glover, Scheduling jobs with linear delay penalties and sequence dependent costs and times, Department of Mechanical Engineering, University of Texas, Austin, TX; also: Appl. Intelligence, to appear.
[17] G.E. Liepens and M.D. Vose, Representational issues in genetic algorithms, Res. Rep., Oak Ridge National Laboratories.
[18] Z. Michalewicz, Vignaux and M. Hobbs, A non-standard genetic algorithm for the nonlinear transportation problem, ORSA J. Comput. 3 (1991) 307–316.
[19] H. Muhlenbein, Parallel genetic algorithms and combinatorial optimization, SIAM J. Optim., to appear.
[20] D. Whitley, T. Starkweather and D. Fuquay, Scheduling problems and traveling salesmen: the genetic edge recombination operator, in: J.D. Schaffer, ed., Proceedings 3rd International Conference of Genetic Algorithms, Fairfax, VA (Morgan Kaufman, San Mateo, CA, 1989) 133.
[21] J. Zhang, N. Kim and L. Lasdon, An improved successive linear programming algorithm, Management Sci. 31 (1985) 1312–1331.