Scatter Search and Path Relinking: Foundations and Advanced Designs

FRED GLOVER[*], MANUEL LAGUNA

Leeds School of Business, University of Colorado, Boulder, CO 80309-0419, USA. Fred.Glover@Colorado.edu

RAFAEL MARTÍ[*]

Departamento de Estadística e Investigación Operativa Universitat de València, 46100 Burjassot, Spain. Rafael.Marti@uv.es

Latest version: June 27, 2002

Abstract — Scatter Search and its generalized form Path Relinking, are evolutionary methods that have been successfully applied to hard optimization problems. Unlike genetic algorithms, they operate on a small set of solutions and employ diversification strategies of the form proposed in Tabu Search, which give precedence to strategic learning based on adaptive memory, with limited recourse to randomization. The fundamental concepts and principles were first proposed in the 1970s as an extension of formulations, dating back to the 1960s, for combining decision rules and problem constraints. (The constraint combination approaches, known as surrogate constraint methods, now independently provide an important class of relaxation strategies for global optimization.) The Scatter Search framework is flexible, allowing the development of alternative implementations with varying degrees of sophistication. Path Relinking, on the other hand, was first proposed in the context of the Tabu Search metaheuristics, but it has been also applied with a variety of other methods. This chapter's goal is to provide a grounding in the essential ideas of Scatter Search and Path Relinking, together with pseudo-codes of simple versions of these methods, that will enable readers to create successful applications of their own.

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

[*] Research partially supported by the Ministerio de Educación, Cultura y Deporte of Spain: PR2002-0060.

1. Introduction

Scatter Search (SS) and Path Relinking (PR) are evolutionary methods that construct solutions by combining others by means of strategic designs that exploit the knowledge on the problem at hand. The goal of these procedures is to enable a solution procedure based on the combined elements to yield better solutions than one based on the original elements. SS and PR are rapidly becoming methods of choice for designing solution procedures for hard combinatorial optimization problems. A comprehensive examination of these methodologies can be found in Glover (1997, 1999) and Glover, Laguna and Martí (2000, 2002).

In common with other evolutionary methods, SS and PR operate with a population of solutions, rather than with a single solution at a time, and employ procedures for combining these solutions to create new ones. The meaning of "combining," and the motivation for carrying it out, has a rather special origin and character in the SS/PR setting, however. One of the distinguishing features of this approaches is its intimate association with the Tabu Search (TS) metaheuristic, and hence its adoption of the principle that search can benefit by incorporating special forms of adaptive memory, along with procedures particularly designed for exploiting that memory. In fact, Scatter Search and Tabu Search share common origins, and initially SS was simply considered one of the component processes available within the TS framework. However, most of the TS literature and the preponderance of TS implementations have disregarded this component, with the result that the merits of Scatter Search did not come to be recognized until quite recently, when a few studies began to look at it as an independent method in the context of evolutionary procedures.

Unlike a "population" in genetic algorithms, the reference set of solutions in SS and PR is relatively small. A typical population size in a genetic algorithm consists of 100 elements, which are sampled by various schemes with a significant reliance on randomization to create combinations. In contrast, SS and PR chooses two or more elements of the reference set in a systematic way with the purpose of systematically creating new solutions. Randomization, where employed, is given a highly controlled and strategic character. (Some of the more recent GA proposals are beginning to incorporate some of the features of early SS and PR designs, including an adoption of some of the SS and PR terminology. However, a number of key elements continue to distinguish the SS and PR methods from GA approaches, with important consequences for performance.)

The process of generating combinations of a set of reference solutions in Scatter Search consists of forming linear combinations that are structured to accommodate discrete requirements (such as those for integer-valued variables). The resulting combinations may be characterized as generating paths between and beyond these solutions, where solutions on such paths also serve as sources for generating additional paths. Path Relinking is based on this conception of the meaning of creating *combinations* of solutions, but making reference to paths between and beyond selected solutions in neighborhood space, rather than in Euclidean space as in the case of Scatter Search. (More precisely, SS operates within a subset of Euclidean space determined by imposing certain constraints according to the problem setting.)

The following principles summarize the foundations of the SS and PR methodologies:

- Useful information about the form (or location) of optimal solutions is typically contained in a suitably diverse collection of elite solutions.
- When solutions are combined as a strategy for exploiting such information, it is important to provide mechanisms capable of constructing combinations that extrapolate beyond the regions spanned by the solutions considered. Similarly, it is also important to incorporate heuristic processes to map combined solutions into new solutions. The purpose of these combination mechanisms is to incorporate both diversity and quality.
- Taking account of multiple solutions simultaneously, as a foundation for creating combinations, enhances the opportunity to exploit information contained in the union of elite solutions.

In this paper we describe the foundations of these closely related methods and discuss some extensions and advanced designs that have been proved effective on solving hard combinatorial optimization problems. Section 2 is devoted to the foundations of both methods, where we provide pseudo-code of

simple versions of each. Extensions and advanced strategies are described in Section 3, to conclude the paper.

2. Foundations

2.1 Scatter Search

Scatter Search consists of five component processes:

1. A *Diversification Generation Method* to generate a collection of diverse trial solutions, using one or more arbitrary trial solutions (or seed solutions) as an input.

2. An *Improvement Method* to transform a trial solution into one or more enhanced trial solutions. (Neither the input nor the output solutions are required to be feasible, though the output solutions are typically feasible. If the input trial solution is not improved as a result of the application of this method, the "enhanced" solution is considered to be the same as the input solution.)

3. A *Reference Set Update Method* to build and maintain a *reference set* consisting of the b "best" solutions found (where the value of b is typically small, e.g., no more than 20), organized to provide efficient accessing by other parts of the solution procedure. Several alternative criteria may be used to add solutions to the reference set and delete solutions from the reference set.

4. A *Subset Generation Method* to operate on the reference set, to produce a subset of its solutions as a basis for creating combined solutions. The most common subset generation method is to generate all pairs of reference solutions (i.e., all subsets of size 2).

5. A *Solution Combination Method* to transform a given subset of solutions produced by the Subset Generation Method into one or more combined solutions. The combination method is analogous to the crossover operator in genetic algorithms but it must be capable of combining two or more solutions.

The structured combinations produced by Scatter Search are designed with the goal of creating weighted centers of selected sub-regions. These include non-convex combinations that project new centers into regions that are external to the original reference solutions. The dispersion patterns created by such centers and their external projections have been found useful in a variety of application areas.

Another important feature relates to the strategies for selecting particular subsets of solutions to combine. These strategies are typically designed to make use of a type of clustering to allow new solutions to be constructed "within clusters" and "across clusters". The method is generally organized to use improving mechanisms that are able to operate on infeasible solutions, removing the restriction that solutions must be feasible in order to be included in the reference set.

The basic procedure for a minimization problem given in Figure 1 starts with the creation of an initial reference set of solutions (*RefSet*). The Diversification Generation Method is used to build a large set of diverse solutions P. The size of P (*PSize*) is typically 10 times the size of *RefSet*. An *Improvement Method* is applied to each generated solution to obtain a better solution, which is added to P (often, though not invariably, as a replacement for the solution it was derived from). Initially, the reference set *RefSet* consists of b distinct and maximally diverse solutions from P. The solutions in *RefSet* are ordered according to quality, where the best solution is the first one in the list. The search is then initiated by assigning the value of TRUE to the Boolean variable *NewSolutions*. In step 3, *NewSubsets* is constructed and *NewSolutions* is switched to FALSE. We focus our attention in this basic illustrative scheme to subsets of size 2, and specify that the cardinality of *NewSubsets* corresponding to the initial reference set is given by $(b^2-b)/2$, which accounts for all pairs of solutions in *RefSet*. The pairs in *NewSubsets* are selected one at a time in lexicographical order and the Solution Combination Method is applied to generate one or more solutions in step 5. The *Improvement Method* is applied again to every newly created solution. If the final solution improves upon the worst solution currently in *RefSet*, the new

solution replaces the worst and *RefSet* is reordered in step 6. The *NewSolutions* flag is switched to TRUE and the subset *s* that was just combined is deleted from *NewSubsets* in steps 7 and 8, respectively.

1. Start with $P = \emptyset$.
   - Use the Diversification Generation Method to construct a solution *y*.
   - Apply the Improvement Method to y. Let x be the final solution.
   - If $x \notin P$ then add *x* to *P* (i.e., $P = P \cup x$), otherwise, discard *x*.
   - Repeat this step until $|P| = PSize$. Build *RefSet* = { x^1, …, x^b } with *b* diverse solutions in *P*.
2. Evaluate the solutions in *RefSet* and order them according to their objective function value such that x^1 is the best solution and x^b the worst. Make *NewSolutions* = TRUE.

while (*NewSolutions*) **do**

3. Generate *NewSubsets*, which consists of all pairs of solutions in *RefSet* that include at least one new solution. Make *NewSolutions* = FALSE.

while (*NewSubsets* $\neq \emptyset$) **do**

4. Select the next subset *s* in *NewSubSets*.
5. Apply the Solution Combination Method to *s* to obtain one or more new solutions *y*.
6. Apply the Improvement Method to *y* and obtain *x*.

if (*x* is not in *RefSet* and $f(x) < f(x^b)$) **then**

6. Make $x^b = x$ and reorder *RefSet*.
7. Make *NewSolutions* = TRUE.

end if

8. Delete *s* from *NewSubsets*.

end while

end while

Figure 1. Outline of a simple scatter search approach.

This illustrative procedure is very aggressive in trying to improve upon the quality of the solutions in the current reference set, to the extent that it sacrifices search diversity. In fact, the Diversification Generation Method is used only once to generate *PSize* different solutions at the beginning of the search and is never employed again. The initial *RefSet* is built by selecting a solution from *P* and then making *b*-1 more selections in order to maximize the minimum distance between the candidate solution and the solutions currently in *RefSet*. That is, for each candidate solution *x* in *P-RefSet* and reference set solution *y* in *RefSet*, we calculate a measure of distance or dissimilarity $d(x,y)$. We then select the candidate solution that maximizes $d_{\min}(x) = \min_{y \in RefSet} \{ d(x,y) \}$.

The updating of the reference set is based on improving the quality of the worst solution and the search terminates when no new solutions are admitted to *RefSet*. The Subset Generation Method is also very simple and consists of generating all pairs of solutions in *RefSet* that contain at least one new solution. This means that the procedure does not allow for two solutions to be subjected to the Combination Method more than once. In the next section, several improvements to this basic scheme are proposed.

2.2 Path Relinking

Path relinking was originally suggested as an approach to integrate intensification and diversification strategies in the context of tabu search (Glover, 1994; Glover and Laguna, 1997). This approach generates new solutions by exploring trajectories that connect high-quality solutions, by starting from one of these solutions, called an *initiating solution*, and generating a path in the neighbourhood space that leads toward the other solutions, called *guiding solutions*. This is accomplished by selecting moves that introduce attributes contained in the guiding solutions.

PR can be considered an extension of the combination mechanisms of Scatter Search. Instead of directly producing a new solution when combining two or more original solutions, PR generates paths between and beyond the selected solutions in the neighborhood space. The character of such paths is easily specified by reference to solution attributes that are added, dropped or otherwise modified by the moves executed. Examples of such attributes include edges and nodes of a graph, sequence positions in a schedule, vectors contained in linear programming basic solutions, and values of variables and functions of variables.

To generate the desired paths, it is only necessary to select moves that perform the following role: upon starting from an *initiating solution*, the moves must progressively introduce attributes contributed by a *guiding solution* (or reduce the distance between attributes of the initiating and guiding solutions). The roles of the initiating and guiding solutions are interchangeable; each solution can also be induced to move simultaneously toward the other as a way of generating combinations. First consider the creation of paths that join two selected solutions x' and x'', restricting attention to the part of the path that lies 'between' the solutions, producing a solution sequence $x' = x(1), x(2), \ldots, x(r) = x''$. To reduce the number of options to be considered, the solution $x(i + 1)$ may be created from $x(i)$ at each step by choosing a move that leaves a reduced number of moves remaining to reach x.

Instead of building a *Reference Set* as in SS, Path Relinking usually starts from a given set of elite solutions obtained during a search process. To simplify the terminology, we will also let *RefSet* (short for "Reference Set"), refer to this set of *b* solutions that have been selected during the application of the embedded search method. This method can be Tabu Search, as in Laguna et al. (1999), GRASP, as in Laguna and Martí (1999), or simply a diversification generator coupled with an improvement method as proposed in Scatter Search. From this point of view, SS and PR can be considered pool-oriented methods (Greistorfer and Voss, 2001) that operate on a set of reference solutions and basically differ in the way in which the Reference Set is constructed, maintained, updated and improved.

In the basic scheme of SS proposed above, all pairs of solutions in the RefSet are subjected to the combination method. Similarly, in this basic version of PR we will consider all pairs in the RefSet to perform a relinking phase. (For each pair (x', x'') two paths are considered; one from x' to x'' and the other from x'' to x'.)

Several studies have experimentally found that it is convenient to add a local search exploration from some of the generated solutions within the relinking path, as proposed in Glover (1994), in order to produce improved outcomes. We refer the reader to Laguna and Martí (1999), Piñana et al. (2001) or Laguna et al. (1999) for some examples. Note that two consecutive solutions after a relinking step are very similar and differ only in the attribute that was just introduced. Therefore, it is generally not efficient to apply an Improvement Method at every step of the relinking process. We introduce the parameter *NumImp* to control its application. In particular, the Improvement Method is applied every *NumImp* steps of the relinking process. (An alternative suggested in Glover (1994) is to keep track of a few "best solutions" generated during the path trace, or of a few best neighbors of the solutions generated, and then return to these preferred candidate solutions to initiate the improvement process.)

Figure 2 shows a simple PR procedure for a minimization problem. It starts with the creation of an initial set of *b* elite solutions (*RefSet*). As in the SS method, the solutions in *RefSet* are ordered according to quality, and the search is initiated by assigning the value of TRUE to *NewSolutions*. In step 3, *NewSubsets* is constructed with all the pairs of solutions in *RefSet*, and *NewSolutions* is switched to FALSE. The pairs in *NewSubsets* are selected one at a time in lexicographical order and the Relinking Method is applied to generate two paths of solutions in steps 5 and 7. The *Improvement Method* is applied every *NumImp* steps of the relinking process in each path (steps 6 and 8). Each of the generated solutions in each path (including also those obtained after the application of the Improvement Method), is checked to see whether it improves upon the worst solution currently in *RefSet*. If so, the new solution replaces the worst and *RefSet* is reordered in step 6. The *NewSolutions* flag is switched to TRUE and the pair (x', x'') that was just combined is deleted from *NewSubsets* in step 11.

1. Obtain a RefSet of b elite solutions.
2. Evaluate the solutions in *RefSet* and order them according to their objective function value such that x^1 is the best solution and x^b the worst. Make *NewSolutions* = TRUE.

while (*NewSolutions*) **do**

3. Generate *NewSubsets*, which consists of all pairs of solutions in *RefSet* that include at least one new solution. Make *NewSolutions* = FALSE.

while (*NewSubsets* $\neq \emptyset$) **do**

4. Select the next pair (x', x'') in *NewSubSets.*
5. Apply the Relinking Method to produce the sequence $x' = x(1), x(2), \ldots, x(r) = x''$

for i=1 **to** i < r / *NumImp* **do**

6. Apply the Improvement Method to *x(NumImp * i)*.

end for

7. Apply the Relinking Method to produce the sequence $x'' = y(1), y(2), \ldots, y(s) = x'$

for i=1 **to** i < s / *NumImp* **do**

8. Apply the Improvement Method to *y(NumImp * i)*.

end for

for (each generated solutions x) **if** (x is not in *RefSet* and $f(x) < f(x^b)$) **then**

9. Make $x^b = x$ and reorder *RefSet*.
10. Make *NewSolutions* = TRUE.

end if, end for

11. Delete (x', x'') from *NewSubsets*.

end while

end while

Figure 2. Outline of a simple path relinking procedure.

As in the illustrative scheme proposed for SS, the updating of the reference set is based on improving the quality of the worst solution and the search terminates when no new solutions are admitted to *RefSet*. Similarly, the Subset Generation Method is also very simple and consists of generating all pairs of solutions in *RefSet* that contain at least one new solution. In the next section we propose strategies to overcome the limitations of this basic design.

3. Advanced Strategies

3.1 Scatter Search

A. REFSET REBUILDING

This update adds a mechanism to partially rebuild the reference set when no new solutions can be generated with the Combination Method. The update is performed when the inner "while-loop" in Figures 1 and 2 fails and the *NewSolutions* variable is FALSE. The *RefSet* is partially rebuilt with a diversification update, which works as follows. Solutions $x^{[b/2]+1}, \ldots, x^b$ are deleted from *RefSet*, where $[v]$ is the largest integer less than or equal to v. The frequency counts in the Diversification Generation Method are updated with the corresponding values from the solutions that remain in the reference set, that is, $x^1, \ldots, x^{[b/2]}$. Then, the generator is used to construct a set P of new solutions. Solutions $x^{[b/2]+1}, \ldots, x^b$ in *RefSet* are sequentially selected from P with the max-min criterion used in step 1 of Figure 1 (and described in the previous section). The min-max criterion is applied against solutions $x^1, \ldots, x^{[b/2]}$ when selecting solution $x^{[b/2]+1}$, then against solutions $x^1, \ldots, x^{[b/2]+1}$ when selecting solution $x^{[b/2]+2}$, and so on.

B. REFSET DYNAMIC UPDATE

In the basic design, the new solutions that become members of *RefSet* are not combined until all pairs in *NewSubsets* are subjected to the Combination Method. We call *Static Update* to this strategy. On the other hand, the Dynamic Update strategy applies the Combination Method to new solutions in a manner that is faster than in the basic design. That is, if a new solution is admitted to the reference set, the goal is to allow this new solution to be subjected to the Combination Method as quickly as possible.

A simple way to implement this strategy is by reducing *NewSubsets* to just one pair of solutions. Then, the procedure returns to the generation of subsets immediately after one application of the Combination

Method. The downside of this intensification phase is that some solutions in *RefSet* could be replaced before being combined. To illustrate this, suppose that the initial *RefSet* = { x^1, x^2, x^3, x^4 } and that the combination of the pair (x^1, x^2) results in a solution *y* for which:

$$f(y) < f(x^3)$$

Then, the reference set is changed in such a way that the updated *RefSet* = { x^1, x^2, *y*, x^3 }. The search continues by combining the members of the pair (x^1, *y*). Clearly, the quick updating of the reference set will eliminate the operation of combining members of the pair (x^1, x^4). Campos et al. (2001) compare this strategy with the basic design in the linear ordering problem.

C. MULTIPLE SOLUTION COMBINATIONS

The combination mechanism in SS is not limited in its general form in combining just two solutions. Glover (1997) proposes a procedure that generates subsets of the RefSet that have useful properties while avoiding the duplication of subsets previously generated. Specifically, four different collections of subsets of RefSet are introduced:

- SubsetType 1 : All 2- element subsets
- SubsetType 2: 3-element subsets derived from 2-element subsets by augmenting each 2-element subset to include the best solution not in this subset.
- SubsetType 3: 4-element subsets derived from 3-element subsets by augmenting each 3-element subset to include the best solution not in this subset.
- SubsetType 4: the subsets consisting of the best i elements, for i=5 to *b*.

These subsets interact with a dynamic update of the RefSet as described above. Algorithms for maintaining the RefSet with these Subsets and avoid duplications are detailed in Glover (1997).

D. HANDLING PROBLEMS WITH ZERO-ONE VARIABLES

When generating linear combinations of points it is important to be sure the results are dispersed appropriately, and to avoid calculations that simply duplicate other – particularly where the duplications occur after mapping continuous values into integer values. This caveat is especially to be heeded in the case of zero-one variables, since each continuous value can only map into one of the two alternatives 0 and 1, and a great deal of effort can be wasted by disregarding this obvious consequence.

Specifically, for the 0-1 case, a useful set of combinations and mappings (except for random variations) of r reference points consists simply of those that create a positive integer threshold $t \leq r$, and require that the offspring will have its i^{th} component $x_i = 1$ if and only if at least t of the r parents have $x_i = 1$. (A preferred way to apply this rule is to subdivide the variables into categories, and to apply different thresholds to different categories – i.e., to different subvectors. Many problems offer natural criteria for subdividing variables into categories. However, in the absence of this, or in the case where greater diversity is desired, the categories can be arbitrarily generated and varied. At the extreme, for example, each variable can belong to its own category. This SS option gives the same set of alternative mappings as the so-called “Bernoulli crossover” introduced into GAs about a decade after the original SS proposals, except that Scatter Search does not select the 0 and 1 values at random as in the Bernoulli scheme, but instead selects these values by relying on frequency memory to achieve intensification and diversification goals.)

To illustrate the consequences of such a threshold mechanism applied to 2 parents (or to selected subvectors of two parents), t = 1 gives the offspring that is the union of the 1's in the parents and t = 2 gives the offspring that is the intersection of the 1's in the parents. But the threshold mechanism is not the only relevant one to consider. Two other kinds of combinations can be produced for r = 2, consisting of those equivalent to subtracting one vector (or subvector) from another and setting $x_i = 1$ if and only if the difference is positive. In this case it is not unreasonable to consider one more linear combination by summing these two offspring to get the symmetric difference between the parents.

To summarize: restricting attention to r = 2 yields the following options of interest:

(1) the intersection of 1's
(2) the union of 1's
(3) the 1's that belong to parent 1 and not to parent 2

(4) the 1's that belong to parent 2 and not to parent 1
(5) the symmetric difference (summing (3) and (4)).

In some contexts the last three options are not as useful as the first two, but in others they can be exceedingly important. An example of the importance of the symmetric difference is given in Glover and Laguna (1997).

An analogous set of options for r = 3 is given by:

(1) the intersection of 1's
(2) the union of 1's
(3) the 1's that belong to a majority of the parents
(4) the 1's that belong to exactly 1 parent but not more
(5) the 1's that belong to exactly 2 parents but not more
(6) the 1's that belong to the union of 2 parents, excluding those that belong to the 3rd parent. (3 different cases)
(7) the 1's that belong to the intersection of 2 parents, excluding those that belong to the 3rd parent. (3 different cases)

Inclusion of the 6^{th} and 7^{th} options, each of which involves 3 different cases, entails a fair amount of effort, which may not be warranted in many situations. However, it would make sense to use (6) and (7) by defining the "3^{rd} parent" to be the worst of the 3, hence the 1's that are excluded are those that belong to the worst parent.

In general, the relevant mappings for linear combinations involving 0-1 variables can be identified in advance by rules such as those indicated, applied either to full vectors or to subvectors. This results in a much more economical and effective process than separately generating a wide range of linear combinations and then performing a mapping operation. The latter can produce multiple duplications and even miss relevant alternatives.

3.2 Path Relinking

A. VARIATION AND TUNNELING

A variant of the Path Relinking approach starts with both endpoints x' and x'' simultaneously, and produces two sequences $x' = x'(1), \ldots, x'(r)$ and $x'' = x''(1), \ldots, x''(s)$. The choices in this case are designed to yield $x'(r) = x''(s)$, for final values of r and s. To progress toward this outcome when $x'(r) = x''(s)$, either $x'(r)$ is selected to create $x'(r + 1)$, as by the criterion of minimizing reducing the number of moves remaining to reach $x''(s)$, or $x'(s)$ is chosen to create $x''(s + 1)$, as by the criterion of minimizing (reducing) the number of moves remaining to reach $x'(r)$. From these options, the move is selected that produces the smallest $c(x)$ value, thus also determining which of r or s is incremented on the next step. Basing the relinking process on more than one neighborhood also produces a useful variation.

The Path Relinking approach also benefits from a tunneling strategy that encourages a different neighborhood structure to be used than in the standard search phase. For example, moves for Path Relinking may be periodically allowed that normally would be excluded due to creating infeasibility. Such a practice is protected against the possibility of becoming 'lost' in an infeasible region, since feasibility evidently must be recovered by the time x'' is reached. The tunneling effect therefore offers a chance to reach solutions that might otherwise be bypassed. In the variant that starts from both x' and x'', priority may be given to keeping at least one of $x'(r)$ and $x''(s)$ feasible.

B. EXTRAPOLATED RELINKING

The Path Relinking approach goes beyond consideration of points 'between' x' and x'' in the same way that linear combinations extend beyond points that are expressed as convex combinations of two endpoints. In seeking a path that continues beyond x'' (starting from the point x') we once again invoke the Tabu Search concept of referring to sets of attributes associated with the solutions generated, as a basis for choosing a move that 'approximately' leaves the fewest moves remaining to reach x''. Let $A(x)$ denote the set of solution attributes associated with ('contained in') x, and let *A_drop* denote the set of solution attributes that are dropped by moves performed to reach the current solution $x'(i)$, starting from x'. (Such attributes may be components of the x vectors themselves, or may be related to these

components by appropriately defined mappings.)

Define a *to-attribute* of a move to be an attribute of the solution produced by the move, but not an attribute of the solution that initiates the move. Similarly, define a *from-attribute* to be an attribute of the initiating solution but not of the new solution produced. Then we seek a move at each step to maximize the number of *to-attributes* that belong to $A(x'') - A(x(i))$, and subject to this to minimize the number that belong to $A\_drop - A(x'')$. Such a rule generally can be implemented very efficiently by appropriate data structures.

Once $x(r) = x''$ is reached, the process continues by modifying the choice rule as follows. The criterion now selects a move to maximize the number of its *to-attributes* not in *A_drop* minus the number of its *to-attributes* that are in *A_drop*, and subject to this to minimize the number of its *from-attributes* that belong to $A(x'')$. The combination of these criteria establishes an effect analogous to that achieved by the standard algebraic formula for extending a line segment beyond an endpoint. (The secondary minimization criterion is probably less important in this determination.) The path then stops whenever no choice remains that permits the maximization criterion to be positive. The maximization goals of these two criteria are of course approximate, and can be relaxed.

C. MULTIPLE PARENTS

New points can be generated from multiple parents in Path Relinking by the following explicit process. Instead of moving from a point x' to (or through) a second point x'', we replace x'' by a collection of solutions X''. Upon generating a point $x(i)$, the options for determining a next point $x(i + 1)$ are given by the union of the solutions in X'', or more precisely, by the union A'' of the attribute sets $A(x)$, for $x \in X''$. A'' takes the role of $A(x)$ in the attribute-based approach previously described, with the added stipulation that each attribute is counted (weighted) in accordance with the number of times it appears in elements $A(x)$ of the collection. Still more generally, we may assign a weight to $A(x)$, which thus translates into a sum of weights over A'' applicable to each attribute, creating an effect analogous to that of creating a weighted linear combination in Euclidean space. Parallel processing can be applied to operate on an entire collection of points $x' \in X'$ relative to a second collection $x'' \in X''$ by this approach. Further considerations that build on these ideas are detailed in Glover (1994), but they go beyond the scope of our present development.

This multiparent Path Relinking approach generates new elements by a process that emulates the strategies of the original Scatter Search approach at a higher level of generalization. The reference to neighborhood spaces makes it possible to preserve desirable solution properties (such as complex feasibility conditions in scheduling and routing), without requiring artificial mechanisms to recover these properties in situations where they may otherwise become lost.

References

Campos, V., Glover, F., Laguna, M. and Martí, R. (2001), "An Experimental Evaluation of a Scatter Search for the Linear Ordering Problem" *Journal of Global Optimization* 21, 397-414

Glover F. (1994). "Tabu Search for Nonlinear and Parametric Optimization (with Links to Genetic Algorithms)," *Discrete Applied Mathematics*, vol. 49, 231-255.

Glover, F. (1997). "A Template for Scatter Search and Path Relinking," in *Lecture Notes in Computer Science*, 1363, J.K. Hao, E. Lutton, E. Ronald, M. Schoenauer, D. Snyers (Eds.), 1-53. (Expanded version available on request.)

Glover F. (1999). "Scatter Search and Path Relinking," In: D Corne, M Dorigo and F Glover (eds.) *New Ideas in Optimisation*, Wiley.

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

Glover, F., M. Laguna and R. Martí (2000), "Fundamentals of Scatter Search and Path Relinking" *Control and Cybernetics*, 29(3), 653-684

Glover, F., M. Laguna and R. Martí (2002). "Scatter Search," to appear in *Theory and Applications of Evolutionary Computation: Recent Trends*, A. Ghosh and S. Tsutsui (Eds.) Springer-Verlag.

Greistorfer, P. and Voss, S. (2001), "Controlled Pool Maintenance in Combinatorial Optimization" Conference on *Adaptive Memory and Evolution: Tabu Search and Scatter Search*, University of Mississippi.

Laguna, M. (2000) "Scatter Search," to appear in *Handbook of Applied Optimization*, P. M. Pardalos and M. G. C. Resende (Eds.), Oxford Academic Press.

Laguna, M. and R. Martí (1999). "GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization," *INFORMS Journal on Computing*, Vol. 11, No. 1, pp. 44-52.

Laguna, M. and R. Martí (2000a). "Experimental Testing of Advanced Scatter Search Designs for Global Optimization of Multi-modal Functions," Working paper, University of Colorado.

Laguna, M. and R. Martí (2000b). "The OptQuest Callable Library," *Optimization Software Class Libraries*, S. Voss and D. L. Woodruff (Eds.)**,** Kluwer, Boston. pp. 193- 218

Laguna, M., R. Martí and V. Campos (1999). "Intensification and Diversification with Elite Tabu Search Solutions for the Linear Ordering Problem," *Computers and Operations Research*, Vol. 26, pp. 1217-1230.