Principles of Scatter Search

RAFAEL MARTÍ

Dpto. de Estadística e Investigación Operativa, Facultad de Matemáticas, Universidad de Valencia, Dr. Moliner 50, 46100 Burjassot (Valencia) Spain. Rafael.Marti@uv.es

MANUEL LAGUNA, FRED GLOVER

Leeds School of Business, University of Colorado, Campus Box 419, Boulder, CO 80309 Laguna@colorado.edu , Fred.Glover@colorado.edu

Abstract

Scatter search is an evolutionary method that has been successfully applied to hard optimization problems. The fundamental concepts and principles of the method were first proposed in the 1970s, based on formulations dating back to the 1960s for combining decision rules and problem constraints. In contrast to other evolutionary methods like genetic algorithms, scatter search is founded on the premise that systematic designs and methods for creating new solutions afford significant benefits beyond those derived from recourse to randomization. It uses strategies for search diversification and intensification that have proved effective in a variety of optimization problems.

This paper provides the main principles and ideas of scatter search and its generalized form path relinking. We first describe a basic design to give the reader the tools to create relatively simple implementations. More advanced designs derive from the fact that scatter search and path relinking are also intimately related to the tabu search (TS) metaheuristic, and gain additional advantage by making use of TS adaptive memory and associated memory-exploiting mechanisms capable of being tailored to particular contexts. These and other advanced processes described in the paper facilitate the creation of sophisticated implementations for hard problems that often arise in practical settings. Due to their flexibility and proven effectiveness, scatter search and path relinking can be successfully adapted to tackle optimization problems spanning a wide range of applications and a diverse collection of structures, as shown in the papers of this volume.

KeyWords

Metaheuristics, Evolutionary Methods, Combination, Path Relinking

1. Introduction

Scatter search (SS) was first introduced in Glover (1977) as a heuristic for integer programming. In the original proposal, solutions are purposely (i.e., non-randomly) generated to take account of characteristics in various parts of the solution space. Scatter search orients its explorations systematically relative to a set of reference points that typically consist of good solutions obtained by prior problem solving efforts, where the criteria for "good" are not restricted to objective function values, and may apply to sub-collections of solutions rather than to a single solution, as in the case of solutions that differ from each other according to certain specifications.

Weighted linear combinations provide the main mechanism to generate new trial points within the space containing the reference points, accompanied by a generalized (successively iterated) rounding mechanism to assure these trial points satisfy integer feasibility conditions in the case where some variables are required to receive integer values. These mechanisms are oriented toward the goal of creating weighted centers of selected sub-regions, including regions external to the convex hull of the reference points.

The *scatter search template* (Glover 1998) has served as the main reference for most of the scatter search implementations to date. The dispersion patterns created by these designs have been found useful in several application areas. Section 2 gives a comprehensive description of the elements and methods of this template based on the formulation given in Laguna and Martí (2003). Following this, section 3 examines different advanced scatter search designs, including some recent features as the 3-Tier reference set update and associated uses of memory. Section 4 is devoted to the path relinking methodology that extends scatter search to the setting of neighborhood spaces, including both basic and advanced implementations, and the paper ends with concluding observations.

2. Basic Scatter Search Design

The scatter search methodology is very flexible, since each of its elements can be implemented in a variety of ways and degrees of sophistication. In this section we give a basic design to implement scatter search based on the well-known "five methods", while the advanced designs are cover in the next section. The advanced features of scatter search are related to the way these five methods are implemented. That is, the sophistication comes from the implementation of the SS methods instead of the decision to include or exclude some elements (like in the case of tabu search, as mentioned above).

The fact that the mechanisms within scatter search are not restricted to a single uniform design allows the exploration of strategic possibilities that may prove effective in a particular implementation. These observations and principles lead to the following template for implementing scatter search that consists of five methods.

1. A *Diversification Generation Method* to generate a collection of diverse trial solutions, using an arbitrary trial solution (or seed solution) 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 will more usually be expected to be so. If no improvement of the input trial solution results, 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 method. Solutions gain membership to the reference set according to their quality or their diversity.
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.
5. A *Solution Combination Method* to transform a given subset of solutions produced by the Subset Generation Method into one or more combined solution vectors.

Figure 1 shows the interaction among these five methods and puts in evidence the central role of the reference set. This basic design starts with the creation of an initial set of solutions P, and then extracts from it the reference set (*RefSet*) of solutions.

The Diversification Generation Method is used to build a large set P of diverse solutions. The size of P (*PSize*) is typically at least 10 times the size of *RefSet*. The initial reference set is built according to the Reference Set Update Method. For example, the Reference Set Update Method could consist of selecting b distinct and maximally diverse solutions from P. A simple mechanism to construct the RefSet is given in the next paragraphs and the next section explores several alternatives for implementing the Reference Set Update Method. Regardless of the rules used to select the reference solutions, 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. The simplest form of the Subset Generation Method consists of generating all pairs of reference solutions. That is, the method would focus on subsets of size 2 resulting in $(b^2\text{-}b)/2$ *NewSubsets*. 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 trial solutions in step 5. These trial solutions are subjected to the Improvement Method, if one is available. The Reference Set Update Method is applied once again in step 6. The simplest form of the application of the Reference Update Method in this step is to build the new *RefSet* with the best solutions, according to the objective function value, from the current *RefSet* and the set of trail solutions. If *RefSet* changes after the application of the reference set update method the *NewSolutions* flag is switched to TRUE in step 7, indicating that at least one new solution has been inserted in the reference set. The subset s that was just subjected to the Combination Method is deleted from *NewSubsets* in step 8.

The basic procedure terminates after all subsets in *NewSubsets* are subjected to the combination method and none of the improved trial solutions are admitted to *RefSet* under the rules of the Reference Set Update Method.

1. Start with $P = \emptyset$. Use the diversification generation method to construct a solution and apply the improvement method. Let x be the resulting solution. If $x \notin P$ then add x to P (i.e., $P = P \cup x$), otherwise, discard x. Repeat this step until $|P| = PSize$.
2. Use the reference set update method to build $RefSet = \{ x^1, \ldots, x^b \}$ with the "best" b solutions in P. Order the solutions in *RefSet* 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* with the subset generation method. Make *NewSolutions* = FALSE.

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

4. Select the next subset s in *NewSubsets*.
5. Apply the solution combination method to s to obtain one or more new trial solutions x. Apply the improvement method to the trial solutions.
6. Apply the reference set update method.

if (*RefSet* has changed) **then**

7. Make *NewSolutions* = TRUE.

end if

8. Delete s from *NewSubsets*.

end while

end while

Figure 1. Basic scatter search procedure

The reference set, *RefSet*, is a collection of both high quality solutions and diverse solutions that are used to generate new solutions by way of applying the Combination Method. In this basic design we can use a simple mechanism to construct an initial reference set and then update it during the search. The size of the reference set is denoted by $b = b_1 + b_2 = |RefSet|$. The construction of the initial reference set starts with the selection of the best b_1 solutions from P. These solutions are added to *RefSet* and deleted from P. For each solution in *P-RefSet*, the minimum of the distances to the solutions in *RefSet* is computed. Then, the solution with the maximum of these minimum distances is selected. This solution is added to *RefSet* and deleted from P and the minimum distances are updated. (In applying this max-min criterion, or any criterion based on distances, it can be important to scale the problem variables, to avoid a situation where a particular variable or subset of variables dominates the distance measure and distorts the appropriate contribution of the vector components.) The process is repeated b_2 times, where $b_2 = b - b_1$. The resulting reference set has b_1 high quality solutions and b_2 diverse solutions.

After the initial reference set is constructed, the Combination Method is applied to the subsets generated as outlined in step 5 of Figure 1. In the basic design we use the so-called *static update* of the reference set after the application of the Combination Method. Trial solutions that are constructed as combination of reference solutions are placed in a solution pool, denoted by *Pool*. After the application of both the Combination Method and the Improvement Method, the *Pool* is full and the reference set is updated. The new reference set consists of the best b solutions from the solutions in the current reference set and the solutions in the pool, i.e., the update reference set contains the best b solutions in $RefSet \cup Pool$.

Of the five methods in the scatter search methodology, only four are strictly required. The Improvement Method is usually needed if high quality outcomes are desired, but a scatter search procedure can be implemented without it. On the other hand, a short term tabu search procedure can be implemented as the improvement method as will be shown in the next section.

3. Advanced Scatter Search Designs

When considering advanced strategies in a metaheuristic framework, the goal of improving performance often conflicts with the goal of designing a procedure that is easy to implement and fine tune. Advanced designs generally, but not always, translate into higher complexity and additional search parameters. As far as we know, there is no simple recipe that can be used to follow a predetermined order in which advanced strategies should be added to progressively improve the performance of a scatter search implementation. Therefore, the order in which these strategies are described in this section does not reflect their importance or ranking. An exhaustive description of advanced designs can be found in Laguna and Martí (2003).

3.1 Dynamic RefSet Updating

The reference set is the heart of a scatter search procedure. If at any given time during the search all the reference solutions are alike, as measured by an appropriate metric, the scatter search procedure will most likely be incapable of improving upon the best solution found even when employing a sophisticated procedure to perform combinations or improve new trial solutions. The Combination Method is limited by the reference solutions that it uses as input. Hence, having the most advanced Combination Method is of little advantage if the reference set is not carefully built and maintain during the search.

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. The new reference set is built with the best solutions in the union of *Pool* and the solutions currently in *RefSet*. This strategy is called the *Static Update* of the reference set. The alternative to the static update is the *Dynamic Update* strategy, which applies the Combination Method to new solutions in a manner that combines new solutions 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. In other words, instead of waiting until all the combinations have been performed to update the reference set, if a new trial solution warrants admission in the reference set, the set is immediately updated before the next combination is performed. Therefore, there is no need for an intermediate pool in this design, since solutions are either discarded or become part of the *RefSet* as soon as they are generated.

The advantage of the dynamic update is that if the reference set contains solutions of inferior quality, these solutions are quickly replaced and future combinations are made with improved solutions. The disadvantage is that some potentially promising combinations are eliminated before they can be considered. The implementation of dynamic updating is more complex than its static counterpart. Also, in the static update the order in which the combinations are performed is not important because the *RefSet* is not updated until all combinations have been performed. In the dynamic updating, the order is quite important because it determines the elimination of some potential combinations. Hence, when implementing a dynamic update of the reference set, it may be necessary to experiment with different combination orders as part of the fine tuning of the procedure.

3.2 RefSet Rebuilding

We introduce now an updating procedure that is triggered when no new trial solutions are admitted to the reference set. This update adds a mechanism to partially rebuild the reference set when the Combination

and Improvement Methods do not provide solutions of sufficient quality to displace current reference solutions.

The *RefSet* is partially rebuilt with a diversification update that works as follows and assumes that the size of the reference set is $b = b_1 + b_2$. Solutions $x^{b_1+1},\ldots,x^{b}$ are deleted from *RefSet*. The Diversification Generation Method is reinitialized considering that the goal is to generate solutions that are diverse with respect to the reference solutions $x^{1},\ldots,x^{b_1}$. Then, the Diversification Generation Method is used to construct a set *P* of new solutions. The b_2 solutions $x^{b_1+1},\ldots,x^{b}$ in *RefSet* are sequentially selected from *P* with the criterion of maximizing the diversity. It is usually implemented with a distance measure defined in the context of the problem being solved. Then, maximize the diversity is achieved by maximizing the minimum distance. The max-min criterion, which is part of the Reference Set Update Method, is applied with respect to solutions $x^{1},\ldots,x^{b_1}$ when selecting solution x^{b_1+1}, then it is applied with respect to solutions $x^{1},\ldots,x^{b_1+1}$ when selecting solution x^{b_1+2}, and so on.

3.3 RefSet Tiers

In lower level scatter search implementations, the reference set is updated by replacing the reference solution having the worst objective function value with a new trial solution having a better objective function value. Since we consider that *RefSet* is always ordered, the best solution is x^1 and the worst solution is x^b. So, when a new trial solution x is generated as a result of the application of the Combination and Improvement Methods, the objective function value of the new trial solution is used to determine whether *RefSet* needs to be updated. This step occurs by setting when x is better than x^b by dropping x^b and inserting x in a position that maintains the indicated ordering of the set. We now explore mechanisms that differentiate solutions using additional measures of merit that are not based on the objective function value.

Instead of waiting until the reference set has converged, that is, until it has reached a state in which no new solutions are admitted, an updating procedure that proactively injects diversification into the search can be used. The updating procedure employs a 2-tier design, where the first tier $RefSet_1$ consists of b_1 high quality solutions and $RefSet_2$ consists of b_2 diverse solutions. The update has the goal of dynamically preserving diversity in the reference set, instead of allowing it to become homogenous by only admitting high quality solutions that in some applications tend to be very similar to each other. Hence, in addition to updating the reference set when new trial solutions of high quality are found with the Combination and Improvement Methods, the reference set is also updated with highly diverse solutions.

Specifically, the update consists of partitioning the reference into two subsets:

$$RefSet_1 = \left\{x^{1},\ldots,x^{b_1}\right\} \text{ and } RefSet_2 = \left\{x^{b_1+1},\ldots,x^{b}\right\}$$

The first subset is referred to as the "high quality" subset and the second is referred to as the "diverse" subset. The solutions in $RefSet_1$ are ordered according to their objective function value and the set is updated with the goal of increasing quality, using the criterion of the basic scatter search design. That is, a new solution x replaces reference solution x^{b_1} if $f(x) < f\left(x^{b_1}\right)$ in a minimization problem. The solutions in $RefSet_2$ are ordered according to their diversity value and the update has the goal of increasing diversity. Therefore, a new solution x replaces reference solution x^b if $d_{\min}(x) > d_{\min}\left(x^{b}\right)$.

The 2-tier update can be used in combination with the rebuilding mechanism. The implementation is straightforward by keeping $RefSet_1$ and reinitializing the Diversification Generation Method in order to rebuild $RefSet_2$ with solutions that are diverse among them and with respect to $RefSet_1$.

Laguna and Martí (2000) propose an extension of this design that maintains a list of the "best generators". A "good generator" is a reference solution that generates high quality trial solutions when used as input to the Combination Method. The 3-tier update uses a reference set of size $b = b_1 + b_2 + b_3$, which is divided into the following three subsets:

$$RefSet_1 = \{x^1, \ldots, x^{b_1}\},\ RefSet_2 = \{x^{b_1+1}, \ldots, x^{b_1+b_2}\} \text{ and } RefSet_3 = \{x^{b_1+b_2+1}, \ldots, x^{b}\}$$

$RefSet_1$ and $RefSet_2$ are updated using the same rules as in the 2-tier update. In order to update $RefSet_3$, we keep track of $g(x)$, the objective function value of the best solution ever created from a combination of $x \in RefSet_1$ and any other reference solution. $RefSet_3$ is ordered according to $g(x)$ in such a way that $g(x^{b_1+b_2+1}) < g(x^{b_1+b_2+2}) < \cdots < g(x^{b})$ for a minimization problem. When x^{b_1} in $RefSet_1$ is replaced with a newly created solution of higher quality, we compare $g(x^{b_1})$ with $g(x^{b})$ and update $RefSet_3$ if appropriate.

This design can be particularly helpful in settings where solutions of relatively low quality are capable of producing high quality solutions, by allowing such "good generators" to participate in additional combinations once they have been replaced from $RefSet_1$. The initialization of $RefSet_1$ and $RefSet_2$ is the same as in the 2-tier design. $RefSet_3$ is initialized with the best solutions in P that were not included in $RefSet_1$.

3.4 Diversity Control

Scatter search does not allow duplications in the reference set, and its combination methods are designed to take advantage of this lack of duplication. Hashing is often used to reduce the computational effort of checking for duplicated solutions. The following hash function, for instance, is an efficient way of comparing solutions and avoiding duplications when dealing with problems whose solutions can be represented with a permutations p of size m:

$$hash(p) = \sum_{i=1}^{m} ip(i)^2$$

Campos, et al. (2001) report the benefits of this form of hashing in the context of the linear ordering problem.

While the simpler scatter search implementations are designed to check that the reference set does not contain duplications, they generally do not monitor the diversity of the b_1 high quality solutions when creating the initial *RefSet*. On the other hand, recall that the b_2 diverse solutions are subjected to a strict diversity check with the max-min criterion. A minimum diversity test can be applied to the b_1 high quality solutions chosen as members of the initial *RefSet* as follows. After the P set has been created, the best solution according to the objective function value is selected to become x^1 in the reference set. Then, x^1 is deleted from P and the next best solution x in P is chosen and added to *RefSet* only if

$$d_{\min}(x) \geq th\_dist.$$

In other words, at each step we add the next best solution in P only if the minimum distance between the chosen solution x and the solutions currently in *RefSet* is at least as large as the threshold value *th_dist*.

3.5 Subset Generation Method

Solution Combination Methods in scatter search typically are not limited to combining just two solutions and therefore the Subset Generation Method in its more general form consists of creating subsets of different sizes. The scatter search methodology assures that the set of combined solutions may be produced in its entirety at the point where the subsets of reference solutions are created. Therefore, once a given subset is created, there is no merit in creating it again. This creates a situation that differs noticeably from those considered in the context of genetic algorithms, where the combinations are typically determined by the spin of a roulette wheel.

The procedure for generating subsets of reference solutions uses a strategy to expand pairs into subsets of larger size while controlling the total number of subsets to be generated. In other words, the mechanism avoids the extreme type of process that creates all the subsets of size 2, then all the subsets of size 3, and so on until reaching the subsets of size b-1 and finally the entire *RefSet*. This approach would clearly not be practical, considering that there are 1013 subsets in a reference set of a typical size $b = 10$. Even for a smaller reference set, combining all possible subsets is not effective because many subsets will be almost identical. The following approach selects representative subsets of different sizes by creating subset types:

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

3.6 Use of Memory

Scatter search incorporates an implicit form of memory as a result of the interactions among the Reference Set Update, the Solution Combination Method and the Subset Generation Method. The Reference Set Update, in its most basic form, is designed to "remember" the best solutions encountered during the search. Selected features of these solutions provide the basis for creating new trial solutions with the Combination Method. Hence, the overall process is instrumental in the transmission of information embedded in the reference solutions.

This implicit type of memory is rather primitive, and may be classified as an *inheritance memory*. All evolutionary methods incorporate an inheritance memory of one sort or another, yet even at this rudimentary level the memory mechanisms of scatter search are somewhat more elaborate than those of other evolutionary approaches. This is due to the Subset Generation Method, which keeps track of the subsets of reference solutions that have been subjected to the combination mechanism from one iteration to the next. When new solutions are admitted to the reference set, the method generates only those subsets that are admissible for combination in the current iteration. The Subset Generation Method performs this operation by a memory structure that identifies the subsets containing new reference solutions. By contrast, traditional evolutionary approaches such as genetic algorithms employ no equivalent use of memory, but select solutions for combination purposes by using some variant of a random sampling scheme. (Quite recently, a few "non-standard" evolutionary approaches have begun to appear that select parent solutions by strategies more nearly resembling those of scatter search. However, they continue to rely predominantly on randomization, and have not yet progressed to incorporate the form of memory-based strategies embodied in the Subset Generation Method.)

In any case, inheritance memory – including the type embodied in scatter search – is not sufficiently focused or purposeful to provide the analytical power necessary to solve complex problems in a consistently effective manner. (If naïve inheritance memory had such properties, human brains would be superfluous.) This is the fundamental reason why scatter search draws upon adaptive memory principles of tabu search, and is often implemented in conjunction with tabu search. The fact that several of the key ideas of the two approaches spring from a common source (Glover, 1977) further strengthens the bonds between them.

As part of its design for introducing strategic adaptive memory into the metaheuristic literature, tabu search makes use of memory that is both *explicit* and *attributive.* Explicit memory records complete solutions, typically consisting of elite solutions visited during the search. Attributive memory records information about solution attributes that change in moving from one solution to another. (See, for example, Glover and Laguna (1997) for a fuller description.)

In applications of scatter search, one of the main uses of explicit memory is in the context of optimizing simulations, as detailed in Laguna and Martí (2002). In this setting, decision variables x in the optimization model are the input factors of the simulation model. The simulation model generates an output $f(x)$ for every set of input values. On the basis of this evaluation, and on the basis of the past evaluations which are integrated and analyzed with the present simulation outputs, a scatter search generates a new set of input values. Because the evaluation of a set of input values by means of the execution of a simulation model may require a large computational effort, recording every solution generated and evaluated during the search is an effective approach in this setting. When a new trial solution is generated with the Solution Combination Method and before it is sent to the simulation model for evaluation, the explicit memory is consulted to make sure that the trial solution is indeed new. Since typical optimization runs in this context make at most ten thousand calls to the Simulation Model, the explicit memory structure does not grow to an unmanageable size. Nonetheless, hashing may be used to limit the memory requirements and the effort associated with checking whether a trial solution is already stored in the explicit memory structure.

Instead of recording full solutions, attributive memory structures are based on recording attributes. Attributive memory is used for implementing both diversification and intensification strategies in TS. The most common attributive memory approaches are recency-based memory and frequency-based memory. Recency-based memory is the most common (though not always the most important) memory structure used in TS implementations. As its name suggests, this memory structure keeps track of solution attributes that have changed during the recent past. Frequency-based memory provides a type of information that complements the information provided by recency-based memory, broadening the foundation for selecting preferred moves.

Laguna and Martí (2003) propose a diversification generator method for nonlinear function optimization in a continuous solution space. This frequency-based memory method falls within the category of a residence measure by viewing the construction of *P* as an iterative process that "visits" one solution per iteration for *PSize* iterations. Campos et al. (2001) introduce a Diversification Generation Method, in the context of the linear ordering problem, based on the notion of constructing solutions employing modified frequencies. The generator exploits the permutation structure of the problem. The residence measure employed in this procedure is such that a frequency counter is maintained to record the number of times a sector appears in a specific position. The frequency counters are used to penalize the "attractiveness" of a sector with respect to a given position.

Search intensification is typically achieved in scatter search with the execution of the Improvement Method. When memory structures are added to the Improvement Method, the method is conceptually and practically transformed from a pure local search heuristic into a metaheuristic given that by definition a metaheuristic refers to a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality. Employing a metaheuristic as an Improvement Method results in a hybrid method that combines two metaheuristics, i.e., scatter search and the one used to improve solutions. An important issue in such a design is how to allocate the total computational effort. In other words, should the search spend most of the time improving solutions or generating new trial solutions with the Diversification Generation and Combination Methods? The balance in the computational effort must be controlled by not only selecting the solutions to be subjected to the Improvement Method but also by choosing a rule to stop the improvement process.

4. Path Relinking

Path relinking (PR) was originally suggested as an approach to integrate intensification and diversification strategies in the context of tabu search (Glover and Laguna, 1993, 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.

Path relinking can be considered an extension of the Combination Method 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.

The approach may be viewed as an extreme (highly focused) instance of a strategy that seeks to incorporate attributes of high quality solutions, by creating inducements to favor these attributes in the moves selected. However, instead of using an inducement that merely encourages the inclusion of such attributes, the path relinking approach subordinates other considerations to the goal of choosing moves that introduce the attributes of the guiding solutions, in order to create a "good attribute composition" in the current solution. The composition at each step is determined by choosing the best move, using customary choice criteria, from a restricted set — the set of those moves currently available that incorporate a maximum number (or a maximum weighted value) of the attributes of the guiding solutions. (Exceptions are provided by aspiration criteria, as subsequently noted.) The approach is called path relinking either by virtue of generating a new path between solutions previously linked by a series of moves executed during a search, or by generating a path between solutions previously linked to other solutions but not to each other.

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 minimizes the number of moves remaining to reach x''. The relinked path may encounter solutions that may not be better than the initiating or guiding solution, but that provide fertile "points of access" for reaching other, somewhat better, solutions. For this reason it is valuable to examine neighboring solutions along a relinked path, and keep track of those of high quality which may provide a starting point for launching additional searches.

The reference set (*RefSet*) can be constructed as previously indicated for scatter search. However, 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* 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, Martí and Campos (1999), GRASP, as in Laguna and Martí (1999), or simply a Diversification Generation Method coupled with an Improvement Method as proposed in scatter search. From this point of view, SS and PR can be considered population-based methods 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 basic (simple) scatter search designs, all pairs of solutions in the *RefSet* are subjected to the Combination Method. Similarly, in a basic version of PR all pairs in the *RefSet* are considered to perform a relinking phase. For each pair (x', x'') two paths can be initiated; 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. Laguna and Martí (1999) and Piñana et al. (2001) provide some examples. Two consecutive solutions obtained by a relinking step are often very similar since they differ only in the attributes that change by a single move. Therefore, it is generally not efficient to apply an Improvement Method at every step of the relinking process. We introduce a parameter *NumImp* and apply the Improvement Method 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.

---

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 and *Pool* = ∅.

while (*NewSubsets* ≠ ∅) **do**

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

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

6. Apply the Improvement Method to $x'(i*NumImp)$ and add solutions to *Pool*.

end for

7. Apply the Relinking Method to produce the sequence $x'' = x''(1), x''(2), \ldots, x''(s) = x'$ and add solutions to *Pool*.

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

8. Apply the Improvement Method to $x''(i*NumImp)$ and add solutions to *Pool*.

end for

for (each solution $x \in Pool$)

if ($x \notin RefSet$ 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. Basic path relinking procedure

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 scatter search, 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. Also in this step, *Pool* is initialized to empty. 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 solutions generated in these steps are added to *Pool*. The Improvement Method is applied every *NumImp* steps of the relinking process in each path (steps 6 and 8). Solutions found during the application of the Improvement Method are also added to *Pool*. Each solution in Pool is examined 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 9. The *NewSolutions* flag is switched to TRUE in step 10 and the pair (x', x'') that was just combined is deleted from *NewSubsets* in step 11.

As in the basic scatter search designs, the updating of the reference set in Figure 2 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. We examine strategies to overcome the limitations of this basic design.

To choose among the different paths that may be possible in going from x' to x''**,** let $f(x)$ denote an objective function which is to be minimized. Selecting unattractive moves relative to $f(x)$*,* from the moves that are candidates to generate the path at each step, will tend to produce a final series of strongly improving moves to complete the path. Correspondingly, selecting attractive moves at each step will tend to produce lower quality moves at the end. (The last move, however, will be improving, or leave $f(x)$ unchanged, if x'' is selected to be a local optimum.) Thus, choosing best, worst or average moves, provides options that produce contrasting effects in generating the indicated sequence. An aspiration criterion may be used as in tabu search to override choices in the last two cases if a sufficiently attractive solution is available. (In general, it appears reasonable to select best moves at each step, and then to allow the option of reinitiating the process in the opposite direction by interchanging x' and x''.) Beyond this, if a sufficiently attractive neighbor is found at any point of the process, an aspiration criterion can allow the relinking to depart from its customary path by moving to such a neighbor.

The choice of one or more solutions $x(i)$ to become reference points for launching a new search phase will preferably be made to depend not only on $f(x(i))$ but also on the $f(x)$ values of those solutions x that can be reached by a move from $x(i)$. The process can be varied to allow solutions to be evaluated other than those that yield $x(i + 1)$ closer to x''. Aspiration criteria again are relevant for deciding whether such solutions qualify as candidates for selection.

The **simultaneous relinking** approach starts with both endpoints x' and x'' simultaneously producing 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 the point where $x'(r) = x''(s)$, the choice rules should be such that the x' path approaches the last solution in the current x'' path and the other way around. The simultaneous relinking may be viewed as a process for which two guiding solutions are dynamically changing until they converge to a single point.

Strategic oscillation is a mechanism used in tabu search to allow the process to visit solutions around a "critical boundary", by approaching such a boundary from both sides. The most common application of strategic oscillation is in constrained problems, where the critical boundary is the feasibility boundary. The search process crosses the boundary from the feasible side to the infeasible side and also from the infeasible side to the feasible side. Path relinking also allows the search to cross the feasibility boundary by way of a **tunneling strategy**. The strategy permits infeasible solutions to be visited while relinking x' and x''. It also allows for either x' or x'' to be infeasible but not both.

The tunneling strategy protects the search from becoming "lost" in the infeasible region, since feasibility evidently must be recovered by the time x'' is reached. When x'' is allowed to be infeasible, the relinking path may stop as soon as it leaves the feasible region or continue until reaching x'', since it is possible (although unlikely in some settings) for the path to go back to the feasible region before reaching x''. The tunneling effect therefore offers a chance to reach solutions that might otherwise be bypassed. If

tunneling is combined with the simultaneous relinking approach at least one of $x'(r)$ and $x''(s)$ may be kept feasible. Nevertheless, it should be stressed that – just as in the case of scatter search – an intermediate solution generated by path relinking need not be feasible in order to be relevant as a starting solution for an improvement procedure, since the latter may be designed to restore feasibility.

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, thus defining the **extrapolated relinking**. In seeking a path that continues beyond x'' (starting from the point x') we invoke a tabu search concept that forbids adding tabu-active attributes back to the current solution. 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 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.

New points can be generated from **multiple guiding solutions** as follows. 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. Promising regions may be searched more thoroughly in path relinking by modifying the weights attached to attributes of guiding solutions, and by altering the bias associated with solution quality and selected solution features.

A natural variation of path relinking occurs by using **constructive neighborhoods** for creating new trial solutions from a collection of initiating and guiding solutions. In this case the guiding solutions consist of subsets of elite solutions, as before, but the initiating solution begins as a partial (incomplete) solution or even as a null solution, where some of the components of the solutions, such as values for variables, are not yet assigned. The use of a constructive neighborhood permits such an initiating solution to "move toward" the guiding solutions, by a neighborhood path that progressively introduces elements contained in the guiding solutions, or that are evaluated as attractive based on the composition of the guiding solutions. The idea of using constructive (and destructive) neighborhood in the context of path relinking was originally described in Glover (1994) with the introduction of the idea of creating *structured combinations*. Such a process relies on three properties.

Property 1 (Representation Property). Each vector represents a set of votes (evaluations that can be non-linear and threshold-based) for particular decisions, such as 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.

Property 2 (Trial solution Property). The votes prescribed by a vector translates into a trial slution to the problem of inerest by a well-defined process. A simple set of 'yes-no' votes for items to include in a set, for example, can be translated into a trial solution according to a designated sequence for processing the votes (such as determined by standard benefit-to-cost ratios) until either the set is full or all votes are considered. More general votes for the same problem may also prescribe an evaluation sequence

to be employed. The vectors giving rise to the votes may not represent feasible solutions to the problems considered, or even represent solutions in a customary sense at all.

Property 3 (Update Property). If a decision is made according to the vots of a given vector, a clearly defined rule updates all vectors for the residual problem so that Properties 1 and 2 continue to hold. For example, upon assigning a specific value to a particular variable, 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.

The foregoing properties are generally easy to establish and thereby give rise to classes of path relinking approaches that can serve as a foundation for multi-start processes. The strategy described next allows further variation on this idea.

Vocabulary building creates structured combinations not only by using the primitive elements of customary neighborhoods, but also building and joining more complex assemblies of such elements. The process receives its name by analogy with the process of building words progressively into useful phrases, sentences and paragraphs, where valuable constructions at each level can be visualized as represented by "higher order words," just as natural languages generate new words to take the place of collections of words that embody useful concepts. The motive underlying vocabulary building is to take advantage of those contexts where certain partial configurations of solutions often occur as components of good complete solutions. A strategy of seeking "good partial configurations"—good vocabulary elements—can help to circumvent the combinatorial explosion that potentially results by manipulating only the most primitive elements by themselves. The process also avoids the need to reinvent (or rediscover) the structure of a partial configuration as a basis for building a good complete solution. In general, vocabulary building relies on destructive as well as constructive processes to generate desirable partial solutions and solution fragments are merged using heuristic or exact procedures.

Clustering, proposed in Glover (1977) as a way to improve scatter search, is equally relevant to path relinking. Intensification is encouraged by selecting the parent solutions from a common cluster, while diversification is encouraged by selecting the parent solutions from different clusters. The more recently developed nitching processes in genetic algorithms bear a partial resemblance to this approach, except that nitches are produced simply by operating with different sub-populations, and use no strategic criterion for determining membership in the sub-populations comparable to the type of sifting and cross-comparing embodied within clustering processes. (For example, clustering can determine membership independently from the source population that produced a particular vector, utilizing measures of similarity and complementarity appropriate to the problem context.) Conditional relationships are extremely important when clustering is used as an adjunct to combinatorial search, as illustrated in Glover and Laguna (1997), and we conjecture that accounting for such relationships can enhance the performance of current strategies. In general, clustering and vocabulary building, taken together, offer strategic possibilities for scatter search and path relinking that deserve fuller attention in future applications.

Conclusions

The focus and emphasis of scatter search and path relinking have a number of implications for the goal of designing improved optimization procedures. These research opportunities carry with them an emphasis on producing systematic and strategically designed rules, rather than following the policy of relegating decisions to random choices, as often is fashionable in evolutionary methods. The strategic orientation underlying scatter search and path relinking is motivated by connections with the tabu search setting where the path relinking ideas were first proposed, and invites the use of adaptive memory structures in determining the strategies produced.

Acknowledgments

This research is 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, and by the Spanish Government under code TIC2000-1750-C06-01.

References

Campos, V., F. Glover, M. Laguna and R. Martí (2001) "An Experimental Evaluation of a Scatter Search for the Linear Ordering Problem," *Journal of Global Optimization*, vol. 21, no. 4, pp. 397-414.

Glover, F. (1977) "Heuristics for Integer Programming Using Surrogate Constraints," *Decision Sciences*, vol. 8, pp. 156-166.

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

Glover, F. (1998) "A Template for Scatter Search and Path Relinking," in *Artificial Evolution, Lecture Notes in Computer Science 1363*, J.-K. Hao, E. Lutton, E. Ronald , M. Schoenauer and D. Snyers (Eds.), Springer, pp. 13-54.

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

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 (2000) "Experimental Testing of Advanced Scatter Search Designs for Global Optimization of Multimodal Functions," Technical Report, University of Colorado at Boulder, http://leeds.colorado.edu/faculty/laguna/articles/advss.html.

Laguna, M. and R. Martí (2002) "The OptQuest Callable Library," *Optimization Software Class Libraries*, S. Voss and D. L. Woodruff (Eds.), Kluwer Academic Publishers, Boston, pp. 193-218.

Laguna, M. and R. Martí (2003) *Scatter Search – Methodology and Implementations in C,* Kluwer Academic Publishers, Boston.

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.

Piñana, E., I. Plana, V. Campos and R. Martí (2001) "GRASP and Path Relinking for the Matrix Bandwidth Minimization," *European Journal of Operational Research*, forthcoming.