Fundamentals of Scatter Search and Path Relinking

Fred Glover
Manuel Laguna[†]

Graduate School of Business and Administration, University of Colorado, Boulder, CO 80309-0419, USA, {Fred.Glover}{Manuel.Laguna}@Colorado.edu

Rafael Martí

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

Latest revision: May 23, 2000

Abstract—The evolutionary approach called Scatter Search, and its generalized form called Path Relinking, have proved unusually effective for solving a diverse array of optimization problems from both classical and real world settings. Scatter Search and Path Relinking differ from other evolutionary procedures, such as genetic algorithms, by providing unifying principles for joining solutions based on generalized path constructions (in both Euclidean and neighborhood spaces) and by utilizing strategic designs where other approaches resort to randomization. Scatter Search and Path Relinking are also intimately related to the Tabu Search metaheuristic, and derive additional advantages by making use of adaptive memory and associated memory-exploiting mechanisms that are capable of being adapted to particular contexts. We describe the features of Scatter Search and Path Relinking that set them apart from other evolutionary approaches, and that offer opportunities for creating increasingly more versatile and effective methods in the future.

[†] Partially supported by the visiting professor fellowship program of the University of Valencia (Grant Ref. No. 42743).

1. Introduction

Scatter Search (SS) and its generalization called Path Relinking (PR) are novel instances of evolutionary methods, because they violate the premise that evolutionary approaches must be based on randomization (see, e.g., Fogel, 1998) — though they likewise are compatible with randomized implementations. SS and PR are also novel, in comparison to the well known Genetic Algorithm (GA) class of evolutionary methods, by being founded on strategies that only piecemeal came to be proposed as augmentations to GAs more than a decade after their debut in Scatter Search. (Today, the methods called "Hybrid GAs" typically derive their main features from a subset of the SS/PR strategies, as will be seen subsequently.) But of greater significance, Scatter Search and Path Relinking embody principles and strategies that are still not emulated by other evolutionary methods, and that prove advantageous for solving a variety of complex optimization problems.

Recent applications of these methods (and of selected component strategies within these methods) that have proved highly successful are shown in Table 1, which expands and updates an earlier table of (Glover, 1999).

Table 1. Illustrative Applications of Scatter Search and Path Relinking Strategies

Application	**Reference**
Vehicle Routing	Rochat and Taillard (1995); Taillard (1996); Rego (1999); Atan and Secomandi (1999)
Arc Routing	Greistorfer (1999)
Quadratic Assignment	Cung et al. (1996, 1977)
Financial Product Design	Consiglio and Zenios (1996)
Neural Network Training	Kelly, Rangaswamy and Xu (1996)
Job Shop Scheduling	Yamada and Nakano (1996); Jain and Meeran (1998a)
Flow Shop Scheduling	Yamada and Reeves (1998, 1999), Jain and Meeran (1998b)
Crew Scheduling	Lourenço, Paixao and Portugal (1998)
Graph Drawing	Laguna and Martí (1999)
Linear Ordering	Laguna, Martí and Campos (1999)
Unconstrained Optimization	Fleurent et al. (1996) Laguna and Martí (2000a)
Bit Representation	Rana and Whitley (1997)
Multi-objective Assignment	Laguna, Lourenço and Martí (2000)
Optimizing Simulation	Glover, Kelly and Laguna (1996)
Tree Problems	Canuto, Resende and Ribeiro (1999) Xu, Chiu and Glover (2000)
Mixed Integer Programming	Glover, Løkketangen and Woodruff (1999)

Improved benchmarks for solving a variety of classical problems have resulted from these applications, along with new advances for solving a significant range of practical business problems, particularly those attended by uncertainty and complex nonlinearities.

In common with other evolutionary methods, Scatter Search and Path Relinking 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 most distinguishing features of this constellation of 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 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. (The word "independent" is relative, because nearly all SS implementations incorporate some of the adaptive memory designs of Tabu Search. Since the TS methodology is better known, we focus in the article on the elements of SS and PR that can be separated from others processes that have a connection to Tabu Search.)

To set the stage for understanding Scatter Search and Path Relinking, and their connections to Tabu Search, we begin by tracing a bit of history of their developments. The following sections, including later discussions of more recent advances, draw on Glover (1997, 1999) and Glover, Laguna and Martí (2000).

2. Historical Background

2.1 Combining Decision Rules

One of the main aspects of SS and PR consists of the manner in which they combine solutions, and undertake to exploit these combinations. As a foundation for seeing the rationale behind these processes, it is useful to examine a related earlier development involving solution strategies based on combining decision rules. This development was launched in the context of scheduling methods to obtain improved local decision rules for job shop scheduling problems (Glover, 1963). The idea was to generate new rules by creating numerically weighted combinations of existing rules, suitably restructured so that their evaluations embodied a common metric.

The approach was motivated by the supposition that information about the relative desirability of alternative choices is captured in different forms by different rules, and that this information can be exploited more effectively when integrated by means of a combination mechanism than when treated by the standard strategy of selecting different rules one at a time, in isolation from each other. In addition, the method departed from the customary approach of stopping upon reaching a local optimum, and instead continued to vary the parameters that determined the combined rules, as a basis for producing additional trial solutions. (This latter strategy is also one of those that became a fundamental part of Tabu Search. See, e.g., Glover and Laguna, 1997.)

The decision rules created from such combination strategies produced better empirical outcomes than standard applications of local decision rules, and also proved superior to a "probabilistic learning approach" (Crowston, *et al*., 1963) that selected different rules probabilistically at different junctures, but without the integration effect provided by generating combined rules.

The next step on the road to the SS/PR framework was to take the idea of creating combinations of elements to produce advantageous new instances by applying it to the domain of systems of constraints.

2.2 Combining Constraints

Soon after the successful introduction of solution strategies based on combining decision rules, associated procedures were introduced for combining constraints. These likewise employed a mechanism of generating weighted combinations, in this case by making use of nonnegative weights to create valid new constraint inequalities, called *surrogate constraints* (Glover, 1965). The approach isolated subsets of constraints that were gauged to be most critical, relative to trial solutions based on the surrogate constraints, and produced new weights that reflected the degree to which the component constraints were satisfied or violated. The combined constraints found particular application in the domains of integer and nonlinear programming.

A principal function of surrogate constraints, in common with the approaches for combining decision rules, was to provide ways to evaluate choices that could be used to generate and modify trial solutions. From this foundation, a variety of heuristic processes evolved that made use of surrogate constraints and their evaluations. Accordingly, these processes led to the complementary strategy of combining solutions, as a *primal* counterpart to the *dual* strategy of combining constraints, which became manifest in Scatter Search and its Path Relinking generalization. (The *primal/dual* distinction stems from the fact that surrogate constraint methods give rise to a mathematical duality

theory associated with their role as relaxation methods for optimization (Greenberg and Pierskalla, 1970, 1973; Glover, 1965, 1975; Karwan and Rardin, 1976, 1979; Freville and Plateau, 1986, 1993).

3. Elements of Scatter Search and Path Relinking

3.1 Scatter Search

The Scatter Search process, building on the principles that underlie the surrogate constraint design, is organized to (1) capture information not contained separately in the original vectors, and (2) take advantage of auxiliary heuristic methods both for selecting the elements to be combined and for generating new vectors.[1].

Scatter Search operates on a set of solutions, the *reference set*, by combining these solutions to create new ones. When the main mechanism for combining solutions is such that a new solution is created from the linear combination of two other solutions, the reference set may evolve as illustrated in Figure 1. This figure assumes that the original reference set of solutions consists of the circles labeled A, B and C. After a non-convex combination of reference solutions A and B, solution 1 is created. More precisely, a number of solutions in the line segment defined by A and B are created; however, in this instance only solution 1 is introduced into the reference set. (The criteria used to select solutions for membership in the reference set are discussed later.) In a similar way, convex and non-convex combinations of original and newly created reference solutions create points 2, 3 and 4. The resulting complete reference set shown in Figure 1 consists of 7 solutions (or elements).

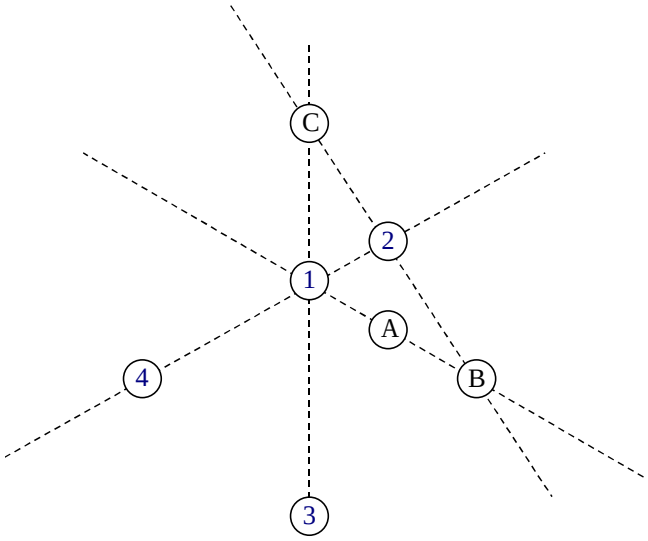


Figure 1. Two-dimensional reference set.

More precisely, Figure 1 shows a precursor form of the resulting reference set. Scatter Search does not leave solutions in a raw form produced by its combination mechanism, but subjects candidates for entry into the reference set to heuristic improvement, as we elaborate subsequently.

Unlike a "population" in genetic algorithms, the reference set of solutions in Scatter Search is relatively small. In genetic algorithms, two solutions are randomly chosen from the population and a "crossover" or combination mechanism is applied to generate one or more offspring. A typical population size in a genetic algorithm consists of 100 elements, which are randomly sampled to create combinations. In contrast, Scatter Search chooses two or more

[1] One group of researchers has argued that the coupling of heuristic improvement with solution combination strategies should be given an entirely new name, and accordingly has inaugurated the term *memetic algorithms* to designate such a coupling (see, e.g., Moscato (1989), Radcliffe and Surrey (1994), Burke, Newall and Weare (1996)).

elements of the reference set in a systematic way with the purpose of creating new solutions. Since the number of two-element to five-element subsets of a reference set, for example, can be quite large, even a highly selective process for isolating preferred instances of these subsets as a basis for generating combined solutions can produce a significant number of combinations, and so there is a practical need for keeping the cardinality of the set small. Typically, the reference set in Scatter Search has 20 solutions or less. In one standard design, if the reference set consists of b solutions, the procedure examines approximately $(3b\text{-}7)b/2$ combinations of four different types. The basic type consists of combining two solutions; the next type combines three solutions, and so on. Note that genetic algorithms need large populations to maintain a desired level of diversity (produce by the random sampling embedded in its search mechanisms), while scatter search systematically injects diversity to the reference set.

Limiting the scope of the search to a selective group of combination types can be used as a mechanism for controlling the number of possible combinations in a given reference set. An effective means for doing this is to subdivide the reference set into "tiers" and to require that combined solutions must be based on including at least one (or a specified number) of the elements form selected tiers.

3.2. Scatter Search Template

By reference to the preceding discussion, the Scatter Search approach may be sketched in basic outline as follows:

1. Generate a starting set of solution vectors to guarantee a critical level of diversity and apply heuristic processes designed for the problem as an attempt for improving these solutions. Designate a subset of the best vectors to be reference solutions. (Subsequent iterations of this step, transferring from Step 4 below, incorporate advanced starting solutions and best solutions from previous history as candidates for the reference solutions.) The notion of "best" in this step is not limited to a measure given exclusively by the evaluation of the objective function. In particular, a solution may be added to the reference set if the diversity of the set improves even when the objective value of the solution is inferior to other solutions competing for admission into the reference set.

2. Create new solutions consisting of structured combinations of subsets of the current reference solutions. The structured combinations are:

   a) chosen to produce points both inside and outside the convex regions spanned by the reference solutions.
   b) modified to yield acceptable solutions. (For example, if a solution is obtained by a linear combination of two or more solutions, a generalized rounding process that yields integer values for integer-constrained vector components may be applied. An acceptable solution may or may not be feasible with respect to other constraints in the problem.)

3. Apply the heuristic processes used in Step 1 to improve the solutions created in Step 2. These heuristic processes must be able to operate on infeasible solutions and may or may not yield feasible solutions.

4. Extract a collection of the "best" improved solutions from Step 3 and add them to the reference set. The notion of "best" is once again broad; making the objective value one among several criteria for evaluating the merit of newly created points. Repeat Steps 2, 3 and 4 until the reference set does not change. Diversify the reference set, by re-starting from Step 1. Stop when reaching a specified iteration limit.

The first notable feature in Scatter Search is that its structured combinations are designed with the goal of creating weighted centers of selected subregions. This adds non-convex combinations that project new centers into regions that are external to the original reference solutions (see, e.g., solution 3 in Figure 1). The dispersion patterns created by such centers and their external projections have been found useful in several application areas.

Another important feature relates to the strategies for selecting particular subsets of solutions to combine in Step 2. 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". Finally, the method is organized to use ancillary 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 following principles summarize the foundations of the Scatter Search methodology:

- 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.

It is to be noted that, in some contexts, high quality solutions are found more often near the boundaries of a feasible region than deep in the interior of the region. For problems that technically have no interior due to the presence of equality constraints, the indicated phenomenon nevertheless can occur relative to a subset of constraints that are inequalities, or relative to a space created by identifying and removing variables that may take the role of slack variables for certain equalities, thus transforming them into inequalities. Then linear combinations may reasonably be biased to generate points that lie within a chosen proximity to the feasible boundary.

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. (This same template also applies to implementing the combinations produced by Path Relinking, as described subsequently.)

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.

Specific processes for carrying out these steps are described in Glover (1997). A numerical example illustrating the use of this template can be found in Glover, Laguna and Martí (2000).

4. Path Relinking

Features that have been added to Scatter Search, by extension of its basic philosophy, are captured in the Path Relinking framework. From a spatial orientation, the process of generating linear combinations of a set of reference

solutions may be characterized as generating paths between and beyond these solutions, where solutions on such paths also serve as sources for generating additional paths. This leads to a broader conception of the meaning of creating *combinations* of solutions. By natural extension, such combinations may be conceived to arise by generating paths between and beyond selected solutions in neighborhood space, rather than in Euclidean space (Glover 1989, 1994a; Glover and Laguna, 1993).

This conception is reinforced by the fact that a path between solutions in a neighborhood space will generally yield new solutions that share a significant subset of attributes contained in the parent solutions, in varying "mixes" according to the path selected and the location on the path that determines the solution currently considered. The character of such paths is easily specified by reference to solution attributes that are added, dropped or otherwise modified by the moves executed in neighborhood space. 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.

The incorporation of attributes from elite parents in partially or fully constructed solutions was foreshadowed by another aspect of Scatter Search, embodied in an accompanying proposal to assign preferred values to subsets of *consistent* and *strongly determined* variables. The theme is to isolate assignments that frequently or influentially occur in high quality solutions, and then to introduce compatible subsets of these assignments into other solutions that are generated or amended by heuristic procedures. (Such a process implicitly relies on a form of frequency-based memory to identify and exploit variables that qualify as consistent.)

Multiparent path generation possibilities emerge in Path Relinking by considering the combined attributes provided by a set of guiding solutions, where these attributes are weighted to determine which moves are given higher priority. The generation of such paths in neighborhood space characteristically “relinks” previous points in ways not achieved in the previous search history, hence giving the approach its name.

4.1 Initial Steps

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 (or more aggressively, a “fewest” number of moves). This policy, even if applied without exception, can permit a significant number of alternative choices for generating the next solution at each step. Consequently, additional criteria are relevant to creating the path, as indicated shortly.

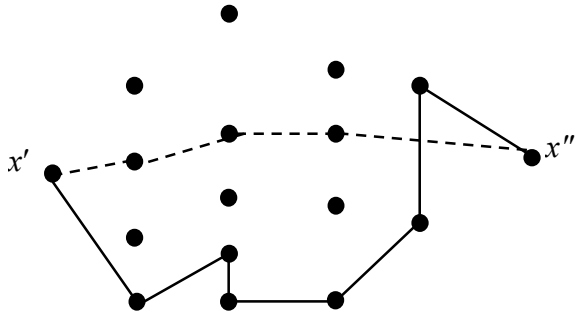


Figure 2. Path Relinking: Original path shown by heavy line and one possible relinked path shown by dotted line.

It is possible, as in applying Scatter Search, that x′ and x″ were previously joined by a search trajectory produced by a heuristic method (or by a metaheuristic such as Tabu Search). In this event, the new trajectory created by Path Relinking is likely to be somewhat different than the one initially established, representing a 'more direct route' between the solutions. An illustration of this is given in Figure 2.

It may also be that x′ and x″ were not previously joined by a search path at all, but were generated on different search paths, which may have been produced either by a heuristic or by a previous relinking process. Such a situation is depicted in Figure 3.

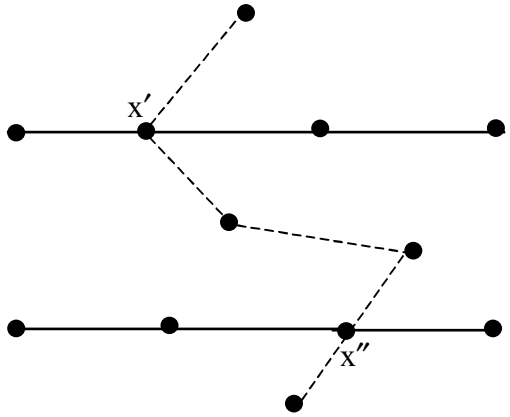


Figure 3. Path Relinking: Previously generated paths shown by heavy lines; relinked path shown by dotted line. (Multiple additional points in the space are not shown.)

In this case, the path between x′ and x″ performs a relinking function by changing the connections that generated x′ and x″ originally. The relinking path of this diagram is shown as extending beyond the points x′ and x″. We discuss this type of construction subsequently under the heading of *extrapolated relinking*.

To choose among the different paths that may be possible in going from x' to x''**,** let $c(x)$ denote an objective function which is to be minimized. Selecting unattractive moves relative to $c(x)$, 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 $c(x)$ 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''.)

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 $c(x(i))$ but also on the values $c(x)$ of those solutions x that can be reached by a move from $x(i)$. The process can additionally 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.

To elaborate the process, let $x^*(i)$ denote a neighbor of $x(i)$ that yields a minimum $c(x)$ value during an evaluation step, excluding $x^*(i) = x(i + 1)$. If the choice rules do not automatically eliminate the possibility $x^*(i) = x(h)$ for $h < i$, then a simple tabu restriction can be used to do this (e.g., see Glover and Laguna, 1997). Then the method selects a solution $x^*(i)$ that yields a minimum value for $c(x^*(i))$ as a new point to launch the search. If only a limited set of neighbors of $x(i)$ are examined to identify $x^*(i)$, then a superior least cost solution $x(i)$, excluding x' and x'', may be selected instead. Early termination becomes possible (though is not compulsory) upon encountering an $x^*(i)$ that yields $c(x^*(i)) < min(c(x'),c(x''),c(x(p))$, where $x(p)$ is the minimum cost $x(h)$ for all $h < i$. The procedure will continue if $x(i)$, in contrast to $x^*(i)$, yields a smaller $c(x)$ value than x' and x'', since $x(i)$ effectively adopts the role of x'.

4.2 Variation and Tunneling

A variant of the Path 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 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 benefits from a tunneling strategy that often 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'', at least one of $x'(r)$ and $x''(s)$ may be kept feasible. An example of tunneling is shown in Figure 4.

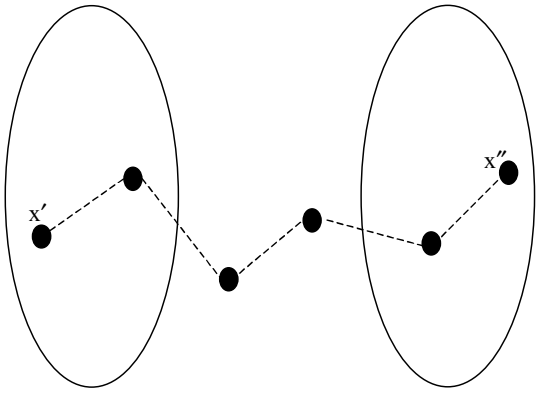


Figure 4. Feasible region consists of disconnected components. The path "tunnels through" the infeasible region to regain feasibility.

As in Tabu Search strategies for achieving intensification and diversification, it is appropriate to select the points x'

and x'' by reference to clusters of solutions that are created according to criteria of similarity or affinity. Choosing x' and x'' from the same cluster then stimulates intensification, while choosing them from two 'widely separated' clusters stimulates diversification. Alternately, parents can be chosen by "anti-clustering," where each parent is selected to be as far as possible from those previously chosen within the same "family" of parents. A separation criterion such as maximizing the minimum distance to previous points can be used, for example.

4.3 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 invoke a Tabu Search concept, 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.

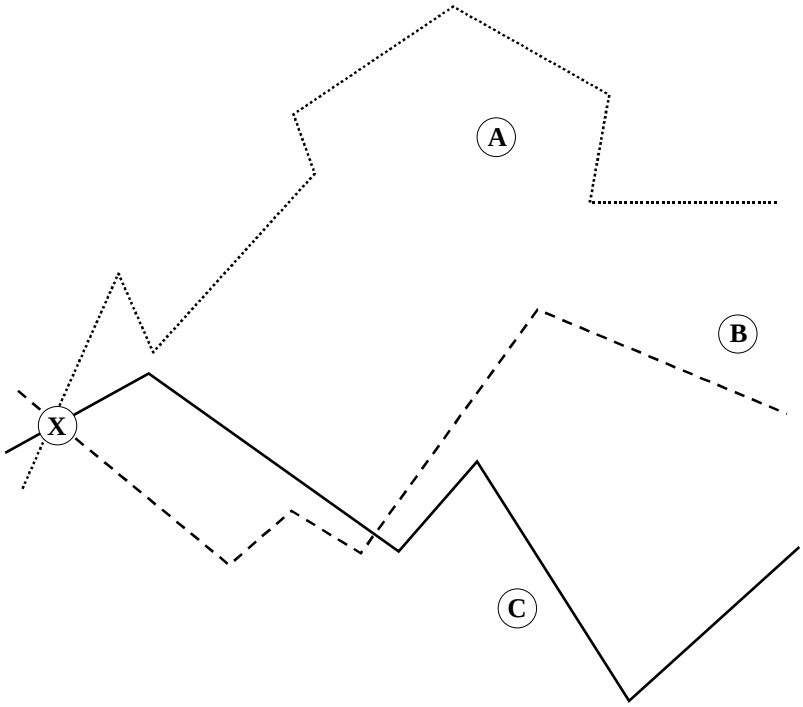
For neighborhoods that allow relatively unrestricted choices of moves, this approach yields a path extending beyond x'' that introduces new attributes, without reincorporating any old attributes, until no move remains that satisfies this condition. The ability to go beyond the limiting points x' and x'' creates a form of diversification analogous to that provided by the original Scatter Search approach. At the same time the exterior points are influenced by the trajectory that links x' and x''.

4.4 Multiple Parents

New points can be generated from multiple parents 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. 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 (1994b), 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.

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. Figure 5 depicts the type of variation that can result, where the point X represents an initiating solution and the points A, B, and C represent guiding solutions. For appropriate choices of the reference points (and neighborhoods for generating paths from them), principles such as those discussed in Glover and Laguna (1997) suggest that additional elite points are likely to be found in the regions traversed by the paths, upon launching new searches from high quality points on these paths.



X = Solution selected to generate a relinked path.
(A, B and C may also interchange roles with X, alternately or simultaneously.)

Figure 5. Neighborhood space paths with different attribute trade-offs.

4.5 Constructive Neighborhoods

A natural variation of Path Relinking occurs by using constructive neighborhoods for creating offspring from a collection of parent 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. (See Alvarez-Valdés, et al. (2000) for an example of this Path Relinking variant.)

The evaluations can be conceived as produced by a process where the guiding solutions *vote* for attributes to be included in the initiating solution. It is possible, for example, that a certain partial configuration may be reached where none of the attributes of the guiding solutions can be incorporated within the existing solution, relative to a

given constructive neighborhood. Then it is important to still be able to select a next constructive step, by relying upon the voting process for evaluating moves. This same consideration can arise in transition neighborhoods, though it is encountered less frequently there.

Combinations created in this way are called *structured combinations*, and their generation rests upon three properties.

> *Property 1* (*representation property*). Each guiding solution represents a vector of votes for particular decisions (e.g., the decision of assigning a specific value to a particular variable).
> *Property 2* (*trial solution property*). The votes prescribed by a guiding solution translate into a trial solution to the problem of interest by a well-defined process (determined by the neighborhood structure).
> *Property 3* (*update property*). If a decision is made according to the votes of a given vector, a clearly defined rule exists to update all voting vectors for the residual problem so that Properties 1 and 2 continue to hold.

Features of these properties in particular contexts may be clarified as follows.

> *Elaboration of Property 1*: Standard solution vectors for many problems can directly operate as voting vectors, or can be expanded in a natural way to create such vectors. For instance, a solution vector for a job shop scheduling problem can be interpreted as a set of 0-1 votes for predecessor decisions in scheduling specific jobs on particular machines.
> *Elaboration of Property 2*: A set of "yes-no" votes for items to include in a knapsack, for instance, can be translated into a trial solution according to a designated sequence for processing the votes (such as determined by benefit-to-weight ratios), until either the knapsack is full or all votes are considered. More general numerical votes for the same problem may additionally prescribe the sequence to be employed, as where knapsack items are rearranged so the votes occur in descending order. (The voting vectors are not required to represent feasible solutions to the problems considered, or even represent solutions in a customary sense at all. Thus, for example, the scheme can also operate to combine decision rules as in the approach for doing this described in section 2.1.)
> *Elaboration of Property 3*. Upon assigning a specific value to a particular variable, all votes for assigning different values to this variable effectively become cancelled. Property 3 then implies that 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.

Concrete illustrations of processes for generating structured combinations by reference to these properties are provided in Glover (1994b). These same kinds of processes can be implemented by reference to destructive neighborhoods – that is, neighborhoods that allow the removal of less attractive elements. Typically, destructive processes are applied to solutions that begin with an "excessive assignment" (such as too many elements to satisfy cardinality or capacity restrictions).

4.6 Vocabulary Building

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.

(The same principle operates in mathematical analysis generally, where basic premises are organized to produce useful lemmas, which in turn facilitate the generation of more complex theorems.)

Vocabulary building has an additional useful feature in some problem settings by providing compound elements linked by special neighborhoods that are more exploitable than the neighborhoods that operate on the primitive elements. For example, a vocabulary-building proposal of Glover (1992) discloses that certain subassemblies (partial "tours") for traveling salesman problems can be linked by exact algorithms to produce optimal unions of these components. Variants of this strategy have more recently been introduced by Aggarwal, Orlin and Tai (1997) as a proposal for modifying traditional genetic algorithms, and have also been applied to weighted clique problems by Balas and Niehaus (1998). A particularly interesting application occurs in the work of Lourenço, Paixao and Portugal (1998), who use such concepts to create "perfect children" for crew scheduling problems.

In general, vocabulary building relies on destructive as well as constructive processes to generate desirable partial solutions, as in the early proposals for exploiting strongly determined and consistent variables – which essentially "break apart" good solutions to extract good component assignments, and then subject these assignments to heuristics to rebuild them into complete solutions. Construction and destruction therefore operate hand in hand in these approaches. An illustration of vocabulary building is depicted in Figure 6.

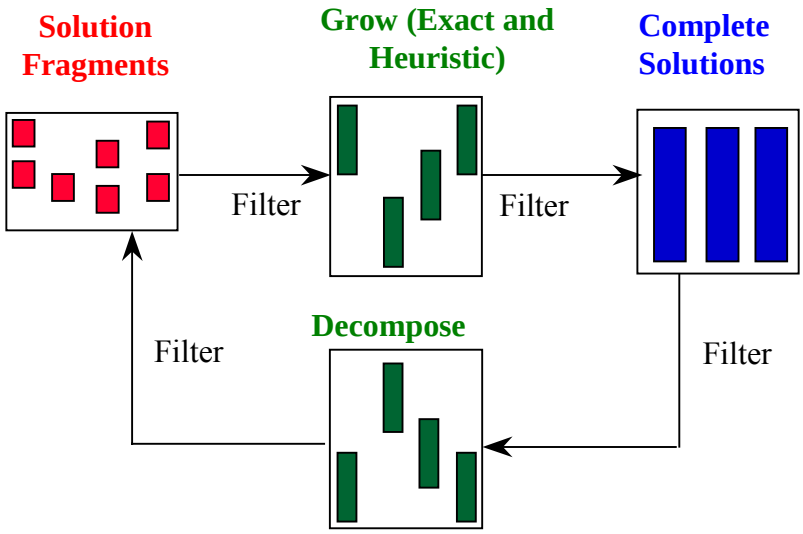


Figure 6. Vocabulary Building Process

5. Illustrative Practical Application – Combining Optimization and Simulation

A variety of applications of Scatter Search and Path Relinking have already been documented in Table 1 of Section 1. Here we describe an application that is especially important in practice.

Many real world problems in optimization are too complex to be given tractable mathematical formulations. Multiple nonlinearities, combinatorial relationships and uncertainties often render challenging practical problems inaccessible to modeling except by resorting to simulation — an outcome that poses grave difficulties for classical optimization methods. In such situations, recourse is commonly made to itemizing a series of scenarios in the hope that at least one will give an acceptable solution. Consequently, a long standing goal in both the optimization and simulation communities has been to create a way to guide a series of simulations to produce high quality solutions, in

the absence of tractable mathematical structures. Applications include the goals of finding:

- the best configuration of machines for production scheduling
- the best integration of manufacturing, inventory and distribution
- the best layouts, links and capacities for network design
- the best investment portfolio for financial planning
- the best utilization of employees for workforce planning
- the best location of facilities for commercial distribution
- the best operating schedule for electrical power planning
- the best assignment of medical personnel in hospital administration
- the best setting of tolerances in manufacturing design
- the best set of treatment policies in waste management

and many other objectives.

The integration of the SS/PR framework with classical optimization has produced a practical software system, called OptQuest[2], that is capable of guiding a series of simulations to uncover optimal or near optimal solution scenarios. Within the brief span since it has come into existence, OptQuest has been used in several thousand real world applications that combine simulation and optimization, and has also been used to determine optimal parameters for other computer based decision tools, to increase their effectiveness. (The optimization engine of OptQuest is implemented as a library of callable functions, as described in Laguna and Martí (2000b).)

OptQuest is designed to search for optimal solutions to the following class of optimization problems:

Max or Min	$f(x)$	
Subject to	$Ax \leq b$	(Constraints)
	$g_l \leq g(x) \leq g_u$	(Requirements)
	$l \leq x \leq u$	(Bounds)

where x can be continuous or discrete with an arbitrary step size.

The objective $f(x)$ may be any mapping from a set of values x to a real value. The set of constraints must be linear and the coefficient matrix A and the right-hand-side values b must be known. The requirements are simple upper and/or lower bounds imposed on a function that can be linear or non-linear. The values of the bounds g_l and g_u must be known constants. All the variables must be bounded and some may be restricted to be discrete with an arbitrary step size.

OptQuest is designed as a general-purpose optimizer that operates completely outside of the system being optimized, which is typically represented by a simulation. That is, OptQuest treats the simulation as a "black-box" to which it sends a set of inputs and from which it reads a set of outputs (Figure 7). A potential disadvantage of this "black box" approach is that the optimization procedure is generic and does not use any problem-specific information. The great advantage, on the other hand, is that the same optimizer can be used for many different simulated systems. That is, the simulation can change and evolve to incorporate additional elements of a complex system, while the optimization routines remain the same

[2] OptQuest is a registered trademark of Optimization Technologies, Inc. (www.opttek.com).

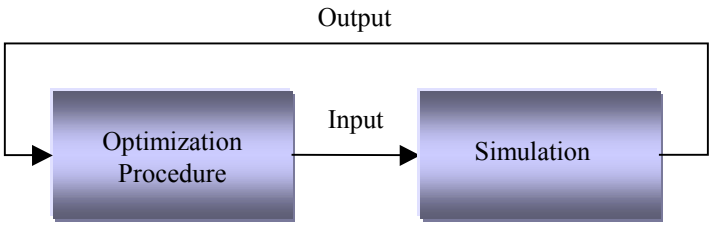


Figure7. Optimizing a simulation with a "black box" approach.

The optimization procedure uses the outputs from the simulation, which measures the merit of the inputs that were fed into the model, to decide upon a new set of input values. In other words, the application of the scatter search elements such as the Diversification and Combination Methods is based on both current and past evaluations of inputs. The procedure carries out a non-monotonic search, where the successively generated inputs produce varying evaluations, not all of them improving, but which over time provide a highly efficient trajectory to the best solutions. The process continues until an appropriate termination criterion is satisfied (usually based on the user's preference for the amount of time to be devoted to the search).

There are two situations that add another level of complexity to the process of optimizing a simulation: (a) when the inputs to the simulation must satisfy a set of linear constraints, and (b) when some output measures must be within specified bounds. That is, the optimization model has both constrains and requirements. The optimization procedure can guarantee feasibility of the inputs by mapping infeasible inputs into feasible ones. However, requirements result in a situation where the feasibility of x is not known prior to the completion of the simulation. This situation is depicted in Figure 8.

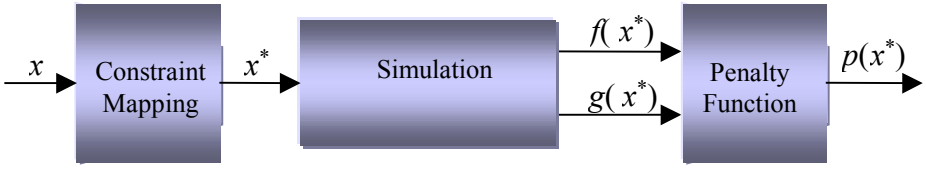


Figure 8. Evaluation with constraints and requirements.

Figure 8 shows that when constraints are included in the optimization model, the evaluation process starts with the mapping $x \rightarrow x^*$. If the only constraints in the model are integrality restrictions, the mapping is achieved with a strategic rounding mechanism that transforms continuous values into allowable discrete values. If the constraints are linear the mapping consists of formulating and solving a linear programming problem that minimizes the distance between x and x^*. Finally, if the constraints are linear and the model also includes discrete variables, then the linear programming formulation becomes a mixed-integer programming problem that must be solved accordingly.

The mapped solution is simulated to obtain a set of performance measures. One of these measures is used as the objective function value $f(x)$ and provides the means for the search to distinguish good from bad solutions. Other measures $g(x)$ associated with the performance of the system are used to define a set of requirements. A requirement is expressed as a bound on the value of a performance measure $g(x)$. Thus, a requirement may be defined as an upper or a lower bound on an output of the simulator. Instead of discarding requirement-infeasible solutions, OptQuest handles them with a composite function $p(x)$ that penalizes the requirement violations. The penalty is proportional to the degree of the violation and is not static throughout the search. OptQuest assumes that the user is interested in finding a requirement-feasible solution if one exists. Therefore, requirement-infeasible solutions are penalized more heavily when no requirement-feasible solution has been found during the search than when one is already available.

To illustrate the evaluation process in this context, consider an investment problem for which x represents the allocation of funds to a set of investment instruments. The objective is to maximize the expected return. Assume that a Monte Carlo simulation is performed to estimate the expected return $f(x)$ for a given fund allocation.

Restrictions on the fund allocations, which establish relationships among the variables, are handled within the linear programming formulation that maps infeasible solutions into feasible ones. Thus, a restriction of the type "the combined investment in instruments 2 and 3 should not exceed the total investment in instrument 7," results in the linear constraint $x_2 + x_3 \leq x_7$. On the other hand, a restriction that limits the variability of the returns (as measured by the standard deviation) to be no more than a critical value c cannot be enforced in the input side of the Monte Carlo simulator. Clearly, the simulation must be executed first in order to estimate the variability of the returns. Suppose that the standard deviation of the returns is represented by $g(x)$, then the requirement in this illustrative situation is expressed as $g(x) \leq c$.

Note that the constraint-mapping mechanism within OptQuest does not handle nonlinear constraints. However, nonlinear constraints can be modeled as requirements and incorporated in the penalty function $p(x)$. For example, suppose that an optimization model includes the following nonlinear constraint:

$$x_1 x_2 - x_1^2 \leq 120$$

Then, the simulator calculates the left-hand side of the nonlinear constraint and communicates the result as one of the outputs. OptQuest uses this output and compares it to the right-hand side value of 120 to determine the feasibility of the current solution. If the solution is not feasible a penalty term is added to the value of the objective function $f(x)$.

6. Implications for Future Developments

The focus and emphasis of the Scatter Search and Path Relinking approaches have a number of specific implications for the goal of designing improved optimization procedures. To understand these implications, it is useful to consider certain contrasts between the highly exploitable meaning of "solution combination" provided by Path Relinking and the rather amorphous concept of "crossover" used in genetic algorithms. Originally, GAs were founded on precise notions of crossover, using definitions based on binary stings and motivated by analogies with genetics. Although there are still many GA researchers who favor the types of crossover models originally proposed with genetic algorithms — since these give rise to the theorems that have helped to popularize GAs — there are also many who have largely abandoned these ideas and who have sought, on a case-by-case basis, to replace them with something different. The well-defined earlier notions of crossover have not been abandoned without a price. The literature is rife with examples where a new problem (or a new variant of an old one) has compelled the search for an appropriate "crossover" to begin anew.[3]

As a result of this lack of an organizing principle, many less-than-suitable modes of combination have been produced, some eventually replacing others, without a clear basis for taking advantage of context — in contrast to the strong context-exploiting emphasis embodied in the concept of search neighborhoods. The difficulty of devising a unifying basis for understanding or exploiting context in GAs was inherited from its original theme, which had the goal of making GAs *context free.*

A few of the more conspicuous features of "genetic crossover" and Path Relinking that embody such contrasts appear in Table 2.

Table 2. Comparison between GA and PR features.

"Genetic Crossover" Features	**Contrasting Path Relinking Features**
Contains no integrated framework	Embodies a unifying "path combination" principle
Each new "crossover" is separate, with no guidance for the next	Each implementation of path relinking derives from a common foundation
No basis exists to systematically exploit context	Context inheres in neighborhood structures and is directly exploitable by them

[3] The disadvantage of lacking a clear and unified model for combining solutions has had its compensations for academic researchers, since each new application creates an opportunity to publish another form of crossover! The resulting abundance of papers has done nothing to tarnish the image of a dynamic and prospering field.

Advances are piecemeal, without clear sources of potential for transfer	Advances in neighborhood search foster advances in path relinking (and reciprocally)
There is no design plan that is subject to analysis or improvement	A cohesive framework exists for developing progressively improved methods

The differences identified in Table 2 have important consequences for research to yield improved methods. Specific areas of research for developing improved solution strategies that emerge directly from the Path Relinking orientation are catalogued in Table 3.

Table 3. Research opportunities.

Research Areas Providing Opportunities for Improved Methods
1. Connections and complementarities between neighborhoods for search methods and neighborhoods for path relinking
2. Rules for generating paths to different depths and thresholds of quality
3. Strategies for generating multiple paths between and beyond reference solutions (with parallel processing applications)
4. Path interpolations and extrapolations that are effective for intensification and diversification goals
5. Strategies for clustering and anti-clustering, to generate candidate sets of solutions to be combined
6. Rules for multi-parent compositions
7. Isolating and assembling solution components by means of constructive linking and vocabulary building

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 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. The learning approach called target analysis (Glover and Laguna, 1997) gives a particularly useful basis for pursuing such research.

7. Intensification and Diversification

A significant feature that distinguishes Scatter Search and Path Relinking from other evolutionary approaches is the fact that intensification and diversification processes are not conceived to be embedded solely within the mechanisms for combining solutions, or within supplementary "mutation" strategies based on randomly varying offspring to produce new solutions.[4]

Evidently, except where hybrids are being created with Tabu Search, alternative evolutionary computation approaches do not undertake to control search paths by adaptive memory strategies such as those based on measures of recency, frequency and influence.

The initial connections between Scatter Search and strategies involving these types of measures have already been noted in reference to exploiting *consistent* and *strongly determined* variables. Such strategies naturally fall in the category of intensification strategies, in the sense that they undertake to take advantage of features associated with good solutions. They are predicated on highly explicit analysis of the frequencies that attributes belong to high quality solutions, supplemented by considerations such as clustering the solutions to give increased meaning to the

[4] Within the last few years, some researchers in the evolutionary computation field have begun to adopt aspects of Scatter Search and Path Relinking by incorporating systematic strategies for achieving intensification and diversification, instead of relying on randomization to achieve less purposeful forms of variation. However, some of the latest literature still disallows this type of approach as a legitimate feature of evolutionary computation. For example, Fogel (1998) says that the main disciplines of evolutionary computation all involve a process whereby "New solutions are created by randomly varying the existing solutions."

frequencies. This stands in notable contrast to the philosophy of other mainstream evolutionary procedures, where the relevance of attribute membership in solutions is left to be "discovered" chiefly by the device of shuffling and combining solutions.

An approach called *strategic oscillation* introduced with the original Scatter Search proposal is important for linking intensification and diversification. The basic idea of this approach is to identify critical regions of search, and to induce the search pattern to visit these regions to various depths within their boundaries, by a variable pattern that approaches and retreats from the boundaries in oscillating waves. Examples of such regions and their associated boundaries are indicated in the following table.

Table 4. Applications of Strategic Oscillation

Regions	Boundary
Feasible and infeasible	Determined by constraints
Partial constructions and (sometimes) "excessive" constructions	A complete construction (tree, clique, tour, etc.)
Underfilled or overfilled schedules	Satisfied schedules (all or an appropriate set of jobs assigned)
Local optima and suboptima	Solutions with no immediate improvement
Elite solution clusters (or partitioned spaces)	Zones between clusters (or between partitions)
Alternative neighborhoods	Transitions among move types

A number of variations of the regions and associated boundaries are evident, such as replacing the feasible/infeasible dichotomy by a focus on selected critical constraints to define varying domains of "partial infeasibility." In each case, the strategic oscillation approach operates by moving through one region to approach the boundary, and then either crosses the boundary or reverses direction to move back into the region just traversed (in the case of a "one-sided" oscillation).

Often there are advantages to crossing boundaries to descend for varying depths within different regions and then doubling back to return again to the boundary. For example, in a number of discrete optimization problems the solutions that are most readily accessible from the feasible and infeasible regions differ — and, quite significantly the types of moves and choice criteria for traversing feasible and infeasible regions also differ. The ability to exploit these differences by rules that are specific to the regions traversed and the direction of movement within these regions (e.g., toward or away from their boundaries) provides an enriched set of options for carrying out the search. Similar characteristics are found in processes that build solutions by constructive processes and then dismantle them by destructive processes. In many settings, classical heuristics have been restricted to constructive processes for generating solutions and in these cases strategic oscillation entails the creation of additional destructive moves to complement the constructive moves.

From the perspective of intensification and diversification, greater diversification is normally achieved by penetrating to greater depths beyond regional boundaries, while greater intensification is normally achieved by spending more time in the vicinity of such boundaries. However, the spatial image of remaining close to a boundary is misleading, because oscillations of small depths can create significant changes. For example, even when only a few destructive moves are made to reverse a constructive process, the portions of the construction dismantled can have a significant influence on the solution structure and composition, and this influence can become magnified after a few oscillation cycles. The guidance of memory as used in Tabu Search allows the oscillations to avoid becoming mired in local optima or in a process that unproductively examines similar points in a common locale.

An extreme application of strategic oscillation in the context of a constructive approach is to employ destructive steps to completely dismantle the solution built by the constructive phase, which thus simply reduces the approach to a "restart" procedure. However, the use of memory to guide the successively restarted constructions produces significantly different outcomes than those produced by customary restarting procedures based on randomization. These outcomes also contrast sharply with those produced by "randomized greedy" construction schemes. The

comparative advantages of memory-based strategies for rebuilding solutions are documented, for example, in Fleurent and Glover (1999).

Intensification is often associated with shallow oscillations because in many settings the best solutions are found on or near the boundary. This is clearly true for multidimensional knapsack and covering problems, for instance, and it is also true by definition where the boundary is established to segregate local optima from suboptimal solutions. In such cases the oscillation process is augmented by spending additional time in the proximity of the boundary, as by shifting from a simple neighborhood to a more complex neighborhood. For instance, simple "add/drop" or "increment/decrement" moves may be augmented by a series of "exchange" or "paired increment/decrement" moves upon reaching (or drawing close to) the boundary. Candidate list strategies are important when complex neighborhoods are used, in order to achieve proper tradeoffs between time spent looking for moves and the quality of the moves found. (Principal approaches of this type are described in Glover and Laguna, 1997.)

7.1 Randomization and the Intensification/Diversification Dichotomy

The emphasis on systematic strategies in achieving intensification and diversification does not preclude the use of randomized selection schemes, which are often motivated by the fact that they require little thought or sophistication to apply. By the same token, deterministic rules that are constructed with no more reflection than devoted to creating a simple randomized rule can be quite risky, because they can easily embody oversights that will cause them to perform poorly. A randomized rule can then offer a safety net, by preventing a bad decision from being applied persistently and without exception.

Yet a somewhat different perspective suggests that deterministic rules can offer important advantages in the longer run. A "foolish mistake" incorporated into a deterministic rule becomes highly visible by its consequences, whereas such a mistake in a randomized rule may be buried from view — obscured by the patternless fluctuations that surround it. Deterministic rules afford the opportunity to profit by mistakes and learn to do better. The character of randomized rules, that provides the chance to escape from repetitive folly, also inhibits the chance to identify more effective decisions.

The concepts of intensification and diversification are predicated on the view that intelligent variation and randomized variation are rarely the same.[5] This clearly contrasts with the prevailing perspective in the literature of evolutionary methods although, perhaps surprisingly, the intensification and diversification terminology has been appearing with steadily increasing frequency in this literature. Nevertheless, a number of the fundamental strategies for achieving the goals of intensification and diversification in Scatter Search and Path Relinking applications have still escaped the purview of other evolutionary methods.

Perhaps one of the factors that is slowing a more complete assimilation of these ideas is a confusion between the terminology of intensification and diversification and the terminology of "exploitation versus exploration" popularized in association with genetic algorithms. The exploitation/exploration distinction comes from control theory, where exploitation refers to following a particular recipe (traditionally memoryless) until it fails to be effective, and exploration then refers to instituting a series of random changes — typically via multi-armed bandit schemes — before reverting to the tactical recipe. The issue of exploitation versus exploration concerns how often and under what circumstances the randomized departures are launched.

By contrast, intensification and diversification are mutually reinforcing (rather than being mutually opposed), and can be implemented in conjunction as well as in alternation. In longer term strategies, intensification and diversification are both activated when simpler tactics lose their effectiveness. Characteristically, they are designed to profit from memory, rather than to rely solely on indirect "inheritance effects."

8. Conclusion

[5] Intelligence can sometimes mean quickly doing something mildly clever, rather than slowly doing something profound. This can occur where the quality of a single move obtained by extended analysis is not enough to match the quality of multiple moves obtained by more superficial analysis. Randomized moves, which are quick, sometimes gain a reputation for effectiveness because of this phenomenon. In such setting, a different perspective may result by investigating comparably fast mechanisms that replace randomization with intelligent variation.

The preceding sections make it clear there are a number of aspects of Scatter Search and its Path Relinking generalization that warrant further investigation. Additional implementation considerations, including associated intensification and diversification processes, and the design of accompanying methods to improve solutions produced by combination strategies, may be found in the *template* for Scatter Search and Path Relinking in Glover (1997) and in the illustration of Scatter Search for a special class of nonlinear problems in Glover, Laguna and Martí (2000).

A key observation deserves to be stressed. The literature often contrasts evolutionary methods — especially those based on combining solutions — with local search methods, as though these two types of approaches are fundamentally different. In addition, evolutionary procedures are conceived to be independent of any reliance on memory, except in the very limited sense where solutions forged from combinations of others carry the imprint of their parents. Yet as previously noted, the foundations of Scatter Search strongly overlap with those of Tabu Search and, in addition, Path Relinking was initiated as a strategy to be applied with the guidance of adaptive memory processes. By means of these connections, a wide range of strategic possibilities exist for implementing Scatter Search and Path Relinking.

Very little computational investigation of these methods has been done by comparison to other evolutionary methods, and a great deal remains to be learned about the most effective implementations for various classes of problems. The highly promising outcomes of studies such as those cited in Section 1 suggest that these approaches may offer a useful potential for applications in areas beyond those investigated up to now.

The ability to use the SS/PR framework in conjunction with classical optimization to deal with exceedingly general kinds of optimization problems that include uncertainty, as illustrated by the OptQuest system described in Section 5, reinforces this supposition and underscores the relevance of such developments for handling real world problems.

References

[1] Aggarwal, C.C., J.B. Orlin and R.P. Tai (1997). "Optimized Crossover for the Independent Set Problem," *Operations Research* 45, 226-234.

[2] Alvarez-Valdés, R., A. Parajón and J. M. Tamarit (2000). "A Tabu Search Algorithm for Large-Scale Guillotine (Un)Constrained Two-dimensional Cutting Problems," Working paper, University of Valencia.

[3] Atan, T. and N. Secomandi (1999). "A Rollout-Based Application of the Scatter Search/Path Relinking Template to the Multi-Vehicle Routing Problem with Stochastic Demands and Restocking," PROS Revenue Management, Inc., Houston, TX.

[4] Balas, E., and W. Niehaus (1998). "Optimized Crossover-Based Genetic Algorithms for the Maximum Cardinality and Maximum Weight Clique Problems," *Journal of Heuristics*, Vol 4:2. 107-124.

[5] Burke, E.K., J.P. Newall and R.F. Weare (1996). "A Memetic Algorithm for University Timetabling," in *The Practice and Theory of Automated Timetabling*, (E.K. Burke and P. Ross eds.). Springer-Verlag Lecture Notes in Computer Science. 1153.

[6] Canuto, S. A., M.G.C. Resende and C.C. Ribeiro, (1999). "Local Search with Perturbations for the Prize-Collecting Steiner Tree Problem in Graphs," AT&T Labs Research Technical Report 99.14.2.

[7] Consiglio, A. and S.A. Zenios (1996). "Designing Portfolios of Financial Products via Integrated Simulation and Optimization Models," Report 96-05, Department of Public and Business Administration, University of Cyprus, Nicosia, CYPRUS, to appear in *Operations Research.*

[8] Consiglio, A. and S.A. Zenios (1997). "A Model for Designing Callable Bonds and its Solution Using Tabu Search," *Journal of Economic Dynamics and Control* 21, 1445-1470.

[9] Coy, S., B. Golden, G. Runger, and E. Wasil (1998). "See the Forest Before the Trees: Finetuned Learning and its Application to the Traveling Salesman Problem," *IEEE Transactions on Systems, Man, and Cybernetics*, 28(4), 454-464.

[10] Coy, S. P., B. L. Golden and E. A. Wasil (1998). "A computational Study of Smoothing Heuristics for the Traveling Salesman Problem," Research report, University of Maryland, to appear in *European Journal of Operational Research*.

[11] Crowston, W.B., F. Glover, G.L.Thompson and J.D. Trawick (1963). "Probabilistic and Parametric Learning Combinations of Local Job Shop Scheduling Rules," ONR Research Memorandum No. 117, GSIA, Carnegie

Mellon University, Pittsburgh, PA.
[12] Cung, V-D., T. Mautor, P. Michelon, A. Tavares (1996). "Scatter Search for the Quadratic Assignment Problem," Laboratoire PRiSM-CNRS URA 1525.
[13] Fleurent, C., F. Glover, P. Michelon and Z. Valli (1996). "A Scatter Search Approach for Unconstrained Continuous Optimization," *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, 643-648.
[14] Fleurent, C. and F. Glover (1999). "Improved Constructive Multistart Strategies for the Quadratic Assignment Problem Using Adaptive Memory," *INFORMS Journal on Computing*, Vol. 11, No. 2, 198-204.
[15] Fogel, David B. (1998). "An Introduction to Evolutionary Computation," Chapter 1 of *Evolutionary Computation: The Fossil Record*, D.B. Fogel, ed., IEEE Press, 1 - 2.
[16] Freville, A. and G. Plateau (1986). "Heuristics and Reduction Methods for Multiple Constraint 0-1 Linear Programming Problems," *European Journal of Operational Research*, 24, 206-215.
[17] Freville, A. and G. Plateau (1993). "An Exact Search for the Solution of the Surrogate Dual of the 0-1 Bidimensional Knapsack Problem," *European Journal of Operational Research*, 68, 413-421.
[18] Glover, F. (1963). "Parametric Combinations of Local Job Shop Rules," Chapter IV, ONR Research Memorandum no. 117, GSIA, Carnegie Mellon University, Pittsburgh, PA.
[19] Glover, F. (1965). "A Multiphase Dual Algorithm for the Zero-One Integer Programming Problem," *Operations Research*, Vol 13, No 6, 879.
[20] Glover, F. (1975). "Surrogate Constraint Duality in Mathematical Programming," *Operations Research*, 23, 434-451.
[21] Glover, F. (1977). "Heuristics for Integer Programming Using Surrogate Constraints," *Decision Sciences*, Vol 8, No 1, 156-166.
[22] Glover, F. (1992). "Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems," University of Colorado. Shortened version published in *Discrete Applied Mathematics*, 1996, 65, 223-253.
[23] Glover, F. (1994a). "Genetic Algorithms and Scatter Search: Unsuspected Potentials," S*tatistics and Computing*, 4, 131-140.
[24] Glover, F. (1994b). "Tabu Search for Nonlinear and Parametric Optimization (with links to genetic algorithms)," *Discrete Applied Mathematics*, 49, 231-255.
[25] Glover, F. (1994c). "Optimization by Ghost Image Processes in Neural Networks," *Computers and Operations Research*, Vol 21, No. 8, 801-822.
[26] 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.), pp. 13-54. (Expanded version available on request.)
[27] Glover F. (1999). "Scatter Search and Path Relinking," In: D Corne, M Dorigo and F Glover (eds.) *New Ideas in Optimisation,* Wiley.
[28] Glover, F., J. P. Kelly and M. Laguna (1996). "New Advances and Applications of Combining Simulation and Optimization," *Proceedings of the 1996 Winter Simulation Conference*, J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain (Eds.), 144-152.
[29] Glover, F. and M. Laguna (1997). *Tabu Search*, Kluwer Academic Publishers.
[30] Glover, F., M. Laguna and R. Martí (2000). "Scatter Search," to appear in *Theory and Applications of Evolutionary Computation: Recent Trends*, A. Ghosh and S. Tsutsui (Eds.) Springer-Verlag.
[31] Glover, F., A. Løkketangen and D. Woodruff (1999). "Scatter Search to Generate Diverse MIP Solutions," Research Report, University of Colorado, Boulder.
[32] Glover, F. and C. McMillan (1986). "The General Employee Scheduling Problem: An Integration of MS and AI," *Computers and Operations Research*, Vol 13, No. 5, 563-573.
[33] Greistorfer, P. (1999). "Tabu and Scatter Search Combined for Arc Routing," presented at MIC'99 - III Metaheuristics International Conference, Brazil, July 19-22.
[34] Greenberg, H. J. and W. P. Pierskalla (1970). "Surrogate Mathematical Programs," *Operations Research*, 18, 924-939.
[35] Greenberg, H. J. and W. P. Pierskalla (1973). "Quasi-conjugate Functions and Surrogate Duality," *Cahiers du Center d'Etudes de Recherche Operationelle*, 15, 437-448.
[36] Gu, J. and X. Huang (1994). "Efficient Local Search With Search Space Smoothing: A Case Study of the Traveling Salesman Problem," *IEEE Transactions on Systems, Man, and Cybernetics*, 24(5) 728-735.
[37] Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor,

MI.
[38] Jain A.S. and S. Meeran (1998a). "An Improved Search Template for Job-Shop Scheduling," INFORMS Spring Meeting, Montreal, Quebec, Canada, April 26-29, Department of Applied Physics, Electronic and Mechanical Engineering, University of Dundee, Dundee, Scotland Department of Applied Physics, Electronic and Mechanical Engineering, University of Dundee, Dundee, Scotland.
[39] Jain A.S. and S. Meeran (1998b). "A Multi-Level Hybrid Framework for the General Flow-Shop Scheduling Problem," Department of Applied Physics, Electronic and Mechanical Engineering, University of Dundee, Dundee, Scotland.
[40] Karwan, M.H. and R.L. Rardin (1976). "Surrogate Dual Multiplier Search Procedures in Integer Programming," School of Industrial Systems Engineering, Report Series No. J-77-13, Georgia Institute of Technology.
[41] Karwan, M.H. and R.L. Rardin (1979). "Some Relationships Between Lagrangean and Surrogate Duality in Integer Programming," *Mathematical Programming*, 17, 230-334.
[42] Kelly, J., B. Rangaswamy and J. Xu (1996). "A Scatter Search-Based Learning Algorithm for Neural Network Training," *Journal of Heuristics*, Vol. 2, pp. 129-146.
[43] Laguna, M., H. Lourenço and R. Martí (2000). "Assigning Proctors to Exams with Scatter Search," in *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, M. Laguna and J. L. González-Velarde (Eds.), Kluwer Academic Publishers, pp. 215-227.
[44] 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.
[45] Laguna, M. and R. Martí (2000a). "Experimental Testing of Advanced Scatter Search Designs for Global Optimization of Multimodal Functions," Working paper, University of Colorado.
[46] Laguna, M. and R. Martí (2000b). "The OptQuest Callable Library," Working paper, University of Colorado.
[47] 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.
[48] Løkketangen, A. and F. Glover (1997). "Surrogate Constraint Analysis – New Heuristics and Learning Schemes for Satisfiability Problems," DIMACS Series in *Discrete Mathematics and Theoretical Computer Science*, Vol. 35, 537-572.
[49] Lourenço, H., J. Paixao and R. Portugal, (1998). "Metaheuristics for the bus-driver scheduling problem," Economic Working Papers Series, no. 304, Universitat Pompeu Fabra, Spain.
[50] Moscato, P. (1989), "On Evolution, Search, Optimization Algorithms and Martial Arts: Towards Memetic Algorithms," Report 826, Caltech Concurrent Computation Program, California Institute of Technology, Pasadena.
[51] Radcliffe, N., and P.D. Surrey (1994), "Formal Memetic Algorithms", *AISB94*.
[52] Rana, S. and D. Whitley (1997). "Bit Representations with a Twist," Proc. 7th International Conference on Genetic Algorithms, T. Baeck ed., pp. 188-196, Morgan Kaufman.
[53] Reeves, C.R. (1997). "Genetic Algorithms for the Operations Researcher," *Journal on Computing*, Vol. 9, No. 3, 231-250 (with commentaries and rejoinder).
[54] Rego, C. (1999). "Integrating Advanced Principles of Tabu Search for the Vehicle Routing Problem," Working Paper, Faculty of Sciences, University of Lisbon.
[55] Rochat, Y. and É. D. Taillard (1995). "Probabilistic diversification and intensification in local search for vehicle routing,". *Journal of Heuristics* 1, 147-167.
[56] Schneider, J., M. Dankersreiter, W. Fettes, I. Morgenstern, M. Schmid, and J. Singer (1997). "Search-space Smoothing for Combinatorial Optimization Problems," *Physica A Statistical and Theoretical Physics*, 243(1-2) 77-112.
[57] Taillard, É. D. (1996). "A heuristic column generation method for the heterogeneous VRP," Publication CRT-96-03, Center de recherche sur les transports, Université de Montréal. To appear in *RAIRO-OR*.
[58] Ulder, N.L.J., E. Pesch, P.J.M. van Laarhoven, H.J. Bandelt and E.H.L. Aarts (1991). "Genetic local search algorithm for the traveling salesman problem," *Parallel Problem solving from Nature* (Edited by R. Maenner and H.P. Schwefel), Lectures in Computer Science, Vol. 496, pp. 109-116. Springer, Berlin.
[59] Whitley, D. (1989). "The GENITOR Algorithm and Selective Pressure: Why Rank Based Allocation of Reproductive Trials is Best," Morgan Kaufmann, J. D. Schaffer, ed., pp. 116-121.
[60] Whitley, D. and J. Kauth, (1988). "GENITOR: A Different Genetic Algorithm," Proceedings of the 1988 Rocky Mountain Conference on Artificial Intelligence.
[61] Xu, J., S. Chiu and F. Glover (2000) "Tabu Search and Evolutionary Scatter Search for 'Tree-Star' Network

Problems, with Applications to Leased-Line Network Design," to appear in *Telecommunications Optimization*, David Corne (Ed.) Wiley.

[62] Yamada, T. and R. Nakano (1996). "Scheduling by Genetic Local Search with Multi-Step Crossover," 4th International Conference on Parallel Problem Solving from Nature, 960-969.

[63] Yamada, T. and C. Reeves (1997). "Permutation Flowshop Scheduling by Genetic Local Search," 2nd IEE/IEEE Int. Conf. on Genetic Algorithms in Engineering Systems (GALESIA '97), pp. 232-238, Glasglow, UK.