Scatter Search

Manuel Laguna

Graduate School of Business and Administration
Campus Box 419, University of Colorado, Boulder, CO 80309-0419, USA.
Manuel.Laguna@Colorado.edu

Latest Revision: August 16, 1999

Abstract — This article explores the meta-heuristic approach called scatter search, which is an evolutionary method that has recently been shown to yield promising outcomes for solving combinatorial and nonlinear optimization problems. Based on formulations originally proposed in the 1960s for combining decision rules and problem constraints, this method uses strategies for combining solution vectors that have proved effective in a variety of problem areas. Scatter search can be implemented in multiple ways, and offers numerous alternatives for exploiting its fundamental ideas. We identify a general design and illustrate the main features in the context of a classical integer programming problem.

1. Scatter Search Background

Scatter search, from the standpoint of meta-heuristic classification, may be viewed as an evolutionary (or also called population-based) algorithm that constructs solutions by combining others. It derives its foundations from strategies originally proposed for combining decision rules and constraints (in the context of integer programming). The goal of this methodology is to enable the implementation of solution procedures that can derive new solutions from combined elements. The way scatter search combines solutions and updates the set of reference solutions used for combination sets this methodology apart from other population-based approaches.

As discussed in Glover (1998 and 1999), the approach of combining existing solutions or rules to create new solutions originated in the 1960s. In the area of scheduling, researchers introduced the notion of combining rules to obtain improved local decisions. Numerically weighted combinations of existing rules, suitably restructured so that their evaluations embodied a common metric, generated new rules. The conjecture that information about the relative desirability of alternative choices is captured in different forms by different rules motivated this approach. The combination strategy was devised with the belief that this information could be exploited more effectively when integrated than when treated in isolation (i.e., when existing selection rules are selected one at a time). In general, the decision rules created from such combination strategies produced better empirical outcomes than standard applications of local decision rules. They also proved superior to a "probabilistic learning approach" that used stochastic selection of rules at different junctures, but without the integration effect provided by generating combined rules.

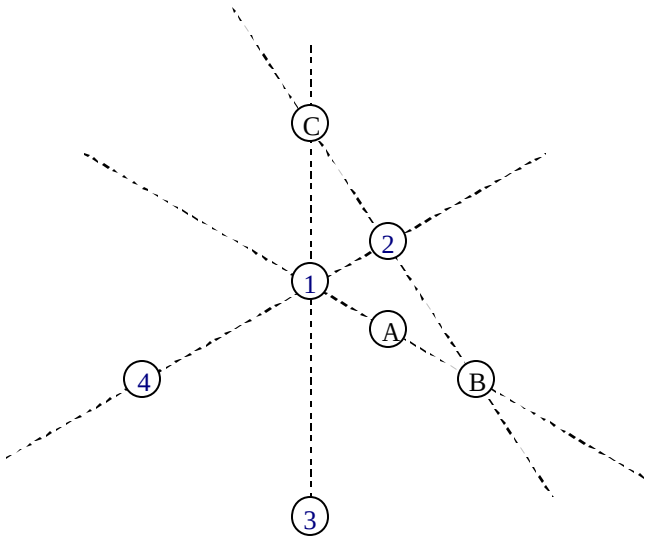
In integer and nonlinear programming, associated procedures for combining constraints were developed, which likewise employed a mechanism for creating weighted combinations. In this case, nonnegative weights were introduced to create new constraint inequalities, called *surrogate constraints*. The approach isolated subsets of constraints that were gauged to be most critical, relative to trial solutions based on the surrogate constraints. This critical subset was used to produce new weights that reflected the degree to which the component constraints were satisfied or violated.

The main function of surrogate constraints was to provide ways to evaluate choices that could be used to create and modify trial solutions. A variety of heuristic processes that employed surrogate constraints and their evaluations evolved from this foundation. As a natural extension, these processes led to the related strategy of combining solutions. Combining solutions, as manifested in scatter search, can be interpreted as the primal counterpart to the dual strategy of combining constraints.

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, only solution 1 is introduced in 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 complete reference set shown in Figure 1 consists of 7 solutions (or elements).

Figure 1. Two-dimensional reference set.



Unlike a "population" in genetic algorithms, the reference set of solutions in scatter search tends to be 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 elements of the reference set in a systematic way with the purpose of creating new solutions. Since the combination process considers at least all pairs of solutions in the reference set, 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 general, if the reference set consists of b solutions, the procedure examines approximately $(3b\text{-}7)b/2$ combinations of four different types (Glover 1998). The basic type consists of combining two solutions; the next type combines three solutions, and so on and so forth. 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.

2. Scatter Search Template

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, (2) take advantage of auxiliary heuristic solution methods to evaluate the combinations produced and to generate new vectors (Glover, 1998). Specifically, the scatter search approach may be sketched 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 considered 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 such solution is inferior to other solutions competing for admission in 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. Note that 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. (Note that 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 (Glover 1998):

- 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.

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.

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.

In the next section, we employ this template to illustrate the design of a simple scatter search procedure for an instance of the well-known 0-1 knapsack problem.

3. A Scatter Search Illustration

Consider the following 0-1 knapsack problem, where the coefficients in the objective function are considered profit values and the coefficients in the constraint are considered weights:

Maximize $11x_1 + 10x_2 + 9x_3 + 12x_4 + 10x_5 + 6x_6 + 7x_7 + 5x_8 + 3x_9 + 8x_{10}$

Subject to $33x_1 + 27x_2 + 16x_3 + 14x_4 + 29x_5 + 30x_6 + 31x_7 + 33x_8 + 14x_9 + 18x_{10} \leq 100$
$x_i = \{0, 1\}$ for $i = 1, \ldots, 10$.

For the purpose of this illustration, we first describe each of the methods that are needed in the overall procedure. We then discuss the iterative nature of the resulting solution method.

Diversification Generation Method

An appropriate diversification generator for this problem is described in Glover (1998). This generator operates as follows. Choose a value for the parameter $h \leq n$-1, where n is the number of variables in the problem. Let us choose h = 5. We also choose the initial seed to be x = (0,0,0, ,0). For each value of h, two solutions are generated. Type 1 solutions are denoted by x' and are initialized as $x'_i = 0$ for all i. Then some of the individual elements are modified as follows:

$$x'_1 = 1 - x_1$$
$$x'_{1+hk} = 1 - x_{1+hk} \text{ for } k = 1, \ldots, n/h.$$

Type 2 solutions are denoted by x'' and obtained as the complement of Type 1 solutions, i.e., $x''_i = 1 - x'_i$. Table 1 shows the 10 solutions generated with this method using the parameters given above.

Table 1. Diversified solutions.

h	x'	x''
1	(1,1,1,1,1,1,1,1,1,1)	(0,0,0,0,0,0,0,0,0,0)
2	(1,0,1,0,1,0,1,0,1,0)	(0,1,0,1,0,1,0,1,0,1)
3	(1,0,0,1,0,0,1,0,0,1)	(0,1,1,0,1,1,0,1,1,0)
4	(1,0,0,0,1,0,0,0,1,0)	(0,1,1,1,0,1,1,1,0,1)
5	(1,0,0,0,0,1,0,0,0,0)	(0,1,1,1,1,0,1,1,1,1)

The 10 solutions in Table 1 can now be used to create the reference set by applying the following Improvement Method.

Improvement Method

Since the solutions generated with the Diversification Generation Method are not guaranteed to be feasible, the Improvement Method should be capable of handling starting solutions that are either feasible or infeasible. To illustrate, we consider a simple procedure that operates as follows:

a) If the trial solution is *infeasible*, then the method attempts to improve it by first finding a feasible solution. A feasible solution is found by changing variable values from one to zero until the constraint is no longer violated. The variables are considered in increasing order of their profit-to-weight ratio, starting with the one with the smallest ratio. Once the solution becomes feasible, the following procedure is applied.

b) If the trial solution is *feasible*, then the method attempts to improve it by changing variable values from zero to one. This is done in a greedy fashion, so that the first variable to be considered is the one with the largest profit-to-weight ratio. (For example, x_4 has the largest ratio value of 12/14 = 0.857 in our problem instance.) The procedure stops when no more variables can be given a value of one without violating the constraint.

Table 2 shows the solutions obtained by the application of the Improvement Method to the trial solutions generated by the Diversification Generation Method.

Table 2. Initial and improved solutions.

Solution	*Trial Solution*	*Objective Value*	*Improved Solution*	*Objective Value*
1	(1,1,1,1,1,1,1,1,1,1)	81†	(0,1,1,1,0,0,0,0,1,1)	42
2	(1,0,1,0,1,0,1,0,1,0)	40†	(1,0,1,1,1,0,0,0,0,0)	42
3	(1,0,0,1,0,0,1,0,0,1)	38	(1,0,0,1,0,0,1,0,0,1)	38
4	(1,0,0,0,1,0,0,0,1,0)	24	(1,0,0,1,1,0,0,0,1,0)	36
5	(1,0,0,0,0,1,0,0,0,0)	17	(1,0,1,1,0,1,0,0,0,0)	38
6	(0,0,0,0,0,0,0,0,0,0)	0	(0,1,1,1,0,0,0,0,0,1)	39
7	(0,1,0,1,0,1,0,1,0,1)	41†	(0,1,0,1,0,1,0,0,0,1)	36
8	(0,1,1,0,1,1,0,1,1,0)	43†	(0,1,1,1,1,0,0,0,1,0)	44*
9	(0,1,1,1,0,1,1,1,0,1)	57†	(0,1,1,1,0,0,0,0,1,1)	42
10	(0,1,1,1,1,0,1,1,1,1)	64†	(0,1,1,1,0,0,0,0,1,1)	42

† denotes an infeasible solution.

* denotes the optimal solution.

Note that the trial solutions 1, 9 and 10 in Table 2 became the same improved solution with an objective value of 42. In these cases, which are not atypical, the Diversification Generation Method can be applied until a desired number of different improved solutions are found. Also note that the Improvement Method was able to find the optimal solution starting from the infeasible trial solution 8. Table 3 shows the moves that the Improvement Method performed to obtain the optimal solution starting from the corresponding infeasible trial solution.

Table 3. Iterations of the Improvement Method.

Iteration	*Current Solution*	*Objective value*	*Total Weight*	*Candidate Moves* (profit/weight)	*Selected Move*
1	(0,1,1,0,1,1,0,1,1,0)	43	149	$x_2 = 0$ (0.370) $x_3 = 0$ (0.563) $x_5 = 0$ (0.345) $x_6 = 0$ (0.200) $x_8 = 0$ (0.152) $x_9 = 0$ (0.214)	$x_8 = 0$
2	(0,1,1,0,1,1,0,0,1,0)	38	116	$x_2 = 0$ (0.370) $x_3 = 0$ (0.563) $x_5 = 0$ (0.345) $x_6 = 0$ (0.200) $x_9 = 0$ (0.214)	$x_6 = 0$
3	(0,1,1,0,1,0,0,0,1,0)	32	86	$x_4 = 1$ (0.857)	$x_4 = 1$
4	(0,1,1,1,1,0,0,0,1,0)	44	100	None	

The iterations in Table 3 show that while the knapsack constraint is violated (i.e., the total weight of the current solution is larger than 100), the candidate moves consist of changing variable values from one to zero. In the first iteration, x_8 is selected because its profit-to-weight ratio is the minimum in the candidate list. After x_6 is also set to zero in iteration 2 using the same criteria as before, the current solution becomes feasible in iteration 3. The procedure

now looks for opportunities to improve the objective function value. The candidates are those variables that are currently set to zero (excluding x_6 and x_8) and whose weight is less than or equal to the current slack (i.e., 100-86 = 14). The only variable that meets such criteria is x_4, which also happens to have the largest profit-to-weight ratio of 0.857. When the value of x_4 is changed to 1, the optimal solution is found. No more moves are available in iteration 4 and the improvement process stops.

Reference Set Update Method

This method is used to create and maintain a set of reference solutions. We divide this set of b solutions into two subsets: the subset of b_1 high-quality solutions and the subset of b_2 diverse solutions. Let us consider a reference set of size b = 5, where b_1 = 3 and b_2 = 2. According to these definitions, the subset of high-quality solutions would consist of the improved solutions 1, 2 and 8.

In order to find a diverse set to complement the subset of high-quality solutions, it is necessary to define a diversity measure. We define the distance between two solutions as the sum of the absolute difference between its corresponding variable values. For example, the distance between the improved solution 1 and the improved solution 8 is calculated as follows:

```
(0,1,1,1,0,0,0,0,1,1)
(0,1,1,1,1,0,0,0,1,0)
_____________________
(0+0+0+0+1+0+0+0+0+1)= 2
```

We then use this distance measure to select the two solutions for the subset of diverse solution. In particular, we look for the solution that is not currently in the reference set and that maximizes the minimum distance to all the solutions currently in the reference set. Table 4 shows the distance values from each candidate solution to each solution currently in the reference set.

Table 4. Distance measures.

	Distance to solution			*Minimum*
Candidate Solution	*1*	*2*	*8*	*distance*
3 (1,0,0,1,0,0,1,0,0,1)	5	4	7	4
4 (1,0,0,1,1,0,0,0,1,0)	5	2	3	2
5 (1,0,1,1,0,1,0,0,0,0)	5	2	5	2
6 (0,1,1,1,0,0,0,0,0,1)	1	4	3	1
7 (0,1,0,1,0,1,0,0,0,1)	3	6	5	3

The list of candidate solutions in Table 4 includes neither solution 9 nor solution 10 because these solutions are the same as solution 1, which is already in the reference set. The distance values show that in order to maximize the minimum distance, improved solution 3 should be added to the reference set. The final member of the reference set becomes solution 7, because solutions 4, 5, 6 and 7 have the same distance (equal to 4) to solution 3. So, solution 7 achieves the maximum minimum distance from the set of candidate solutions.

It is important to note that the reference set does not consist of all the “best” solutions as measured by the objective function value only. Because we are interested in a balance between high quality solutions and diverse solutions, improved solution 6 with an objective value of 39

is by-passed in favor to other solutions that will add more diversity to the set. As indicated in Table 4, the distance between solution 1 and solution 6 is only one unit, making this solution unattractive from the standpoint of diversification.

Subset Generation Method

This method consists of generating the subsets that will be used for creating structured combinations in the next step. The method is typically designed to generate the following types of subsets:

1) All 2-element subsets.
2) 3-element subsets derived from the 2-element subsets by augmenting each 2-element subset to include the best solution (as measured by the objective value) not in this subset.
3) 4-element subsets derived from the 3-element subsets by augmenting each 3-element subset to include the best solution (as measured by the objective value) not in this subset.
4) The subsets consisting of the best i elements (as measured by the objective value), for i = 5 to b.

Table 5 shows the subsets generated using our current reference set.

Table 5. Subset generation.

Type	*Subsets*
1	(1,2) (1,3) (1,7) (1,8) (2,3) (2,7) (2,8) (3,7) (3,8) (7,8)
2	(1,2,8) (1,3,8) (1,7,8) (2,3,8) (2,7,8) (3,7,8)
3	(1,2,3,8) (1,2,7,8) (1,3,7,8)
4	(1,2,3,7,8)

As expected, there are 10 possible subsets of type 1 in Table 5. Type 2 subsets are formed by augmenting the 2-element subsets. The first 2-element subset is formed by solutions 1 and 2, and we add solution 8 to form the 3-element subset (1,2,8). This is done starting from each of the 2-element subsets. After eliminating repetitions, there are 6 subsets of type 2. Subsets of type 3 are formed in a similar way, starting this time from the 3-element subsets.

Solution Combination Method

This method uses the subsets generated in the previous step to combine the elements in each subset with the purpose of creating new trial solutions. The Combination Method is typically problem-specific since is directly related to the solution representation. Depending on the specific form of the Solution Combination Method, each subset can create one or more new solutions. For illustration purposes, we use a combination method that creates only one solution from each subset. Specifically, our Solution Combination Method calculates a score for each variable, based on the solutions in the subset and their corresponding objective

values. The score for variable i that corresponds to the solutions in subset S is calculated with the following formula:

$$score(i) = \frac{\sum_{j \in S} OV(j) * x_i^j}{\sum_{j \in S} OV(j)}.$$

Where $OV(j)$ is the objective value of solution j and x_i^j is the value of the ith variable in solution j. Then, the trial solution is constructed by rounding the score for each variable to the nearest integer, i.e.,

$$x_i' = \begin{cases} 1 & \text{if } score(i) > 0.5 \\ 0 & \text{if } score(i) \leq 0.5 \end{cases}.$$

Note that the combination mechanism constructs solutions that may be infeasible. This does not represent a problem in general, since the Improvement Method is always applied to each trial solution created after the application of the Solution Combination Method. Recall that the Improvement Method must be designed to deal with either feasible or infeasible starting solutions.

To illustrate the use of the Solution Combination Method, consider the subset type 2 given by solutions 3, 7 and 8. The objective value of solution 3 is 38, the objective value of solution 7 is 36 and the objective value of solution 8 is 44, i.e., $OV(3) = 38$, $OV(7) = 36$, and $OV(8) = 44$. The score for each variable can be calculated as shown in Table 6.

Table 6. Score calculation for subset combination (3,7,8).

Solution	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
3	0.322	0.000	0.000	0.322	0.000	0.000	0.322	0.000	0.000	0.322
7	0.000	0.305	0.000	0.305	0.000	0.305	0.000	0.000	0.000	0.305
8	0.000	0.373	0.373	0.373	0.373	0.000	0.000	0.000	0.373	0.000
Total	0.322	0.678	0.373	1.000	0.373	0.305	0.322	0.000	0.373	0.627

After rounding the scores in Table 6, the new solution becomes `(0,1,0,1,0,0,0,0,0,1)` with an objective value of 30 and a total weight of 59. This feasible solution can be improved with the Improvement Method. In this case, the value of x_5 is changed from zero to one, because x_5 has the highest profit-to-weight ratio of the variables that are currently set to zero. The improved solution has an objective value of 40 and a total weight of 88. Since no more variable values can be switched from zero to one without violating the constraint, the improvement process stops.

The new solutions constructed in this step of the procedure are considered for membership in the reference set. A solution may become a member if its objective value is better than the objective value of any of the solutions in the high-quality subset. Alternatively, if a new solution improves the diversity of the reference set, this solution can replace one that is currently in the diverse set. If the reference set is modified, then the Subset Generation

Method, the Solution Combination Method and the Improvement Method are applied in sequence. The application of the methods continues until the reference set converges (i.e., no elements in the set are replaced). At this point, the Diversification Method can be applied again from a different seed and the search continues until termination criteria are satisfied.

4. A Summary of the Procedure

In the previous section, we illustrated the operations that take place within each of the methods that shape a scatter search implementation. We now finish our introduction to scatter search with an overall view of the procedure. This outline (or pseudo-code) uses the following parameters:

PSize =	the size of the set of diverse solutions generated by the Diversification Generation Method
b =	the size of the reference set.
b_1 =	the size of the high-quality subset.
b_2 =	the size of the diverse subset.
MaxIter =	maximum number of iterations.

The procedure consists of the steps in the outline of Figure 2, where *P* denotes the set of solutions generated with the Diversification Generation Method and *RefSet* is the set of solution in the reference set.

The procedure starts with the generation of *PSize* distinct solutions. These solutions are originally generated to be diverse and subsequently improved by the application of the Improvement Method (step 1). The set *P* of *PSize* solutions is ordered in step 2, in order to facilitate the task of creating the reference set in step 3. The reference set (*RefSet*) is constructed with the first b_1 solutions in *P* and b_2 solutions that are diverse with respect to the members in *RefSet*.

The search consists of three main loops: 1) a “for-loop” that controls the maximum number of iterations, 2) a “while-loop” that monitors the presence of new elements in the reference set, and 3) a “for-loop” that controls the examination of all the subsets with at least one new element. In step 4, the number of subsets with at least one new element is counted and this value is assigned to *MaxSubset*. Also, the Boolean variable *NewElements* is made FALSE before the subsets are examined, since it is not known whether a new solution will enter the reference set in the current examination of the subsets. The actual generation of the subsets occurs in step 5. Note that only subsets with at least one new element are generated in this step. A solution is generated in step 6 by applying the Combination Method. Step 7 attempts to improve this solution with the application of the Improvement Method. If the improved solution from step 7 is better (in terms of the objective function value) than the worst solution in $RefSet_1$, then the improved solution becomes a new element of *RefSet*. Note that $RefSet_1$ is the subset of the reference set that contains the best solutions as measured by the objective function value. The solution is added in step 8 and the *NewElements* indicator is switched to TRUE in step 9.

If a solution is not admitted to the *RefSet* due to its quality, the solution is tested for its diversity merits. If a solution adds diversity to RefSet2, then the solution is added to the reference set and the less diverse solution is deleted (as indicated in steps 10 and 11). Finally, step 12 is performed if additional iterations are still available. This step provides a seed for set

P by adding the solutions in $RefSet_1$ before a new application of the Diversification Generation Method.

Note that the general procedure outlined in Figure 2 can be modified in different ways. One possibility is to eliminate the outer "for-loop" along with step 12. In this case, set *P* is generated only one time and the process stops after no new elements are admitted in the reference set. That is the search is abandoned when the "while-loop" that contains steps 4 to 11 becomes false. This variation is useful for problems in which the search has to be performed within a relatively small computer time. Also, there are some settings in which a large percentage of the best solutions are found during the first iteration (i.e., when *Iter* = 1). In this case, bypassing additional iterations has a small effect on the average performance.

A second variation consists of eliminating steps 10 and 11 and the if-statement associated with these steps. This variation considers that the reference set will be initially constructed with both high-quality solutions and diverse solutions. However after step 3, membership to the reference set is obtained only due to quality. Therefore, in addition to eliminating steps 10 and 11, step 8 and its associated if-statement are modified as follows:

If (x_s^* is not in *RefSet* and the objective function value of x_s^* is better than the objective function value of the worst element in *RefSet*) **then**

8. Add x_s^* to *RefSet* and delete the worst element currently in *RefSet*. (The worst element is the solution with worst objective value.)

That is, all references to $RefSet_1$ are substituted with references to *RefSet*. In other words, after step 3, the elements of *RefSet* are always ordered according to the objective function value, making element 1 the best and element *b* the worst.

Implementing both of these variations at the same time results in a very aggressive search method that attempts to find high quality solutions fast. While this may be desirable in some settings, there are also settings in which a more extensive search can be afforded, and therefore making the full procedure outlined in Figure 2 more attractive.

Figure 2. Scatter Search Outline.

1. Start with $P = \emptyset$. Use the Diversification Generation Method to construct a solution x. Apply the Improvement Method to x to obtain the improved solution x^*. 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. Order the solutions in P according to their objective function value (where the best overall solution is first on the list).

For (*Iter* = 1 **to** *MaxIter*)

3. Build $RefSet = RefSet_1 \cup RefSet_2$ from P, with $|RefSet| = b$, $|RefSet_1| = b_1$ and $|RefSet_2| = b_2$. Take the first b_1 solutions in P and add them to $RefSet_1$. For each solution x in *P-RefSet* and y in *RefSet*, calculate a measure of distance or dissimilarity $d(x,y)$. Select the solution x' that maximizes $d_{\min}(x)$, where $d_{\min}(x) = \min_{y \in RefSet} \{d(x,y)\}$. Add x' to $RefSet_2$, until $|RefSet_2| = b_2$. Make *NewElements* = TRUE.

While (*NewElements*) **do**

4. Calculate the number of subsets (*MaxSubset*) that include at least one new element. Make *NewElements* = FALSE.

For (*SubsetCounter* = 1, ..., *MaxSubset*) **do**

5. Generate the next subset s from *RefSet* with the Subset Generation Method. This method generates one of four types of subsets with number of elements ranging from 2 to $|RefSet|$. Let subset $s = \{ s_1, ..., s_k \}$, for $2 \le k \le |RefSet|$. (We consider that the Subset Generation Method skips subsets for which the elements considered have not changed from previous iterations.)
6. Apply the Solution Combination Method to s to obtain one or more new solutions x_s.
7. Apply the Improvement Method to x_s, to obtain the improved solution x_s^*.

If (x_s^* is not in *RefSet* and the objective function value of x_s^* is better than the objective function value of the worst element in $RefSet_1$) **then**

8. Add x_s^* to $RefSet_1$ and delete the worst element currently in $RefSet_1$. (The worst element is the solution with worst objective value.)
9. Make *NewElements* = TRUE.

Else

If (x_s^* is not in $RefSet_2$ and $d_{\min}(x_s^*)$ is larger than $d_{\min}(x)$ for a solution x in $RefSet_2$) **then**

10. Add x_s^* to $RefSet_2$ and delete the worst element currently in $RefSet_2$. (The worst element is the solution x with the smallest $d_{\min}(x)$ value.)
11. Make *NewElements* = TRUE.

End if

End if

End for

End while

If (*Iter* < *MaxIter*) **then**

12. Build a new set P using the Diversification Generation Method. Initialize the generation process with the solutions currently in $RefSet_1$. That is, the first b_1 solutions in the new P are the best b_1 solutions in the current *RefSet*.

End if

End for

Bibliography

Campos, V., M. Laguna and R. Martí (1999) "Scatter Search for the Linear Ordering Problem," to appear in *New Methods in Optimisation*, D. Corne, M. Dorigo and F. Glover (Eds.), McGraw-Hill.

Glover, F. (1977) "Heuristics for Integer Programming Using Surrogate Constraints," *Decision Sciences*, Vol. 8, No. 1, 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. (1999) "Scatter Search and Path Relinking," to appear in *New Methods in Optimisation*, D. Corne, M. Dorigo and F. Glover (Eds.), McGraw-Hill.