The Scatter Search Methodology

MANUEL LAGUNA
Leeds School of Business, University of Colorado at Boulder, USA
laguna@colorado.edu

RAFAEL MARTÍ
Departamento de Estadística e Investigación Operativa, Universidad de Valencia, Spain
Rafael.Marti@uv.es

MICAEL GALLEGO
Departamento de Ciencias de la Computación, Universidad Rey Juan Carlos, Spain.
Micael.Gallego@urjc.es

ABRAHAM DUARTE
Departamento de Ciencias de la Computación, Universidad Rey Juan Carlos, Spain.
Abraham.Duarte@urjc.es

ABSTRACT

Scatter search (SS) is an evolutionary approach for optimization. It has been applied to problems with continuous and discrete variables and with a single or multiple objectives. The success of scatter search as an optimization technique is well documented in a constantly growing number of journal articles and book chapters. This chapter first focuses on the basic scatter search framework, which is responsible for most of the outcomes reported in the literature, and then covers advanced elements that have been introduced in a few selected papers, such as the hybridization with tabu search, a well-known memory-based metaheuristic. We consider the maximum diversity problem to illustrate the search elements, methods and strategies described here.

1. Introduction

Scatter Search (SS) is an evolutionary method that has been successfully applied to hard optimization problems. SS was first introduced by Prof. Fred Glover as a heuristic for integer programming and it was based on strategies presented at a management science and engineering management conference held in Austin, Texas in September of 1967. Unlike genetic algorithms, it operates on a small set of solutions and employs diversification strategies of the form proposed in Tabu Search (Glover and Laguna, 1997), which give precedence to strategic learning based on adaptive memory, with limited recourse to randomization.

The fundamental concepts and principles were first proposed in 1977 (Glover 1977) as an extension of formulations, dating back to the 1960s, for combining decision rules and problem constraints. (The constraint combination approaches, known as surrogate constraint methods, now independently provide an important class of relaxation strategies for global optimization.) The Scatter Search framework is flexible, allowing the development of alternative implementations with varying degrees of sophistication. In addition, SS can be combined with other methods to enhance their effectiveness. For example SS has been integrated with Particle Swarm Optimization and adaptive memory strategies of Tabu Search to produce a method called Cyber Swarm Optimization that has provided improvements over its particle swarm component (Yin et al., 2010).

As described in Glover (1998) and Laguna and Martí (2003), SS consists of five methods:

1. Diversification Generation
2. Improvement
3. Reference Set Update
4. Subset Generation
5. Solution Combination

The **diversification generation method** is used to generate a set of diverse solutions that are the basis for initializing the search. The most effective diversification methods are those capable of creating a set of solutions that balances diversification and quality. It has been shown that SS produces better results when the diversification generation method is not purely random and constructs solutions by reference to both a diversification measure and the objective function.

The **improvement method** transforms solutions with the goal of improving quality (typically measured by the objective function value) or feasibility (typically measured by some degree of constraint violation). The input to the improvement method is a single solution that may or may not be feasible. The output is a solution that may or may not be better (in terms of quality or feasibility) than the original solution. The typical improvement method is a local search with the usual rule of stopping as soon as no improvement is detected in the neighborhood of the current solution. There is the possibility of basing the improvement method on procedures that use a neighborhood search but that they are able to escape local optimality. Tabu search, simulated annealing and variable neighborhood search qualify as candidates for such a design. This may seem as an attractive option as a general approach for an improvement method, however, these procedures do not have a natural stopping criterion. The end result is that choices need to be made to control the amount of computer time that is spent improving solutions (by running a metaheuristic-based procedure) versus the time spent outside the improvement method (e.g., combining solutions). In general, local search procedures seem to

work well and most SS implementations do not include mechanisms to escape local optimality within the process of improving a solution.

The **reference set update** method refers to the process of building and maintaining a reference set of solutions that are used in the main iterative loop of any scatter search implementation. While there are several implementation options, this element of scatter search is fairly independent from the context of the problem. The first goal of the reference update method is to build the initial reference set of solutions from the population of solutions generated with the diversification method. Subsequent calls to the reference update method serve the purpose of maintaining the reference set. The typical design of this method builds the first reference set by blending high quality solutions and diverse solutions. While choosing diverse solution, reference needs to be made to a distance metric that typically depends on the solution representation. That is, if the problem context is such that continuous variables are used to represent solutions, then diversification may be measured with Euclidean distances. Other solution representations (e.g., binary variables or permutations) result in different ways of calculating distances and in turn diversification. The updating of the reference set during the scatter search iterations is customarily done on the basis of solution quality.

The **subset generation method** produces subsets of reference solutions which become the input to the combination method. The typical implementation of this method consists of generating all possible pairs of solutions. The scatter search framework considers also the generation of larger subsets of reference solutions; however, most SS implementations have been limited to operate on pairs of solutions. Clearly, no context information is needed to implement the subset generation method.

The **solution combination method** uses the output from the subset generation method to create new solutions. New trial solutions are the results of combining, typically two but possibly more, reference solutions. The combination of reference solutions is usually designed to exploit problem context information and solution representation. The strategic use of linear combinations of solutions in heuristic search has often been employed for discrete and nonlinear optimization problems, since such uses were first proposed in Glover (1977). Several proposals for combining solutions represented by permutations have also been applied (Martí, Laguna and Campos 2005). The strategy known as path relinking, originally proposed within the tabu search methodology (Glover and Laguna 1997), has also played a relevant role in designing combination methods for scatter search implementations.

As mentioned above, we will describe both the main scatter search elements and the advanced search strategies, and use the maximum diversity problem, MDP, (Gallego et al., 2009) to illustrate some implementation details. The MDP consists of selecting a subset of m elements from a set of n elements in such a way that the sum of the distances between the chosen elements is maximized. The definition of distance between elements is customized to specific applications. As mentioned in Kuo, Glover and Dhir (1993) and Glover, Kuo and Dhir (1998), the maximum diversity problem has applications in plant breeding, social problems, ecological preservation, pollution control, product design, capital investment, workforce management, curriculum design and genetic engineering. In most applications, it is assumed that each element can be represented by a set of attributes. Let s_{ik} be the state or value of the k^{th} attribute of element i, where $k = 1, \ldots, K$. Then the distance between elements i and j may be defined as:

$$d_{ij} = \sqrt{\sum_{k=1}^{K} \left(s_{ik} - s_{jk}\right)^2}$$

In this case, d_{ij} is simply the Euclidean distance between *i* and *j*. The distance values are then used to formulate the MDP as a quadratic binary problem, where variable x_i takes the value 1 if element *i* is selected and 0 otherwise, $i = 1, .., n$:

$$\text{Maximize} \qquad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j$$

$$\text{Subject to} \qquad \sum_{i=1}^{n} x_i = m$$

$$x_i = \{0, 1\} \qquad 1 \le i \le n$$

We finally would like to highlight the Scatter Search library provided in Laguna and Martí (2003). It consists of a C code that implements the search mechanisms that are discussed throughout their book. The C code can be used "as is" to replicate practical illustrations or can be modified and adapted to other optimization problems of interest. Several SS applications have been implemented from that source code.

2. Basic SS Design

Laguna (2009) summarizes the basic scatter search framework as follows (see Figure 1). The search starts with the application of the diversification and improvement methods (step 1 in Figure 1). The typical outcome consists of a set of about 100 solutions that is referred to as the population (denoted by P). In most implementations, the diversification generation method is applied first followed by the improvement method. If the application of the improvement method results in the shrinking of the population (due to more than one solution converging to the same local optimum) then the diversification method is applied again until the total number of improved solutions reaches the desired target. Other implementations construct and improve solutions, one by one, until reaching the desired population size.

```
1. Diversification generation and improvement methods
2. while (stopping criteria not satisfied) {
3.          Reference set update method
4.          while(new reference solutions) {
5.                      Subset generation method
6.                      Combination method
7.                      Improvement method
8.                      Reference set update method
9.          }
10.         Rebuild reference set
11. }
```

Fig. 1. Scatter search framework

The main scatter search loop is shown in lines 2 to 11 of Figure 1. The input to the first execution of the reference set update method (step 3) is the population of solutions generated in step 1 and the output is a set of solutions known as the reference set (or *RefSet*). Typically, 10 solutions are chosen from a population of 100. The first 5 solutions are chosen to be the best solutions (in terms of the objective function value) in the population. The other 5 are chosen to

be the most diverse with respect to the solutions in the reference set. If the diverse solutions are chosen sequentially, then the 6th solution is the most diverse with respect to the 5 best solutions that were chosen first. The 7th solution is the most diverse with respect to the first six and so on until the 10th one is added to the reference set.

The inner while-loop (lines 4 to 9) is executed as long as at least one reference solution is new in the *RefSet*. A solution is considered new if it has not been subjected to the subset generation (step 5) and combination (step 6) methods. If the reference set contains at least one new solution, the subset generation method builds a list of all the reference solution subsets that will become the input to the combination method. The subset generation method creates new subsets only. A subset is new if it contains at least one new reference solution. This avoids the application of the combination method to the same subset more than once, which is particularly wasteful when the combination method is completely deterministic. Combination methods that contain random elements may be able to produce new trial solutions even when applied more than once to the same subset of reference solutions. However, this is generally discouraged in favor of introducing new solutions in the reference set by replacing some of the old ones in the rebuilding step (line 10).

The combination method (step 6) is applied to the subsets of reference solutions generated in the previous step. Most combination methods are designed to produce more than one trial solution from the combination of the solutions in a subset. These trial solutions are given to the improvement method (step 7) and the output forms a pool of improved trial solutions that will be considered for admission in the reference set (step 8).

If no new solutions are added to the reference set after the execution of the reference set update method, then the process exits the inner while-loop. The rebuilding step in line 10 is optional. That is, it is possible to implement a scatter search procedure that terminates the first time that the reference set does not change. However, most implementations extend the search beyond this point by executing a *RefSet* rebuilding step. The rebuilding of the reference set entails the elimination of some current reference solutions and the addition of diverse solutions. In most implementations, all solutions except the best are replaced in this step. The diverse solutions to be added may be either population solutions that have not been used or new solutions constructed with the generation diversification method. Note that only 10 solutions out of 100 are used from the population to build the initial reference set and therefore the remaining 90 could be used for rebuilding purposes.

The process (i.e., the main while-loop in lines 2 to 11) continues as long as the stopping criteria are not satisfied. Possible stopping criteria include number of rebuilding steps or elapsed time. When scatter search is applied in the context of optimizing expensive black boxes, a limit on the number of calls to the objective function evaluator (i.e., the black box) may also be used as a criterion for stopping. We now expand our description and provide examples of each of the scatter search methods.

3. The Maximum Diversity Problem

Three of the five scatter search elements (the diversification-generation, the improvement and the combination methods) are **problem-dependent** and should be designed specifically for the problem at hand (although it is possible to design “generic” procedures, it is more effective to base the design on the specific idiosyncrasies of the problem setting). The other two, the reference-set-update and the subset-generation methods, are **problem-independent**, and usually follow a standard implementation. In this section we describe efficient implementations

of the three problem-dependent methods for the MDP. In the next section we cover the two problem-independent methods.

The definition of distance between solutions is a key design issue in scatter search implementations. Distance is used to measure how diverse one solution is with respect to a set of solutions. Specifically, for the MDP, let x_i^r be the value of the i^{th} variable for the reference solution r (i.e., $r \in$ *RefSet*). Also let x_i^t be the value of the i^{th} variable for the trial solution *t.* Then, the distance between the trial solution *t* and the solutions in the *RefSet* in our SS implementation is defined as:

$$distance(t, RefSet) = bm - \sum_{r=1}^{b} \sum_{i:x_i^t=1} x_i^r \ .$$

The formula simply counts the number of times that each selected element in the trial solution *t* appears in the reference solutions and subtracts this value from the maximum possible distance (i.e., *bm*). The maximum distance occurs when no element that is selected in the trial solution *t* appears in any of the reference solutions. When choosing solutions to rebuild the reference set, we select the trial solution that has the maximum distance between itself and the solutions currently in the *RefSet*. Since the solutions are added one at a time, the distance calculations have to be updated before the next solution is selected.

3.1 Diversification Generation Method

Duarte and Martí (2007) proposed a GRASP Hybrid (Resende and Ribeiro 2001) for generating MDP diverse solutions. It is based on randomizing *D*-2, a deterministic destructive heuristic developed by Glover et al. (1998). *D*-2 starts with the infeasible solution for which $x_i = 1$ for all *i*. That is, all *n* elements are originally selected. In order to reduce the set of selected elements to *m*, the procedure performs *n*-*m* steps. At each step, the procedure deselects element i^* (i.e., x_{i*} is set to zero), where i^* is such that:

$$D(i^*) = \underset{i:x_i=1}{Min}(D(i)),$$
$$\text{where } D(i) = \sum_j d_{ij} x_i x_j \ .$$

The randomization of *D*-2 that is employed within *RD*-2 consists of selecting i^* from a reduced candidate list formed by all those elements *i* such that $D(i) \le (1+\alpha)D(i^*)$. The value of α is initially set to 0.5 and decreased by 0.1 —to a minimum of 0.1— after a pre-specified CPU time is consumed without improving the incumbent. We set this value to a 20% of the total CPU time in our implementation.

3.2 Improvement Method

The Local Search method LS (Ghosh 1996) scans the set of selected elements in search of the best exchange to replace a selected element with an unselected one. The method performs moves as long as the objective value increases and it stops when no improving exchange can be found. The Improved Local Search method, I_LS, (Duarte and Martí 2007) selects the element i^* ($x_{i*} = 1$) that provides the smallest contribution to the objective function value of the current solution. Then, it searches for an element *j* ($x_j = 0$) to be exchange with element i^*. The first

element *j* that results in an improving move is selected and the exchange is performed without examining the remaining unselected elements. If no improving move can be found to exchange element i^*, then the selected element with the next smallest contribution is examined. This process continues until no improving exchange can be found.

3.3 Combination Method

This method consists of the application of the destructive heuristic *D*-2 (Glover, Kuo and Dhir, 1998) to the union of the elements in the reference solutions being combined. The method starts with the selection of all elements in the union and then it deselects one element at a time until there are only *m* selected elements remaining. The element *i* that is deselected at each step is the one with the minimum *D*(*i*) value.

4. Problem-independent methods

We now discuss some generic (i.e., context–independent) implementations of the reference set update and the subset generation. While these designs are not customized for a particular problem setting, the solution representation (e.g., binary strings, permutations, real numbers, etc.) does play an important role in their development and implementation. The following two subsections have been taken from Laguna (2009).

4.1 Reference Set Update

The execution of the diversification generation and improvement methods in line 1 of Figure 1 results in a population *P* of solutions. The reference set update method is executed in two different parts of the scatter search procedure. The first time that the method is called, its goal is to produce the initial *RefSet,* consisting of a mix of *b* (typically 10) high quality and diverse solutions drawn from *P*. The mix of high quality and diverse solutions could be considered a tunable parameter; however, most implementations populate the initial reference set with half of the solutions chosen by quality and half chosen by diversity.

Choosing solutions by quality is straightforward. From the population *P*, the best (according to the objective function value) *b*/2 solutions are chosen. These solutions are added to *RefSet* and deleted from *P*. To choose the remaining half, an appropriate measure of distance *d*(*r,p*) is needed, where *r* is a reference solution and *p* is a solution in *P*. The distance measure depends on the solution representation, but must satisfy the usual conditions for a metric, that is:

$d(r,p) = 0$	if and only if $r = p$
$d(r,p) = d(p,r) > 0$	if $r \neq p$
$d(r,q) + d(q,p) \geq d(r,p)$	triangle inequality

For instance, the Euclidean distance is commonly used when solutions are represented by continuous variables:

$$d(r,p) = \sqrt{\sum_{i=1}^{n} (r_i - p_i)^2}$$

Likewise, the Hamming (1950) distance is appropriate for two strings of equal length. The distance is given by the number of positions for which the corresponding symbols are different.

In other words, the distance is the number of changes required to transform one string into the other:

$$d(r,p) = \sum_{i=1}^{n} v_i \qquad v_i = \begin{cases} 0 & r_i = p_i \\ 1 & r_i \neq p_i \end{cases}$$

This distance has been typically used for problems whose solution representation is given by binary vectors (e.g., the max-cut and knapsack problems) and permutation vectors (e.g., the traveling salesman, quadratic assignment and linear ordering problems).

The distance measure is used to calculate the minimum distance $d_{\min}(p)$ of a population solution and all the reference solutions *r*:

$$d_{min}(p) = \min_{r \in RefSet} d(r,p)$$

Then, the next population solution *p* to be added to *RefSet* and deleted from *P* is the one with the maximum $d_{\min}$ value. That is, we want to choose the population solution p^* that has the maximum minimum distance between itself and all the solutions currently in *RefSet*:

$$p^* = \left\{ p : \max_{p \in P} d_{min}(p) \right\}$$

The process is repeated *b*/2 times to complete the construction of the initial *RefSet*. Note that after the first calculation of the $d_{\min}$ values, they can be updated as follows. Let p^* be the population solution most recently added to the *RefSet*. Then, the $d_{\min}$ value for a population solution *p* is given by:

$$d_{min}(p) = \min\left(d_{min}(p), d(p,p^*)\right)$$

Alternatively, the diverse solutions for the initial reference set may be chosen by solving the maximum diversity problem (MDP) introduced above. The special version of the MDP that must be solved includes a set of elements that have already been chosen (i.e., the high quality solutions). Mathematically, the problem may be formulated as follows:

$$\text{Maximize} \quad \sum_{(p,p') \in P : p \neq p'} d(p,p')\, x_p x_{p'}$$

$$\text{Subject to} \quad \sum_{p \in P} x_p = b$$

$$x_p = 1 \qquad \forall\, p \in RefSet$$

$$x_p = \{0,1\} \qquad \forall\, p \in P$$

The formulation assumes that a subset of high quality solutions in *P* have already been chosen and added to *RefSet*. The binary variables indicate whether a population solution *p* is chosen (x_p = 1) or not (x_p = 0). The second set of constraints in the formulation force the high-quality solutions to be included in the set of *b* solutions that will become the initial *RefSet*. This nonlinear programming model has been translated into an integer program for the purpose of solving it as well as for showing that the MDP is NP-hard. Martí, Duarte and Laguna (2009) embed the MDP in a scatter search procedure for the max-cut problem. Instead of solving the MDP exactly, they employ the GRASP_C_2 procedure developed by Duarte and Martí (2007). The

procedure was modified to account for the high-quality solutions that are chosen before the subset of diverse solutions is added to the *RefSet*. The procedure is executed for 100 iterations and the most diverse *RefSet* is chosen to initiate the scatter search.

The reference set update method is also called in step 8 of Figure 1. This step is performed after a set of one or more trial solutions has been generated by the sequential calls to the subset generation, combination and improvement methods (see steps 5, 6 and 7 in Figure 1). The most common update consists of the selection of the best (according to the objective function value) solutions from the union of the reference set and the set of trial solutions generated by steps 5-7 in Figure 1. Other updates have been suggested in order to preserve certain amount of diversity in the *RefSet*. These advanced updating mechanisms are beyond the scope of this tutorial chapter but the interested reader is referred to chapter 5 of Laguna and Martí (2003) for a detailed description and to Laguna and Martí (2005) for experimental results.

4.2 Subset Generation

This method is in charge of proving the input to the combination method. This input consists of a list of subsets of reference solutions. The most common subset generation consists of creating a list of all pairs (i.e., all 2-subsets) of reference solutions for which at least one of the solutions is new. A reference solution is new if it hasn't been used by the combination method. The first time that this method is called (step 5 in Figure 1), all the reference solutions are new, given that the method is operating on the initial *RefSet*. Therefore, the execution of the subset generation method results in the list of all 2-subsets of reference solutions, consisting of a total of (b^2-b)/2 pairs. Because the subset generation method is not based on a sample but rather on the universe of all possible pairs, the size of the *RefSet* in scatter search implementation must be moderate (e.g., less than 20). As mentioned in the introduction, a typical value for b is 10, resulting in 45 pairs the first time that the subset generation method is executed.

When the inner while-loop (steps 5-8 in Figure 1) is executing, the number of new reference solutions at the time that the subset generation method is called depends on the strategies implemented in the reference set update method. Nonetheless, the number of new solutions decreases with the number of iterations within the inner while-loop. Suppose that b is set to 10 and that, after the first iteration of the inner while-loop, 6 solutions are replaced in the reference set. This means that the reference set that will serve as the input to the subset generation method will consist of 4 "old" solutions and 6 "new" solutions. Then, the output of the subset generation method will be 39 2-subsets. In general, if the reference set contains n new solutions and m old ones, the number of 2-subsets that the subset generation method produces is given by:

$$nm + \frac{n^2 - n}{2}$$

The scatter search methodology also considers the generation of subsets with more than two elements for the purpose of combining reference solutions. As described in Laguna and Martí (2003), the procedure uses a strategy to expand pairs into subsets of larger size while controlling the total number of subsets to be generated. In other words, the mechanism does not attempt to create all 2-subsets, then all 3-subsets, and so on until reaching the b-1-subsets and finally the entire *RefSet*. This approach would not be practical because there are 1013 subsets in a reference set of size b = 10. Even for a smaller reference set, combining all possible subsets

would not effective because many subsets will be very similar. For example, a subset of size four containing solutions 1, 2, 3, and 4 is almost the same as all the subsets with four solutions for which the first three solutions are solutions 1, 2 and 3. And even if the combination of subset {1, 2, 3, 5} would generate a different solution than the combination of subset {1, 2, 3, 6}, these new trial solutions would likely reside in the same basin of attraction and therefore converge to the same local optimum after the application of the improvement method. Instead, the approach selects representative subsets of different sizes by creating subset types:

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

Campos et al. (2001) designed an experiment with the goal of assessing the contribution of combining subset types 1 to 4 in the context of the linear ordering problem. The experiment undertook to identify how often, across a set of benchmark problems, the best solutions came from combinations of reference solution subsets of various sizes. The experimental results showed that most of the contribution (measured as the percentage of time that the best solutions came from a particular subset type) could be attributed to subset type 1. It was acknowledge, however, that the results could change if the subset types were generated in a different sequence. Nonetheless, the experiments indicate that the basic SS that employs only subsets of type 1 is quite effective and explains why most implementations do not use subset types of higher dimensions.

5. Hybridizing Scatter Search with Tabu Search

Evolutionary approaches, such as scatter search, implicitly make use of memory. This is evident if one examines how the Reference Set Update, Solution Combination and Subset Generation Methods operate. The Reference Set Update Method, in its most basic form, is designed to "remember" the best solutions encountered during the search. Some features of these solutions are used to create new trial solutions with the Combination Method. Hence, this method is instrumental in the transmission of information embedded in the reference solutions. But we can add extra memory structures to scatter search by hybridizing it with tabu search. Specifically, in this section we briefly describe how to design tabu search based procedures to implement the diversification generation, the improvement and the combination methods for the maximum diversity problem.

5.1 Diversification Generation Method

The diversification generator within the Tabu Search methodology, *MD*-2, is also based on the destructive procedure *D*-2. At each step of the procedure, the element i^* to be deselected is such that:

$$D(i^*) = \min_{i:x_i=1}\left(D(i) - \beta(range)\left(\frac{f(i)}{f_{\max}}\right) + \delta(range)\left(\frac{q(i)}{q_{\max}}\right)\right),$$

where $range = \max_{i:x_i=1}(D(i)) - \min_{i:x_i=1}(D(i))$

In this modified distance calculation, $f(i)$ indicates the frequency in which element i has appeared in previous solutions and $q(i)$ is the average quality (as measured by the objective function value) of past solutions that included element i. The f_{max} and q_{max} are the maximum values of f and q over all elements. The penalty factors β and δ are respectively set to 0.1 and 0.0001 according to Gallego et al. (2009).

5.2 Improvement Method

LS_TS (Duarte and Martí 2007) implements a short-term tabu search method also based on exchanges. An iteration of this method begins with a random selection of an element i ($x_i = 1$). The probability of selecting element i is inversely proportional to $D(i)$. The list of unselected elements is scanned and the first improving move that exchanges elements i and j ($x_j = 0$) is selected. If no improving move is found, then the least non-improving move is chosen. The chosen exchange is performed and both elements participating in the exchange are classified tabu-active for a number of iterations (known as the tabu tenure). Tabu-active elements are not allowed to participate in any exchanges. The LS_TS method stops if after a number of consecutive iterations the incumbent solution is not modified.

5.3 Combination Method

This method consists of the application of the *MD*-2 procedure to the union of the elements of the reference solutions being combined. This method uses information about solutions generated in the past as well as information associated with those solutions combined in previous iterations. Once we obtain the set of elements selected in the solutions being combined, we only need to apply MD-2 with the modified evaluations according to the penalty terms:

$$\beta(range)\left(\frac{f(i)}{f_{\max}}\right) \quad \text{and} \quad \delta(range)\left(\frac{q(i)}{q_{\max}}\right)$$

The element with minimum D value is deselected. The union of elements updated, iteration are performed until the number of selected elements matches m.

6. Experiments with the Maximum Diversity Problem

In this section we summarize some of the experiments presented in Gallego et al. (2009) for the MDP. The scatter search implementation follows the basic framework outlined in Figure 1. The diversification generation, combination and improvement methods are those corresponding to the descriptions in Section 3. All the experiments were conducted on a Pentium 4 computer at 3 GHz with 3 GB of RAM. The procedures were coded in Java and executed in the Java Runtime Environment 1.5. We use the same four data sets employed in Duarte and Martí (2007):

SOM: This data set consists of 20 matrices with random numbers between 0 and 9 generated from an integer uniform distribution. The problem sizes are such that for n = 100, m = 10, 20, 30 and 40; for n = 200, m = 20, 40, 60 and 80; for n = 300, m = 30, 60, 90 and 120; for n = 400, m = 40, 80, 120, and 160;

and for n = 500, m = 50, 100, 150 and 200. These instances were generated by Silva, Ochi and Martins (2004).

GKD: This data set consists of 20 matrices for which the values were calculated as the Euclidean distances from randomly generated points with coordinates in the 0 to 10 range. The number of coordinates for each point is also randomly generated between 2 and 21. Glover, Kuo and Dhir (1998) developed this data generator and constructed instances with n = 30. We generated instances with n = 500 and m = 50.

Type I: Diversity values are real numbers uniformly distributed between 0 and 1000. We generate 30 instances of Type I with n = 500, 2000 and m = 50, 200.

Type II: Diversity values are real numbers uniformly distributed between 0 and 10. We generate 20 instances of Type II, as described in section 2, with n = 500 and m = 50.

The purpose of the first experiment is identifying the most effective method for generating subsets of reference solutions that are in turn the input to the combination method. For this experiment, we consider combinations of 2, 3, 4 and 5 solutions. Our subset generation method operates as described in Section 4. All subsets of size 2 (SG1) are considered. That is, all pairs of reference solutions are added to the list of subsets. Subsets of size 3 (SG2) are constructed by considering each subset of size 2 and adding the best reference solution that is not part of the subset. Subsets of higher dimensions are constructed following the same logic. That is, subsets of size 4 (SG3) are based on subsets of size 3. Likewise, subsets of size 5 (SG4) are constructed by adding a solution to subsets of size 4. This mechanism avoids the exponential explosion in the number of subsets generated had we considered all possible subsets of size 3, 4 and 5. The results of running this experiment on 20 instances (10 problems of each type, I and II, and with n = 500 and m = 50) are summarized in Table 1. We use the outcomes of our experiments to calculate the average percent deviation (*Average Deviation*) of the solutions obtained by each procedure when compared to the best solutions during the given experiments. We also report on the number of best solutions (*Number of Best*) found by each method.

Subset Generation Method	Average Deviation	Number of Best
SG1	0.0000%	20
SG2	0.0017%	19
SG3	0.0000%	20
SG4	0.0042%	18

Table 1. Alternative subset generation methods

The results of this preliminary experiment indicate that there is no additional gain that could be realized by generating and combining subsets with more than 2 solutions. Hence, we perform step 2 (see Figure 1) of the scatter search implementation by limiting the subset generation to all pairs of reference solutions. These results are in line with similar experiments for other combinatorial optimization problems (Campos, et al. 2001).

The objective of the second experiment is to compare the relative merit of the scatter search variants described in Sections 3 and 5 respectively. We set a time limit of 3 CPU minutes and we

run each procedure with and without the improvement method. The results are summarized in Table 2.

Procedure	Improvement	Average Deviation	Number of Best
Scatter Search	No	1.16%	0
(Section 3)	Yes	0.18%	2
Tabu Search Hybrid	No	1.07%	0
(Section 5)	Yes	0.00%	20

Table 2. Different scatter search designs

The results in Table 2 indicate the advantage of using an improvement method within our design. In terms of average deviation from the best solutions, the improvement method has the largest impact in the case of the Base design. Also, the improvement method makes a significant difference in the number of best solutions found in the Tabu Search Hybrid. In general, we conclude that the procedures embedded in the Tabu Search Hybrid variant results in the best scatter search configuration. For the next experiment, the scatter search that we use is the one that implements the Tabu Search Hybrid procedures (including the improvement method), as described in Section 5.

In the final experiment we compare the proposed procedure to state-of-the-art methods for solving the maximum diversity problem. In particular we consider:

- KLD (Silva, Ochi and Martins, 2004) with local search (Ghosh, 1996)
- KLDv2 (Silva, Ochi and Martins, 2004) with local search (Ghosh, 1996)
- Tabu_*D*-2 with LS_TS (Duarte and Martí, 2007)

In this experiment, we observed the solution quality obtained by each method after 30 seconds and after 3 minutes of search time. Table 3 reports the average percentage deviation of the solution obtained with each method with respect to the best solution known.

Data Set	Time	KLD	KLDv2	Tabu_D-2	SS
SOM	30 sec.	1.056%	1.463%	0.138%	0.002%
	3 min.	0.178%	0.187%	0.095%	0.000%
GKD	30 sec.	0.000%	0.000%	0.000%	0.000%
	3 min.	0.000%	0.000%	0.000%	0.000%
Type II	30 sec.	0.857%	1.083%	0.245%	0.010%
	3 min.	0.525%	0.607%	0.203%	0.000%
Type I	30 sec.	9.807%	100.000%	0.453%	0.453%
	3 min.	9.807%	9.828%	0.331%	0.331%

Table 3. Comparison of average percent deviation at two times during the search

Table 3 shows the merit of the proposed procedure. The scatter search implementation consistently produces the best solutions with percent deviations that in some cases are orders of magnitude smaller than those of the competing methods. The problem instances in the GKD set do not provide a way of differentiating the performance of the methods that we are comparing. They are either easy to solve and all the methods are capable of finding the optimal solutions in a very short period of time or the problems are difficult and all the methods are attracted to the same local optima

7. Conclusions

Our goal was to introduce the scatter search framework at a level that would make it possible for the reader to implement a basic but robust procedure. A number of extensions are possible and some of them have already been explored and reported in the literature. It is not possible within the limited scope of this encyclopedic article to detail completely many of the aspects of scatter search that warrant further investigation. Additional implementation considerations, including those associated with intensification and diversification processes, and the design of accompanying methods to improve solutions produced by combination strategies are found in several of the references listed below.

Acknowledgments

This research has been partially supported by the *Ministerio de Ciencia e Innovación* of Spain (TIN2009-07516).

References

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

Duarte, A. and R. Martí (2007) “Tabu Search and GRASP for the Maximum Diversity Problem,” *European Journal of Operational Research* 178, 71-84.

Gallego, M., A. Duarte, M. Laguna y R. Martí (2009) “Hybrid heuristics for the maximum diversity problem”, Computational Optimization and Applications 44(3), 411-426

Ghosh, J. B. (1996) “Computational Aspects of the Maximum Diversity Problem,” *Operations Research Letters*, vol. 19, pp. 175-181.

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

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 and D. Snyers (eds.), pp. 13-54.

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., C. C. Kuo, and K. S. Dhir (1998) “Heuristic Algorithms for the Maximum Diversity Problem,” *Journal of Information and Optimization Sciences*, vol. 19, no. 1, pp. 109-132.

Glover, F., M. Laguna (1997) *Tabu Search*. Kluwer Academic Publisher.

Hamming, R. W. (1950) “Error Detecting and Error Correcting Codes,” *The Bell System Technical Journal,* vol. 26, no. 2, pp. 147-160.

Kuo, C. C., F. Glover and K. S. Dhir (1993) “Analyzing and Modeling the Maximum Diversity Problem by Zero-One Programming,” *Decision Sciences*, vol. 24, no. 6, pp. 1171-1185.

Laguna (2009) “Scatter Search,” to appear in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques,* 2nd *Edition,* E. K. Burke and G. Kendall (eds.)

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

Laguna, M. and R. Martí (2005) "Experimental Testing of Advanced Scatter Search Designs for Global Optimization of Multimodal Functions," *Journal of Global Optimization*, vol. 33, pp. 235-255.

Martí, R., A. Duarte and M. Laguna (2009) "Advanced Scatter Search for the Max-Cut Problem," *INFORMS Journal on Computing*, vol. 21, no. 1, pp. 26-38.

Martí, R., M. Laguna and V. Campos (2005) "Scatter Search vs. Genetic Algorithms: An Experimental Evaluation with Permutation Problems," *Metaheuristic Optimization Via Adaptive Memory and Evolution: Tabu Search and Scatter Search*, C. Rego and B. Alidaee (eds.), Norwell, MA: Kluwer Academic Publishers, pp. 263-282.

Resende, M.G.C., C.C. Ribeiro (2001) Greedy randomized adaptive search procedures. F. Glover, G. Kochenberger, eds. *State-of-the-art Handbook in Metaheuristics*, Kluwer Academic Publishers, Boston, MA. 219–250.

Silva, G. C., L. S. Ochi and S. L. Martins (2004) "Experimental Comparison of Greedy Randomized Adaptive Search Procedures for the Maximum Diversity Problem," *Lecture Notes in Computer Science* 3059, Springer: Berlin Heidelberg, pp. 498-512.

Yin, P.Y., F. Glover, M. Laguna and J-X Zhu (2010) Cyber Swarm Algorithms – Improving Particle Swarm Optimization Using Adaptive Memory Strategies, European Journal of Operational Research 201 (2), 377-389.