Scatter Search

Fred Glover[1], Manuel Laguna[1] and Rafael Martí[2]

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

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

Abstract: The evolutionary approach called scatter search originated from strategies for creating composite decision rules and surrogate constraints. Recent studies demonstrate the practical advantages of this approach for solving a diverse array of optimisation problems from both classical and real world settings. Scatter search contrasts with other evolutionary procedures, such as genetic algorithms, by providing unifying principles for joining solutions based on generalised path constructions in Euclidean space and by utilising strategic designs where other approaches resort to randomisation. Additional advantages are provided by intensification and diversification mechanisms that exploit adaptive memory, drawing on foundations that link scatter search to tabu search. The main goal of this chapter is to demonstrate the development of a scatter search procedure by demonstrating how it may be applied to a class of non-linear optimisation problems on bounded variables. We conclude the chapter highlighting key ideas and research issues that offer promise of yielding future advances.

1. Introduction

Scatter search derives its foundations from earlier strategies for combining decision rules and constraints, with the goal of enabling a solution procedure based on the combined elements to yield better solutions than one based only on the original elements. An examination of these origins sheds light on the character of these methods.

Historically, the antecedent strategies for combining decision rules were introduced in the context of scheduling methods to obtain improved local decision rules for job shop scheduling problems [1]. New rules were generated 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 also became a fundamental component of tabu search. See, e.g., [2].)

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" that selected different rules probabilistically at different junctures, but without the integration effect provided by generating com bined rules [3].

The associated procedures for combining constraints likewise employed a mechanism of generating weighted combinations, in this case applied in the setting of integer and nonlinear programming, by introducing nonnegative weights to create new constraint inequalities, called *surrogate constraints* [4]. 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.

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 generalisation. 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 optimisation. E.g., see [4] - [11].

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 labelled A, B and C. After a non-convex combination of reference solutions A and B, solution 1 is created. In fact, 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 resulting complete reference set shown in Figure 1 consists of 7 solutions (or elements).

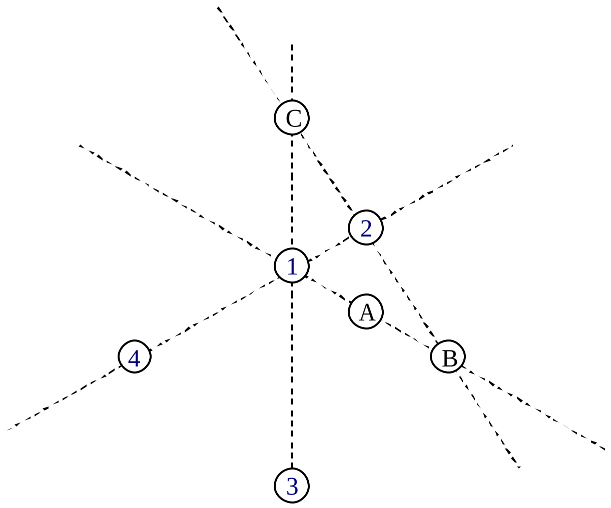

Fig. 1. Two-dimensional reference set.

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 chosen from the population by a process that relies heavily on randomization and a "crossover" mechanism, likewise involving recourse to randomization, is applied to generate one or more offspring. A typical population size in a genetic algorithm consists of 100 elements. In contrast, scatter search chooses two or more elements of the reference set in a systematic way with the purpose of creating new solutions strategically. (Randomization can be invoked, but in a way that is subordinate to this strategic emphasis.) 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 $(3b-7)b/2$ combinations of four different types [12]. The basic type consists of combining two solutions; the next type combines three solutions, and so on.

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.

A two-tier example, which subdivides the reference set into two components, is illustrated in section 3.3

2. Scatter Search Template

The scatter search process, building on the principles that underlie the surrogate constraint design, is organised 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 [12]. 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 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 generalised 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 centres of selected subregions. This adds non-convex combinations that project new centres into regions that are external to the original reference solutions (see, e.g., solution 3 in Figure 1). The dispersion patterns created by such centres 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 organised 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 summarise 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.

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), organised 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 scatter search procedure for unconstrained non-linear optimisation problems. The success of scatter search and related strategies is evident in a variety of application areas such as vehicle routing, arc routing, quadratic assignment, financial product design, neural network training, job shop scheduling, flow shop scheduling, crew scheduling, graph drawing, linear ordering, unconstrained optimisation, bit representation, multi-objective assignment, optimising simulation, tree problems, mixed integer programming, as reported in [12].

3. Scatter Search Tutorial

In this tutorial section we develop a scatter search procedure for the following class of optimisation problems:

Min $f(x)$
Subject to $l \leq x \leq u$

where $f(x)$ is a non-linear function of x. For the purpose of illustrating the solution procedure, we will apply our design to the following problem instance:

$$\text{Minimize} \quad 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2 + 90\left(x_4 - x_3^2\right)^2 + (1 - x_3)^2 + 10.1\left((x_2 - 1)^2 + (x_4 - 1)^2\right) + 19.8(x_2 - 1)(x_4 - 1)$$

$$\text{Subject to} \quad -10 \leq x_i \leq 10 \quad \text{for } i = 1, \ldots, 4$$

This problem is “Test Case #6” in [13]. The best solution found in [13] has an objective function value of 0.001333, reportedly found after 500 000 runs of a genetic algorithm (GA)[1].

3.1 Diversification Generation Method

Our illustrative diversification method employs controlled randomisation (the emphasis on “controlled” is important) drawing upon frequency memory to generate a set of diverse solutions. We accomplish this by dividing the range of each variable $u_i - l_i$ into 4 sub-ranges of equal size. Then, a solution is constructed in two steps. First a sub-range is randomly selected. The probability of selecting a sub-range is inversely proportional to its frequency count. Then a value is

[1] A single run of the GA in [13] consists of 1000 iterations. An iteration requires 70 evaluations of the objective function, where 70 is the size of the population. Therefore, GA required 35 billion objective function evaluations to find this solution.

randomly generated within the selected sub-range. The number of times sub-range *j* has been chosen to generate a value for variable *i* is accumulated in *freq*(*i*, *j*).

The diversification generator is used at the beginning of the search to generate a set of *P* solutions with *PSize*, the cardinality of the set, generally set at max(100, 5**b*) diverse solutions, where *b* is the size of the reference set. Although the actual scatter search implementation would generate 100 solutions for this example, we have limited *P* in our tutorial to the ten solutions shown in Table 1.

Table 1. Diverse solutions

Solution	x_1	x_2	x_3	x_2	$f(x)$
1	1.1082	0.8513	9.4849	-6.3510	835534.3
2	-9.5759	-6.5706	-8.8128	-2.2674	1542087.0
3	-8.3565	0.7865	7.8762	-2.6978	854129.3
4	8.8337	-8.4503	4.5242	3.1800	775473.7
5	-6.2316	7.4765	5.9955	7.8018	171451.1
6	0.1975	-3.6392	-5.2999	-7.0332	114021.1
7	-3.0909	6.6189	-2.3250	-3.1240	7468.8
8	-6.0775	0.6699	-6.4774	1.4775	279100.3
9	-1.9659	8.1258	-5.6343	8.0178	54538.5
10	3.1131	-1.9358	5.8964	6.8859	83607.1

The 100 solutions generated by the actual procedure are diverse with respect to the values that each variable takes in each of the sub-ranges. Note, however, that the generation is done without considering the objective function. In other words, the Diversification Generation Method focuses on diversification and not on the quality of the resulting solutions, as evident from the objective function values in Table 1. In fact, the objective function values for the entire set of 100 solutions range from 1689.7 to 1 542 087 with an average of 392 032.8. These objective function values are very large considering that the optimal solution to this problem has an objective function value of zero. To verify that the Generation method is operating as expected, we show in Table 2 the frequency values corresponding to the complete set of 100 solutions.

Table 2. Frequency counts.

Range	x_1	x_2	x_3	x_4
-10 to 5	19	25	26	29
-5 to 0	25	18	22	21
0 to 5	26	27	29	25
5 to 10	30	30	23	25

Note that the frequency counts in Table 2 are very similar for each range and variable, with a minimum value of 18 and a maximum value of 30. The target frequency value for each range is 25 for a set of 100 solutions. This value is observed four times in Table 2.

3.2 Improvement Method

Since we are developing a solution procedure for a class of unconstrained optimisation problems, the solutions constructed with the Diversification Generation Method are guaranteed to be feasible. Improvement Methods, however, must be able to handle starting solutions that are either feasible or infeasible.

The Improvement Method we use in this tutorial consists of a classical local optimiser for unconstrained non-linear optimisation problems. In particular, we apply Nelder and Mead's simplex method [14] to each solution in Table 1. After the application of this local optimiser, the solutions in Table 3 are transformed to the solutions in Table 2.

Table 3. Improved solutions.

Solution	x_1	x_2	x_3	x_2	$f(x)$
1	-0.8833	0.7514	1.1488	1.3231	3.7411
2	1.3952	1.9411	0.3424	0.0977	0.9809
3	0.0921	0.0668	-1.3826	1.8998	7.2004
4	0.6172	0.3937	-1.3036	1.641	5.9422
5	-1.1008	1.2423	0.9054	0.7429	5.0741
6	2.3414	5.9238	0.5721	-0.5716	210.8007
7	-1.2979	1.6781	-0.6035	0.2775	8.7473
8	-2.2507	4.8302	-1.8987	3.2877	408.1172
9	-2.2492	4.1608	-2.9055	8.0508	1164.5050
10	0.8406	0.6755	-1.0425	1.0697	4.9854

The objective function values now range between 0.9809 and 1164.5050 for the improved solutions in Table 3. It is interesting to point out that when the 100 solutions are subjected to the Improvement Method, 5 solutions converge to the same local minimum, thus effectively reducing the cardinality of *P*. As shown in Section 4 we could apply the Diversification Method again until *PSize* different solutions are found after executing the Improvement Method. Incidentally, the convergence point for those 5 solutions turns out to be the global optimum that sets $x = (1, 1, 1, 1)$ for an objective function value of zero. Hence for this small example, a single application of the Diversification and Improvement Methods is sufficient to find the optimal solution.

3.3 Reference Set Update Method

The reference set, *RefSet*, is a collection of both high quality solutions and diverse solutions that are used to generate new solutions by way of applying the Solution Combination Method. Specifically, the reference set consists of the union of two subsets, $RefSet_1$ and $RefSet_2$, of size b_1 and b_2, respectively. That is, $|RefSet| = b = b_1 + b_2$. The construction of the initial reference set starts with the selection of the best b_1 solutions from P. These solutions are added to *RefSet* and deleted from P.

For each improved solution in *P-RefSet*, the minimum of the Euclidean distances to the solutions in *RefSet* is computed. Then, the solution with the maximum of these minimum distances is selected. This solution is added to *RefSet* and deleted from P and the minimum distances are updated. This process is repeated b_2 times. The resulting reference set has b_1 high-quality solutions and b_2 diverse solutions.

Let us apply this initialisation procedure to our example, using the following parameter values: $b = 5$, $b_1 = 3$ and $b_2 = 2$, and considering that P is limited to the set of improved solutions in Table 3. The best 3 solutions in Table 3 are shown in Table 4.

Table 4. High-quality subset of *RefSet*.

Solution	x_1	x_2	x_3	x_4	$f(x)$
2	1.3952	1.9411	0.3424	0.0977	0.9809
1	-0.8833	0.7514	1.1488	1.3231	3.7411
10	0.8406	0.6755	-1.0425	1.0697	4.9854

We then calculate the minimum distance $d_{\min}(x)$ between each solution x in *P-RefSet* and the solutions y currently in *RefSet*. That is,

$$d_{\min}(x) = \underset{y \in RefSet}{Min} \{ d(x, y) \},$$

where $d(x, y)$ is the Euclidean distance between x and y. For example, the minimum distance between solution 3 in Table 3 (x^3) and the *RefSet* solutions in Table 4 (x^2, x^1, and x^{10}) is calculated as follows:

$$d_{\min}\left(x^3\right) = Min\left\{ d\left(x^3, x^2\right), d\left(x^3, x^1\right), d\left(x^3, x^{10}\right) \right\}$$
$$d_{\min}\left(x^3\right) = Min\{3.38, 2.86, 1.32\} = 1.32$$

The maximum $d_{\min}$ value for the solutions in *P-RefSet* corresponds to solution 9 in Table 3 ($d_{\min}(x^9) = 8.6$). We add this solution to *RefSet*, delete it from P and update the $d_{\min}$ values. The new maximum $d_{\min}$ value of 4.91 corresponds to solution 8 in Table 3, so the diverse subset of *RefSet* is as shown in Table 5.

Table 5. Diverse subset of *RefSet*.

Solution	x_1	x_2	x_3	x_4	$f(x)$
9	-2.2492	4.1608	-2.9055	8.0508	1164.5
8	-2.2507	4.8302	-1.8987	3.2877	408.2

After the initial reference set is constructed, the Solution Combination Method is applied to the subsets generated as outlined in the following section. The reference set is dynamically updated during the application of the Solution Combination Method. A newly generated solution may become a member of the reference set if either one of the following conditions is satisfied:

– The new solution has a better objective function value than the solution with the worst objective value in $RefSet_1$.
– The new solution has a better $d_{\min}$ value than the solution with the worst $d_{\min}$ value in $RefSet_2$.

In both cases, the new solution replaces the worst and the ranking is updated to identify the new worst solution in terms of either quality or diversity. The reference set is also regenerated when the Combination Method is incapable of creating solutions that can be admitted to *RefSet* according to the rules outlined above. The regeneration consists of keeping $RefSet_1$ intact and using the Diversification Generation Method to construct a new diverse subset $RefSet_2$.

3.4 Subset Generation Method

This method consists of generating the subsets that will be used for creating new solutions with the Solution Combination Method. The Subset Generation 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.

Note that due to the rules for generating subsets of type 2, 3 and 4, the same subset may be generated more than once. Simple and efficient procedures outlined in [12] can be used to generate all the unique subsets of each type.

For the purpose of our tutorial, we limit our scope to type-1 subsets consisting of all pairwise combinations of the solutions in *RefSet*. There are $(b^2-b)/2$ type-1 subsets, which in the case of our example amounts to a total of $(5^2-5)/2 = 10$.

3.5 Solution Combination Method

This method uses the subsets generated with the Subset Generation Method to combine the elements in each subset with the purpose of creating new trial solutions. Generally, the Solution Combination Method is a problem-specific mechanism, since it 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. Let us consider the following Combination Method for solutions that can be represented by bounded continuous variables.

The method consists of finding linear combinations of reference solutions. The number of solutions created from the linear combination of two reference solutions depends on the membership of the solutions being combined. These combinations in our illustration are based on the following three types, assuming that the reference solutions are x' and x'':

C1: $x = x' - d$
C2: $x = x' + d$
C3: $x = x'' + d$

where $d = r(x'' - x')/2$ and r is a random number in the range (0, 1). The following rules are used to generate solutions with these three types of linear combinations:

- If both x' and x'' are elements of $RefSet_1$, then generate 4 solutions by applying C1 and C3 once and C2 twice.
- If only one of x' and x'' is a member of $RefSet_1$, then generate 3 solutions by applying C1, C2 and C3 once.
- If neither x' nor x'' is a member of $RefSet_1$, then generate 2 solutions by applying C2 once and randomly choosing between applying C1 or C3.

To illustrate the combination mechanism, consider the combination of the first two solutions in $RefSet_1$ (i.e., solutions 2 and 1 in Table 4). Table 6 shows the solutions generated from combining these two high quality reference points.

Table 6. New solutions from combining x^1 and x^2.

Type	r	x_1	x_2	x_3	x_4	$f(x)$
C1	0.1330	1.5468	2.0202	0.2888	0.0162	15.480
C2(a)	0.3822	0.9598	1.7138	0.4965	0.3319	63.937
C2(b)	0.6862	0.6134	1.5329	0.6191	0.5181	135.83
C3	0.3551	-1.2879	0.5401	1.2920	1.5407	132.09

Solutions generated with the Solution Combination Method are subjected to the Improvement Method before they are considered for membership in the reference set. After applying the Improvement Method to the solutions in Table 6, the solutions are transformed to those shown in Table 7.

Table 7. Improved new solutions.

Type	x_1	x_2	x_3	x_4	$f(x)$
C1	1.2565	1.5732	0.6746	0.4464	0.3125
C2(a)	-0.4779	0.2308	-1.2065	1.4939	8.1029
C2(b)	1.0354	1.0746	0.9625	0.9266	0.0055
C3	0.8506	0.7171	1.1408	1.3101	0.0956

The best solution in Table 7 is C2(b) with an objective function value of 0.0055. According to our updating rules for the reference set, this solution should replace solution 10 in Table 4, because solution 10 is the worst in the $RefSet_1$ subset.

The search continues in a loop that consists of applying the Solution Combination Method followed by the Improvement Method and the Reference Update Method. This loop terminates when the reference set does not change and all the subsets have already been subjected to the Solution Combination Method. At this point, the Diversification Generation Method is used to construct a new $RefSet_2$ and the search continues [2].

4. An Outline of the Procedure

In the previous section, we illustrated the operations that take place within each of the methods in the scatter search framework. 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.
b1 =	the size of the high-quality subset.
b2 =	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

[2] A copy of an experimental C code of the procedure described in this tutorial section can be obtained from the authors.

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 application of the Solution 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*. As previously indicated, *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 *RefSet*$_2$, 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.

The general procedure outlined in Figure 2 can be modified in a variety of ways. One possibility is to eliminate the outer "for-loop" along with step 12. In this case, set P is generated one time only and the process stops after no new elements are admitted into 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 amount of 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 of the procedure (as measured by the quality of the best solution found).

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, making the full procedure outlined in Figure 2 more attractive. A reasonable approach could be to begin with the more aggressive variant and then to shift to the more through variant (or alternate between these variants).

```
1. Start with $P = Ø$. 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 maximises
     $d_{\min}(x)$, 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.
```

Fig. 2. Scatter Search Outline

```
        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.
            Initialise 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
```

Fig. 2. (Continued) Scatter Search Outline

5. Implications for Future Developments

The focus and emphasis of the scatter search approach have a number of specific implications for the goal of designing improved optimisation procedures. To understand these implications, it is useful to consider certain contrasts between the highly exploitable meaning of “solution combination” provided by scatter search 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 favour the types of crossover models originally proposed with genetic algorithms – since these give rise to the theorems that have helped to popularise 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 organising 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 structured combinations. 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*.

Specific areas of research for developing improved solution strategies that emerge directly from the scatter search orientation are:

- Strategies for clustering and anti-clustering, to generate candidate sets of solutions to be combined
- Rules for multi-parent compositions
- 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 scatter search is motivated by connections with the tabu search setting and invites the use of adaptive memory structures in determining the strategies produced. The learning approach called target analysis [2] gives a particularly useful basis for pursuing such research.

[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.

6. Randomisation and the Intensification/Diversification Dichotomy

The emphasis on systematic strategies in achieving intensification and diversification does not preclude the use of randomised selection schemes, which are often motivated by the fact that they require little thought or sophistication to apply (as illustrated by the Solution Combination Method of Section 3.5). By the same token, deterministic rules that are constructed with no more reflection than devoted to creating a simple randomised rule can be quite risky, because they can easily embody oversights that will cause them to perform poorly. A randomised 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 randomised 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 randomised 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 randomised variation are rarely the same.[4] 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 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" popularised 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 randomised departures are launched.

[4] 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 settings, a different perspective may result by investigating comparably fast mechanisms that replace randomization with intelligent variation.

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 [2], rather than to rely solely on indirect "inheritance effects."

7. Conclusion

It is not possible within the limited scope of this chapter to detail completely the aspects of scatter search 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 [12].

However, 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. By means of these connections, a wide range of strategic possibilities exist for implementing scatter search.

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 [15] suggest that these approaches may offer a useful potential for applications in areas beyond those investigated up to now.

References

1. Glover F. Parametric Combinations of Local Job Shop Rules. ONR Research Memorandum no. 117, GSIA, Carnegie Mellon University, Pittsburgh, PA, 1963, Chapter IV.
2. Glover F. and Laguna M. *Tabu Search*, Kluwer Academic Publishers, Boston, 1997.
3. Crowston WB, Glover F, Thompson GL and Trawick JD. Probabilistic and Parametric Learning Combinations of Local Job Shop Scheduling Rules. ONR Research Memorandum No. 117, GSIA, Carnegie Mellon University, Pittsburgh, PA, 1963.
4. Glover F. A Multiphase Dual Algorithm for the Zero-One Integer Programming Problem, *Operations Research* 1965; 13(6): 879.
5. Greenberg HJ and Pierskalla WP. Surrogate Mathematical Programs. *Operations Research* 1970; 18: 924-939.

6. Greenberg HJ and Pierskalla WP. Quasi-conjugate Functions and Surrogate Duality. *Cahiers du Centre d'Etudes de Recherche Operationelle* 1973; 15: 437-448.
7. Glover F. Surrogate Constraint Duality in Mathematical Programming. *Operations Research* 1975; 23: 434-451.
8. Karwan MH and Rardin RL. Surrogate Dual Multiplier Search Procedures in Integer Programming. School of Industrial Systems Engineering, Report Series No. J77-13, Georgia Institute of Technology, 1976.
9. Karwan MH and Rardin RL. Some Relationships Between Lagrangean and Surrogate Duality in Integer Programming. *Mathematical Programming* 1979; 17: 230-334.
10. Freville A and Plateau G. Heuristics and Reduction Methods for Multiple Constraint 0-1 Linear Programming Problems. *European Journal of Operational Research* 1986; 24: 206-215.
11. Freville A and Plateau G. An Exact Search for the Solution of the Surrogate Dual of the 0-1 Bidimensional Knapsack Problem. *European Journal of Operational Research* 1993; 68: 413-421.
12. Glover F. A Template for Scatter Search and Path Relinking. In: Hao JK, Lutton E, Ronald E, Schoenauer M, and Snyers D (eds.) *Lecture Notes in Computer Science* 1997; 1363: 1-53.
13. Michlewicz Z and Logan TD. Evolutionary Operators for Continuous Convex Parameter Spaces. In: Sebald AV and Fogel LJ (eds.) *Proceedings of the 3rd Annual Conference on Evolutionary Programming*. World Scientific Publishing, River Edge, NJ, 1994, pp. 84-97.
14. Nelder JA and Mead R. A Simplex Method for Function Minimisation. *Computer Journal* 1965; 7: 308.
15. Glover F. Scatter Search and Path Relinking. In: D Corne, M Dorigo and F Glover (eds.) *New Ideas in Optimisation*, McGraw-Hill, 1999.