Working Paper Series

HEARIN CENTER
FOR
ENTERPRISE SCIENCE

HCES-10-00

A Scatter Search Tutorial for Graph-Based Permutation Problems

by
César Rego
Pedro Leão

The University of Mississippi

Director, Keith Womer
School of Business Administration
The University of Mississippi
Post Office Box 1848
University, MS 38677-1848
(662) 915-5820
http://hces.bus.olemiss.edu

A Scatter Search Tutorial for Graph-Based Permutation Problems

César Rego[a,†] and Pedro Leão[b]

[a] Hearin Center for Enterprise Science, School of Business Administration, University of Mississippi, University, MS 38677, USA. crego@bus.olemiss.edu

[b] Universidade Portucalense, Departamento de Informática, Rua Dr. António Bernardino de Almeida 541-619, 4200, Porto, Portugal. pedro@upt.pt

Latest Revision: November 21, 2001.

Abstract — Scatter search is an evolutionary method that has proved highly effective in solving several classes of non-linear and combinatorial optimization problems. Proposed early 1970s as a primal counterpart to the dual surrogate constraint relaxation methods, scatter search has recently found a variety of applications in a metaheuristic context. Because both surrogate constraint methods and scatter search incorporate strategic principles that are shared with certain components of tabu search methods, scatter search provides a natural evolutionary framework for adaptive memory programming. The aim of this paper is to illustrate how scatter search can be effectively used for the solution of general permutation problems that involve the determination of optimal cycles (or circuits) in graph theory and combinatorial optimization. In evidence of the value of this method in solving constrained optimization problems, we identify a general design for solving vehicle routing problems that sets our approach apart from other evolutionary algorithms that have been proposed for various classes of this problem.

Keywords: Metaheuristics, scatter search, combinatorial optimization, permutation problems, vehicle routing problems.

[†] Corresponding author.
The research of the first author has been supported in part by the Office of Naval Research (ONR) grants N000140010598 and N000140010769.

1. Introduction

Scatter search is an evolutionary method proposed by Glover (1977) as a primal counterpart to the dual approaches called surrogate constraint methods, which were introduced as mathematical relaxation techniques for discrete optimization problems (Glover 1965). As opposed to eliminating constraints by plugging them into the objective function as in Lagrangean relaxations, for example, surrogate relaxations have the goal of generating new constraints that may stand in for the original constraints. The method is based on the principle of capturing relevant information contained in individual constraints and integrating it into new surrogate constraints as a way to generate composite decision rules and new trial solutions. Scatter search combines vectors of solutions in place of the surrogate constraint approach of combining vectors of constraints, and likewise is organized to capture information not contained separately in the original vectors. Also, in common with surrogate constraint methods it is organized to take advantage of auxiliary heuristic solution methods to evaluate the combinations produced and generate new vectors. As any evolutionary procedure, the method maintains a population of solutions that evolves in successive generations.

A number of algorithms based on the scatter search approach have recently been proposed for various combinatorial problems (see Kelly, Rangaswamy and Xu 1996, Fleurent *et al.* 1996, Cung *et al.* 1999, Laguna, Martí and Campos 1999, Campos *et al.* 1999, Glover, Løkketangen and Woodruff 1999, Atan and Secomandi 1999, Laguna, Lourenço and Martí 2000, Xu, Chiu and Glover 2000).

In this study we provide a general scatter search design for problems dealing with the optimization of cycles (or circuits) on graphs. The traveling salesman problem (TSP) which consists of finding the shortest Hamiltonian *cycle* (or *circuit* in the asymmetric case) is a typical example of this class of problems. It is well-known that the TSP is NP-Complete so that efficient heuristic methods are required to provide high quality solutions for large problem instances (see Johnson and McGeoch 1997, Rego 1998). A direct extension of the TSP is the vehicle routing problem (VRP) where side constraints force the creation of multiple Hamiltonian cycles (*routes*) starting and ending at a central node (representing a hub or depot). Both TSP and VRP problems provide a general model for a wide range of practical applications and are central in the fields of transportation, distribution and logistics (see Reinelt 1994, Laporte and Osman 1995). Empirical studies have shown that the VRP is significantly harder to solve than TSPs of similar sizes (see Gendreau, Hertz and Laporte 1994, Rochat and Taillard 1995, and Rego 1998, 2000, for some of the current best heuristic algorithms for the VRP). Because of its theoretical and practical relevance we use the vehicle routing problem to illustrate the various procedures involved in the scatter search template.

Also, while other tutorials on scatter search exist (see, e.g. the articles by Laguna (2001) and Glover, Laguna and Martí (2001)), there have been no exposition of the approach that disclose its operation within the setting of routing.

The remainder of this paper is organized as follows. An overview of the scatter search method is presented in section 2. Section 3 defines the vehicle routing problem and presents formulations that are relevant for the context of the paper. Section 4 illustrates the design of the scatter search procedure for an instance of the problem. Section 5 summarizes and discusses possible generalizations of the proposed template.

2. Scatter Search Overview

Scatter search operates on a set of reference solutions to generate new solutions by weighted linear combinations of structured subsets of solutions. The reference set is required to be made up of high-quality and diverse solutions and the goal is to produce weighted centers of selected subregions that project these centers into regions of the solution space to be explored by auxiliary heuristic procedures. Depending on whether convex or nonconvex combinations are used, the projected regions can be respectively internal or external to the selected subregions. For problems where vector components are required to be integer a rounding process is used to yield integer values for such components. Rounding can be achieved either by a generalized rounding method or iteratively, using updating to account for conditional dependencies that can modify the rounding options. Regardless the type of combinations employed, the projected regions are not required to be feasible so that the auxiliary heuristic procedures are usually designed to incorporate a double function of mapping an infeasible point to a feasible trial solution and then to transform this solution into one or more trial solutions. (Although, the auxiliary heuristic commonly includes the function of restoring feasibility, this is not an absolute requirement since scatter search can be allowed to operate in the infeasible solution space.) From the implementation standpoint the scatter search method can be structured to consist of the following subroutines:

Diversification Generation Method - Used to generate diverse trial solutions from one or more arbitrary *seed solutions* used to initiate the method.

Improvement Method - Transform a trial solution into one or more enhanced trial solutions. (If no improvement occurs for a given trial solution, the enhanced solution is considered to be the same as the one submitted for improvement.)

Reference Set Update Method - Creates and maintains a set of *reference solutions* consisting of the best according to the criteria under consideration. The goal is to ensure diversity while keeping high-quality solutions as measured by the objective function.

Subset Generation Method - Generates subsets of the reference set as a basis for creating combined solutions.

Solution Combination Method - Uses weighted structured combinations to transform each subset of solutions produced by the subset generation method into one or more combined solutions.

A general template for a scatter search algorithm can be organized in two phases outlined as follows.

Initial Phase
1. Diversification Generation Method
2. Improvement Method
3. Reference Set Update Method
4. Repeat this initial phase until producing a desirable level of high-quality and diverse solutions.

Scatter Search Phase
5. Subset Generation Method
6. Solution Combination Method
7. Improvement Method
8. Reference Set Update Method

9. Repeat this scatter search phase while the reference set converges or until a specified cutoff limit on the total number of iterations.

3. The Vehicle Routing Problem

The vehicle routing problem is a classic in Combinatorial Optimization. To establish some basic notation, and to set the stage for subsequent illustrations, we provide a formal definition of the vehicle routing problem in this section. We also present two mathematical formulations which are relevant for the scatter search design introduced in the next section.

3.1. Problem definition

In graph theory terms the classical VRP can be defined as follows. Let $G = (V, A)$ be a graph where $V = \{v_0, v_1, \ldots, v_n\}$ is a vertex set, and $A = \{(v_i, v_j) \mid v_i, v_j \in V, i \neq j\}$ is an arc set. Vertex v_0 denotes a *depot*, where a fleet of m identical vehicles of capacity Q are based, and the remaining vertices $V' = V \setminus \{v_0\}$ represent n *cities* (or client locations). A nonnegative *cost* or distance matrix $C = (c_{ij})$ which satisfies the triangle inequality ($c_{ij} \leq c_{ik} + c_{kj}$) is defined on A. When $c_{ij} = c_{ji}$ for all $(v_i, v_j) \in A$ the problem is said to be symmetric and it is then common to replace A with the edge set $E = \{(v_i, v_j) \mid v_i, v_j \in V, i \neq j\}$. It is assumed that $m \in [\underline{m}, \overline{m}]$ with $\underline{m} = 1$ and $\overline{m} = n - 1$. The value of m can be a decision variable or can be fixed depending on the application.

Vehicles make *collections* or deliveries but not both. With each vertex v_i is associated a quantity q_i $(q_0 = 0)$ of some goods to be delivered by a vehicle. The VRP consists of determining a set of m vehicle routes of minimal total cost, starting and ending at a depot v_0, such that every vertex in $v_i \in V'$ is visited exactly once by one vehicle and the total quantity assigned to each route does not exceed the capacity Q of the vehicle which services the route.

We define a solution for the VRP as a set of m routes, $S = \{R_1, \ldots, R_m\}$, $R_k = (v_0, v_{k_1}, v_{k_2}, \ldots v_0)$ where R_k is an ordered set representing consecutive vertices in the route k. Consequently, we denote $v_i \in R_k$ if v_i is component of R_k and similarly $(v_i, v_j) \in R_k$ if v_i, v_k are two consecutive vertices in R_k. Finally, the cost of a solution S is defined as $C(S) = \sum_{1 \leq k \leq m} \sum_{(i,j) \in R_k} c_{ij}$.

3.2. Vehicle Flow Formulation

$$\min \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} \sum_{k=1}^{m} x_{ij}^{k}$$

$s.t.$

$$\sum_{i=1}^{n} q_i y_i^k \le Q_k, \qquad k = 1,\ldots,m \qquad (1)$$

$$\sum_{k=1}^{m} y_i^k = \begin{cases} m, & i = 0 \\ 1, & i = 1,\ldots,n \end{cases} \qquad (2)$$

$$\sum_{i=0}^{n} x_{ij}^k = \sum_{i=0}^{n} x_{ji}^k = y_i^k, \qquad \begin{matrix} j = 0,\ldots,n \\ k = 1,\ldots,m \end{matrix} \qquad (3)$$

$$\sum_{i,j \in U}^{n} x_{ij}^k \le |U| - 1 \qquad \begin{matrix} 1 \le |U| \le n-1 \\ i = 0,\ldots,n \\ k = 1,\ldots,m \end{matrix} \qquad (4)$$

$$y_i^k, x_{ij}^k \in \{0,1\} \qquad \begin{matrix} i,j = 1,\ldots,n \\ k = 1,\ldots m \end{matrix} \qquad (5)$$

In this formulation (Fisher and Jaikumar 1981), variables x_{ij}^k indicates whether or not (v_i, v_j) is an edge of route k in the optimal solution, and similarly y_i^k specifies whether route k contains vertex v_i. Constraints (1) guarantee that vehicle capacity is not exceeded; constraints (2) ensure that each city is visited by exactly one vehicle and m vehicles visit the depot; constraints (3) ensure that every city is entered and left by the same vehicle; subtour elimination are specified by constraints (4); and equations (5) are the integrality constraints.

This formulation contains two well-known problems: the generalized assignment problem (GAP) specified by constraints (1), (2) and (5); and the traveling salesman problem (TSP). The TSP results when the y_i^k's are fixed to satisfy the GAP constraints for a given k, whereupon constraints (3) and (5) define a TSP for vehicle k. Specifically, the GAP is the subproblem responsible for assigning clients to vehicle routes, and the sequence in which clients are visited by a vehicle is determined by the TSP solution over each individual route.

Laporte, Nobert, and Desrochers (1985) show that a two-index formulation can be derived from a three-index formulation by aggregating all x_{ij}^k variable into a single variable x_{ij} which does not refer specifically to a particular route, but simply indicates whether or not an arc (v_i,v_j) is in the optimal solution. For the symmetrical case the value of the x_{ij} variable indicates the number of times (0,1, or 2) that the corresponding edge is used in the optimal solution. We will see later that this vehicle-flow formulation and the index-reduction concept proves useful in several parts of our scatter search model.

3.3. A Permutation-Based Formulation

In combinatorial optimization it is usually useful to distinguish two classes of problems according to whether the objective is to find (1) an optimal *combination* of discrete objects, or (2) an optimal *permutation* of objects. In the first case, a solution is a non-ordered set while in the second a solution is a ordered set (or a sequence). For instance, the 0-1 knapsack problem and the set covering problem are typical examples of combination problems where a solution can be interpreted to consist simply of a set of elements. Permutation problems on the other hand, are exemplified by job scheduling, traveling salesman, and vehicle routing problems where different arrangements of the same set represent different solutions (allowing for a sequence to be considered equivalent to its reverse ordering in the case of symmetric problems). The differentiation of these two types of combinatorial problems is particularly relevant because the specialized methods that apply to these problems, especially those methods based on graph theoretic representations, differ dramatically in the two cases. We will address to these features later on when describing the proposed scatter search template.

The VRP, which is the primary focus of this tutorial, seeks to determine an optimal permutation according to the following formulation:

$$\min \sum_{k=1}^{m} \sum_{i=0}^{n_k} c_{\pi_i^k, \pi_{i+1}^k}$$

$s.t.$

$$\sum_{i=1}^{n_k} q_{\pi_i^k} \le Q_k, \quad k = 1,\ldots,m \quad (1)$$

$$\sum_{k=1}^{m} n_k \le n \quad (2)$$

Here, a solution is defined by the permutation $\{\pi_1^1,\ldots,\pi_{n_1}^1,\ldots,\pi_1^m,\ldots,\pi_{n_m}^m\}$ of vertices $\{v_1,\ldots,v_n\}$. Vertex v_0 is implicitly represented by inserting it at the beginning and end of each subsequence k of the permutation (and may be expressed by "dummy elements" π_0^k and $\pi_{n_k+1}^k$). The permutation is partitioned into subsequences by indexes n_k and may be interpreted to consist of m separate routes, where the sequence of nodes on a given route identifies the order in which clients (vertices) are visited by a vehicle assigned to that route.

4. Scatter Search Template

Consider the following VRP instance with $n = 14$, $Q = 30$, $q = (q_i)(i = 0,\dots,14) = (0,6,20,8,9,10,8,7,5,5,4,3,4,7)$, $m \in [1,14]$, and a cost matrix $C = (c_{ij})(i, j = 1,\dots,14, i < j)$ defined as follows:

$$C = \begin{bmatrix}
0 & 7.85 & 10.82 & 8.18 & 9.71 & 4.53 & 3.13 & 5.11 & 6.40 & 7.54 & 6.40 & 9.30 & 8.51 & 4.61 & 7.96 \\
 & 0 & 14.40 & 4.36 & 4.19 & 9.41 & 9.39 & 2.77 & 1.57 & 1.12 & 2.82 & 1.73 & 8.98 & 12.04 & 15.74 \\
 & & 0 & 10.89 & 12.52 & 15.24 & 13.62 & 12.48 & 13.04 & 14.96 & 15.02 & 16.10 & 19.19 & 13.20 & 13.54 \\
 & & & & 1.84 & 11.47 & 10.80 & 4.40 & 3.78 & 5.34 & 6.48 & 5.76 & 12.59 & 12.79 & 15.88 \\
 & & & & 0 & 12.60 & 12.12 & 5.30 & 4.30 & 5.30 & 6.82 & 5.12 & 13.05 & 14.30 & 17.54 \\
 & & & & & 0 & 1.71 & 7.31 & 8.46 & 8.60 & 6.85 & 10.26 & 5.02 & 4.15 & 8.01 \\
 & & & & & & 0 & 6.96 & 8.22 & 8.75 & 7.14 & 10.48 & 6.60 & 2.94 & 6.91 \\
 & & & & & & & 0 & 1.30 & 2.72 & 2.60 & 4.34 & 8.27 & 9.44 & 13.04 \\
 & & & & & & & & 0 & 1.92 & 2.70 & 3.26 & 8.85 & 10.74 & 14.35 \\
 & & & & & & & & & 0 & 1.81 & 1.77 & 7.88 & 11.50 & 15.30 \\
 & & & & & & & & & & 0 & 3.40 & 6.24 & 9.98 & 13.86 \\
 & & & & & & & & & & & 0 & 8.95 & 13.26 & 17.07 \\
 & & & & & & & & & & & & 0 & 9.14 & 12.87 \\
 & & & & & & & & & & & & & 0 & 3.97 \\
 & & & & & & & & & & & & & & 0
\end{bmatrix}$$

A viewgraph of the clients' spatial distribution and the corresponding vertex coordinates that gave rise to the matrix *C* is provided in the appendix. We will use this special representation of the problem to illustrate different procedures used in the scatter search design.

Now, we can proceed to the illustration of the scatter search design using the defined VRP instance as an example of a permutation problem. We first describe in the context of the VRP each of the methods considered in the general scatter search template. Then, we present the various procedures integrated in the overall algorithm.

Initial Phase

Diversification Generation Method

Scatter search starts with an initial set of trial solutions which are required to be different, thus the use of a systematic procedure to generate these solutions is appropriate. Regarding the VRP as permutation problem we consider the generator of combinatorial objects described in Glover (1997), which operates as follows. A trial permutation *P* is used as a seed to generate subsequent permutations. Define the subsequence *P(h:s)* where *s* is a positive integer between *1* and *h*, so that $P(h : s) = (s, s + h, s + 2h, \dots, s + rh)$, where *r* is the largest nonnegative integer such that $s + rh \le n$. Finally, define the permutation *P(h)* for *h<n*, to be $P(h) = (P(h : h), P(h : h - 1), \dots, P(h : 1))$. In the VRP context we consider permutations in *n*-vectors where components are all vertices $v_i \in V \setminus \{v_0\}$. Consider for illustration $h = 4$, and a *seed* permutation $P = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14\}$ given by the sequence of clients ordered by their indices. The recursive application of *P(4:s)* for $s = 4,\dots,1$ results the subsequences, $P = \{4,8,12\}$, $P = \{3,7,11\}$, $P = \{2,6,10,14\}$, and $P = \{1,5,9,13\}$, hence $P(4) = \{4,8,12,3,7,11,2,6,10,14,1,5,9,13\}$. For illustration purposes we consider the set of initial vertices assignments A_h to vehicle routes derived from permutations *P(h)* $(h = 1,\dots,10)$ where each cluster of vertices in a route is obtained by successively assigning a vertex $v_i (i \in P(h))$ to a route R_{h_k} (initially *k=1*) until the cumulative quantity

$Q_k = \sum_{v_i \in R_{h_k}} q_i$ does not exceed Q with the insertion of a new vertex v_{k_j}. As soon as such a cutoff limit is attained a new assignment is created by incrementing k by one unit, and the process goes on until all vertices have been assigned. Table 2 shows the assignment A_4 derived from permutation $P(4)$, where shadowed cells indicate the Q_k value associated with the insertion of vertex v_i in a route R_k.

P(4)	*i*	*4*	*8*	*12*	*3*	*7*	*11*	*2*	*6*	*10*	*14*	*1*	*5*	*9*	*13*
R_k	*q_i*	*8*	*7*	*3*	*9*	*8*	*4*	*20*	*10*	*5*	*7*	*6*	*9*	*5*	*4*
R_1		*8*	*15*	*18*	*27*										
R_2						*8*	*12*								
R_3								*20*							
R_4										*5*	*12*	*18*	*27*		
R_5														*5*	*9*

Table 2. Sequential assignment of clients to vehicle routes

In fact, the result obtained can be viewed as a generalized assignment process which does not rely on the order in which clients are visited, though it ensures that all the initial solutions that can be created are feasible and different (since they derive from distinct permutations). Vehicle routes can thus be determined by using any traveling salesman algorithm on the subgraph defined by the previous assignments.

For illustration, we consider the standard *2-exchange* procedure (Lin 1965) to find a *2-optimal* TSP tour for each individual route. The 2-optimal procedure is a local search improvement method, hence an initial feasible TSP solution for each route R_k is required to start the method. A straightforward method to create this solution can be obtained by successively linking vertices in the order they appear in the permutation and attaching the initial and ending vertices to the depot. Then, the method proceeds by replacing two edges $(v_i, \overline{v}_i)$ and $(v_j, \overline{v}_j)$ by two others (v_i, v_j) and $(\overline{v}_i, \overline{v}_j)$ where $\overline{v}$ denotes the successor of v in a given orientation of the route. Hence, in order to maintain the feasible orientation of the route, the subpath $(\overline{v}_i, \ldots, v_j)$ needs to be reversed, so that the subpath $(v_i, \overline{v}_i, \ldots, v_j, \overline{v}_j)$ becomes $(v_i, v_j, \ldots, \overline{v}_i, \overline{v}_j)$. For convenience, denote the c_{ij} values by $c(v_i, v_j)$ so that the solution cost change produced by a 2-exchange move can be expressed as $\Delta_{ij} = c(v_i, v_j) + c(\overline{v}_i, \overline{v}_j) - c(v_i, \overline{v}_i) - c(v_j, \overline{v}_j)$. A *2-optimal* (or *2-opt*) solution is obtained by iteratively applying 2-exchange moves until no possible move yield a negative Δ value.

As the purpose of the diversification generation method is to generate diverse solutions rather than high quality solutions, it is convenient to make the method fast, therefore we have adopted a *first-improvement* (as opposed to a *best-improvement*) strategy for the TSP heuristic.

Table 3 shows the four iterations carried out to create the solution S_4. Specifically, the first column shows the initial trial solution T_4 created from the assignment A_4 (defined in Table 2) and the final solution S_4. Figures within parenthesis indicate the iteration number. The second column shows vertices v_i, v_j that define the move from one iteration to another. (Note that a 2-exchange move is completely identified by vertices v_i, v_j since the two other vertices are necessarily their successors, respectively $\overline{v}_i$ and

$\overline{v}_j$.) The remaining columns show the routes in each solution, the solution cost, and the value associated with the move that has led to the solution shown in the next row. Boldfaced numbers indicate the changes carried out by the corresponding move. The result of these changes are illustrated in Figure 2, which depicts the initial trial solution T_4 and the final solution S_4.

Solution	Move	R_1	R_2	R_3	R_4	R_5	Cost	Δ_{ij}
$T_4^{(1)}$	v_0,v_{12}	*4 8 12 3*	*7 11*	*2 6*	*10 14 1 5*	*9 13*	*163.54*	*-11.94*
$T_4^{(2)}$	v_{10},v_1	***12 8 4*** *3*	*7 11*	*2 6*	*10 14 1 5*	*9 13*	*151.60*	*-12.45*
$T_4^{(3)}$	v_1,v_5	*12 8 4 3*	*7 11*	*2 6*	*10* ***1 14*** *5*	*9 13*	*139.15*	*-2.90*
$T_4^{(4)}$	v_0,v_1	*12 8 4 3*	*7 11*	*2 6*	*10 1* ***5 14***	*9 13*	*136.25*	*-1.10*
S_4		*12 8 4 3*	*7 11*	*2 6*	***1 10*** *5 14*	*9 13*	*135.15*	

Table 3. Application of 2-opt procedure to T_4 trial solution

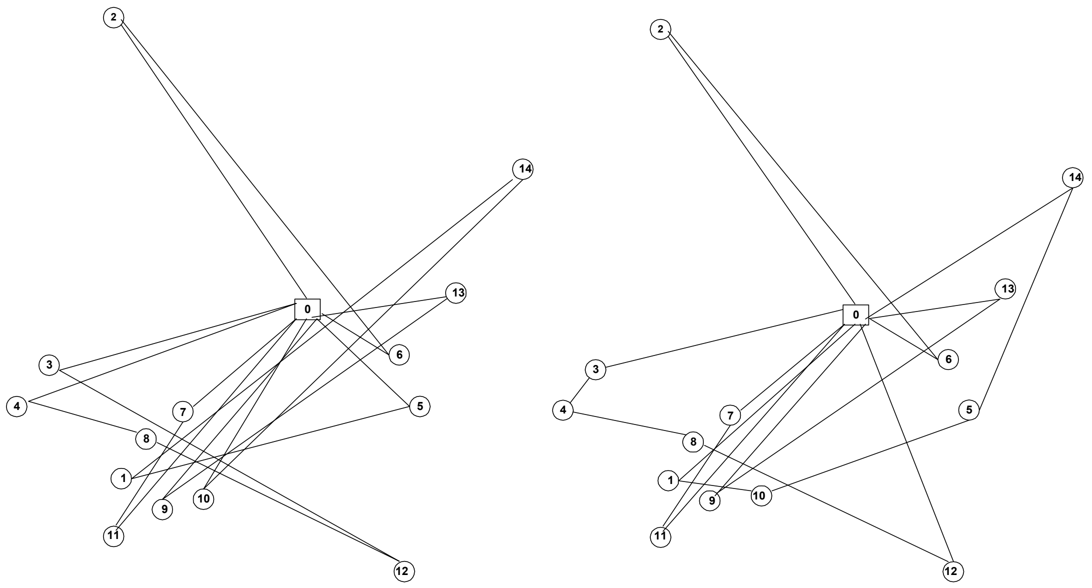


Figure 2. Solutions T_4 (left-hand side) and S_4 (right-hand side)

The shape of the solutions in the figure clearly show that routes R_1 and R_4 were significantly improved. Repeating the process for the rest of the permutations we obtain the ten solutions produced by the diversification generation method as illustrated in Table 4.

Solution	Routes	Cost
T1	**0** 1 2 **0** 3 4 5 **0** 6 7 8 9 **0** 10 11 12 13 14 **0**	120,90
S1	**0** 1 2 **0** 3 4 5 **0** 6 9 8 7 **0** 11 10 12 14 13 **0**	120,83
T2	**0** 2 4 **0** 6 8 10 12 **0** 14 1 3 **0** 5 7 9 11 13 **0**	132,28
S2	**0** 2 4 **0** 6 12 10 8 **0** 14 3 1 **0** 7 11 9 5 13 **0**	122,74
T3	**0** 3 6 9 12 **0** 2 5 **0** 8 11 14 1 **0** 4 7 10 13 **0**	157,24
S3	**0** 6 12 9 3 **0** 2 5 **0** 8 1 11 14 **0** 7 4 10 13 **0**	128,27
T4	**0** 4 8 12 3 **0** 7 11 **0** 2 6 **0** 10 14 1 5 **0** 9 13 **0**	163,54
S4	**0** 12 8 4 3 **0** 7 11 **0** 2 6 **0** 1 10 5 14 **0** 9 13 **0**	135,15
T5	**0** 5 10 4 9 **0** 14 3 8 13 **0** 2 7 **0** 12 1 6 11 **0**	149,08
S5	**0** 5 10 9 4 **0** 8 3 14 13 **0** 2 7 **0** 1 11 12 6 **0**	119,50
T6	**0** 6 12 5 11 **0** 4 10 3 9 **0** 2 8 **0** 14 1 7 13 **0**	140,97
S6	**0** 6 5 12 11 **0** 3 4 9 10 **0** 2 8 **0** 14 13 1 7 **0**	113,74
T7	**0** 7 14 6 13 **0** 5 12 4 11 **0** 3 10 **0** 2 9 **0** 1 8 **0**	139,83
S7	**0** 7 6 13 14 **0** 5 12 11 4 **0** 3 10 **0** 2 9 **0** 1 8 **0**	130,47
T8	**0** 8 7 6 **0** 14 5 13 4 **0** 12 3 11 **0** 2 10 **0** 1 9 **0**	146,83
S8	**0** 7 8 6 **0** 14 13 5 4 **0** 12 11 3 **0** 2 10 **0** 1 9 **0**	136,29
T9	**0** 9 8 7 6 **0** 5 14 4 13 **0** 3 12 **0** 2 11 1 **0** 10 **0**	148,42
S9	**0** 7 8 9 6 **0** 14 13 5 4 **0** 3 12 **0** 2 11 1 **0** 10 **0**	137,16
T10	**0** 10 9 8 7 **0** 6 5 4 **0** 14 3 13 **0** 2 12 1 **0** 11 **0**	148,65
S10	**0** 10 9 8 7 **0** 6 5 4 **0** 3 14 13 **0** 2 1 12 **0** 11 **0**	135,92

Table 4. Trial solutions (T_k) and final solutions (C_k) produced by the diversification generation method

Improvement Method

The improvement method used in the initial phase may or may not be the same method used in the scatter search phase. This decision usually depends on the context and on the search strategy one might want to implement. For the purpose of this tutorial we use the same procedure in both phases, though in the scatter search phase the method is required to deal with some additional features.

We consider an iterative improvement method based on local search. The method is designed to directly operate on the problem graph using a very straightforward *neighborhood structure* consisting of removing a vertex from its current position and inserting it between two other vertices. Specifically, denoting respectively by $\underline{v}$ and $\overline{v}$ the predecessor and successor of a vertex v, the insertion of a vertex v_i between vertices v_p and v_q operates by inserting edges $(\underline{v}_i,\overline{v}_i)$, (v_p,v_i), (v_i,v_q), and by removing edges $(\underline{v}_i,v_i)$, $(v_i,\overline{v}_i)$ and $(v_p,\overline{v}_p)$, where $v_q=\overline{v}_p$. Hence, the solution cost change is given by $\Delta_{ip}=c(\underline{v}_i,\overline{v}_i)+c(v_p,v_i)+c(v_i,\overline{v}_p)-c(\underline{v}_i,v_i)-c(v_i,\overline{v}_i)-c(v_p,\overline{v}_p)$. Applying the procedure on each S_i solution (in Table 4) we obtain the corresponding improved solutions reported in Table 5. (A graphic illustration of the method is shown in the scatter search phase where some additional features of the method are included.)

Solution	Routes	Cost
S1	**0** 2 **0** 3 4 5 **0** 6 9 8 7 **0** 1 11 10 12 14 13 **0**	109,67
S2	**0** 2 **0** 3 4 1 8 **0** 6 5 13 14 **0** 7 11 9 10 12 **0**	92,51
S3	**0** 6 5 12 9 **0** 2 **0** 8 1 11 13 14 **0** 7 3 4 10 **0**	106,37
S4	**0** 8 4 3 **0** 7 1 11 9 10 **0** 2 **0** 12 5 6 **0** 13 14 **0**	96,84
S5	**0** 5 12 10 9 4 **0** 7 3 14 13 **0** 2 **0** 8 1 11 6 **0**	111,52
S6	**0** 6 5 **0** 3 4 9 10 12 **0** 2 8 **0** 14 13 11 1 7 **0**	106,30
S7	**0** 6 5 13 14 **0** 3 4 11 10 12 **0** 2 **0** 9 1 8 7 **0**	92,48
S8	**0** 7 8 1 9 **0** 14 13 5 6 **0** 12 10 11 4 3 **0** 2 **0**	92,48
S9	**0** 7 8 **0** 14 13 5 6 **0** 3 4 9 10 12 **0** 2 11 1 **0**	102,11
S10	**0** 10 9 11 8 7 **0** 6 5 **0** 4 3 14 13 **0** 2 1 12 **0**	107,74

Table 5. Improved Solutions

Reference Set Update Method

This method is used to create and maintain a set of reference solutions. As in any evolutionary (population-based) method, a set of solutions (population of individuals) containing high evaluation combinations of attributes replaces less promising solutions at each iteration (generation) of the method in order to enhance the quality of the population. In genetic algorithms, for example, the updating process relies on randomized selection rules which select individuals according to their relative fitness value. In scatter search the updating process relies on the use of memory and is confined to maintain a good balance between *intensification* and *diversification* of the solution process. In advanced forms of scatter search reference solutions are selected based on the use of memory which operates by reference to different dimensions as defined in tabu search. Depending on the context and the search strategy, different types of memory can be used. The way they are integrated to achieve both intensification and diversification is generically called *adaptive memory programming*. (See Glover and Laguna 1997 for a detailed explanation of various forms and uses of memory within search processes.) For the purpose of this tutorial we use a simple rule to update the set of reference solutions, where intensification is achieved by the selection of high-quality solutions (in terms of the objective function value) and diversification is basically induced by including diverse solutions from the current candidate set *CS*. Thus the reference set *RS* can be defined by two distinct subsets *B* and *D*, representing respectively the subsets of high-quality and diverse solutions, hence $RS = B \cup D$. Also, in the context of our example the terms “highest evaluated solution” and “best solution” are interchangeable and refer to the solution that best fits the evaluation criterion under consideration.

Consider the candidate set $CS = \{S_1,\ldots,S_{10}\}$ defined by the improved solutions, and a reference set of size $|RS| = 6$ where $|B| = |D| = 3$[1]. Before proceeding to the creation of the reference set we first need to eliminate possible repeated solutions in the current set of

[1] The cardinality of *B* and *D* does not need to be identical and can vary during the search. For instance, relatively higher values of *B* (*D*) can be appropriate during a phase that is more strongly characterized by an intensification (diversification) emphasis. Also different schemes can be chosen to implement these variations. A dynamic variation of these values can be implemented by a perturbation scheme conducted strategically rather than randomly. For example, a strategic oscillation can be implemented by using critical event memory as an indicator of the order of magnitude of the relative variations. Note that since the cardinality of subsets *B* and *D* are complementary in relation to the *RS* size and the decision can be uniquely based on the variation of either one.

trial solutions of Table 5. We can see that *S₇* and *S₈* represent the same solution, so we drop *S₈* from the solution candidate set, making $CS = CS \setminus \{S_8\}$. Now, to create *RS* we start by selecting the three best solutions S_7, S_2 *and* S_4 in *CS* to generate *B*, then the set *D* of diverse solutions is generated by successively selecting the solution which mostly differs from the ones currently belonging to *RS*. As a diversity measure we define $d_{ij} = |(S_i \cup S_j) \setminus (S_i \cap S_j)|$ as the distance between solutions S_i and S_j, which gives the number of edges by which the two solutions differ from each other. For example, solution S_3 contains *6* edges *(1,8)*, *(3,7)*, *(4,10)*, *(9,0)*, *(9,12)*, *(11,13)*, which are not in solution S_4, and solution S_4 has *7* edges *(1,7)*, *(3,0)*, *(4,8)*, *(9,10)*, *(9,11)*, *(12,0)*, *(13,0)*, which are not in the solution S_3. Hence the distance between the two solutions is *13*. Table 6 shows d_{ij} values for each pair of solutions $S_i \in RS$ and $S_j \in CS$.

Candidate solutions are included in *RS* according to the *Maxmin* criterion which maximizes the minimum distance of each candidate solution to all the solutions currently in the reference set. The method starts with $RS = B$ and at each step *RS* is extended with a solution $S_j \in CS$, to be $RS = RS \cup \{S_j\}$, and consequently *CS* is reduced to $CS = CS \setminus \{S_j\}$. Then the distance of solution S_j to every solution currently in the reference set is computed to make possible the selection of a new candidate solution according to the *Maxmin* criterion. More formally, the selection of a candidate solution is given by $S_j = arg\,max \min_{i=1,\dots,|RS|} \{d_{ij} : j = 1,\dots,|CS|\}$. The process is repeated until $|RS|$ is achieved. In the table, boldfaced figures represent the maxmin values obtained at each step of the method.

	Candidate solutions (*CS*)					
Reference set (*RS*)	S_1	S_3	S_5	S_6	S_9	S_{10}
S_7	*16*	*16*	*22*	*16*	*12*	*20*
S_2	*20*	*16*	*18*	*12*	*10*	*18*
S_4	*19*	*13*	*17*	*11*	*13*	*15*
	16	*13*	***17***	*11*		*15*
S_5	*18*	*16*		*16*	*18*	*18*
	16	*13*		*11*		*15*
S_1		*22*		*18*	*16*	*18*
		13		*11*		***15***

Table 6. Distances between solutions

This results in an initial reference set formed by solutions S_7, S_2, S_4, S_5, S_1, S_{10}. For convenience, we reorder the solution indexes by setting $RS = \{S_1, S_2, S_3, S_4, S_5, S_6\}$ as illustrated in Table 7.

Solution	Routes	Cost
S1	**0** 6 5 13 14 **0** 3 4 11 10 12 **0** 2 **0** 9 1 8 7 **0**	92,48
S2	**0** 2 **0** 3 4 1 8 **0** 6 5 13 14 **0** 7 11 9 10 12 **0**	92,51
S3	**0** 8 4 3 **0** 7 1 11 9 10 **0** 2 **0** 12 5 6 **0** 13 14 **0**	96,84
S4	**0** 5 12 10 9 4 **0** 7 3 14 13 **0** 2 **0** 8 1 11 6 **0**	111,52
S5	**0** 2 **0** 3 4 5 **0** 6 9 8 7 **0** 1 11 10 12 14 13 **0**	109,67
S6	**0** 10 9 11 8 7 **0** 6 5 **0** 4 3 14 13 **0** 2 1 12 **0**	107,74

Table 7. Initial reference set.

It is important to note that a balance between intensification and diversification is achieved by an evaluation criterion that causes the highest-evaluated solutions considered for the reference to include qualities other than a “good” (small) objective function value. Solution S_9 with a cost of *102.11* is by-passed in favor of other solutions that will add more diversity to the set. As indicated in Table 6, the distance between S_9 to solutions in the reference set are relatively lower, making this solution unattractive from the standpoint of diversification.

Scatter Search Phase

Subset Generation Method

This method consists of generating subsets of reference solutions to create structured combinations in the next step. The method is typically designed to organize subsets of solutions to cover different promising regions of the solution space. In a spatial representation, the convex-hull of each subset delimits the solution space in subregions containing all possible convex combinations of solutions in the subset. In order to achieve a suitable intensification and diversification of the solution space, three types of subsets are required to be organized:

1) subsets containing only solutions in *B*,
2) subsets with only solutions in *D*, and
3) subsets mixing in solutions in *B* and *D* in different proportions.

Subsets defined by solutions of type 1 are conceived to intensify the search in regions of high-quality solutions while subsets of type 2 are created to diversify the search to unexplored regions. Finally, subsets of type 3 integrate both high-quality and diverse solutions with the aim of exploiting solutions across these two types of subregions.

Again, adaptive memory can be used to appropriately define combined rules for clustering elements in the various types of subsets. This has the advantage of incorporating additional information about the search space and problem context.

Since the use of sophisticated memory features is beyond the scope of this study, we consider a simpler yet systematic procedure to generate the following types of subsets:

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

Table 8 shows the subset generated using the current reference set.

Type	Subsets
1	*(S1, S2), (S1, S3), (S1, S4), (S1, S5), (S1, S6), (S2, S3), (S2, S4), (S2, S5), (S2, S6), (S3, S4), (S3, S5), (S3, S6), (S4, S5), (S4, S6), (S5, S6)*
2	*(S1, S2, S3), (S1, S2, S4), (S1, S2, S5), (S1, S2, S6), (S1, S3, S4), (S1, S3, S5), (S1, S3, S6), (S1, S4, S5),*

	(S1, S4, S6), (S1, S5, S6)
3	*(S1, S2, S3, S4), (S1, S2, S3, S5), (S1, S2, S3, S6), (S1, S2, S4, S5), (S1, S2, S4, S6), (S1, S2, S5, S6);*
4	*(S1, S2, S3, S5, S6), (S1, S2, S3, S4, S5, S6)*

Table 8. Subset generation

Subsets of type 1 are the fifteen possible combinations of two solutions. For type 2 subsets, we start by adding solution S_3 to the subset (S_1, S_2). Then since adding solution S_2 to the subset (S_2, S_3) leads to a repeated subset, the procedure jumps to the next subset, adding S_2 to the subset (S_1, S_4), and so forth. Type 3 subsets are created in a similar way, now starting by adding solution S_4 to the subset (S_1, S_2, S_3). Finally, subsets of type 3 consist of successively adding solutions S_5 and S_4 to subsets (S_1,S_2,S_3,S_6) and (S_1,S_2,S_3,S_6,S_5).

Solution Combination Method

This method is designed to explore subregions within the convex-hull of the reference set. We consider solutions encoded as vectors of variables x_{ij} representing edges (v_i,v_j). New solutions are generated by weighted linear combinations which are structured by the subsets defined in the last step. In order to restrict the number of solutions only one solution is generated in each subset by a convex linear combination defined as follows. Let E be a subset defined in RS, $|E| = r$, and let $H(E)$ denote the convex-hull of E. We generate solutions $S \in H(E)$ represented as

$$S = \sum_{t=1}^{r} \lambda_t S_t$$

$$\sum_{t=1}^{r} \lambda_t = 1$$

$$\lambda_t \geq 0 \qquad j = 1,\ldots,r$$

where the multiplier λ_t represents the weight assigned to solution S_t. We compute these multipliers by

$$\lambda_t = \frac{\frac{1}{C(S_t)}}{\sum_{t=1}^{r} \frac{1}{C(S_t)}}$$

so that the better (lower cost) solutions receive higher weight than less attractive (higher cost) solutions. Then, we calculate the score of each variable x_{ij} relative to the solutions in E by computing

$$score(x_{ij}) = \sum_{t=1}^{r} (\lambda_t x_{ij}^t)$$

where x_{ij}^t means that x_{ij} is an edge in the solution S_t. Finally as variables are required to be binary, the value is obtained by rounding its score to give $x_{ij} = \lfloor score(x_{ij}) + .5 \rfloor$. The computation of the value for each variable in E results in the linear combination of the solutions in E. Table 9 shows the computation of the linear combination of solutions S_1, *S₂*, *S₃*, and *S₄* in the initial reference set illustrated in Table 8. For the sake of simplicity only edges that appear in at least one solution are represented in the table, since the remaining variables correspond to $x_{ij} = 0$.

Solution	**S_1**	**S_2**	**S_3**	**S_4**		
λ_t	**0,2643**	**0,2642**	**0,2524**	**0,2191**		
Edges					***score(x_{ij})***	***x_{ij}***
(1,4)		0,2642			0,2642	0
(1,7)			0,2524		0,2524	0
(1,8)	0,2643	0,2642		0,2191	0,7476	1
(1,9)	0,2643				0,2643	0
(1,11)			0,2524	0,2191	0,4715	0
(2,0)	0,2643	0,2642	0,2524	0,2191	1,0000	1
(3,0)	0,2643	0,2642	0,2524		0,7809	1
(3,4)	0,2643	0,2642	0,2524		0,7809	1
(3,7)				0,2191	0,2191	0
(3,14)				0,2191	0,2191	0
(4,0)				0,2191	0,2191	0
(4,8)			0,2524		0,2524	0
(4,9)				0,2191	0,2191	0
(4,11)	0,2643				0,2643	0
(5,0)				0,2191	0,2191	0
(5,6)	0,2643	0,2642	0,2524		0,7809	1
(5,12)			0,2524	0,2191	0,4715	0
(5,13)	0,2643	0,2642			0,5285	1
(6,0)	0,2643	0,2642	0,2524	0,2191	1,0000	1
(6,11)				0,2191	0,2191	0
(7,0)	0,2643	0,2642	0,2524	0,2191	1,0000	1
(7,8)	0,2643				0,2643	0
(7,11)		0,2642			0,2642	0
(8,0)		0,2642	0,2524	0,2191	0,7357	1
(9,0)	0,2643				0,2643	0
(9,10)		0,2642	0,2524	0,2191	0,7357	1
(9,11)		0,2642	0,2524		0,5166	1
(10,0)			0,2524		0,2524	0
(10,11)	0,2643				0,2643	0
(10,12)	0,2643	0,2642	0,2524	0,2191	1,0000	1
(12,0)	0,2643	0,2642			0,5285	1
(13,0)			0,2524	0,2191	0,4715	0
(13,14)	0,2643	0,2642	0,2524	0,2191	1,0000	1
(14,0)	0,2643	0,2642			0,5285	1

Table 9. Linear combinations of solution vectors

It is important to observe the effect of this weighting on the resulting solutions. As previously intimated, the least cost solution *S₁* has the largest weight in the combination. Also, note that both edges *(5,12)* and *(5,13)* appear in two solutions, though only the variable x_{513} is considered for the resulting solution.

A new solution can now be created using the edges associated with variables $x_{ij} = 1$. Nevertheless, the set of these edges does not necessarily (and usually doesn't) represent a feasible graph structure for a VRP solution. Instead, it produces a subgraph containing vertices with a degree different than two. Such subgraphs can be viewed as fragments of solutions (or partial routes). For example, the previous linear combination produced the following solution fragments: *(1,8,0)*, *(2,0)*, *(0,3,4)* *(0,6,5,13,14,0)*, *(0,11,9,10,12,0)*, and *(7,0)*. To create a feasible solution subgraph we can simply link vertices *1*, *2*, *4*, and *7* (which have a degree equal to 1) directly to the depot. This has the advantage that the algorithm always deals with feasible structures. However, it is also possible that the subgraph resulting from a linear combination contains vertices of degree greater than two. In these cases, a very straightforward technique consists of successively dropping edges with the smallest scores, from among those incident at these vertices, until the degree of each vertex becomes equal to two. By doing so, the subgraph obtained will be either feasible or fall into the case where some of the vertices have degree 1, which can be handled as already indicated.

Improvement Method

Even though the solution combination method creates feasible subgraphs for the VRP, the solution may still not be feasible in relation to the other problem constraints. Therefore, the improvement method must be able to deal with infeasible solutions. As already noted, the solution combination method has generated 33 new solutions (one per each subset). Let $C_1, \ldots, C_{33}$ denote these solutions indexed by the order they appear in Table 8. In the whole set, combinations C_{26}, …, C_{31} and C_{33} repeat previously generated solutions, so we first drop these solutions before applying the improvement method. The relatively high rate of repeated solutions results from the fact that the example deals with a very small instance. However, some number of repeated solutions may still be expected to appear when solving more realistic problems. Thus, it is valuable to have a fast procedure to identify repeated solutions which can be achieved in this context by using an appropriate *hash function*.

Table 10 shows an example of the improvement method applied to solution C_{12} where the local optimum C^*_{12} is found after 3 iterations. This is the same improvement method used in the initial phase of the scatter search algorithm, though the illustration shows how the method deals with infeasible solutions.

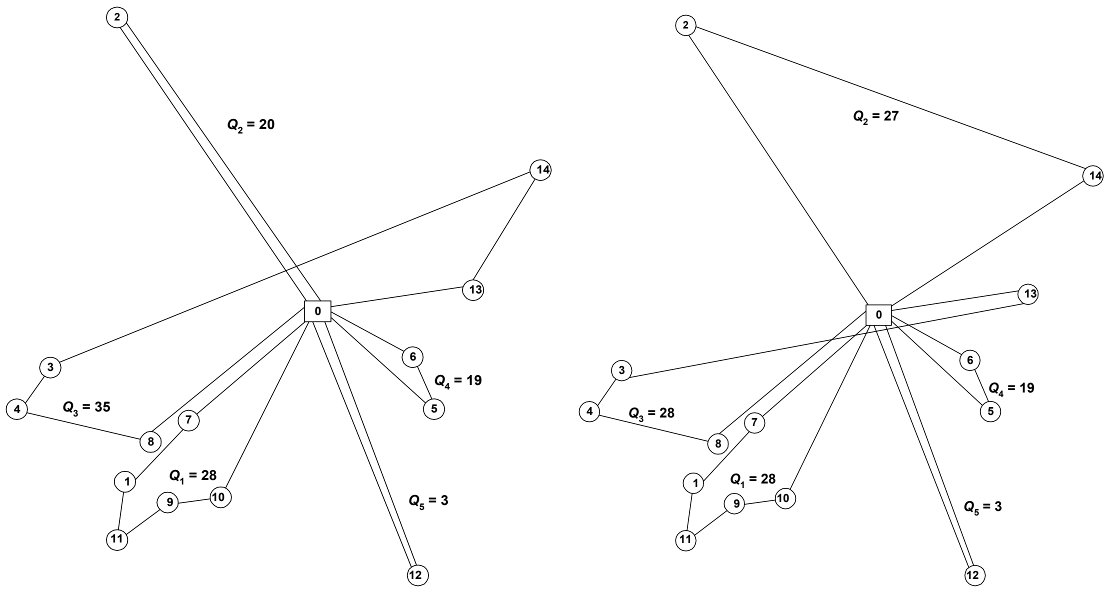
The method works in two stages. The first stage is concerned with making the solution feasible while choosing the most favorable move (relative to the objective function cost), and the second stage is the improvement process that operates only on feasible solutions. In general, several variants can be used to deal with infeasible solutions. These techniques are usually based on varying certain penalty factors associated with the problem constraints. Some constraints are potentially “less damaging” when violated than others, and usually the choice of penalty values should take this into consideration. Also, the way these penalties are modified can make the search more or less aggressive for moving into the feasible region. High penalty values are usually employed for an intensification strategy, and lower values for a diversification approach that allows the search keep going longer in the infeasible region. An interplay between intensification and diversification can be carried out by changing penalties according to certain patterns. For example, such patterns can be based on critical event memory within a strategic oscillation approach as suggested in tabu search. As previously noted, different balances between intensification and diversification of the search can also be controlled in the construction of the reference set, and in the solution combination method (as by considering non-convex linear combinations to extrapolate to solutions outside the convex hull of a subset). Our illustrative method, however, doesn’t use a penalty factor but rather identifies the most violated route and makes the best (cost reducing) move that consists of removing a vertex from this route and feasibly inserting it in another route. It is possible that an additional route will be created if such a move is not possible or if all routes are over their capacity.

Solution Iteration	***k***	R_k	Q_k	$C(R_k)$	$C(C_{12})$
Initial Solution C_{12}	1	0 7 1 11 9 10 0	28	19,58	
	2	0 2 0	20	21,63	
	3	0 13 14 3 4 8 0	**35**	37,00	
	4	0 5 6 0	19	9,36	
	5	0 12 0	3	17,02	104,60†
Iteration 1		Drop vertex 14 from R_3 and insert it into R_2			
	1	0 7 1 11 9 10 0	28	19,58	
	2	0 **2 14 0**	27	32,32	
	3	0 **13 3** 4 8 0	28	29,94	
	4	0 5 6 0	19	9,36	
	5	0 12 0	3	17,02	108,23
Iteration 2		Drop vertex 12 from R_5 and insert it into R_4			
	1	0 7 1 11 9 10 0	28	19,58	
	2	0 2 14 0	27	32,32	
	3	0 13 3 4 8 0	28	29,94	
	4	0 **12 5** 6 0	22	18,37	100,21
Iteration 3		Drop vertex 13 from R_3 and insert it into R_4:			
Final Solution C^*_{12}	1	0 7 1 11 9 10 0	28	19,58	
	2	0 2 14 0	27	32,32	
	3	**0 3** 4 8 0	24	20,72	
	4	0 12 5 **6 13** 0	26	22,79	95,41

† infeasible solution, $Q_3 = 35 \geq Q = 30$

Table 10. Improving Method

A graphic representation of the procedure is illustrated in Figure 3.



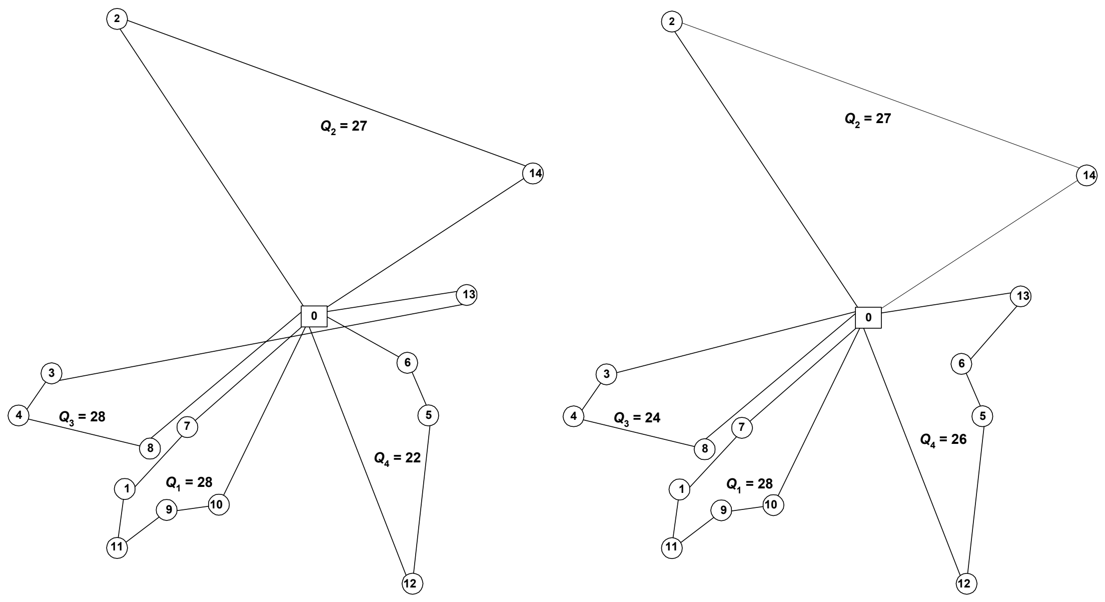


Figure 3. Graphical illustration of the improvement method

In our example, we apply the improving method to all the remaining (non-repeated) solutions to obtain a set of improved solutions. The best solution results from the improvement of C_{16} associated with a combination involving subset (S_1,S_2,S_3) . C_{16} is the solution *(0,1,8,0,2,0,3,4,0,6,5,13,14,0,7,0,12,10,9,11,0)* with cost $C(C_{16}) = 115.94$, which is transformed by the improvement method into the solution S_{16} with cost $C(S_{16}) = 91.01$. We have also solved the problem using an exact method (which requires substantially greater computation time, even for this simple example, an verify that S_{16} is in fact the optimal solution. This solution is illustrated in Figure 4.

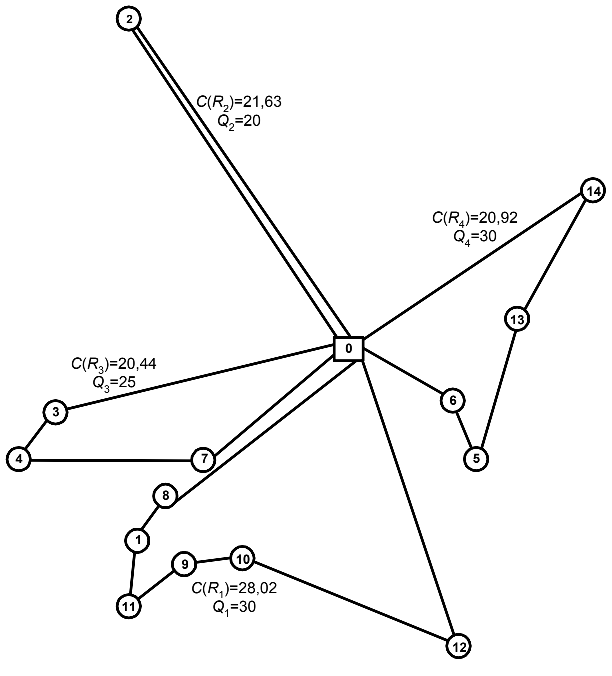


Figure 4. Final solution

In the absence of external confirmation that an optimal solution has been found, the method could continue to iterate in the scatter search phase by updating the reference set in the same way as in the initial phase. In this succeeding phase, however, the current reference set and the solutions produced by the improving method are all considered together before determining the new *B* and *D* sets that will form the updated reference set for the next iteration. The method stops when no elements in the current reference are replaced (i.e. when the reference set has not changed in two successive iterations).

5. Summary and conclusion

We have developed a scatter search template for circuit-based graph problems and have shown its application to the classical vehicle routing problem under capacity restrictions.

Our intention was not to provide a computational exploration of scatter search. Rather, the numerous recent studies demonstrating the computational and practical efficacy of the approach provided one of the key motivations for the present paper. Another primary motivation for this tutorial paper was the fact that, in spite of its growing applications, scatter search has not yet been described in a step by step manner in application to the routing context, accompanied by a detailed numerical illustration. For this reason, the access to applying the method to routing problems is undoubtedly somewhat less than it might otherwise be, and the amount of effort to launch a new scatter search study in this domain is also accordingly somewhat greater than would be necessary, due to the need to digest the elements of the method apart from an illustrated presentation of their nature and relevance.

The illustrative example used to demonstrate the scatter search provides a new evolutionary approach and a general framework for designing scatter search algorithms for vehicle routing. Moreover, because the problem constraints are handled separately from the solution generation procedures, and are therefore independent of the problem context, our scatter search design can be directly used to solve other classes of vehicle routing problems by applying any domain-specific (local search) heuristic that is able to start from infeasible solutions.

Finally, our illustrative approach affords various possibilities for using adaptive memory programming as a basis for creating effective scatter search algorithms for solving practical instances. In particular, we show the relevance of using memory to dynamically modify weights in the solution and to introduce an appropriate balance between intensification and diversification of the search.

References

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

Campos, V., F. Glover, M. Laguna and Martí (1999) "An Experimental Evaluation of a Scatter Search for the Linear Ordering Problem," Research Report HCES-06-99, Hearin Center for Enterprise Science, School of Business Administration, University of Mississippi.

Cung, V-D., T. Mautor, P. Michelon, A. Tavares (1996) "Scatter Search for the Quadratic Assignment Problem," Proceedings of the 1996 *IEEE International Conference on Evolutionary Computation*, pp 165–169.

Fleurent, C., F. Glover, P. Michelon and Z. Valli (1996) "A Scatter Search Approach for Unconstrained Continuous Optimization," Proceedings of the 1996 *IEEE International Conference on Evolutionary Computation*, pp 643–648.

Gendreau, M., A. Hertz, G. Laporte (1994) "Tabu Search Heuristic for the Vehicle Routing Problem," *Management Science*, Vol. 40, pp 1276-1290.

Glover, F. (1965) "A Mutiphase-Dual Algorithm for Zero-One Integer Programming Problem," *Operations Research*, Vol. 13, pp 879-919.

Glover, F. (1977) "Heuristics for Integer Programming Using Surrogate Constraints," *Decision Sciences*, Vol 8, No 1, pp 156-166.

Glover, F. (1994) "Genetic Algorithms and Scatter Search: Unsuspected Potentials," *Statistics and Computing*, Vol. 4, pp 131-140.

Glover, F. (1997) "A Template for Scatter Search and Path Relinking," in Lecture Notes in Computer Science, 1363, J.K. Hao, E. Lutton, E. Ronald, M. Schoenauer, D. Snyers (Eds.), pp 13-54.

Glover, F., A. Løkketangen and D. Woodruff (1999) "Scatter Search to Generate Diverse MIP Solutions," Research Report, University of Colorado, Boulder.

Glover F., M. Laguna (1997) "Tabu Search," Kluwer Academic Publishers, Boston, MA.

Glover F., M. Laguna, R. Martí (2001) "Scatter Search," to appear in Theory and Applications of Evolutionary Computation: Recent Trends, A. Ghosh and S. Tsutsui (Eds.), Springer-Verlag.

Johnson, D.S., L. A. McGeoch, (1997) "The Traveling Salesman Problem: A Case Study in Local Optimization," in Local Search in Combinatorial Optimization, E. H. L. Aarts and J. K. Lenstra (Eds), John-Wiley and Sons, Ltd., pp. 215-310.

Kelly, J., B. Rangaswamy and J. Xu (1996) "A Scatter Search-Based Learning Algorithm for Neural Network Training," *Journal of Heuristic*s, Vol. 2, pp. 129-146.

Laguna, M. (2001) “Scatter Search,” to appear in Handbook of Applied Optimzation, P.M. Pardalos and M.G.C. Resende (Eds), Oxford Academic Press.

Laguna, M., H. Lourenço and R. Martí (2000) “Assigning Proctors to Exams with Scatter Search,” in Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research, M. Laguna and J. L. González-Velarde (Eds.), Kluwer Academic Publishers, pp. 215-227.

Laguna, M. and R. Martí (2000) “Experimental Testing of Advanced Scatter Search Designs for Global Optimization of Multimodal Functions,” Working paper, University of Colorado, Boulder.

Laporte, G. and I.H. Osman (1995) “Routing problems: A bibliography,” *Annals of Operations Research*, Vol. 61, pp 227–262.

Rego, C. (1996) “Relaxed Tours and Path Ejections for the Traveling Salesman Problem,” in Tabu Search Methods for Optimization, *European Journal of Operational Research*, 106, pp 522-538.

Rego, C. (1998) “A Subpath Ejection Method for the Vehicle Routing Problem, Management Science,” Vol. 44, No 10, pp 1447-1459.

Rego, C. (2000) “Node Ejection Chains for the Vehicle Routing Problem: Sequential and Parallel Algorithms,” *Parallel Computing*, forthcoming.

Rochat Y., E. Taillard (1995) “Probabilistic Diversification and Intensification in Local Search for Vehicle Routing,” *Journal of Heuristics*, Vol. 1, pp 147-167.

Xu, J., S. Chiu and F. Glover (2000) “Tabu Search and Evolutionary Scatter Search for ‘Tree-Star’ Network Problems, with Applications to Leased-Line Network Design,” to appear in *Telecommunications Optimization*, David Corne (Ed.) Wiley.

Appendix

Table 1 shows the data of the problem used along this tutorial. The depot is represented by index *0*, therefore q_0 is set to be zero.

v_i	x_i	y_i	q_i
0	11,5	14,4	0
1	5,8	9,0	6
2	5,5	23,4	20
3	3,5	12,7	9
4	2,3	11,3	8
5	14,9	11,4	9
6	14,3	13,0	10
7	7,6	11,1	8
8	6,5	10,4	7
9	6,8	8,5	5
10	8,6	8,7	5
11	5,5	7,3	4
12	14,4	6,4	3
13	16,0	15,4	4
14	18,2	18,7	7

Table 1. Problem data for the VRP instance

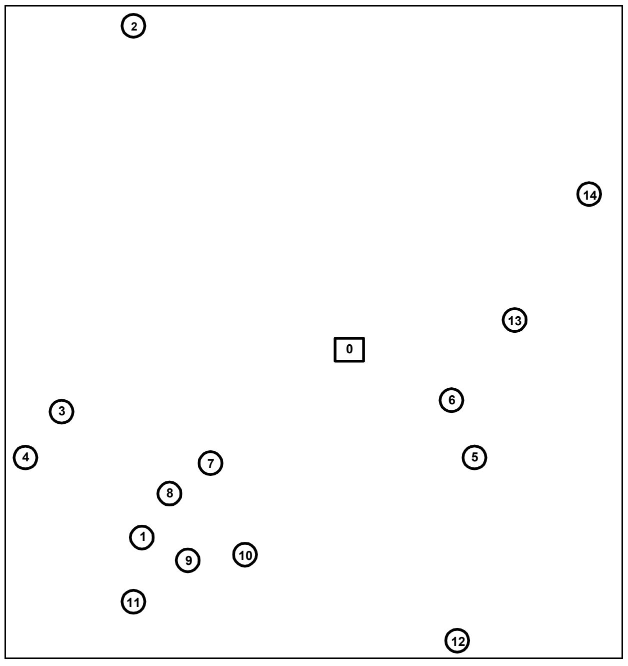


Figure 1. Spatial distribution of the problem vertices