J Math Model Algor
DOI 10.1007/s10852-008-9080-2

Multi-objective Meta-heuristics for the Traveling Salesman Problem with Profits

Nicolas Jozefowiez · Fred Glover · Manuel Laguna

Received: 1 March 2007 / Accepted: 21 December 2007

Abstract We introduce and test a new approach for the bi-objective routing problem known as the traveling salesman problem with profits. This problem deals with the optimization of two conflicting objectives: the minimization of the tour length and the maximization of the collected profits. This problem has been studied in the form of a single objective problem, where either the two objectives have been combined or one of the objectives has been treated as a constraint. The purpose of our study is to find solutions to this problem using the notion of Pareto optimality, i.e. by searching for efficient solutions and constructing an efficient frontier. We have developed an ejection chain local search and combined it with a multi-objective evolutionary algorithm which is used to generate diversified starting solutions in the objective space. We apply our hybrid meta-heuristic to synthetic data sets and demonstrate its effectiveness by comparing our results with a procedure that employs one of the best single-objective approaches.

Keywords Routing problem · Multi-objective optimization · Ejection chain · Evolutionary algorithm · Hybrid method

Mathematics Subject Classifications (2000) 90B06 · 90C27 · 90C29 · 68T20

N. Jozefowiez (✉)
INSA, LAAS-CNRS, Université de Toulouse, 7 Avenue du Colonel Roche,
F-31077 Toulouse cedex 4, France
e-mail: nicolas.jozefowiez@laas.fr

F. Glover · M. Laguna
Leeds School of Business, University of Colorado at Boulder, CO, USA

F. Glover
e-mail: fred.glover@colorado.edu

M. Laguna
e-mail: manuel.laguna@colorado.edu

1 Introduction

We investigate the solution of the traveling salesman problem with profits (TSPP) [7] by means of an ejection chain procedure combined with a multi-objective evolutionary algorithm. The TSPP is a generalization of the traveling salesman problem (TSP) where a profit is realized when a customer is visited. The problem can be described as follows. Let $G = (V, E)$ be a graph where $V = \{v_1, \ldots, v_n\}$ is a set of n nodes and E is a set of edges. We associate a profit p_i with each node $v_i \in V$ (with $p_1 = 0$) and a distance c_{ij} with each edge $(v_i, v_j) \in E$. The TSPP consists in determining a tour on a subset of V that includes v_1. Performance is measured both in terms of the collected profit and the tour length, creating two conflicting objectives: (1) to minimize the length of the tour; (2) to maximize the total profit. From the perspective of single-objective optimization, three associated problems have been addressed in the literature:

1. Both objectives are combined in the objective function and the goal is to find a solution that minimizes the tour length minus the collected profit. Dell'Amico et al. [5] refer to this version of the TSPP as the profitable tour problem (PTP).
2. A maximum allowed tour length $c_{\max}$ is imposed as a bound and the goal is to find a tour that maximizes the total collected profit subject to satisfy this bound. This problem is called the orienteering problem (OP). The OP has also been referred to as the selective traveling salesman problem (STSP) [18] and the maximum collection problem [15].
3. A minimum allowed profit $p_{\min}$ is imposed as a bound and the goal is to find a minimal length tour whose total collected profit is not smaller than this bound. This problem is called the prize-collection traveling salesman problem (PCTSP) [2] or the quota traveling salesman problem (QTSP) [1].

More information about these problems (formulations, methods, and applications) can be found in the survey by Feillet et al. [7]. It is important to note that, while these variants of the TSPP are multi-objective in nature, they have never been studied from a multi-objective point of view [3]. An attempt to address the TSPP in its explicitly multi-objective form was made by Keller [16] and Keller and Goodchild [17], who referred to the problem as the multi-objective vending problem, but their approaches consisted of sequentially solving single-objective versions of the problem and they did not try to generate a set of non-dominated solutions.

In its general form, a multi-objective problem can be stated as follows:

$$(MOP) = \begin{cases} \min\ F(x) = (f_1(x),\ f_2(x), \ldots,\ f_n(x)) \\ s.t.\ x \in D \end{cases} \tag{1}$$

with $n \geq 2$ being the number of objective functions; $x = (x_1, x_2, \ldots, x_r)$, the decision variable vector; D, the feasible solution space; and $F(x)$, the objective vector. The set $O = F(D)$ corresponds to the feasible solutions in the objective space, and

$y = (y_1, y_2, \ldots, y_n)$, where $y_i = f_i(x)$, is a solution. A MOP solution is the set of the non-dominated solutions called the Pareto set (PS). Dominance is defined as follows:

Definition 1.1 A solution $y = (y_1, y_2, \ldots, y_n)$ dominates (denoted $\prec$) a solution $z = (z_1, z_2, \ldots, z_n)$ if and only if $\forall\, i \in \{1 \ldots n\}$, $y_i \leq z_i$ and $\exists\, i \in \{1 \ldots n\}$, such that $y_i < z_i$.

Definition 1.2 A solution y found by an algorithm A is said to be potentially Pareto optimal (PPS), relative to A, if A does not find a solution z, such that z dominates y.

Evolutionary algorithms and local search methods have been proposed to approximate PS [6]. Such heuristics must be designed with two goals in mind: (2) the algorithm must converge toward the PS, and (2) the solutions identified as efficient must provide a good coverage of the frontier.

Our goal is to apply multi-objective optimization techniques to the TSPP. The main contribution of this paper is the development of a hybrid meta-heuristic (HM) that finds high-quality approximations of the efficient frontier for this class of bi-objective problems. This approach provides a means for addressing all the single objective problems mentioned above and avoids a priori parameterization of the TSPP. The first step of the design of the HM was the definition of a neighborhood search process. To do that, two difficulties in solving the TSPP must be overcome: (1) to solve many traveling salesman problems on different sets of nodes, (2) to select different subsets of nodes to be visited. This is solved by using two sets of neighborhood moves in an ejection chain (EC) process. To provide starting solutions for the EC process, a multi-objective evolutionary algorithm (MOEA) has been developed. We have chosen a MOEA because it is one of the most studied methods for multi-objective problems and it remains, apart from the specific operators, an easy-to-implement solution. However, other methods or techniques could be chosen instead of the MOEA. The coupling of the MOEA and the EC process provides a means for both exploring and approaching the optimal Pareto set, as the EC process is an efficient way to bring solutions toward the optimal Pareto set, while the MOEA is able to provide solutions in the complete objective space thanks to its population, and, therefore, to diversify the search.

The EC process is presented in Section 2. The MOEA and the HM are described in Section 3. Experimental testing is reported in Section 4 and conclusions are summarized in Section 5.

2 Ejection Chain Process

The EC process uses two sets of moves. The first set originates from a reference structure initially proposed by Glover for the TSP [11]. These moves allow the search to be efficient in the solution of the TSP aspect of the problem. The reference structure is presented in Section 2.1 and the set of moves in Section 2.2. The second set of moves is used to modify the set of visited nodes as it is the only way to modify the profit generated by the solution. This set is described in Section 2.3. A goal programming strategy is used to guide the search in the bi-objective space. The

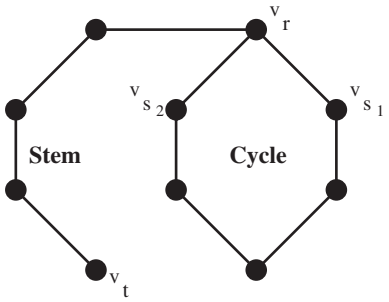

Fig. 1 The stem-and-cycle reference structure

goals are dynamically computed according to the starting solutions as explained in Section 2.4. The improvement procedure for the TSPP, called IP-TSPP, is described in Section 2.5.

2.1 The Stem-and-cycle Reference Structure

Our improvement procedure uses the stem-and-cycle reference structure proposed by Glover [11] for the TSP, which is a spanning sub-graph of G consisting of a path called a stem ST = $\{v_t, \ldots, v_r\}$ connected to a cycle CY = $\{v_r, v_{s_1}, \ldots, v_{s_2}, v_r\}$. An example of such a structure is shown in Fig. 1. Node v_r is common to both the stem and the cycle and is referred to as the *root*. The two nodes (v_{s_1} and v_{s_2}) of the cycle adjacent to v_r are known as *subroots*. Node v_t is the *tip* of the stem.

The structure obtained through the application of an ejection move does not usually represent a feasible tour (unless $v_t = v_r$, i.e. if a unique node is at the same time the root and the tip); thus, a trial move is required to generate a feasible TSP solution. Trial solutions are obtained by inserting an edge (v_t, v_s), where v_s is one of the subroots, and removing the edge (v_r, v_s).

2.2 First Set of Moves

The first set is composed of moves rearranging the sequence in which the nodes are visited. Two different ejection chain moves are possible depending on the positions of the nodes in the *stem-and-cycle* structure. The first move is called a *cycle-ejection* move. An edge of the cycle (v_p, v_q) is removed, v_p is linked to v_t and v_q becomes the new tip. The second move is called a *stem-ejection* move. The move inserts an edge between v_t and v_p where v_p is a node of the stem. Then, the edge (v_q, v_p) is deleted where v_q is the node before v_p on the subpath $(v_t, \ldots, v_p)$. Node v_q becomes the new tip.

It should be noted that a standard TSP tour can be seen as a stem-and-cycle structure where the stem is empty and the cycle is the tour. The root can be any node in the tour.

2.3 Second Set of Moves

The second set is composed of three moves which correspond to the basic operations that can be done in order to modify the set of visited nodes and the profit. These moves work as follows.

1. The first move removes a node visited in the solution. If the node is the tip, the next node in the stem becomes the new tip. Otherwise, the nodes before and after the removed node are connected together to repair the solution. In this move, v_1 and v_r cannot be removed.
2. The second move adds a node that is not visited in the solution. The node is inserted so that the increase in length is minimal. In this move, it is possible to add a node as a new tip.
3. The third move exchanges a node currently in the tour with a node outside the tour. If the removed node is the tip, then the added node becomes the tip. In this move, v_1 or v_r cannot be exchanged.

2.4 Construction of the Goal Point

A goal point is used to guide the search in the multi-objective space. The choice of the goal point is important as it should attract the search toward the section of the Pareto frontier that dominates the solution that is the starting point of the local search.

To construct the goal point, we start by calculating the upper bound on the total profit associated with the m most profitable nodes that are not currently part of the solution. In our implementation, m is fixed to $\lfloor \frac{n}{10} \rfloor$ (where n is the number of nodes). This is done to initially favor moves that result in solutions that do not differ much (in terms of the subset of visited nodes) from the starting solutions. The tabu restrictions (discussed below) diversify the subset of visited nodes that are associated with the solutions in the late stages of the search.

The difficulty with creating a goal in terms of tour length is that the impact of removing m nodes from the starting solution is not really known, because the tour will change. Therefore, the goal for the length objective is computed as follows. Let g_p be the goal value for the profit, s_l the length of the starting solution, s_p the profit of the starting solution, ub_p the maximum profit (i.e. the profit obtained when all the nodes are visited), and ub_l the length upper bound (i.e. the worst length in the archive A containing all the non-dominated solutions found so far). Then, the goal value for the length g_l is given by $g_l = s_l - \text{ub}_l \times \frac{g_p - s_p}{\text{ub}_p}$. In this way, the distance between the starting solution length and g_l is the same as the one between the starting solution profit and g_p if the values are normalized.

2.5 Improvement Procedure for the TSPP (IP-TSPP)

The improvement procedure that we have developed is similar to the PSEC LS proposed by Rego [19] for the TSP. A high level description of the improvement procedure is shown in Algorithm 1. IP-TSPP takes as an input a solution S for the TSPP. First, it computes the goal point according to S. At each iteration, itselects a new node to become the root node of the stem-and-cyle procedure. This is necessary as the explored neighborhood highly depends on the definition of the root node. The process is iterated until all the nodes have been used as the root without finding a

solution closer to the goal point. When a better solution is found, the set of possible root nodes is reset so that every node can be chosen again as a root. A solution s_1 is said to be better than another solution s_2 if the Euclidean distance between $F(s_1)$ and g is smaller than the Euclidean distance between $F(s_2)$ and g. The search also updates an archive A which contains the non-dominated solutions found during the search. This archive is the final result of IP-TSPP. It allows to find solutions obtained thanks to an oscillation around the local optimum found by the procedure.

Algorithm 1 IP-TSPP(Solution S)

```
Compute the goal point g according to S.
S* ← S
C ← V (C is the set of candidates to become the root node)
A ← ∅ (A is the archive of the non-dominated solutions found)
while C ≠ ∅ do
   Select v_r randomly from C and set v_r as the root node
   C ← C \ {v_r}
   S' ← core_step(S*, g, v_r, A)
   Apply a trial move on S' (i.e. build a feasible tour as explained in Section 2.1)
   if S' is better (i.e, closer to g) than S* then
      S* ← S'
      C ← V
   end if
end while
Return A
```

The core step of IP-TSPP is detailed in Algorithm 2. First, the stem-and-cycle structure is completely defined by making the root node also the tip node. The starting solution is saved as the best solution found so far and as the current solution.

Algorithm 2 core_step (Solution S, Goal g, Root v_r, Archive A)

```
v_t ← v_r (Initially, the root is also the tip)
k ← 1
S* ← S
S_c ← S
repeat
   Find the best feasible move on S_c, i.e., the moves allowing the smallest Euclidean
   distance between the resulting solution and g
   Perform the move on S_c
   Update the tabu short term memory structure
   Try to include S_c in A
   if S_c is better than S* then
      S* ← S_c
   end if
   k ← k + 1
until k > k_max
Return S*
```

Then, the process is run for $k_{\max}$ iterations or until no move is possible. $k_{\max}$ is fixed to n. The move to perform is chosen first by considering the first set of moves and then the second set of moves. The chosen move is the one which provides the closest point in the objective space to the goal point when it is combined with a trial move. In case of a tie, the move from the first set is preferred, otherwise the choice is made randomly. During the move selection phase, the move is not performed as it is not necessary to do so to know the values of the objectives. The selected move operates on the current solution Sc and the tabu memory structure is updated. A short term memory is used to record the attribute of the (θ) most recent moves. For the first set of moves, we record the edge that was removed or inserted in order to prevent the reversal of such a move for (θ) iterations. If the selected move belongs to the second set, we record the node that was removed (added) in order to prevent the node from being added (removed) in the next (θ) iterations. The tabu status are reset between each core step as it has shown to provide better results. The current solution is also tried for inclusion in the archive A. S_c is included in the archive if no solution from the archive dominates it. All solutions from A dominated by S_c are removed. An iteration ends with the updating of the best solution found if the current solution is closer to the goal point.

3 The Multi-objective Evolutionary Algorithm and the Hybrid Meta-heuristic

To generate starting solutions for IP-TSPP, we use a multi-objective evolutionary algorithm (MOEA). The multi-objective evolutionary algorithm we propose for the TSPP is a steady-state variant of NSGA II [4]. The procedure operates on a finite population of solutions (or individuals) that we initialize as described in Section 3.1. Then, the population evolves from one generation to the next. In our MOEA, an iteration consists of the following steps (where the numbers in parentheses indicate the section number where the step is described in greater detail). In our MOEA, an iteration runs as follows:

1. Evaluation (Section 3.2): The *fitness* (quality) of the individuals in the population is evaluated.
2. Parent selection (Section 3.2): A pair of solutions (*parents)* is selected according to their fitness.
3. Crossover (Section 3.4): The pair of parents combines to produce a new solution (*offspring*). We generate only one offspring because we are implementing a steady-state variant of NSGA II.
4. Mutation (Section 3.5): The offspring is randomly modified with a given probability.
5. Population update (Section 3.2): The offspring may replace the worst individual in the current population.

These steps are explained in details below. In Section 3.6, we explain how solutions generated by the MOEA are selected for the IP-TSPP procedure and how the output from the IP-TSPP procedure is merged with the MOEA population. This forms our hybrid meta-heuristic.

3.1 Initialization of the Population

We build the initial population of N individuals by means of heuristically solving several STSPs with the *Insert and Shake* procedure of Gendreau et al. [9]. This method combines the TSP tour extension heuristic described in Rosenkrantz et al. [20] and GENIUS, a TSP heuristic developed by Gendreau et al. [8]. *Insert and Shake* gradually extends a tour T until no other node can be added without violating the length limit $c_{\max}$. Then, GENIUS is applied in an attempt to obtain a shorter tour on the nodes in T. If GENIUS fails to produce a better tour, the procedure terminates. Otherwise, more node insertions are attempted and the process is repeated. Thus, the steps to build an initial population of solutions are:

STEP 1 The IP-TSP local search described in Section 2 is applied to find a tour, considering all nodes in V, and this solution is included in the population. Let ub_l be the length of the tour. Set $c_{\max} \leftarrow \text{ub}_l - \frac{\text{ub}_l}{N-1}$ and $i \leftarrow 1$.

STEP 2 If i is equal to $N+1$, terminate. Otherwise, generate a solution by means of *Insert and Shake* and add this solution to the population.

STEP 3 Set $c_{\max} \leftarrow \frac{c_{\max} - ub_l}{N-1}$ and $i \leftarrow i+1$. Go to STEP 2.

3.2 Population Management

At each generation, NSGA II computes two values for each solution i: a rank r_i, which measures solution quality, and a crowding distance metric d_i, which estimates search diversification. To do that, NSGA II sorts the population into different non-domination levels. In this ranking phase, the non-dominated individuals in the population obtain rank 1 and form the subset E_1. Rank k is given to the solutions only dominated by the individuals belonging to the subset $E_1 \cup E_2 \cup \cdots \cup E_{k-1}$. Then, a fitness equal to its rank (1 is the best level) is assigned to each solution.

The *crowding distance metric* gives an estimate of the density of the solutions surrounding a solution i in the population and is expressed by approximating the perimeter of the cuboid formed by the nearest neighbors of i.

Two parents are selected by means of a tournament that favors the best ranked solutions and uses the crowding distance to break ties. One offspring is generated from the selected parents using the genetic operators described below. Solutions may not appear more than once in the population, following an orientation introduced in scatter search [10, 13]. This means that a recently created offspring is added to the current population if and only if it does not already exist. A new offspring replaces the solution in the current population that has the worst rank, with the crowding distance serving as the tie-breaking mechanism. The population does not change if the newly created offspring is already in the population.

3.3 Archive and Stopping Criterion

The potentially Pareto optimal solutions are stored in an archive A. In the implementation, the archive is common with IP-TSPP. At the end of the process, this archive contains our solution to the problem. This archive is also used to implement a rule that makes the search stop if the archive does not change for M generations in a row.

3.4 The Combination Operator

The main difficulty associated with designing a method to combine solutions for the TSPP (loosely referred to as *crossover* in GA parlance) is that two parent solutions may not have much in common. For instance, when considering the set of nodes visited in each parent solution, their intersection may consist of only v_1 and their cardinality may be different. The following multi-phase combination operator was designed taking these observations into account. It works by transforming the parents into parents visiting the same set of nodes such that a classic operator for the TSP can be applied. More precisely, in the first phase, the parents are modified by eliminating all nodes that are not common to both parents. Then the edge recombination crossover (ERX) [21] is applied to the modified parents. Finally, nodes that have been discarded during the first phase are tested for inclusion in the offspring. This constitutes a simple instance of the structured combination approach proposed in connection with scatter search and tabu search [12, 14].

3.4.1 Phase 1

Let s_1 and s_2 be the two parents, where s_1 (respectively s_2) visits the nodes $V_1 \subseteq V$ (respectively $V_2 \subseteq V$) following the tour σ_1 (respectively σ_2). Let $V' = V_1 \cap V_2$, identifying the set of nodes that are common to s_1 and s_2. If $|V'| = 1$, the crossover is aborted and no offspring is generated. Otherwise, we build a solution s'_1 from s_1 by keeping only the nodes in V' and building a tour σ'_1 that preserves the visiting order in σ_1. For every node pair v_i, v_j such that v_j is visited just after v_i in σ'_1, two arcs (v_i, v_j) and (v_j, v_i) are defined. Arc (v_i, v_j) receives two values: (1) the length of the path from v_i to v_j in σ_1; (2) the sum of the profits of the nodes on the path between v_i and v_j in σ_1, excluding v_i. (v_j, v_i) receives similar values by interchanging the roles of v_i and v_j. It should be noted that the arcs (v_i, v_j) and (v_j, v_i) have the same length but not the same profit as they stand for the same path but traverse in two different orders and that the profit of the starting node is not considered in the profit associated to the arc. A solution s'_2 is built in the same way from s_2. An example of this process is provided in Fig. 2, where, in the left-hand side graphs, the values next to the nodes are profit and the values next to the edges are length. In the right-hand side graphs, the values on the arcs are length/profit.

3.4.2 Phase 2

During this phase, a modified form of ERX is applied, with s'_1 and s'_2 as parents. For each node in $v_i \in V'$, a list l_i of the nodes adjacent to v_i in s'_1 and s'_2 is built. A node v_j can appear twice if the arcs (v_i, v_j) and/or (v_j, v_i) appear in both parents. (The direction of the arc is also saved.)

The offspring is created by adding one node at a time beginning with v_1. Let v_i be the last node added to the offspring. Then, v_i is removed from every list where it appears. If the list l_i associated with v_i is empty, the structured combination process stops. We point out that this operator may not add all the nodes that appear in the parent solutions. However, in practice, it has been observed that ERX builds a solution that includes all the parent nodes about 95 % of the time [21]. When ERX fails to generate a solution that includes all nodes in the parents, a node that is not

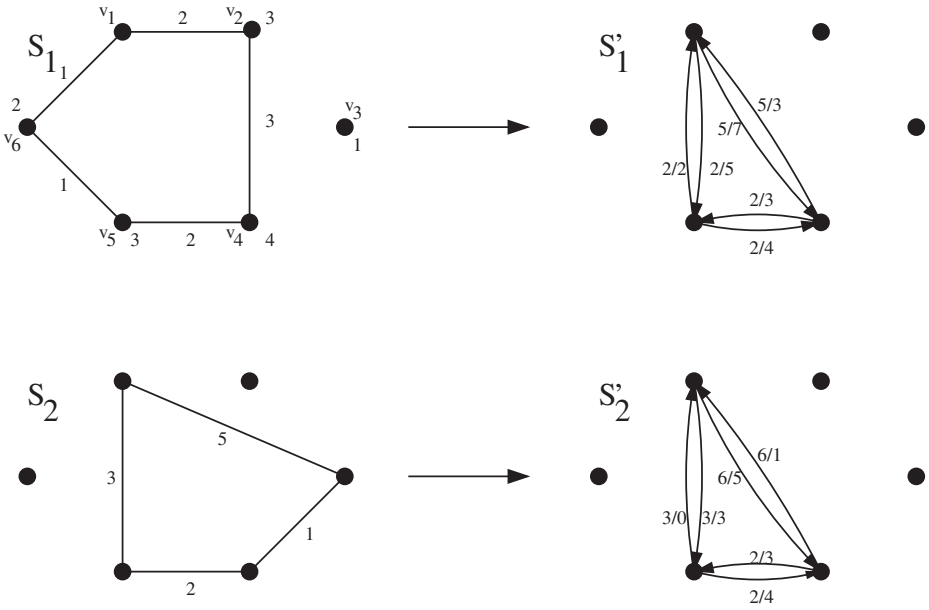

Fig. 2 An example of the phase 1 of the crossover

already in the offspring is randomly chosen and the process continues. In the context of the TSPP, we require no repairing mechanism because a solution is feasible even if all nodes are not included.

When l_i is not empty, the following rules build the set of candidates from which the next node is randomly selected:

1. Let C_1 be the subset of nodes v_j in l_i whose associated list l_j is not empty. If the l_j list associated with every node v_j in l_i is empty, then let C_1 consist of all v_j in l_i.
2. For every node v_j in C_1, let a_j be the arc between v_j and v_i, and assign node v_j a rank equal to the number of nodes v_k such that the values of a_j are dominated in the Pareto sense by a_k. Let C_2 be the subset of nodes in C_1 having rank 0.
3. Let C_3 be the subset of C_2 nodes whose associated lists tie for having the least number of elements.
4. Let C_4 be the subset of nodes v_j in C_3 such that (v_i, v_j) and/or (v_j, v_i) appears in both parents s'_1 and s'_2. If such a node v_j in C_3 does not exist, let $C_4 = C_3$.
5. Choose a node randomly from C_4.

The first step favors the adjacent nodes with available neighbors such that the tour can be extended after its application. The second step is a greedy criterion such that the current offspring at this moment is not dominated by another offspring of the same length. Experiments have shown that this choice has a positive impact. Step 3 is the standard criterion from ERX [21]. Step 4 is to encourage the choice of information present in both parents. For instance, if we have the choice to include two nodes v_j and v_k after v_i. If the arc (v_i, v_j) appears in both parents while (v_i, v_k) only appears in one parent, the former is selected.

3.4.3 Phase 3

A node v_j is added immediately after a node v_i during the second phase when either arc (v_i, v_j) or arc (v_j, v_i) appears in one of the parents s'_1 and s'_2. Let α be the added arc, s' be the modified parent where α appears and s be the original parent that was transformed into s' during phase 1. According to the construction process used in phase 1, α may stand for a path composed of several nodes in s. The nodes in such a path are added to the offspring while preserving the direction of the path between v_i and v_j in s. This is done for all the edges (v_i, v_j) in the offspring. For instance, if we consider the first parent in Fig. 2 and if the arc (v_1, v_5) has been added to the offspring from this parent, then, in phase 3, it is extended to the path $v_1 v_6 v_5$.

3.5 Mutation and Improvement Operators

After an offspring is generated, we allow it to be modified with probability P_m. There are three methods to achieve this:

Add/Remove	A node is randomly selected from V. If the node appears in the tour, it is removed and its predecessor and successor are connected to repair the tour. Otherwise, the node is added so that the length of the resulting tour is minimal.
Exchange	A node in the tour is randomly selected and replaced by another randomly selected node that is currently not in the tour.
Local search	An EC process is applied to the solution. It is the same process as IP-TSPP except that the possible moves are limited to the first set and the trial move. The goal point is defined as $(0, p)$ where p is the profit of the solution to be mutated. In that way, this operator may improve the length of the tour without modifying the set of visited nodes (i.e., with no change in the total profit).

These operators have been selected because they closely relate to the four main operations that may be used to transform a tour in the context of the TSPP.

3.6 Definition of the Hybrid Meta-heuristic

IP-TSPP is applied to every generated solution s that enters the archive A during the GA process, that is, to every solution that appears to be of good quality and promising (i.e. solutions which are potentially Pareto optimal at a given time). Then, if such a solution exists at the end of IP-TSPP, s is replaced in P by one of the solutions that dominates s found during the execution of IP-TSPP.

Then, the complete method works as follows (cf. Fig. 3): first, an initial population P is generated by a heuristic for the STSP. An archive A containing all the non-dominated solutions found in P is created. Every generated solution s will be tested for inclusion in A, i.e., if no solution from A dominates s, then s is included in A. At the same time, every solution from A dominated by s is removed from A. Then, the MOEA is run until the number of generations without a solution included in A (*nbstall*) reaches a given number of generations M. A generation works as follows: two parents are selected in the population and genetic operators are applied to obtain an offspring o. This offspring is only considered if it is not already in the population.

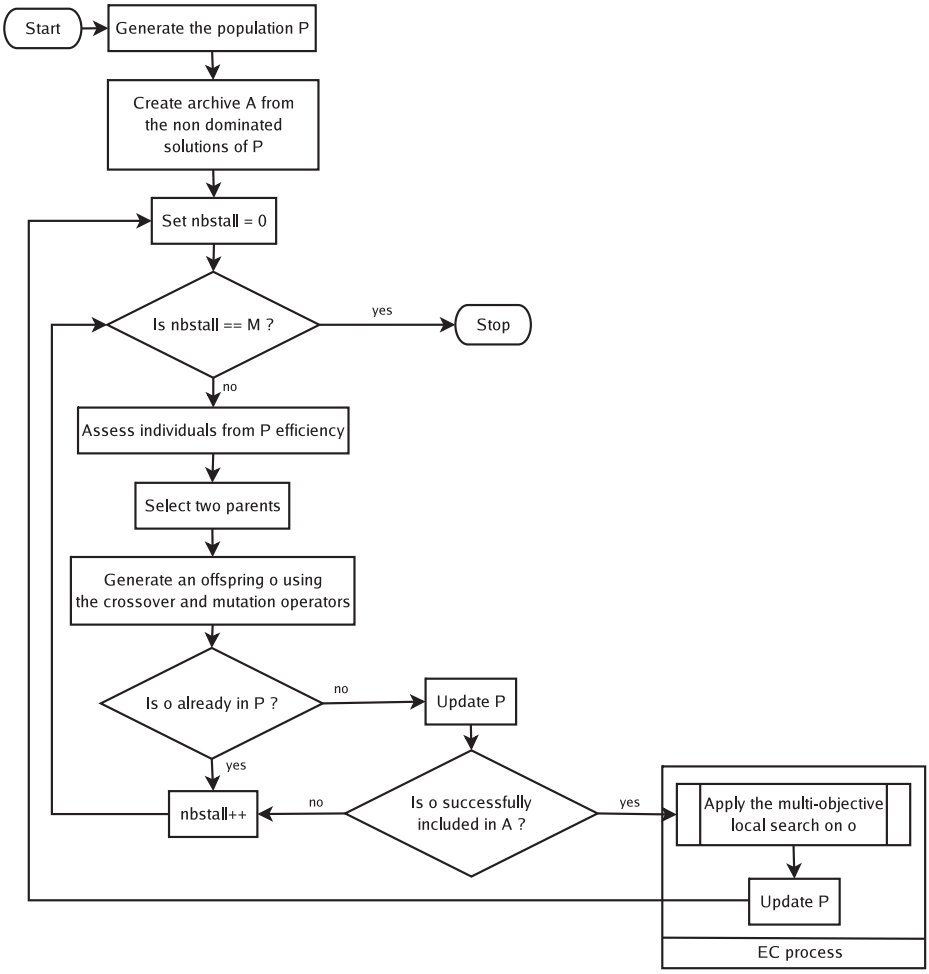

Fig. 3 Flowchart of the hybrid meta-heuristic

If o is successfully added to the archive, the EC process is executed with o as the starting solution. In any case, the worst solution in P is replaced by o or a better solution found by the EC process if it has been used.

4 Computational Results

4.1 An ϵ-constraint Method

As far as we are able to determine, there is no other method in the literature that addresses the TSPP as a bi-objective problem. However, a number of studies have investigated the single-objective variants [7], and we have undertaken to compare our procedure to the Tabu search (TS) for the STSP designed by Gendreau et al. [9]

which is reported to be one of the most efficient procedures for the STSP (see Feillet et al. [7]).

To generate solutions to the TSPP with the TS method of Gendreau et al., we use the following ϵ-constraint approach. In the bi-objective case, the ϵ-constraint method adds a new constraint to the problem: $f_i(x) \leq \epsilon$ (or $f_i(x) \geq \epsilon$), where f_i is an objective to be minimized (respectively maximized) and ϵ is a given value. The single-objective procedure is then asked to optimize the other (unconstrained) objective. Varying the ϵ parameter allows for the exploration of the bi-objective space. In the context of the STSP, the objective to be optimized is the maximization of profit and the constrained objective is the minimization of the length. The following procedure is used to choose ϵ:

STEP 1 Compute $\epsilon < 0$ such that any improvement on any tour is smaller than ϵ. This can be computed by considering the smallest possible improvement in the length. This is computed by considering the different possible moves: a node n is added or removed or exchanged with another node while being between any two nodes or swapped from any two nodes with another node which would be between any two nodes. α is computed once at this step considering all the possibilities.

STEP 2 Solve the TSP (heuristically) on V. Save the resulting solution as a potential Pareto optimal solution. Set $c_{\max} \leftarrow \text{ub}_l + \epsilon$, where ub_l is the length of the solution.

STEP 3 If $c_{\max} < 0$, stop. Solve the STSP by means of the TS. Let s be the resulting solution with a length of s_l. Save s as a potential Pareto optimal solution. Set $c_{\max} \leftarrow s_l + \epsilon$. Reiterate STEP 3.

If the TS method of Gendreau et al. were able to find the optimal solution at each iteration, then the set built during the search would be the optimal Pareto set. Furthermore, the number of executions of their TS method will not exceed the number of solutions in the optimal Pareto set in the parametric we are using.

4.2 Benchmarks and Parameters

Experiments were conducted on a series of randomly-generated instances. To generate the node set, $|V|$ different points were generated in a $[0, 100] \times [0, 100]$ square with a uniform distribution. A randomly generated integer profit between 1 and 100 was assigned to each node of $V \setminus \{v_1\}$. For each value of $|V| = 75, 100, 125, 150$, five instances were generated.

The evolutionary algorithm MOEA and the hybrid meta-heuristic HM were executed ten times on each instance. The parameters used were: θ was randomly chosen in [5], $N = 256$, $M = 2{,}500$, and $P_m = 0.2$. Preliminary experimentation disclosed these parameters to be effective.

We have compared the methods according to their computational times in seconds (Time) and the number of potentially Pareto optimal solutions (NB) they were able to find. The $\mathcal{S}$ metric [22] was also used to compare the methods, based on computing the volume (area in the bi-objective case) dominated by a given Pareto-front approximation. The $\mathcal{S}$ metric requires a reference point Z_{ref} consisting in a reference value for each of both objectives. For the length value, we used the worst length found by all methods; for the profit, we took the value 0 as it is the worst

Table 1 Times (in seconds)

n	ϵ-Constraint	HM		MOEA	
	method	Mean	SD	Mean	SD
75	5,114	270	33	121	40
75	5,172	388	77	120	32
75	5,610	236	19	107	21
75	4,191	305	40	147	32
75	5,603	379	22	130	25
100	20,373	1,122	132	445	128
100	20,990	1,178	156	272	67
100	17,447	1,415	127	529	185
100	15,754	1,404	350	465	80
100	17,757	1,796	745	489	112
125	39,828	4,613	542	997	237
125	36,770	2,960	434	972	182
125	39,656	4,131	761	1,206	399
125	38,313	3,496	437	876	270
125	32,800	3,687	619	1,277	464
150	55,258	7,330	1,162	2,462	635
150	59,203	8,729	1,975	2,380	494
150	53,381	5,239	670	1,998	426
150	50,395	6,637	1,467	2,999	383
150	54,780	7,224	1,409	3,220	654

possible profit. We have normalized the $\mathcal{S}$ metric so that the maximal theoretical area is 1.0. The results are respectively reported in Tables 1, 2, and 3, for the TS-based ϵ-constraint method, the MOEA alone, and the hybrid meta-heuristic (HM),

Table 2 Number of potentially Pareto optimal solutions

n	ϵ-Constraint	HM		MOEA	
	method	Mean	SD	Mean	SD
75	305	645.7	17.3	516.0	39.2
75	435	723.9	42.2	569.5	50.9
75	326	660.2	11.2	539.0	28.4
75	308	669.7	12.8	516.2	33.3
75	380	733.9	10.0	607.2	45.1
100	580	1,142.9	40.6	869.7	37.9
100	444	867.4	13.8	657.0	28.3
100	445	1,271.9	29.0	886.4	121.1
100	481	1,012.2	31.3	891.2	50.9
100	562	1,093.1	31.6	854.7	61.9
125	561	1,752.9	25.1	1,382.1	147.5
125	510	1,531.8	33.8	1,068.9	85.3
125	601	2,005.8	35.5	1,318.6	172.0
125	505	1,381.4	17.8	1,129.6	102.6
125	478	1,692.0	45.1	1,366.3	99.5
150	464	1,864.6	39.0	1,490.9	167.1
150	527	2,344.8	149.4	1,767.6	261.2
150	435	1,671.0	72.4	1,341.8	100.5
150	439	2,028.1	75.6	1,326.5	89.6
150	719	2,283.6	46.8	1,829.0	178.7

Table 3 $\mathcal{S}$ metric values

n	ϵ-CM	HM			MOEA		MOEAu	
		Max.	Mean	Min.	Max.	Mean	Max.	Mean
75	0.585067	0.586240	0.585888	0.585420	0.585468	0.583486	0.585561	0.583986
75	0.593236	0.594540	0.593858	0.590694	0.590843	0.588162	0.591399	0.589119
75	0.576733	0.577810	0.577765	0.577670	0.577333	0.577040	0.577494	0.577299
75	0.596526	0.597674	0.597607	0.597486	0.596054	0.595351	0.596182	0.595912
75	0.581920	0.583361	0.583007	0.582194	0.582334	0.580350	0.582417	0.581113
100	0.606018	0.607229	0.606574	0.605043	0.605763	0.603865	0.606657	0.605292
100	0.591636	0.592637	0.592344	0.592243	0.591934	0.591287	0.592143	0.591838
100	0.611364	0.612885	0.611928	0.611251	0.610975	0.608407	0.611155	0.609311
100	0.595497	0.597554	0.597039	0.596057	0.594981	0.594168	0.596677	0.595223
100	0.594143	0.595550	0.594759	0.593889	0.593215	0.589779	0.592664	0.590929
125	0.587292	0.590175	0.589944	0.588946	0.587899	0.586793	0.588643	0.587425
125	0.595876	0.598160	0.597566	0.596947	0.595562	0.594876	0.596203	0.595445
125	0.630162	0.633343	0.632389	0.629455	0.631595	0.629748	0.631395	0.630464
125	0.600385	0.603044	0.602341	0.599631	0.601820	0.600550	0.602253	0.601144
125	0.614249	0.617887	0.616088	0.614819	0.614379	0.612510	0.614921	0.613707
150	0.587704	0.595406	0.594031	0.591555	0.592332	0.588543	0.591752	0.587566
150	0.611467	0.620409	0.619362	0.614182	0.617709	0.613773	0.616726	0.614095
150	0.614186	0.617686	0.616531	0.615642	0.614569	0.611352	0.613949	0.611781
150	0.609805	0.618804	0.618209	0.617180	0..616121	0.612857	0.616281	0.614079
150	0.629149	0.632400	0.631189	0.629091	0.631050	0.629373	0.631292	0.629490

i.e. IP-TSPP with the MOEA providing the starting solutions. We have also used the $\mathcal{C}$ metric [22] to compare the average ratio of solutions found by a given method dominated by solutions from the hybrid meta-heuristic. In Table 4, C(A, B) is the average ratio of solutions from B dominated by solutions from A.

4.3 Efficiency of the MOEA

Because the genetic algorithm literature proposes that a solution recombination method should constitute a solution procedure in itself, and not simply as an intensification or diversification process for a local search, we investigated the MOEA procedure in isolation from the local search procedure. The efficiency of the MOEA was assessed in comparison with the TS-based ϵ-constraint method. According to the $\mathcal{S}$ metric, the ϵ-constraint method was able to find better quality approximation than the MOEA for most problem instances of size smaller or equal to 125. However, the MOEA has a better average $\mathcal{S}$ value than the $\mathcal{S}$ value for the ϵ-constraint method for 4 out of 5 instances when $|V| = 150$. Furthermore, for 11 instances, i.e. more than fifty percent of the instances, there is at least one run during which the MOEA found a better approximation than the ϵ-constraint method. This seems to indicate that the MOEA is still an interesting method. Indeed, the running times of the MOEA remain particularly low compared to the TS-based ϵ-constraint method, and therefore, it can be used to obtain quickly a first approximation to the problem.

4.4 Efficiency of the Hybrid Meta-heuristic

As it can be expected, the full HM procedure which incorporates the MOEA as a process to generate starting solutions considerably increases the computational

Table 4 $\mathcal{C}$ metric values

n	HM/ϵ-CM		HM/MOEA		HM/MOEAu	
	C(HM,ϵ-CM)	C(ϵ-CM,HM)	C(HM,MOEA)	C(MOEA,HM)	C(HM,MOEAu)	C(MOEAu,HM)
75	0.26	0.03	0.42	0.04	0.40	0.04
75	0.33	0.02	0.59	0.02	0.55	0.03
75	0.32	0.01	0.38	0.02	0.28	0.02
75	0.39	0.01	0.50	0.01	0.46	0.01
75	0.40	0.06	0.56	0.04	0.49	0.04
100	0.39	0.08	0.67	0.07	0.50	0.07
100	0.34	0.01	0.41	0.01	0.30	0.01
100	0.48	0.02	0.67	0.08	0.56	0.08
100	0.58	0.07	0.67	0.02	0.57	0.07
100	0.34	0.03	0.71	0.03	0.61	0.03
125	0.63	0.01	0.71	0.01	0.63	0.01
125	0.60	0.06	0.76	0.03	0.70	0.03
125	0.61	0.05	0.72	0.01	0.64	0.02
125	0.67	0.09	0.60	0.09	0.48	0.09
125	0.48	0.03	0.73	0.06	0.67	0.06
150	0.70	0.04	0.79	0.05	0.79	0.06
150	0.70	0.02	0.80	0.03	0.78	0.02
150	0.52	0.03	0.84	0.05	0.81	0.04
150	0.78	0.01	0.83	0.02	0.78	0.07
150	0.39	0.14	0.58	0.06	0.53	0.07

time. However, as shown in Table 1, the time remains significantly smaller than the time needed by the ϵ-constraint method. From a positive point of view, IP-TSPP significantly improves the quality of the solutions. It increases the number of potentially Pareto optimal solutions found by the HM when it is compared to the MOEA. It should be noted that the average $\mathcal{S}$ value for the HM is always better than the best value for the MOEA. Moreover, the worst $\mathcal{S}$ metric value for the HM is better on 17 instances than the average $\mathcal{S}$ value for the MOEA and it is better on 10 instances than the best $\mathcal{S}$ value for the MOEA.

Another test we have done is to compare the HM with the MOEA if the latter was allowed to run for a time equal to the average of the times of the HM on each instance. The results for the $\mathcal{S}$ metric for ten runs on each instance is reported in Table 3 and the new implementation of the MOEA is denoted MOEAu. It appears that the average $\mathcal{S}$ metric value is better for the HM for all the instances and this value is also better than the one for the best run of MOEAu on 19 instances out of 20. The worst $\mathcal{S}$ metric value for the HM is better on 16 instances than the average $\mathcal{S}$ value for MOEAu and it is better on 9 instances than the best $\mathcal{S}$ value for MOEAu. Therefore, it appears that even if we let the MOEA run longer, the HM is still a better choice as it can provide better solutions on average and, in general, it seems it would be able to provide better solutions.

When compared to the ϵ-constraint method, the HM also performs well. As disclosed by Table 1, it is an order of magnitude faster than the Tabu-based ϵ-constraint approach (on average 9.2 times faster). It is also able to find many more potentially Pareto optimal solutions, which can offer more possibilities to the decision maker. Even if the number of solutions found may not be able to allow to determine which method is better, the fact that the ϵ-constraint method found fewer solutions employing significantly more computer time indicates the limitations of

Table 5 Kruskal–Wallis statistical test results

n	HM vs. ϵ-CM	HM vs. MOEAu	ϵ-CM vs. MOEAu
75	$\prec$	$\prec$	$\equiv$
75	$\prec$	$\prec$	$\succ$
75	$\prec$	$\prec$	$\prec$
75	$\prec$	$\prec$	$\succ$
75	$\prec$	$\prec$	$\equiv$
100	$\equiv$	$\prec$	$\prec$
100	$\prec$	$\prec$	$\prec$
100	$\prec$	$\prec$	$\succ$
100	$\prec$	$\prec$	$\succ$
100	$\prec$	$\prec$	$\prec$
125	$\prec$	$\prec$	$\equiv$
125	$\prec$	$\prec$	$\succ$
125	$\prec$	$\prec$	$\prec$
125	$\prec$	$\prec$	$\prec$
125	$\prec$	$\prec$	$\equiv$
150	$\prec$	$\prec$	$\equiv$
150	$\prec$	$\prec$	$\prec$
150	$\prec$	$\prec$	$\succ$
150	$\prec$	$\prec$	$\prec$
150	$\prec$	$\prec$	$\equiv$

applying an iterated single objective method to this kind of problems and shows the need to develop multi-objective methods for them. Furthermore, the large number of potentially Pareto optimal solutions found by the HM is another indicator of the interest to solve this problem as a bi-objective one. Our approach is also able to generate a better quality approximation than the ϵ-constraint method as the average $\mathcal{S}$ metric values of the HM are always better than those of the other method. Moreover, the worst $\mathcal{S}$ value of the HM is better than the $\mathcal{S}$ value for the ϵ-constraint method on 13 instances out of 20.

Additionally, the results in Table 4 show that the HM is able to dominate large parts of the approximation of the other meta-heuristics while the other methods only dominate a marginal number of solutions identified by the HM. This fact combined with the previous one that the HM found more non-dominated solutions reinforce our conclusion that the HM is the most efficient meta-heuristic proposed here.

Finally, statistical tests have been conducted to see if the results related to the $\mathcal{S}$ metric were relevant. To do that, we used the Kruskal–Wallis statistical test with a p value of 5% and compared HM with ϵ-CM and MOEAu as well as ϵ-CM with MOEAu. The results are reported in Table 5. In this table, according to the methods under comparison (A vs. B), $\prec$ means that A is significantly better than B, $\succ$ that B is significantly better than A, and $\equiv$ that there is no significant difference between both. It appears that the HM is always significantly better than MOEAu and that there is only one instance where it is not significantly better than the ϵ-CM. At the same time, no definitive conclusion can be drawn between the ϵ-CM and MOEAu, this indicates that the hybridization is able to improve the robustness of the method and it corroborates our conclusion that the HM is the most efficient meta-heuristic we have designed for the TSPP.

5 Conclusions

We have shown how our new approach to the traveling salesman problem with profits, which uses a bi-objective representation and an ejection chain process with a multi-objective evolutionary algorithm to generate starting solutions, yields an effective method for generating a high quality Pareto set. A computational comparison with an iterated ϵ-constraint implementation of one of the best meta-heuristics for the selective traveling salesman problem shows our method has advantages as the size of the problem increases.

We observe two primary opportunities to improve the performance of our method in future research: first, the possibility to upgrade the Insert and Shake procedure for generating an initial population of tours in the evolutionary approach by incorporating ejection chain strategies to improve the quality of the TSP tours produced. Second, we could employ an alternative evolutionary approach by making use of scatter search and path relinking, which have been shown in a variety of studies to perform more effectively than a genetic algorithm design.

Acknowledgements Nicolas Jozefowiez was supported by a Fulbright grant. This support is gratefully acknowledged. Thanks are also due to F. Semet for providing the code for the Tabu search for the selective traveling salesman problem. Thanks are also due to the referees for their valuable comments.

References

1. Awerbuch, B., Azar, Y., Blum, A., Vempala, S.: New approximation guarantees for minimum-weight *k*-trees and prize-collection salesmen. SIAM J. Comput. **28**, 254–262 (1998)
2. Balas, E.: The prize-collecting traveling salesman problem. Networks **19**, 621–636 (1989)
3. Boffey, B.: Multiobjective routing problems. Top **3**, 167–220 (1995)
4. Deb, K., Pratap, A., Agarwal, S., Meyarvan, T.: A fast and elitist multiobjective genetic algorithm: NSGA II. IEEE Trans. Evolution. Comput. **6**, 182–197 (2002)
5. Dell'Amico, M., Maffioli, F., Värbrand, P.: On prize-collecting tours and the asymmetric travelling salesman problem. Int. Trans. Oper. Res. **2**, 297–308 (1995)
6. Ehrgott, M., Gandibleux, X.: A survey and annotated bibliography of multi-objective combinatorial optimization. OR Spektrum **22**, 425–460 (2000)
7. Feillet, D., Dejax, P., Gendreau, M.: Traveling salesman problems with profits. Trans. Sci. **39**, 188–205 (2005)
8. Gendreau, M., Hertz, A., Laporte, G.: New insertion and postoptimization procedures for the traveling salesman problem. Oper. Res. **40**, 1086–1094 (1992)
9. Gendreau, M., Laporte, G., Semet, F.: A tabu search heuristic for the undirected selective travelling salesman problem. Eur. J. Oper. Res. **106**, 539–545 (1998)
10. Glover, F.: Heuristics for integer programming using surrogate constraints. Decis. Sci. **8**, 156–166 (1977)
11. Glover, F.: New ejection chain and alternating path methods for the traveling salesman problems. Comput. Sci. Oper. Res. 449–509 (1992)
12. Glover, F.: Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). Discrete Appl. Math. **49**, 231–255 (1994)
13. Glover, F., Laguna, M.: Modern heuristic techniques for combinatorial problems. Chapt. Tabu search, pp. 71–140. Blackwell Scientific Publishing (1993)
14. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Press (1997)
15. Kataoka, S., Morito, S.: An algorithm for the single constraint maximum collection problem. J. Oper. Res. Soc. Jpn. **31**, 515–530 (1988)
16. Keller, C.P.: Multiobjective routing through space and time: the MVP and TDVRP problems. Ph.D. thesis, Department of Geography, University of Western Ontario. London, Ontario, Canada (1985)
17. Keller, C.P., Goodchild, M.: The multiobjective vending problem: a generalization of the traveling salesman problem. Environ. Plann., B. Plann. Des. **15**, 447–460 (1988)
18. Laporte, G., Martello, S.: The selective traveling salesman problem. Discrete Appl. Math. **26**, 193–207 (1990)
19. Rego, C.: Relaxed tours and path ejections for the traveling salesman problem. Eur. J. Oper. Res. **106**, 522–538 (1998)
20. Rosenkrantz, D.J., Stearns, R.E., Lewis II, P.M.: An analysis of several heuristics for the traveling salesman problem. SIAM J. Comput. **6**, 563–581 (1977)
21. Whitley, D., Starkweather, T., Fuquay, D.: Scheduling problems and traveling salesman: the genetic edge recombination operator. In: Schaffer J. (ed.) Proceedings of the Third International Conference on Genetic Algorithms, pp. 133–140 (1989)
22. Zitzler, E.: Evolutionary algorithm for multiobjective optimization: methods and applications. Ph.D. thesis, Swiss Federal Institute of Technology (ETH). Zurich, Switzerland (1999)