Finding Pareto-Optimal Set by Merging Attractors for a Bi-Objective Traveling Salesmen Problem

Weiqi Li

School of Management, University of Michigan-Flint, 303 East Kearsley Street,
Flint, Michigan 48502, U.S.A.
weli@umflint.edu

Abstract. This paper presents a new search procedure to tackle multi-objective traveling salesman problem (TSP). This procedure constructs the solution attractor for each of the objectives respectively. Each attractor contains the best solutions found for the corresponding objective. Then, these attractors are merged to find the Pareto-optimal solutions. The goal of this procedure is not only to generate a set of Pareto-optimal solutions, but also to provide the information about these solutions that will allow a decision-maker to choose a good compromise solution.

1 Introduction

A multi-objective optimization seeks to optimize a vector of non-commensurable and often competing objectives. In other words, we whish to find a set of values for the decision variables that optimizes a set of objective functions. The general multi-objective combinatorial optimization problem can be formulated as:

$$\textit{optimize} \quad f(x) = \begin{Bmatrix} f_1(x) = z_1 \\ f_2(x) = z_2 \\ \vdots \\ f_k(x) = z_k \end{Bmatrix} = z \in Z \tag{1}$$

$$\textit{subject to} \quad x = (x_1, x_2, \ldots, x_n) \in X$$

where x is the decision vector, or *solution*, and $X \in \Re^n$ is the n-dimensional *decision space*, consisting of a finite set of feasible solutions. The objective function $f(x)$ maps x into $Z \in \Re^k$, the k-dimensional *objective space*, where k is the number of objectives. Whereas a single-objective problem is typically studied in decision space, multi-objective optimization is mostly studied in objective space. The image of a solution in the objective space is a point, $z = [z_1, z_2, \ldots, z_k]$. A point, z, is attainable if there exists a solution $x \in X$ such that $z = f(x)$. The set of all attainable points is denoted as Z. The ideal objective vector z^* is defined as $z^* = [\text{opt}f_1(x), \text{opt}f_2(x), \ldots, \text{opt}f_k(x)]$, which is obtained by optimizing each of the objective functions individually. Normally, the ideal objective vector is not attainable because of the conflict among the objectives.

Therefore, there will not exist a single optimal solution to the multi-objective combinatorial problem. Instead, we must look for "trade-off" solutions when dealing with a multi-objective optimization problem.

Objective vectors are compared according to the concept of Pareto-optimality and dominance relation. A partial ordering can be applied to solutions to the problem by the dominance criterion. A solution $x^a \in X$ is said to dominate a solution $x^b \in X$ if x^a is superior or equal in all objectives and at least superior in one objective. Mathematically, the concept of *Pareto optimality* is as follows [21]: assume, without loss of generality, a minimization problem, and consider two decision vectors, x^a, $x^b \in X$, then x^a is said to dominate x^b (often written as $x^a \succ x^b$) if and only if

$$\begin{aligned} &\forall i \in \{1,2,...,k\} : f_i(x^a) \le f_i(x^b) \quad \wedge \\ &\exists j \in \{1,2,\ldots,k\} : f_j(x^a) < f_j(x^b) \end{aligned} \tag{2}$$

The solution x^a is said to be indifferent to a solution x^b, if neither solution is dominating the other one. When no a priori preference is defined among the objectives, dominance is the only way to determine if one solution performs better than the other does. The concept of Pareto optimality almost gives us a set of solutions called the *Pareto-optimal set*. The solutions in the Pareto-optimal set are also called *nondominated*, characterized by the fact that starting from a solution within the set, one objective can only be improved at the expense of at least one other objective being deteriorated. The curve formed by joining the Pareto-optimal solutions is known as a *Pareto-optimal front*. The goal of solving multi-objective problem is to find the Pareto-optimal set for the decision-maker to choose the most preferred solution. A solution selected by the decision-maker always represents a compromise between the different objectives.

The bounds on the Pareto-optimal set in the objective space can be defined by the ideal point and the nadir point [16]. The ideal objective vector, z^*, denotes an array of the lower bound of all objective functions. For each of the k conflicting objectives, there exists one different optimal solution. An objective vector constructed with these individual optimal objective values constitutes the ideal objective vector z^*. In general, the ideal objective vector corresponds to a non-existent solution. This is because the optimal solution for each objective function need not be the same solution. The nadir objective vector, z^{nad}, represents the upper bounds of each objective in the entire Pareto-optimal set.

The problem of finding the true Pareto-optimal set is NP-hard [5]. Thus, the goal of the multi-objective combinatorial optimization is to approximate the Pareto-optimal set. Over the years, the work of a considerable number of researchers has produced an important number of techniques to deal with multi-objective optimization problems [4], [7], [16], [22].

The TSP is the most well-known of all NP-hard combinatorial optimization problems. Multi-objective TSP is even harder than its corresponding single-objective version. Some researches have specifically treated the multi-objective TSP. Fischer and Richter [8] used a branch and bound approach to solve a TSP with two (sum) criteria. Gupta and Warburton [9] used the 2- and 3-opt heuristics for the max-ordering TSP. Sigal [20] proposed a decomposition approach for solving the TSP

with respect to the two criteria of the route length and bottlenecking, where both objectives are obtained from the same cost matrix. Tung [23] used a branch and bound method with a multiple labeling scheme to keep track of possible Pareto-optimal tours. Melamed and Sigal [14] suggested an *e*-constrained-based algorithm for bi-objective TSP. Ehrgott [6] proposed an approximation algorithm with worst case performance bound. Hansen [10] applied the tabu search algorithm to multi-objective TSP. Borges and Hansen [2] used the weighted sums program to study the global convexity for multi-objective TSP. Jaszkiewicz [11] proposed the genetic local search which combines ideas from evolutionary algorithms, local search with modifications of the aggregation of the objective functions. Paquete and Stützle [18] proposed the two-phase local search procedure to tackle bi-objective TSP. During the first phase, a good solution to one single objective is found by using an effective single objective algorithm. This solution provides the starting point for the second phase, in which a local search algorithm is applied to a sequence of different aggregations of the objectives, where each aggregation converts the bi-objective problem into a single objective one. Yan et al. [24] used an evolutionary algorithm to solve multi-objective TSP. Angel, Bampis and Gourvès [1] proposed the dynasearch algorithm which uses local search with an exponential sized neighborhood that can be searched in polynomial time using dynamic programming and a rounding technique. Paquete, Chiarandini and Stützle [17] suggested a Pareto local search method which extends local search algorithm for the single objective TSP to bi-objective case. This method uses an archive to hold non-dominated solutions found in the search process.

This study proposes a new search procedure to tackle multi-objective TSP. Fig. 1 sketches the schematic of the search procedure. This procedure incorporates the relationship between the problem presentation and data structure into the algorithm design. For a TSP with *k* objectives, this procedure first constructs the solution attractor for each of objectives individually. The solution attractor contains a set of the best solutions found for the corresponding objective. It is also reasonable to believe that the attractor consists of a large proportion of low-cost edges for the objective. These edges are in the extreme for that objective function. Then the procedure combines these *k* attractors in order to mix the edges contained in these attractors. Finally, the Pareto-optimal solutions can be found from these mixed edges.

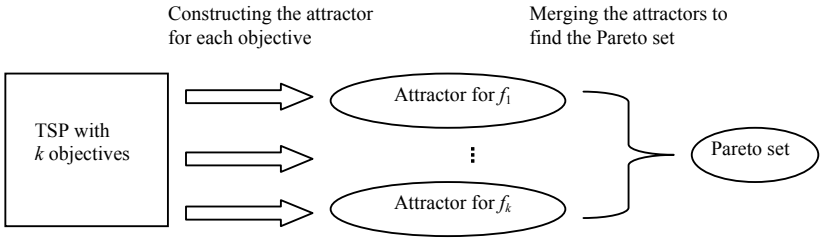

Fig. 1. Schematic of the search procedure

The remaining of the paper is organized as follows. Next section introduces a method for constructing a solution attractor for a single-objective TSP. Section 3 describes the proposed procedure for finding the Pareto-optimal solutions for multi-objective TSP and presents the computational results for a bi-objective TSP instance. The goal of this procedure is not only to generate a set of Pareto-optimal solutions effectively, but also to provide information about these solutions that will allow a decision-maker to choose a good compromise solution. The final section concludes this paper.

2 Constructing Solution Attractor for TSP

Local search heuristics is a widely used general approach to find reasonable solutions to hard combinatorial optimization problems. Local search algorithms are simple to implement and quick to execute, but they have the main disadvantage that they are locally convergent.

When we apply a local search algorithm to the TSP, the common opinion about local optima is that the set of local optima forms a “big valley” structure in the solution space [3], [13], [15], [19]. In fact, it is a *solution attractor*, i.e., a set of fixed points, that drives the local search trajectories into the small region of the solution space [12]. The attractor is formed by the set of all local optimal tours. Since the global solution is a special case of local optimal solutions, the global tour is expected to be included in the attractor.

Li [12] suggests a procedure for constructing a solution attractor for single-objective TSP. For a TSP instance, the solution space contains the tours that the salesman may traverse. Li's procedure uses an $n \times n$ matrix E, called *hit-frequency matrix*, to record the number of hits on each edge by a set of local optimal tours. The hit-frequency matrix explores the information on edges and thus stores rich information about the solution attractor for the TSP instance. If we use a local search algorithm and generate all possible search trajectories for the TSP instance, when all search trajectories reach their local optima we could obtain the real attractor for the problem. Then, when all involved edges are recorded in the hit-frequency matrix, we should immediately recognize the attractor and easily identify the global optimal tour. Unfortunately, this "all possible search trajectories" scenario is unrealistic, due to the enormous amount experimental data required. A more realistic goal would be gathering a moderate sample of local optima to construct the solution attractor and infer statistical properties of the attractor.

We denote all edges that are contained in the global optimal tour as G-edges. When a local search process searches for an optimal solution, the search trajectory is constantly adjusting itself by disregarding unfavorable edges and trying to collect G-edges. If it successfully collects all of the G-edges, the final tour is the global optimal tour. If it only collects some of the G-edges, it ends up at a local optimum. If a tour contains none of the G-edges, the search process can always improve the tour by exchanging edges. Local heuristic algorithms cause individual search trajectories explore only a tiny fraction of the enormous solution space when n is large. Thus, it is difficult for a particular search trajectory to select G-edges globally, and a search

trajectory often ends at a local optimum which contains most *G*-edges and some unfavorable edges. The more *G*-edges a local optimal tour contains, the closer to the global optimum it is. Local optimal tours are actually linked together by sharing the *G*-edges. When a solution attractor is constructed by a large number of local optimal tours, the attractor should consist of all *G*-edges and some unfavorable edges, called *noise*.

```
procedure TSP-Attractor(Q)
begin
 repeat
    s_i = Initial_tour();
    s_j = Local_Search(s_i);
    Update(E(s_j));
 until StoppingCriterion = M;
 A = Find_Core(E);
   Exhausted_Search(A)
end
```

Fig. 2. The procedure for contracting solution attractor of local search in TSP

Fig. 2 presents the procedure for constructing solution attractor for TSP. The procedure is very straightforward: randomly generating a sample of local optimal solutions to construct the attractor, and then removing the noise contained in the attractor to identify the core of the attractor. In the procedure, Q is a TSP instance. s_i is an initial solution generated by `Initial_Tour()`. s_j is a local optimum outputted by a local search process `Local_Search()`. E is the hit-frequency matrix to record M local optima. Each time when a search trajectory reach a local optimal solution s_j, the function `Update()` records the edges contained in s_j into E. Since a solution attractor contains *G*-edges and noise, the function `Find_Core()` is used to try to remove the noise. The remaining edges form the core of the attractor and are recorded into the matrix A. Finally, the matrix A is searched by an exhausted enumeration process `Exhausted_Search()` to generate all solutions in the attractor core. The hit-frequency matrix E plays an important role for collecting all information about the solution attractor. It acts as an input/output data table where the entry e_{ij} records the number of times that the corresponding edge is hit by the set of local optimal tours.

A solution representation is a mapping from the solution space of a possible solution to a solution space of encoded configuration within a particular data structure. For a TSP with n cities, there are many ways to represent a tour in the computer. One way is to make an ordered list of the cities to visit, with a return to the home city being implied. Another way is with an $n \times n$ matrix $E = [e_{ij}]$, such that $e_{ij} = 1$ if and only if city j follows city i in the tour. A tour therefore must always have exactly one "1" in every row and every column. The matrix E is an effective data structure that allows the local search process to maintain a functional link among the local optimal tours. The basic idea of the hit-frequency matrix E used in the procedure is to build a probabilistic representation of the solution attractor based on the M local optimal tours and then generate new candidate solutions based on the knowledge contained in the attractor.

The hit-frequency value in e_{ij} represents the probability that the corresponding edge in the TSP will be hit by a local optimal tour. If M is large enough, this value also can be viewed as the probability that the corresponding edge is a G-edge. For example, if an edge is hit by 73 percent of M local optimal tours, although each local optimum selects the edge based on its neighborhood structure, the edge is globally superior since the edge is reached by these individual optima from different search trajectories. The hit-frequency matrix gives us important insights into the nature of the search space and provides an opportunity for us to concentrate the search in the region that contains the most promising solutions. When we restrict our attention to a smaller solution space represented by the attractor, the number of possibilities is no longer prohibitive.

The hit-frequency matrix E has capacity of learning. The basis of learning in the matrix is the generation of long-lived memory and statistics-based pattern. The matrix not only plays a fundamental role in the organization of memory, but also provides a powerful way of discovering the pattern in the searched edges. We can exploit this information to generate more local optimal tours, and even the global optimal tour.

Fig. 3 uses a simple 20-city TSP example to explain the attractor-construction procedure. This example generates $M = 100$ random initial tours. Since these initial tours are randomly produced, the edges should have an equal probability to be selected. The darkened elements in the matrix shown in Fig. 3(a) represents the union of the edges found in these initial tours. After applying the 2-opt local search algorithm to the initial tours, we obtain 100 local optimal tours. Fig. 3(b) marks the union of the edges hit by these 100 local optimal tours. Each of these marked elements also contains a value which is the number of hit by the local optimal tours.

It is interesting to see how the search space is reduced. During the local search process, the only thing the process is doing on a particular search trajectory is to replace bad edges with good ones. As result, the search process causes the elements in the matrix E to have unequal hit frequency. Good edges are selected by many search trajectories; bad edges are displaced and therefore contain low or zero hit frequency. After search trajectories reach their local optimal points, they leave their "final footprints" in E. The darken area in Fig. 3(a) is reduced to the one shown in Fig. 3(b), which exhibits the structure of solution attractor for the TSP. Comparing to the full solution space, the size of the attractor is very small.

The attractor constructed by local optima contains G-edges and noise. The function `Find_Core()` groups the edges into clusters in an attempt to remove the noise. A *cluster* is defined here as a set of edges that contain the hit frequency within a certain range. The cluster with the highest range of hit frequency constitutes the core of the attractor, while the cluster with the lowest range can be regarded as noise. In each column (or row) of the hit-frequency matrix E, the value of the maximum hit $MaxV$ is identified. Knowing that the range of possible value for an edge in that column can vary from zero to $MaxV$, we could divide this range into r equal portions. Our example chooses $r = 3$ to cluster the edges in each column. Fig. 3(c) illustrates the clustering process for the column 18 in E. Fig. 3(d) displays the cluster in which the edges are within the highest range of hit frequency. The darkened elements form the attractor core. In this way the attractor is further reduced into a core, an even smaller region. The most-hit edges in the core form the most promising region for search. Now

it is possible to use an exhausted-enumeration algorithm to find all solutions in the core. In our example, the function `Exhausted_Search()` found 32 solutions in the core.

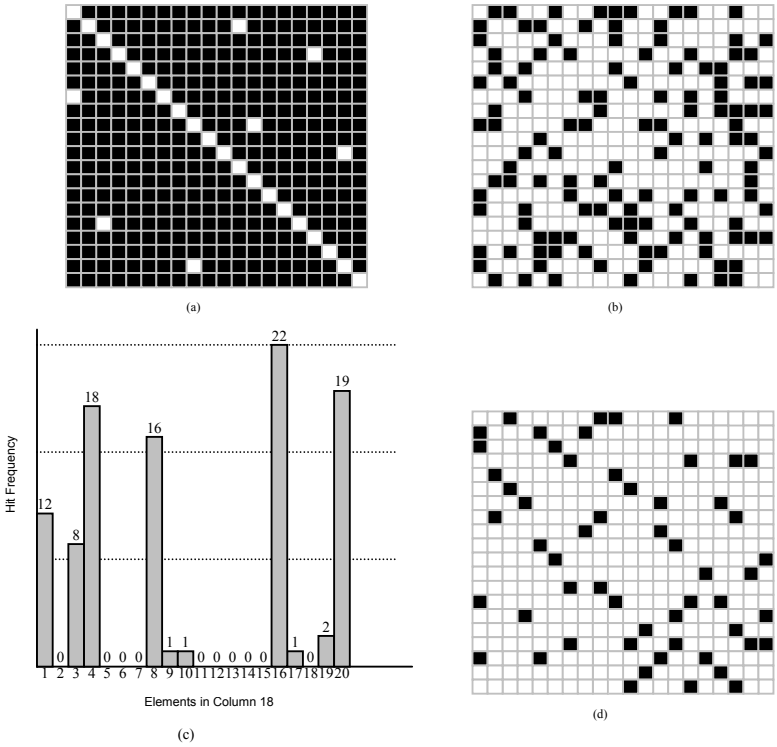

Fig. 3. A 20-city TSP example for illustrating attractor construction. (a) presents the union of the edges in the initial tours; (b) marks the union of the edges hit by the local optimal tours; (c) illustrates the clustering process for the column 18 of *E*; and (d) displays the attractor core.

3 Finding Pareto-Optimal Set for a Bi-Objective TSP

3.1 The TSP and Search Procedure

In a multi-objective setting, the TSP becomes even more difficult and complex. The general multi-objective TSP can be formulated as follows:

$$\min f(x) = \left\{ \begin{array}{l} f_1(x) = c_1(n,1) + \sum_{i=1}^{n-1} c_1(i,i+1) = z_1 \\ f_2(x) = c_2(n,1) + \sum_{i=1}^{n-1} c_2(i,i+1) = z_2 \\ \vdots \\ f_k(x) = c_k(n,1) + \sum_{i=1}^{n-1} c_k(i,i+1) = z_k \end{array} \right\} \quad \textbf{(3)}$$

$$\textit{subject to} \quad x(1,2,\ldots,n) \in X$$

where n is the number of cities, $c_q(i, j)$ is the cost between city i and j according to the q-th objective, $q = 1,\ldots k$, and the decision variable x holds a cyclic permutation of the n cities. In practical applications the cost factors may correspond to distance, travel time, expenses, tourist attractiveness, energy consumed, degree of risk, or other relevant considerations for the tour. The goal is to find the "minimal" Hamiltonian circuit of the graph in terms of Pareto optimality. In a TSP, if the cost weights of the edges satisfy the triangle inequality, the problem is called the *metric TSP*. A special case is when the cities are points on the plan, and the cost weights are the Euclidean distances between the points. When the cost weights satisfy $c(i, j)=c(j, i)$, it is called the *symmetric TSP*, which has many practical applications.

The design of a test problem is always important in designing any new search algorithm. The context of problem difficulty naturally depends on the nature of problems that the underlying algorithm is trying to solve. In the context of solving multi-objective optimization problems, we are interested in designing the features that makes a problem difficult for the proposed multi-objective optimization algorithm. In this study, the test problem instance is designed based on several considerations. First, the size of problem should be large, since the TSP instances as small as 200 cities must now be considered to be well within the state of the global optimization art. The instance must be considerably larger than this for this study to be sure that the proposed approach is really called for. Second, the instance should be multi-modal, that is, with many local Pareto-optimal regions. Third, there is no any pre-known information related to the result of the experiment, since in a real-world problem one does not usually have any knowledge of the Pareto-optimal front. Fourth, the problem instance should be general, understandable and easy to formulate so that the experiments are repeatable and verifiable, but difficult to solve.

This study generates a general symmetric TSP instance, which consists of $n = 1000$ cities with two cost matrixes c_1 and c_2. The cost matrices are generated at random, where each cost element $c(i, j) = c(j, i)$ is assigned a random number in the interval [1, 1000]. This study uses two objectives primarily because of the ease in which two-dimensional Pareto-optimal front can be visually demonstrated.

Fig. 4 presents a general search procedure used in this study. Q is a TSP instance with a set of cost matrices $\{c_1, c_2,\ldots, c_k\}$ with respect to objective $f_1, \ldots, f_k$. The function `TSP_Attractor()` finds the core of the attractor for each of objective functions respectively. This study uses the 2-opt algorithm [13], which is one of the earliest local search algorithms for the TSP.

```
procedure TSP_Pareto_Set(Q)
begin
 q = 1;
 while (q ≤ k) do
    E_q = TSP_Attractor(c_q);
    q = q + 1;
 end while
 E = Merge(E_1,…,E_k);
 L = Find_Pareto_Set(E);
 Analize_Pareto_Set(L);
end
```

Fig. 4. The procedure for searching Pareto-optimal set in multi-objective TSP

The information about the attractor core for objective q (q = 1,…, k) is stored in matrix E_q, in which we mark the elements to represent the corresponding edges that are hit by the core. In this study, only 15 best solutions in each attractor core are selected and stored in E_q. And also the objective vector values $Z_q(z_1,\ldots,z_k)$ for the best solution in the E_q is calculated. The vector $Z_q(z_1,\ldots,z_k)$ can help us to determine the ideal objective vector z^* and nadir objective vector z^{nad}. After k attractor cores with respect to objective f_1, …, f_k are constructed, the function `Merge()` stores the union of all marked elements in the matrixes $E_1,\ldots,E_k$ into the matrix E. Then the function `Find_Pareto-Set()` finds all solutions in E through an exhausted search method, and outputs all *non-dominated solutions* into list L. Finally, the function `Analize_Paroto_Set()` analyzes the Pareto set and generate the information about each of the solutions.

This procedure intuitively reflects the idea of finding solutions around the extreme ends of the Pareto-optimal front and then mixing their characteristics (edges) to find other trade-off solutions in the Pareto-optimal region. By considering each of objective functions separately, this procedure generates high-quality solutions in the solution attractor corresponding to that objective. The merge of these attractors will form a well-distributed set of the Pareto-optimal solutions, each of which takes part of high-quality edges from each of the attractors. Fig. 5 illustrates examples. Suppose that there are two local optimal tours (1,2,3,4,5,6) and (5,1,2,4,3,6) in the attractor 1 that corresponds to objective f_1 (see Fig. 5(a)), and there is one local optimum (5,3,1,4,2,6) in the attractor 2 corresponding to objective f_2 (Fig. 5(b)). If we merge these two attractors into one matrix, as illustrated in Fig. 5(c), we can identify other two solutions (1,2,4,5,3,6) and (1,4,2,3,6,5). The first solution takes four edges from attractor 1, one edge from attractor 2 and one common edge shared by both attractors. The second solution takes three edges from attractor 1, one edge from attractor 2 and two common edges. Even in the case in which two objectives are mutually exclusive (i.e., the two cost matrixes are mutually disjunctive), we will see that solutions in different attractors do not share edges, but we still can find the solutions that mix edges from attractor 1 and 2. For example, suppose that there are two solutions (1,3,5,2,6,4) and (1,3,6,4,2,5) in attractor 1 (Fig. 5(d)) and one solution (1,2,3,4,5,6) in attractor 2 (Fig. 5(e)). When we merge these two attractors into a matrix, as shown

in Fig. 5(f), it is easy to see that no common edge is shared by both attractors. In addition to the three original solutions, we can find other four new solutions (1,2,3,5,6,4), (1,2,3,6,4,5), (1,3,4,2,5,6) and (1,3,4,5,2,6), each of them combines three edges from attractor 1 and three edges from attractor 2.

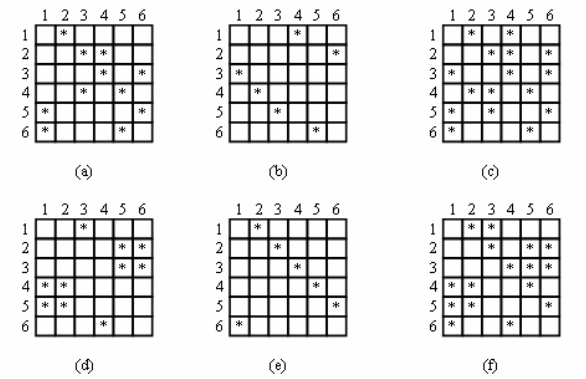

Fig. 5. Examples of merging two attractors

3.2 The Experiment Results

To guarantee the diversity of the sample of local optima, this study generated M = 2000 initial points, and, as a result, generated 2000 local optima for each objective. This experiment also relied heavily on randomization. All initial tours were randomly constructed. In the 2-opt local search, the algorithm randomly generated a solution in the neighborhood of the current solution. A move that gave the first improvement was chosen. The great advantage of first-improvement pivoting rule is to produce randomized local optima. The local search process on each search trajectory terminated when no improvement had been achieved during 1000 iterations.

This study used two 1000×1000 matrixes, E_1 and E_2, to store 15 best solutions taken from each of the attractor cores, respectively. Of course, the number of solutions selected from the attractor core affects the size of the Pareto-optimal set, the coverage of the set, and the computational resources needed to generate the set. Fig. 6 illustrates the solution points from each objective in the objective space. This figure also indicates the lower bound z^* and upper bound z^{nad} on the Pareto-optimal set to display the topology of the set.

Then these solutions were combined into the matrix E, in which 40.4% of the marked edges belong to attractor 1, 41.1% to attractor 2, and 18.5% are shared by both attractors. The fraction of edges that are common to both attractors can be defined as the overlap between the two attractors. This study then used an exhausted search algorithm to find all solutions in E. After discarding all dominated solutions, we obtained 31 Pareto-optimal solutions, as illustrated in Fig. 7.

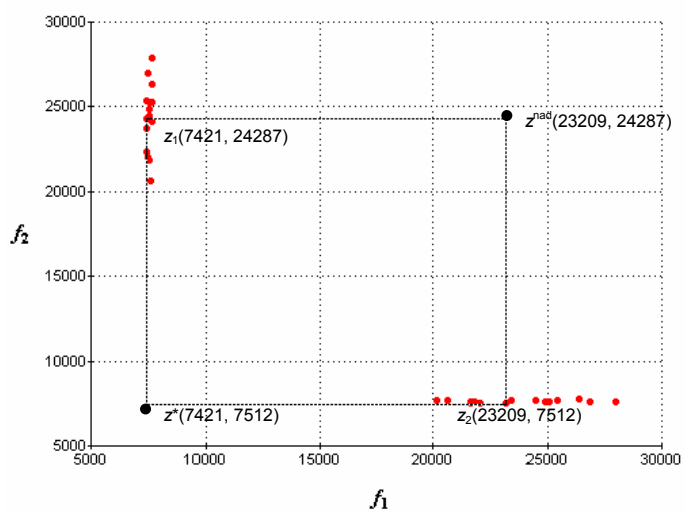

Fig. 6. Solutions in each of the attractor core

For the purpose of comparison, this study also applied the aggregation approach to the same TSP instance. This study varied systematically the weights in the fashion of [0, 1], [0.04, 0.96], [0.08, 0.92], …, [1, 0], and then use the 2-opt algorithm to search on the aggregated objective function. The search process in each run terminated when no improvement had been made during 1500 iterations. These multiple runs generated 26 points, among which 23 points were nondominated. These nondominated points are also displayed in Fig. 7. It is clear that the solutions generated by the proposed procedure maintain good properties of convergence and diversity. These solutions probably provide a more accurate description of the true Pareto-optimal set.

Mathematically, the multi-objective optimization problem is considered to be solved when the Pareto-optimal set is found, and all the Pareto-optimal points are equally acceptable solutions to the problem. However, it is not enough in many practical cases. The ultimate goal is to select the single *best compromise solution*. Selecting one solution out of the Pareto-optimal set calls for a decision-maker, who has better insight into the problem and can express preference relations between different objectives. However, finding a reliable important solution is difficult in the absence of any knowledge about the Pareto-optimal solutions. An important question related to this issue is how to present the Pareto-optimal solutions to the decision-maker in a meaningful way. This requires a multi-objective optimization algorithm not only to be capable of finding multiple and diverse Pareto-optimal (or near Pareto-optimal) solutions, but also to be able to provide necessary information about the obtained solutions. In our case, what would be more desirable is the information about each obtained tour. More specifically, we want to know that in a particular tour, what percentage of edges comes from attractor 1, what percentage from attractor 2, and what percentage are shared by both attractors. This kind of information can aid the deci-

sion-maker in arriving at a final decision. The characteristics of the solutions are essential decision elements when people look for the best compromise solution, and they are implicitly included in the common-sense notion of optimality.

The function `Analize_Pareto_Set()` calculated the distribution of edges for each obtained tour, as illustrated in Table 1. For instance, among 1000 edges in the tour 1, 28 edges are shared by both attractors, and other 972 edges are taken from attractor 1. For the tour 12, 164 edges are shared by both attractors, and among other edges, 574 edges are taken from attractor 1 and 262 edges are taken from attractor 2. It can be interpreted as: If we choose the tour 12, it will correspond to 57.4% preference of the objective f_1, 26.2% preference of the objective f_2, and 16.4% preference of both objectives at same time. However, if we choose the tour 1, 97.2% of this solution will satisfy the objective f_1, and 2.8% will satisfy both objectives simultaneously. No doubt, such information is useful to the decision makers for comparing multiple optimal solutions and choosing the best compromise solution, as and when required.

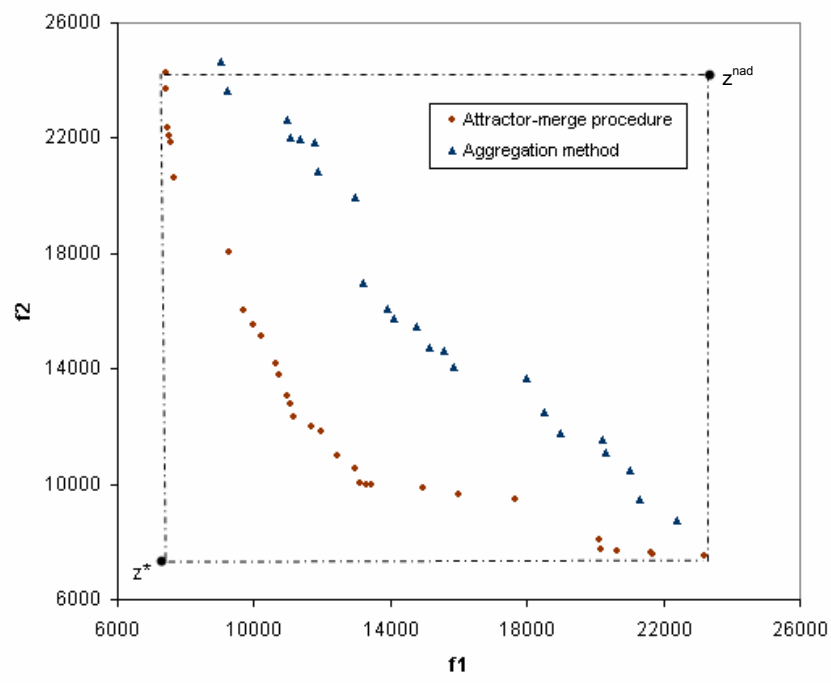

Fig. 7. The Pareto-optimal sets for the test TSP

Table 1. Characteristics of the obtained Pareto-optimal set

	Solution Value for		Edges from		
Tour #	Objective 1	Objective 2	Attractor 1	Attractor 2	Shared
1	7421	24287	972	0	28
2	7435	23731	967	0	33
3	7465	22343	934	0	66
4	7529	22060	919	0	81
5	7551	21838	906	0	94
6	7645	20629	893	0	107
7	9285	18021	744	97	159
8	9698	16001	698	153	149
9	9963	15530	661	172	167
10	10205	15120	634	214	152
11	10648	14205	587	248	165
12	10720	13789	574	262	164
13	10978	13075	531	312	157
14	11054	12775	512	334	154
15	11155	12335	495	353	152
16	11674	12001	446	399	155
17	11973	11847	433	401	166
18	12456	11007	409	425	166
19	12986	10548	374	452	174
20	13099	10048	329	507	164
21	13294	10001	286	552	162
22	13455	9987	260	577	163
23	14976	9877	227	615	158
24	15987	9645	205	644	151
25	17654	9465	147	702	151
26	20101	8073	14	871	115
27	20170	7719	0	901	99
28	20651	7697	0	906	94
29	21644	7639	0	917	83
30	21675	7553	0	933	67
31	23209	7512	0	964	36

4 Conclusion

The multi-objective nature of most real-world problems makes multi-objective optimization a very important research topic. The increasing complexity of typical search space demands new strategies in solving multi-objective optimization problems. This study provides one possibility. This study shows that the use of simple local search, together with an effective data structure, can identify high quality Pareto-optimal solutions. Although the procedure was applied to a bi-objective TSP in this study, it can be expected that, with little modification, this procedure can be used to deal with

the TSP with three or more objectives. Even if generalization cannot be claimed, this work provides a new search strategy that casts new light into multiple-objective optimization and might serve as a basis to build new algorithm for solving other multiple objective problems.

References

1. Angel, E., Bampis, E., Gourvès, L.: A Dynasearch Neighborhood for the Bicriteria Traveling Salesman Problem. In: Gandibleux, X., Sevaux, M., Sörensen, K., T'kindt, V. (eds.): Metaheuristics for Multiobjective Optimization, Lecture Notes in Economics and Mathematical Systems 535. Springer-Verlag Berlin (2004) 153-176.
2. Borges, P. C., Hansen, M. P.: A Study of Global Convexity for a Multiple Objective Traveling Salesman Problem. In Ribeiro, C. C., Hansen, P. (Eds.): Essays and Surveys in Metaheuristics. Kluwer Academic Publishers Norwell, MA (2002) 129-150.
3. Boese, K. D., Kahng, A. B., Muddu, S.: A New Adaptive Multistart Technique for Combinatorial Global Optimizations. Operations Research Letters 16:2 (1994) 101-113.
4. Collette, Y., Siarry, P.: Multiobjective Optimization – Principles and Case Studies. Springer-Verlag Berlin (2003).
5. Ehrgott, M.: Multicriteria Optimization. Springer-Verlag Berlin (2000).
6. Ehrgott, M.: Approximation Algorithms for Combinatorial Multi-Criteria Problems. International Transactions in Operations Research 7 (2000) 5-31.
7. Ehrgott, M., Gandibleux, X.: A Survey and Annotated Bibliography of Multiobjective Combinatorial Optimization. OR Spectrum 22:4 (2000) 425-460.
8. Fisher, R., Richter, K.: Solving a Multiobjective Traveling Salesman Problem by Dynamic Programming. Mathematische Operationsforschung und Statistik, Series Optimization 13:2 (1982) 247-252.
9. Gupta, A., Warburton, A.: Approximation Methods for Multiple Criteria Traveling Salesman Problems. In: Sawaragi, Y. (ed.): Towards Interactive and Intelligent Decision Support Systems: Proceedings of the 7th International Conference on Multiple Criteria Decision Making. Springer-Verlag Berlin (1986) 211-217.
10. Hansen, M. P.: Use of Substitute Scalarizing Functions to Guide a Local Search Based Heuristics: The Case of MOTSP. Journal of Heuristics 6 (2000) 419-431.
11. Jaszkiewicz, A.: Genetic Local Search for Multiple Objective Combinatorial Optimization. European Journal of Operational Research 137:1 (2002) 50-71.
12. Li, W.: Attractor of Local Search in the Traveling Salesman Problem. Journal of Heuristics, Forthcoming.
13. Lin, S.: Computer Solutions to the Traveling Salesman Problem. Bell Systems Technical Journal 44 (1965) 2245-2269.
14. Melamed, I. I., Sigal, I. K.: The Linear Convolution of Criteria in the Bicriteria Traveling Salesman Problem. Computational Mathematics and Mathematical Physics 37:8 (1997) 902-905.
15. Mezard, M., Parisi, G.: A Replica Analysis of the Traveling Salesman Problem. Journal of Physique 47 (1986) 1285-1296.
16. Miettinen, K. M.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers Dordrecht (1999).
17. Paquete, L. Chiarandini, M., Stützle, T.: Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem: An Experimental Study. In: Gandibleux, X., Sevaux, M., Sörensen, K., T'kindt, V. (eds.): Metaheuristics for Multiobjective Optimization, Lecture Notes in Economics and Mathematical Systems 535. Springer Berlin (2004) 177-199.

18. Paquete, L., Stützle, T.: A Two Phase Local Search for the Biobjective Traveling Salesman Problem. In: Fonseca, C. M., Fleming, P. J., Zitzler, E., Deb, K., Thiele, L. (eds.): Evolutionary Multi-Criterion Optimization, Proceedings of Second International Conference, EMO2003. Springer, Berlin (2003) 479-493.
19. Raidl, G. R., Kogydek, G., Julstrom, B. A.: On Weight-Biased Mutation for Graph Problems. In: Merelo-Guervós, J. J., Adamidis, P., Beyer, H. G., Fernáandez-Villacaňas J. L., Schwefel, H. P. (eds.): Parallel Problem Solving from Nature: PPSN VII, LNCS2439. Springer Berlin (2002) 204-213.
20. Sigal, I. K.: Algorithm for Solving the Two-Criterion Large-scale Traveling Salesman Problem. Computational Mathematics and Mathematical Physics 34:1 (1994) 33-43.
21. Steuer, R. E.: Multiple Criteria Optimization – Theory, Computation and Application. John Wiley & Sons New York (1986).
22. Tan, K. C., Lee, T. H., Khor, E. F.: Evolutionary Algorithms for Multi-objective Optimization: Performance Assessments and Comparisons. Artificial Intelligence Review 17 (2002) 253-290.
23. Tung, C. T.: A Multicriteria Pareto-optimal Algorithm for the Traveling Salesman Problem. Asia-Pacific Journal of Operational Research 11 (1994) 103-115.
24. Yan, Z., Zhang, L., Kang, L., Lin, G.: A New MOEA for Multi-objective TSP and Its Convergence Property Analysis. In: Fonseca, C. M., Fleming, P. J., Zitzler, E., Deb, K., Thiele, L. (eds.): Evolutionary Multi-Criterion Optimization, Proceedings of Second International Conference, EMO2003. Springer Berlin (2003) 342-354.