FACHHOCHSCHULE TRIER
Hochschule für Technik, Wirtschaft und Gestaltung
University of Applied Sciences
Informatik

Informatik-Bericht Nr. 2007-4

Schriftenreihe Fachbereich Informatik, Fachhochschule Trier

First Come, First Served – Tour Scheduling with Priorities

Heinz Schmitz and Sebastian Niemann

Fachhochschule Trier, Department of Computer Science,
Schneidershof, D–54293 Trier, Germany

Abstract. This paper introduces a bicriteria version of the classical Traveling Salesman Problem (TSP) which is motivated by various applications in the context of service delivery. The additional objective allows to take priorities among locations into account while minimizing the costs of traveling. For this, cities in the input are given in a strict ordering, e.g., due to arrivial times of delivery requests.
After making the notion of priorities precise, we present a local-search algorithm to approximate the set of non-dominated solutions. While still being conceptionally easy, our algorithm employs different means of intensification and diversification in a way we call *breadth-first local search.* We maintain one candidate solution for each possible value of the additional objective in a polynomially-sized archive, and try to improve this set towards the Pareto front. Experimental results with test data from TSPLIB show that this is a reasonable approach to attack the problem.

Keywords. Multi-objective discrete optimization, bicriteria traveling salesman problem, local search, metaheuristic.

1 Introduction

The classical Traveling Salesman Problem (TSP) has numerous real-world applications, for a recent overview we refer to [1]. A typical one is the minimum-cost tour scheduling to fulfill delivery requests from different locations.

Example 1. Let $\{1, \ldots, m\}$ be a set of customer locations such that the earliest request is from location 1, the second earliest from 2 and so on. Obviously, minimizing traveling costs is a reasonable objective for the shipping company. However, if delivery time is crucial, then the tour $(m, m-1, \ldots, 1)$ cannot be considered optimal from a customers perspective since requests are fulfilled in reverse order.

One way around this is to incorporate the *first-come-first-served*-policy. If locations have priorities according to arrivial times of delivery requests one can additionally try to maintain this ordering while minimizing traveling costs. Obviously, both objectives are conflicting in general. Due to the many applications of TSP there are also other settings where the new objective emerges in a natural way. We have encountered the following situation in a real-world project, where we used the approach presented in this paper.

Example 2. The manager of a single-machine production line wants to schedule tasks such that the overall makespan is minimized. Since there are sequence-dependent setup times this is equivalent to solving TSP instances. On the other hand, there also needs to be achieved some level of service for different sales departments that place orders. To avoid lengthy discussions about what jobs are more important than others, all participants agreed on the *first-come-first-served*-policy. So in fact, the production-line manager needs to solve TSP instances with priorities, i.e., trade-offs between both objectives have to be balanced.

Once priorities are considered in general, they can also be used to implement other preferences, e.g., due to some customer-specific or order-specific properties.

Our Contribution. Motivated by the above examples we give a formal definition of the *Traveling Salesman Problem with Priorities* (TSPwP) which is a straightforward and natural extension of classical TSP (Section 2). We observe some easy facts about this bicriteria problem.

Next we design a local-search algorithm called *breadth-first local search* that combines the two typical means of heuristic search, i.e., intensification and diversification, in a novel way (Section 3). To intensify the search we explore the neighborhood of an ordered archive of candidate solutions using different neighborhood structures and a variable search-depth (Section 3.1). The polynomially-sized archive is such that it keeps one candidate solution for each possible value of the additional objective. A well-balanced amount of diversification is achieved using tabu lists together with a random choice between two problem-specific perturbation operators (Section 3.2). Experimental results suggest that this is a promising approach to solve TSPwP (Section 4).

Additionally, our algorithm can be seen as an implementation of a more abstract algorithmic pattern that can be used to solve similar bicriteria problems, namely problems where the range of at least one objective function is polynomially bounded in the input size.

Related Work. Our algorithmic approach is in the same line with other recently published variants of local search, such as Pareto Local Search [2] and Pareto Iterated Local Search [3], but it also differs in a number of aspects. On one hand we also incorporate perturbation operators [4], we make use of multiple neighborhood structures [5], and we keep an archive of candidate solutions.

On the other hand we exploit that the range of the additional objective function is polynomially bounded to obtain a *dense* approximation for every such value, and we do not restrict the archive to locally Pareto optimal solutions. Moreover, we combine the mentioned aspects with tabu lists which is a well established method in combinatorial optimization [6]. It has been successfully applied to classical TSP as well as to other multiple-objective problems, e.g. [7, 8]. Other multiple-objective variants of TSP that have been studied in the literature mainly consider multiple cost matrices, e.g. [9].

2 Problem Statement

We define the *Traveling Salesman Problem with Priorities* (TSPwP) as follows. As in case of classical TSP an instance $x = (m, C)$ with m cities consists of some cost matrix $C = (c_{i,j})_{m \times m}$ with $c_{i,j} \in \mathbb{N}$. A solution for x is any permutation $t = (a_1, \ldots, a_m)$ of $\{1, \ldots, m\}$ having costs

$$z_1(t) = c_{a_m,a_1} + \sum_{i=1}^{m-1} c_{a_i,a_{i+1}}.$$

Additionally we assume w.l.o.g. that cities are labeled according to some priority rule, i.e., the city with highest (lowest) priority has label 1 (respectively, m). No extra input data is needed. To measure violations of these priorities we define the penalty resulting from the i-th city on tour t as $p_i = \max\{i - a_i, 0\}$. E.g., if $a_5 = 3$ then city with priority 3 is visited in fifth place and hence $p_5 = 2$. As the second objective function we set

$$z_2(t) = \sum_{i=1}^{m} p_i.$$

Note that this takes the quantity of each penalty into account. In analogy to the tardiness measure in machine scheduling one could also count the number of non-zero penalties or minimize the largest penalty. We do not study these alternatives here.

It is easy to see that $t = (1, 2, \ldots, m)$ is the unique tour with $z_2(t) = 0$, and that

$$z_2(m, m-1, \ldots, 1) = \begin{cases} (m^2 - 1)/4 & \text{if } m \text{ is odd} \\ m^2/4 & \text{otherwise.} \end{cases}$$

Since no solutions with larger z_2-values exist, we have for all tours t that

$$0 \leq z_2(t) \leq m^2/4. \tag{1}$$

For notational convenience assume that m is even for the remainder of this paper.

We associate with every tour t the vector $z(t) = (z_1(t), z_2(t))$ where both components need to be minimized. As is common in multicriteria optimization we say a tour t *dominates* t' if $z_1(t) \leq z_1(t')$, $z_2(t) \leq z_2(t')$ and $z(t) \neq z(t')$. If there is no t that dominates t' we call t' *Pareto-optimal* or *efficient*. Observe from (1) that for each input x the number of Pareto-optimal solutions t with pairwise different vectors $z(t)$ is polynomially bounded in the length of x. The optimization goal for instances of TSPwP is to compute such a set of Pareto-optimal tours.

It is easy to see that TSPwP is not a special case of bicriteria TSP where a second cost matrix is given. Just note that usually $z_2(t) \neq z_2(t')$ if t' is a cyclic shift of the permutation t. There are also some similarities to the single-machine scheduling problem $1|s_{fg}|\#(C_{\max}, \sum T_j)$ with sequence-dependent setup times where the overall makespan and the total tardiness both need to be minimized. However, it is not clear how a reduction from TSPwP to this problem can be achieved such that the quality of solutions is preserved.

3 Algorithm Design

We describe the main design ideas of our algorithm. The overall structure is rather simple: We keep an archive A of candidate solutions, and try to improve its quality via alternation of intensifying and diversifying phases during the search. To organize this, we instantiate the pool template [10, 11] (Algorithm 2) to control the behaviour of an iterated local-search procedure (Algorithm 3).

To be more precise, solutions in the polynomially-sized set $A = (t_0, t_1, \ldots)$ always have the property that

$$z_2(t_k) = k \text{ for } 0 \leq k \leq m^2/4. \tag{2}$$

Hence we can understand A as a *dense* approximation of the Pareto front since it contains one candidate for each possible value in the range of function z_2. It is not until the final step of the algorithm that a (locally) efficient set of solutions is extracted from the set of best solutions that appeared during the search.

There is a straightforward way to generate a first version of the archive such that (2) holds. Starting with $(1, 2, 3, \ldots, m)$ we move the first city to the last position in the permutation via successive transpositions that increase the penalty one-by-one. Then we move city 2 in $(2, 3, \ldots, m, 1)$ to the second last position resulting in $(3, \ldots, m, 2, 1)$. This is repeated until we finally get $(m, \ldots, 3, 2, 1)$. Note that not every transposition during this procedure strictly increases the penalty, e.g., when turning $(2, 3, \ldots, m, 1)$ into $(3, 2, \ldots, m, 1)$.

For $t = (a_1, \ldots, a_m)$ denote by $exchng(t, i, j)$ the transposition of a_i and a_j. Then we can state the following algorithm.

Algorithm 1: `init`()

```
begin
    $t := (1, 2, \ldots, m)$;
    $t_0 := t$; $A := \{t_0\}$;
    $k := 0$;
    for $lastpos := m$ downto 2 do
        for $pos := 1$ to $(lastpos - 1)$ do
            $t := exchng(t, pos, pos + 1)$;
            if $z_2(t) = k + 1$ then
                $k := k + 1$;
                $t_k := t$; $A := A \cup \{t_k\}$
            end
        end
    end
    return $A$
end
```

Next we describe what the instantiation of the pool template looks like. One kind of diversification we use is a collection T of tabu lists $T(k)$ for each $0 \leq k \leq m^2/4$. We take these lists to ensure that a subsequent iteration of the local search yields an archive A' with solutions that all have z_1-values different from the ones of previous iterations. So if $A = (t_0, t_1, \ldots)$ is the content of the archive j iterations ago, we let

$$T(k) = (z_1^1, \ldots, z_1^l) \text{ with } z_1^j = z_1(t_k).$$

By storing values instead of solutions every tabu-list entry excludes numerous other tours. The duration of this effect can immediately be controlled by the length-parameter l. It is one out of just two search parameters, the other one being the number of iterations for the halting condition.

After initializing the archive and the tabu lists, we repeatedly call the local-search procedure `intensify`, we remember the best solutions found so far, update the tabu lists and we apply one out of two randomly-chosen perturbation operators. Inspired by the way intensification is organized (left-right sweeps, see next subsection) we call our approach *breadth-first local search* (BFLS). Together, we have the following algorithmic pattern.

Algorithm 2: `BFLS`(*maxIterations*, l)

```
begin
    $A :=$ init();
    init $T$ with $T(k) := \emptyset$ for $0 \le k \le m^2/4$;
    repeat
        $A'$:=intensify ($A$, $T$);
        update bestArchive with $A'$;
        update $T$ with $A'$;
        choose $r \in \{1, 2\}$ randomly;
        $A := \mathcal{O}_r(A')$
    until maxIterations reached ;
    return efficient solutions in bestArchive
end
```

A detailed description of the `intensify`-procedure is given in Section 3.1 below, while the perturbation operators $\mathcal{O}_1$ and $\mathcal{O}_2$ are explained in Section 3.2. They form the second kind of diversification we use in the algorithm.

For $A = (t_0, t_1, \ldots)$ denote its coordinates in objective space as $z(A) = (z(t_0), z(t_1), \ldots)$. Then the progress of `BFLS` can be depicted as shown in Fig. 1.

3.1 Intensification

The execution of a single call of the `intensify`-procedure starts with an archive A and performs several left-right sweeps. During each sweep neighborhoods $\mathcal{N}(t_k)$ of $t_k \in A$ for $k = 0, 1, \ldots, m^2/4$ are completely investigated in this order. Whenever a tour $t \in \mathcal{N}(t_k)$ with $z_2(t) = k'$ is found such that

$$z_1(t) \notin T(k') \text{ and } z_1(t) < z_1(t_{k'}) \qquad (3)$$

for some $t_{k'} \in A$, then $t_{k'}$ is replaced by t (see Fig. 2).

It has been observed in the literature that the combined use of different neighborhood functions yields better results compared to a single function [5, 3].

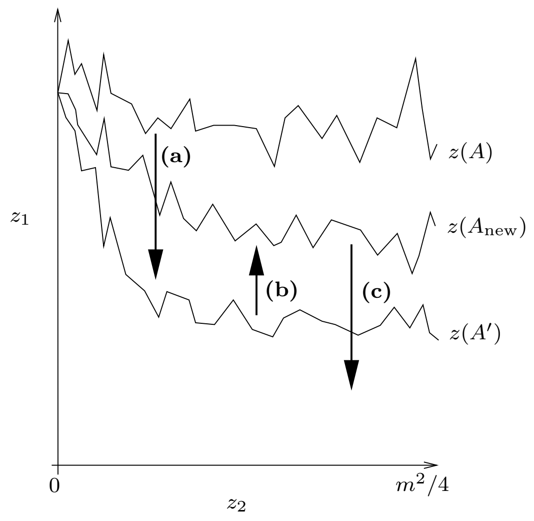

Fig. 1. Every iteration has an `intensify`-step (a) followed by the application of a perturbation operator (b). Due to forbidden values, the next iteration (c) returns a vector $z(A'_{\mathrm{new}})$ which is componentwise different from $z(A')$.

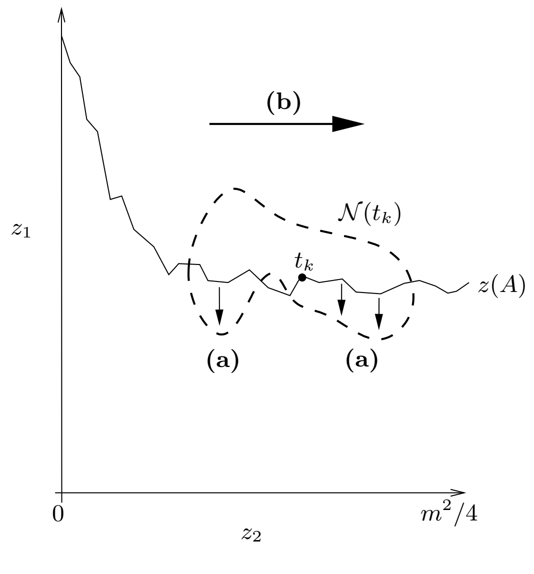

Fig. 2. Archive A is updated (a) whenever a non-forbidden but better solution is found in $\mathcal{N}(t_k)$. Each left-right sweep (b) repeats this for $k = 0, 1, \ldots, m^2/4$.

We apply two such functions that are efficiently computable but yet effective. The first is simply the exchange neighborhood

$$\mathcal{N}_1(t) = \{exchng(t, i, j) \mid 1 \leq i < j \leq m\}.$$

For a second one we move every a_i to some position j, i.e., if $t = (\ldots, a_i, \ldots, a_j, \ldots)$ let $move(t, i, j) = (\ldots, a_{i-1}, a_{i+1}, \ldots, a_j, a_i, \ldots)$ and

$$\mathcal{N}_2(t) = \{move(t, i, j) \mid 1 \leq i, j \leq m\}.$$

The `intensify`-procedure alternates between $\mathcal{N}_1$ and $\mathcal{N}_2$ after every completion of a left-right sweep. This is repeated until A is locally optimal with respect to both neighborhood functions.

Algorithm 3: `intensify`(A,T)

```
begin
    $s := 1$;
    repeat
        for $k := 0$ to $m^2/4$ do
            foreach $t \in \mathcal{N}_s(t_k)$ do
                $k' := z_2(t)$;
                if (3) holds then
                    replace $t_{k'}$ by $t$ in $A$;
                end
            end
        end
        $s := (s \bmod 2) + 1$;
    until $A$ is locally optimal ;
    return $A$
end
```

As a result of this procedure we obtain an archive A such that no $t_k \in A$ can be further improved within

$$\mathcal{N}(A) = \bigcup_{t \in A} \bigcup_{s \in \{1,2\}} \mathcal{N}_s(t).$$

So every t_k is not only locally optimal for the value $z_2(t_k)$ within $\mathcal{N}_s(t_k)$, but it is also a best tour for this z_2-value in all other neighborhoods $\mathcal{N}_s(t)$ with $t \in A$ and $s \in \{1, 2\}$.

There are two more aspects worth noticing. First, even tours already known to be dominated may contribute with their neighborhood to an improvement of the archive (see again Fig. 2). This is because we do not restrict A to locally Pareto-optimal solutions after each iteration.

Secondly, due to the overlapping of neighborhoods there is a variable search-depth during intensification. If some $t_{k+\delta}$ for $\delta > 0$ is improved to $t'_{k+\delta}$ when looking at $\mathcal{N}_s(t_k)$, the search continues with $\mathcal{N}_s(t'_{k+\delta})$ later during the same sweep. If $t_{k-\delta}$ has changed, then $\mathcal{N}_s(t'_{k-\delta})$ is considered during the next sweep.

It turns out that δ can be as large as $m-1$ for both neighborhood functions. To see this let $t = (1, \ldots, m)$ and observe that

$$z_2(t) + (m-1) = z_2(exchng(t, 1, m)) = z_2(move(t, 1, m)).$$

3.2 Perturbation Operators

When A becomes a locally-optimal fixpoint during intensification, the `BFLS` algorithm makes a random choice between two perturbation operators in order to diversify the archive while maintaining other (parts of) solutions at the same time.

The first operator simply performs a swap operation by exchanging the first half with the second half of a tour. So if $t = (a_1, \ldots, a_i, a_{i+1}, \ldots, a_m)$ with $i = m/2$ then

$$\mathcal{O}_1(t) = (a_{i+1}, \ldots, a_m, a_1, \ldots, a_i).$$

This does not necessarily generate a new solution for *every* z_2-value in the archive, but it can be experimentally observed that a reasonable large fraction of A is rebuilt. Note that this swap operator has the property that penalties usually change but

$$z_1(t) = z_1(\mathcal{O}_1(t)).$$

For our second operator $\mathcal{O}_2$ we would like to obtain just the dual behaviour: It should change the z_1-value of a tour $t \in A$ but $z_2(t) = z_2(\mathcal{O}_2(t))$. The design of such an operator is more subtle. To our knowledge, it is not clear how many permutations t' exist with $z_2(t') = z_2(t)$ for some given t, and how they can be computed without exhaustive search. The numbers $S(m, k)$ of permutations t of $\{1, \ldots, m\}$ such that $z_2(t) = k$ are know as integer sequence A062869 from [12].

The idea for $\mathcal{O}_2$ is to define an equivalence relation such that some random $t' = (a'_1, \ldots, a'_m)$ with $z_2(t') = z_2(t)$ can be chosen efficiently from t's equivalence class when $t = (a_1, \ldots, a_m)$ is given. Let

$$P(t) = \{1 \leq i \leq m \mid p_i = 0\}$$

be the set of positions in t that do not contribute to $z_2(t)$. We say that t' is a *variant* of t, in symbols $t \approx t'$, if and only if for $1 \leq i \leq m$ it holds that

i) $i \in P(t) \Rightarrow p'_i = 0$ and
ii) $i \notin P(t) \Rightarrow a'_i = a_i$.

So t' is obtained from t by permuting elements from $V(t) = \{a_j \mid j \in P(t)\}$ without introducing new penalties. It is easy to see that $\approx$ is an equivalence relation and that $t \approx t'$ implies $z_2(t) = z_2(t')$. Observe furthermore that by ii) the part of a tour t that is responsible for t's penalty is carried over to every variant of t.

In general, there are permutations t having $\approx$-equivalence classes of exponential size, e.g, if $t = (m, m-1, \ldots, 1)$ then every permutation of the cities $m, m-1, \ldots, m/2$ yields a variant of t. However, the following producer-consumer type of algorithm efficiently computes $\mathcal{O}_2(t)$ by making a uniform random choice from $[t]_\approx$. A 0-1-vector $(d_1, \dots, d_m)$ stores the candidates from $V(t)$ (producer) that can be put at position $i \in P(t)$ (consumer) as i decreases from m to 1.

Algorithm 4: $\mathcal{O}_2(t)$

```
begin
    $t' := t$;
    $(d_1, \dots, d_m) := (0, \dots, 0)$;
    for $i := m$ downto 1 do
        if $i \in V(t)$ then $d_i := 1$ end;
        if $i \in P(t)$ then
            choose $r \in \{i \leq j \leq m \mid d_j = 1\}$ randomly;
            $a'_i := r$;
            $d_r := 0$
        end
    end
    return $t'$
end
```

It must be noticed that there are permutations with $[t]_\approx = \{t\}$, e.g, if $t = (2, 1, 4, 3, \dots, m, m-1)$. In such an undesirable case $\mathcal{O}_2$ has no effect. We finally prove in this section how many permutations have single-elemented equivalence classes. To do so, we first show the following characterization.

Lemma 1. *Let $t = (a_1, \dots, a_m)$ be a permutation of $\{1, \dots, m\}$. Then it holds that $[t]_\approx = \{t\}$ if and only if $p_j > 0$ for all $i \in P(t)$ and j with $i < j \leq a_i$.*

Proof. We prove both implications by contraposition. So first assume that there is some $i \in P(t)$ and some j with $i < j \leq a_i$ such that $p_j = 0$. If we put a_i in t at position j there is penalty 0 at this position because $a_i \geq j$. On the other hand, we get from $p_j = 0$ that $a_j \geq j > i$. So we can also place a_j at position i in t while having penalty 0 there as well. Since $i \neq j$ the transposition of a_i and a_j yields a strict variant of t.

Conversely, let $t' = (a'_1, \dots, a'_m) \in [t]_\approx$ with $t' \neq t$. So there must be some $a_i \in V(t)$ and $i, j \in P(t)$ with $a_i = a'_j$ but $i \neq j$. We may assume w.l.o.g. that $j > i$ since it cannot be the case that $j \leq i$ for all $a_i \in V(t)$. Because $p'_j = 0$ it holds that $a_i = a'_j \geq j$ and with $P(t) = P(t')$ we see that also $p_j = 0$. Together, we identified some $i \in P(t)$ and j with $i < j \leq a_i$ such that $p_j = 0$. □

Next we want to count all permutations with the above property. Assume $P(t) = \{i_1, \dots, i_k\}$ for some $0 \leq k \leq m$ and $i_1 < \cdots < i_k$. For every $i_l \in P(t)$ it holds that $a_{i_l} \in \{i_l, \dots, m\}$ and due to the previous lemma we have $i_{l+1} \in \{a_{i_l}, \dots, m\}$. In order to count these possibilities we consider a tree $B(m)$ with its root labeled 0, and such that every node with label n has successors labeled $n+1, \dots, m$. Then a node with label n at depth d means $a_{i_d} = n$ and $i_{d+1} = n + 1$, and every leaf of $B(m)$ corresponds uniquely to a permutation t with $[t]_\approx = \{t\}$. An easy induction shows that $B(m)$ has 2^{m-1} leaves.

Theorem 1. *There are 2^{m-1} permutations of $\{1, \dots, m\}$ with $[t]_\approx = \{t\}$.*

Although exponential, this is a fast decreasing fraction of the size of the solution space $m!$ as can be seen as follows. Recall that w.l.o.g. m is even.

$$\begin{aligned} m! &= 2^{m-1} \cdot \left(\frac{m}{2}\right)! \cdot \prod_{i=1}^{m/2-1} (i + 1/2) \\ &> 2^{m-1} \cdot \left(\frac{m}{2}\right)! \cdot \left(\frac{m}{2} - 1\right)! \end{aligned}$$

So, e.g., for $m = 48$ this means $2^{m-1}/m! \leq 10^{-46}$.

4 Experimental Results

In order to test our algorithm we first compare its results with optimal solutions of some randomly-generated smaller instances. After that we investigate larger problem instances from TSPLIB [13].

It turns out that `BFLS` yields very good approximations of the complete Pareto front for randomly-generated instances with $m \leq 13$. In fact, in almost all cases the results differ from the optimal curve only slightly and only for very few z_2-values (optimal values being calculated via a multicritera branch-and-bound algorithm). We observe that deviations are not biased towards the minimal-cost tour. Furthermore, all computations are done reasonable fast. In many cases it can even be observed that the optimal curve is met exactly. One such example is shown in Fig. 3 for a randomly-generated instance with $m = 12$ cities.

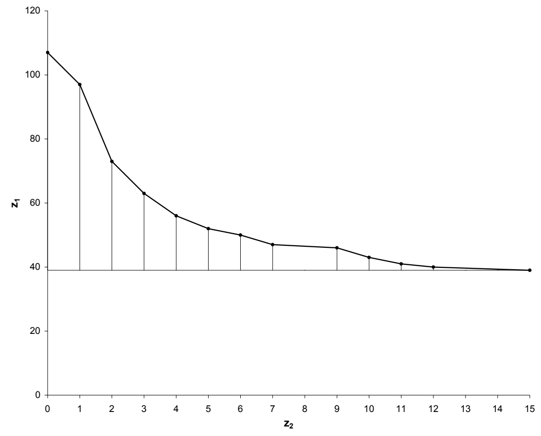

Fig. 3. Pareto front and `BFLS` results coincide in case of this randomly-generated instance with $m = 12$ cities. The BFLS algorithm uses $maxIterations = 20$ and $l = 3$.

For larger m we study instances from TSPLIB. Not surprisingly, we can neither compute the exact Pareto front in a reasonable amount of time on our own, nor are all of these values stored in the library. However, the minimum-cost value for the objective function z_1 is provided in the library and we take it as slight evidence to judge the approximation quality of the BFLS algorithm. We choose asymmetric instances to match the situation of Examples 1 and 2.

Figure 4 shows the initial archive A as generated by Algorithm 1, and the approximation of the Pareto front for the instance ry48p as finally provided by BFLS. Although the optimal z_1-value of 14422 is not exactly reached in this case, a solution t is found within 0.6 percent of this value. For this solution we have $z_2(t) = 400$ while $m^2/4 = 576$.

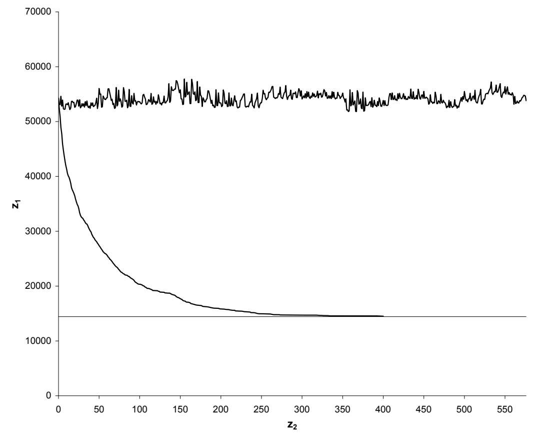

Fig. 4. Initial archive and final result of BFLS with lower bound for z_1 for the instance ry48p from TSPLIB.

As another example we display the results on instance ftv100. Figure 5 again shows the initial archive and the approximated Pareto front returned by the BFLS algorithm (resticted to the more interesting part $z_2 \leq 550$). For this instance with $m = 101$ cities a minimum-cost tour t with the optimal value $z_1(t) = 1788$ is found.

Finally, Table 1 compares the best found values for z_1 with the known optimal values for some more instances from TSPLIB. As can be seen the results of BFLS for this single point of the Pareto front differ only slightly from optimal values. Together with our initial observations on random instances we conclude that one can expect a rather close approximation of the complete Pareto front using the BFLS approach. Only very few iterations and short tabu lists were necessary to achieve these results. They were all computed using $maxIterations = 25$ and $l \in \{3, 4, 5\}$.

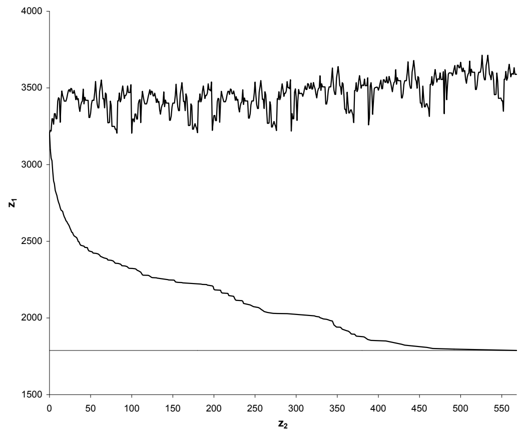

Fig. 5. Initial archive and final result of `BFLS` including an optimal minimum-cost tour for instance `ftv100` from TSPLIB.

Table 1. Comparison of optimal z_1-values with the ones found by `BFLS` for some instances from TSPLIB.

Instance	m	Optimum	`BFLS`	Difference
br17	17	39	39	0%
ftv33	34	1286	1347	5%
ftv35	36	1473	1475	0.1%
ftv38	39	1530	1532	0.1%
p43	43	5620	5620	0%
ftv44	45	1613	1636	1.2%
ry48p	48	14422	14507	0.6%
ftv47	48	1776	1855	4.4%
ft53	53	6905	6909	0.1%
ftv70	71	1950	1961	0.6%
ftv90	91	1579	1579	0%
kro124p	100	36230	37813	4.5%
ftv100	101	1788	1788	0%
ftv130	131	2307	2354	2.0%
ftv150	151	2611	2729	4.5%

The most expensive part of these computations is the call of the `intensify`-procedure. In order to improve its performance we label each solution t_k in the archive to indicate whether there was an improvement to some t'_k during the last sweeps or not. In this way we avoid recalculation of neighborhoods that have already been explored.

5 Conclusions

We propose the local-search based algorithm `BFLS` and argue that it is a reasonable approach to solve the bicriteria version TSPwP of classical TSP. The algorithm is conceptually easy and can be summarized as follows. A dense (with respect to z_2) collection of candidate solutions is iteratively improved via left-right sweeps involving two neighborhood structures and a variable search-depth. A well-balanced amount of diversification is achieved using a collection of tabu lists and problem-specific perturbation operators. Further investigations could involve the performace of this approach on other bicriteria optimization problems. A necessary prerequisite for the type of archive used here is that the range of at least one objective function is polynomially bounded in the input size.

References

1. Gutin, G., Punnen, A.P., eds.: The traveling salesman problem and its variations. Volume 12 of Combinatorial Optimization. Springer (2007)
2. Paquete, L., Chiarandini, M., Stützle, T.: Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In: Metaheuristics for Multiobjective Optimisation. Volume 535 of Lecture Notes in Economics and Mathematical Systems., Springer Verlag (2004)
3. Geiger, M.J.: Foundations of the pareto iterated local seach metaheuristic. In: Proceedings of the 18th International Conference on Multiple Criteria Decision Making, Chania, Greece, June 19-23 (2006)
4. Lourenço, H.R., Martin, O., Stützle, T.: Iterated local search. In: Handbook of Metaheuristics. Volume 57 of International Series in Operations Research & Management Science., Kluwer Academic Publishers (2003)
5. Hansen, P., Mladenović, N.: Variable neighborhood search. In: Handbook of Metaheuristics. Volume 57 of International Series in Operations Research & Management Science., Kluwer Academic Publishers (2003)
6. Glover, F., Laguna, M.: Tabu search. Kluwer Academic Publishers (1997)
7. Misevicius, A.: Using iterated tabu search for the travelling salesman problem. Information technology and control **32**(3) (2004) 29–40
8. Armentano, V.A., Arroyo, J.E.C.: An application of a multi-objective tabu search algorithm to a bicriteria flowshop problem. Journal of Heuristics **10**(5) (September 2004) 463–481
9. Hansen, M.P.: Use of substitute scalarizing functions to guide a local search based heuristic: The case of moTSP. Journal of Heuristics **6**(3) (August 2000) 419–431
10. Greistorfer, P., Voß, S.: Controlled pool maintenance in combinatorial optimization. In: Metaheuristic Optimization via Memory and Evolution Tabu Search and Scatter Search. Volume 30 of Operations Research/Computer Science Interfaces., Springer Verlag (2005) 382–424
11. Voß, S.: Hybridizing metaheuristics: The road to success in problem solving!?! In: Proceedings of the 8th EU/MEeting on Metaheuristics in the Service Industry, Stuttgart, Germany (2007)
12. The on-line encyclopedia of integer sequences: Sequence A062869. http://www.research.att.com/∼njas/sequences/A062869
13. Reinelt, G.: TSPLIB95. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/