The Node Weight Dependent Traveling Salesperson Problem: Approximation Algorithms and Randomized Search Heuristics

Jakob Bossek
School of Computer Science
The University of Adelaide, Australia

Katrin Casel
Chair for Algorithm Engineering
Hasso Plattner Institute, Germany

Pascal Kerschke
Department of Information Systems
University of Münster, Germany

Frank Neumann
School of Computer Science
The University of Adelaide, Adelaide, Australia

ABSTRACT

Several important optimization problems in the area of vehicle routing can be seen as variants of the classical Traveling Salesperson Problem (TSP). In the area of evolutionary computation, the Traveling Thief Problem (TTP) has gained increasing interest over the last 5 years. In this paper, we investigate the effect of weights on such problems, in the sense that the cost of traveling increases with respect to the weights of nodes already visited during a tour. This provides abstractions of important TSP variants such as the Traveling Thief Problem and time dependent TSP variants, and allows to study precisely the increase in difficulty caused by weight dependence. We provide a 3.59-approximation for this weight dependent version of TSP with metric distances and bounded positive weights. Furthermore, we conduct experimental investigations for simple randomized local search with classical mutation operators and two variants of the state-of-the-art evolutionary algorithm EAX adapted to the weighted TSP. Our results show the impact of the node weights on the position of the nodes in the resulting tour.

CCS CONCEPTS

• **Theory of computation → Theory of randomized search heuristics**;

KEYWORDS

Evolutionary algorithms, dynamic optimization, running time analysis, theory.

ACM Reference format:
Jakob Bossek, Katrin Casel, Pascal Kerschke, and Frank Neumann. 2020. The Node Weight Dependent Traveling Salesperson Problem: Approximation Algorithms and Randomized Search Heuristics. In *Proceedings of Genetic and Evolutionary Computation Conference, Cancun, Mexico, July 8–12, 2020 (GECCO '20),* 9 pages.
https://doi.org/10.1145/3377930.3390243

GECCO '20, July 8–12, 2020, Cancun, Mexico

ACM ISBN 978-1-4503-7128-5/20/07…$15.00
https://doi.org/10.1145/3377930.3390243

1 INTRODUCTION

Evolutionary algorithms have been used for many complex optimization problems, but it is very hard to understand the complexity of the considered problems as well as the performance of evolutionary algorithms dependent on important problem characteristics. Complex optimization problems often involve several interacting components that determine the value of a solution. Often these problems are called multi-component problems [4] and the goal is to compute an overall high quality solution which might be quite different from good solutions of the underlying silo problems. The Traveling Thief Problem (TTP) has been introduced in [3] as an example problem which combines two of the most well studied $\mathcal{NP}$-hard problems in combinatorial optimization, namely the Traveling Salesperson Problem (TSP) and the Knapsack Problem (KP). The TTP searches for a TSP tour and a packing plan such that the overall benefit of the tour is maximized. Here, the overall benefit is given by the profit of the collected items minus the cost of the tour which takes into account that costs are increasing with the weight of the collected items. The problem has obtained significant attention in the evolutionary computation literature in recent years. Different types of evolutionary and other heuristic approaches have been designed for the TTP [10, 11, 26–28] and various types of studies have been carried out to understand the interaction of the two underlying subproblems [16, 24, 25]. Furthermore, the TTP has been subject to various competitions over the last 5 years using a benchmark set that combines popular classes of benchmarks for the TSP and KP [21].

The goal of this paper is to study such interactions of having increasing weights for the TSP from a theoretical perspective. In the case of TTP, previous studies have focused on the theoretical investigations of the underlying packing problem when the tour is kept fixed. An exact approach based on dynamic programming and a fully polynomial time approximation scheme have been presented in [18]. These studies show that the underlying packing problem (although $\mathcal{NP}$-hard) is relatively easy to solve by these approaches. Dealing with the traveling salesperson part of TTP seems to be the harder problem. The traveling salesperson part of TTP involves traveling costs that are increasing with the weight of the items collected. Motivated by the TTP, we study a variant of the TSP, which we call node weight dependent TSP (W-TSP), where there are additional weights on the nodes and the cost of a tour increases with the weight of already visited nodes. Its increased difficulty in comparison to classical TSP shows that TTP is not just more difficult than TSP because of the added knapsack problem. W-TSP

highlights the challenges of solving TTP that do not originate from packing decisions. Aside from these investigative purposes, W-TSP is a natural model for many practical applications. For example, it can be used to model the effect of a vehicle's loaded weight on its gas consumption, e.g., in case of flights, liner ship movements or waste disposal.

The W-TSP is related to other variants of the TSP. Both definitions of the time dependent TSP (TDTSP) also consider change of costs with respect to the already traveled tour. In [20], the authors consider traveling costs that depend on the position in the tour (TDTSP1), a different version (TDTSP2) studied in [15] defines cost with respect to the distance traveled. The very general form of time dependence in both versions results in optimization problems that are very hard to analyze, hence there are few known positive results. Most studies on the TDTSP1 focus on exact algorithms based on integer linear programming, e.g. [1]. For the restriction to distances in $\{1, 2\}$, a $(2 - 2/3n)$-approximation for the TDTSP1 has been presented in [6]. Furthermore, for TDTSP2 a genetic algorithm approach has been studied in [23].

Our more restrictive set of weights for W-TSP turns out to be much closer related to the so-called minimum latency problem (MLP). This problem was introduced in [2] to model a cost function for tours from a customer perspective. Formally, the objective is to minimize the average distance from a given start city to all other cities in a TSP tour. At first glance, latency may seem very different from weight dependence, but we will see that these two additional dependencies on the cost of a TSP tour have useful similarities. We pay particular attention to the connection of the classical TSP and the MLP to the W-TSP. Our goal is to examine what types of successful algorithmic approaches for the TSP can be translated to the W-TSP.

1.1 Our contribution

We study the W-TSP from a theoretical perspective and are particularly interested in how the problem changes dependent on the weights that are part of the input. We start by introducing approximation algorithms that make use of known methods for the MLP. Particularly, we investigate the restriction to metric distances and show that if there is an α-approximation for MLP then there is a 3α-approximation for W-TSP restricted to all weights equal to 1. By adapting the techniques used to approximate the MLP and further structural properties, we derive a 3.59-approximation for metric W-TSP with bounded integer weights. Afterwards, we investigate the quality of approximations for the {1,2}-TSP and show how this translates into a 1.75-approximation for {1,2}-W-TSP when all weights are 1.

Our theoretical investigations are complemented by experimental investigations that systematically investigate the performance of randomized local search using different mutation operators. Furthermore, we investigate the high performing evolutionary algorithm EAX for the classical TSP and its adaptation to the W-TSP. We study the effect of increasing weights for the W-TSP and point out differences that occur in the tours for the TSP and the W-TSP when using EAX on these two problems. For randomized local search, we observe that the inversion operator is preferred over jump operations although even symmetric TSP instances lead to non symmetric instances for W-TSP. For EAX, the results for $n = 50$ cities show that the performance when using the best TSP tour computed by EAX and the best tour for W-TSP might differ by a factor of up to 2.75 in terms of quality for W-TSP. For the considered instances having 1000 nodes, we regularly observe a difference by a factor of 1.75.

The paper is structured as follows. In Section 2, we formally introduce the weighted Traveling Salesperson Problem. In Section 3, we provide theoretical approximation guarantees for the metric case and provide improved results for the case where TSP costs are 1 or 2 in Section 4. In Section 5, we study experimentally different mutation operators for randomized local search as well as the difference of the quality of solutions for the classical TSP and weighted TSP obtained by EAX. Finally, we conclude the paper and point out several promising future research directions.

2 PROBLEM FORMULATION

We consider a version of the symmetric TSP where cities have additional weights. The cost traveled along an edge depends on the weight of the cities visited so far and the distance of the edge. Let $\pi = (\pi_1, \dots \pi_n)$ be a permutation of the n cities. We assume that we always start at city 1 for the evaluation of a permutation, i.e., we have $\pi_1 = 1$. If this is not the case, we simply rotate the permutation prior to the fitness evaluation such that city 1 is the first city in the permutation.

For distance function d and weight function w on a set of n cities, we aim to find a permutation π that minimizes the weighted TSP cost, denoted by $\mathcal{W}(\pi)$, formally given by the expression

$$\mathcal{W}(\pi) = d(\pi_n, \pi_1)\left(\sum_{j=1}^{n} w(\pi_j)\right) + \sum_{i=1}^{n-1} d(\pi_i, \pi_{i+1})\left(\sum_{j=1}^{i} w(\pi_j)\right).$$

We call this optimization problem the node weight dependent TSP (W-TSP). Note that the standard (unweighted) TSP is the special case where $w(\pi_1) = 1$ and $w(\pi_i) = 0$, $2 \leq i \leq n$. Generally, the distance $d(\pi_i, \pi_{i+1})$ is multiplied by the weight at city π_i, which we abbreviate with $\omega(i) = \sum_{j=1}^{i} w(\pi_j)$.

To analyze the properties of W-TSP, we consider the following variants. With uniform weighted TSP (UWDTSP) we refer to the restriction that each city (except for the fixed start city) has the same weight, formally, $w_i = a$, $2 \leq i \leq n$, for some fixed value $a \geq 0$. Also, 1-weighted TSP (1W-TSP) denotes the further restriction to $a = 1$.

This formal definition of the W-TSP and its variations relates to known variations of the TSP as follows. Time dependence as defined in [20], considers a collection of distance values $d_{i,j,\ell}$, $1 \leq i, j \leq n$, $1 \leq \ell < n$ with the interpretation that the cost of traveling from city i to city j in a tour where i is the ℓth city to be visited is $d_{i,j,\ell}$. UWDTSP with unit weight $a > 0$ can be modeled by such a time dependent formulation by setting $d_{i,j,\ell} = d(i, j)(n - \ell + 1)a$. For the general W-TSP however, the cost of a transition in the weight dependent TSP does not only depend on its position in the tour, but also on the cities that were previously visited (their respective weights to be precise), hence a form of dependency that can not purely be modeled in relation to the position in the tour.

Another definition of time dependence given in [15] defines for the transition from i to j, a cost that varies with respect to the

time that has passed (i.e. the traveling cost) until the tour reaches city i. This version of time dependence is similar to our weight dependence in the sense that it also models distance variation with respect to the partial tour traversed before reaching a city. With time dependence however, distance and time dependence are inherently entangled while weight dependence retains a stronger separation between distance and weight-effects.

1W-TSP has an interesting relationship with the minimum latency problem (MLP) introduced in [2]. We give a formal definition for the MLP and discuss the relevant known results in more detail in the next section.

3 APPROXIMATION ALGORITHMS

In this section we consider W-TSP restricted to distances that satisfy the triangle inequality. We call this variant metric W-TSP. Without this restriction, the W-TSP, like most variants of the TSP, can not be approximated within any constant factor; this immediately follows from the standard reduction from the $\mathcal{NP}$-hard Hamiltonian cycle problem.

Aside from these complexity theoretic reasons, restriction to metric distances is a standard assumption for the TSP. Sometimes, triangle inequality is also indirectly implied by the objective of finding a shortest tour that visits each city at least once, first introduced by [9] as *graphical* TSP.

3.1 Connections to the Minimum Latency Problem

We explore the connection between the 1W-TSP and the MLP. Formally, the Minimum Latency Problem (also called *delivery-man*, *school-bus driver*, or *traveling repairman problem*) models the task to find, for a given set of cities with distance function d and a fixed start city p, a path starting at p which visits all cities and minimizes the sum of waiting times. Formally, with a solution again modeled as a permutation $\pi = (\pi_1, \ldots, \pi_n)$ with $p = \pi_1$, MLP minimizes

$$\mathcal{L}(\pi) = \sum_{i=2}^{n} \ell(\pi_i) = \sum_{i=2}^{n} \sum_{j=1}^{i-1} d(\pi_j, \pi_{j+1})$$

The shorthand $\ell(i)$ describes the *latency* of city π_i as it models the distance passed until city π_i is reached. Although the MLP asks for a path and not a round-trip, it is possible to relate it to 1W-TSP in case of metric distances.

For 1W-TSP, the cost of a permutation $\pi = (\pi_1, \ldots, \pi_n)$ can be rewritten to $\mathcal{W}(\pi) = nd(\pi_n, \pi_1) + \sum_{i=1}^{n-1} id(\pi_i, \pi_{i+1})$. Rewriting the summation to compute the latency of a permutation yields the following connection to 1W-TSP:

$$\begin{aligned}\mathcal{L}(\pi) &= \sum_{i=1}^{n-1} (n-i) d(\pi_i, \pi_{i+1}) = \sum_{j=1}^{n-1} j d(\pi_{n-j}, \pi_{n-j+1}) \\ &= \mathcal{W}(\pi_n, \pi_{n-1}, \ldots, \pi_1) - nd(\pi_1, \pi_n)\end{aligned}$$

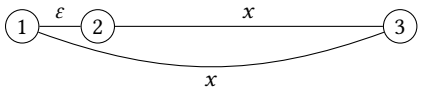
Reversing the order of a permutation reveals a strong connection between latency and W-TSP. As already observed by [2], combined with a shift to start both tours at the fixed start city π_1, it follows that the cost of a node weighted TSP tour can be interpreted as the sum of a reversed latency tour and a classic TSP tour, formally:

$$\mathcal{L}(\pi) = \mathcal{W}(\pi_1, \pi_n, \ldots, \pi_2) - \sum_{i=1}^{n-1} d(\pi_i, \pi_{i+1}) - d(\pi_1, \pi_n)$$

Since all distances are non-negative, this relation shows that the optimum value for MLP gives a lower bound for the optimum for 1W-TSP. Triangle inequality implies $d(\pi_1, \pi_n) \leq \sum_{i=1}^{n-1} d(\pi_i, \pi_{i+1})$, which together with the rough bound $\mathcal{L}(\pi) \geq \sum_{i=1}^{n-1} d(\pi_i, \pi_{i+1})$ yields:

$$\mathcal{W}(\pi_1, \pi_n, \ldots, \pi_2) \leq \mathcal{L}(\pi) + 2 \sum_{i=1}^{n-1} d(\pi_i, \pi_{i+1}) \leq 3\mathcal{L}(\pi)$$

In the worst case, this relation is tight as seen by the example below, where $\mathcal{L}(1,2,3) = x + 2\varepsilon$ and $\mathcal{W}(1,3,2) = x + 2x + 2\varepsilon$ which for large x and small ε yields the worst-case factor of 3 between 1W-TSP and MLP (observe that the respective permutations are optimum solutions).



In general this connection only yields that any α-approximation for metric MLP can be used to approximate metric 1W-TSPwith ratio 3α. A direct application of the techniques used for MLP allows to derive better approximation results for 1W-TSP.

Over the past 25 years, approximation algorithms for MLP have been gradually improved from the initial 144-approximation in [2] to the currently best 3.59-approximation in [7]. All such approximations have the same underlying idea of appending a certain set of tours starting and ending at the fixed start city. These tours are approximate solutions to k-MST, the problem of finding a minimum cost tree spanning k vertices which is an obvious lower bound on the latency of the k-th vertex in an optimal MLP tour. These constructions hence always calculate with the cost of a tour that returns to the start, so they can also be interpreted as a solution to 1W-TSP. The basic idea of our following approximation is to alter the procedure that picks the approximate solutions to the k-MST problem according to the objective of 1W-TSP. The formal description with technical details of this idea are given in the proof below.

Theorem 3.1. *Metric 1-weighted TSP can be approximated within a ratio of at most 3.59 in polynomial time.*

Proof. We adapt the strategy in [7] as follows. Consider a given metric instance of 1W-TSP with distance d on n cities. First assume that we have tours T_k that are a 2-approximation to the k-MST problem, for each $1 \leq k \leq n$; which we will also refer to as *good k-tours*. As a first difference to the approximation for latency, in this set of good k-tours, we construct as n-tour a 1.5-approximate solution to TSP, calculated from the algorithm of [8], to have a certain approximation ratio for the last tour.

As already mentioned, the final solution is constructed by appending a subset of the good k-tours. To find a good sequence to build this final solution, we also create an auxiliary graph H that contains a node for each good tour T_k and weights on directed arcs that reflect the cost produced by appending these tours in a solution,

and search for a shortest path from T_1 to T_n. To now reflect the cost of the 1W-TSP instead of the MLP, we change the cost of a path from the node corresponding to T_i to the node corresponding to T_j for any $1 \leq i < j \leq n$ in the weighted graph H to $(n - \frac{i+j}{2} + 1)c(T_j)$, where $c(T_j)$ denotes the cost of the tour T_j. This additional cost of $c(T_j)$ compared to the construction used for the latency problem gives exactly the cost of latency plus TSP tour, hence the cost for the 1W-TSP.

Consider, like in the original approach, appending the following set of subtours. For some $c > 1$ let T_{n_i} be the good tour of length at most $2bc^i$ that contains the largest number of vertices, with b set to be c^U, for a random variable U uniformly distributed between 0 and 1. Append these tours T_{n_i} for $i = 1, 2, \ldots$ in this order for all values of i for which $2bc^i$ is strictly smaller than $c(T_n)$, then append T_n as the last tour. Skipping multiple occurrences of cities in this tour then yields a valid solution to 1W-TSP. With $c = 3.59$, the bounds on the latency of each city in the resulting tour remains 3.59 with exactly the calculations as presented in [7]. For the additional TSP-cost, we claim that the constructed tour is at most 3.07 times as long as an optimal TSP tour. Let j and d be such that $d \leq 1$ and that the cost of an optimal TSP tour is dc^j. Regardless of the value of b, the last tour appended by the algorithm is the n-path created by Christofides' algorithm, so it has a cost of at most $1.5dc^j$. Similar to the computations for the latencies, the other appended tours depend on the relation between d and b.

If $d < b$, the algorithm appends (aside from the n-tour), $j - 1$ tours, up to cost $2bc^{j-1}$, with combined cost of at most

$$2\sum_{\ell=1}^{j-1} bc^\ell < \frac{2bc^j}{c-1}.$$

If $d > b$, the algorithm appends j tours, up to cost $2bc^j$, with combined cost

$$2\sum_{\ell=1}^{j} bc^\ell < \frac{2bc^{j+1}}{c-1}.$$

With expectation over U, the expected length of the tour up to T_n is at most:

$$\int_{\log_c d}^{1} \frac{2c^U c^j}{c-1}\,\mathrm{d}U + \int_{0}^{\log_c d} \frac{2c^U c^{j+1}}{c-1}\,\mathrm{d}U = \frac{2dc^j}{\ln c}$$

Overall, the expected ratio between the constructed tour and an optimal TSP tour is at most $1.5 + \frac{2}{\ln c} < 3.07$.

At last, the primal-dual procedure described in [7] only gives a set of good k-tours for a subset of $\{1, \ldots, n\}$, not for the whole set as we assumed in the beginning. Exactly as shown for the latency problem, the tours for the missing values of k can be replaced by phantom tours which then are replaced by existing ones since our distance function shows the same behaviour as the original one with respect to the interpolation used for the phantom tours. □

3.2 Bounded Integer Weights

So far, we only derived approximation results for the W-TSP for the restriction to all weights equal to 1 with the help of the MLP. While there exist generalizations of MLP, there are no known approximation results which translate to W-TSP. In [12], a variation of the MLP with a service time at each city which adds to the latency of the following city has been investigated. Considering the reverse tour, this is very different from W-TSP. Weights other than 1 in a reverse tour could however be interpreted as the importance of a city, given as multiplicative factor on the penalty of its waiting cost. To the best of our knowledge, this generalization of the MLP has not been studied.

In order to generalize the approximation for 1W-TSP to different weights, we exploit structural properties of optimal solutions. This approach yields the following result.

Lemma 3.2. *For any $\alpha > 1$, an α-approximation for metric 1-weighted TSP can be used to derive an α-approximation for metric weighted TSP with polynomially bounded, non-zero, integer weights.*

Proof. Consider an instance of W-TSP given by distances d and polynomially bounded non-negative integer weights w on n cities. Create an instance of 1W-TSP by including $w(i)$ copies of city i, for each city $i \in \{1, \ldots, n\}$. Denote for the new instance formally the set of cities by $\{\{i_1, \ldots, i_{w(i)}\} \colon 1 \leq i \leq n\}$. We further define the distances $\hat{d}$ for the transformed instances by

$$\hat{d}(i_r, j_s) = \begin{cases} 0, & \text{if } i = j \\ d(i,j), & \text{else} \end{cases}$$

Observe that this definition yields a metric distance. Further, since the weights are polynomially bounded, this construction is polynomial. Denote by r the number of cities in this new instance, and assign the weight 1 to each of these cities.

Observe that any permutation $(\pi_1, \ldots, \pi_n)$ for the original instance can be translated to a permutation of the same weighted cost for the new instance by replacing π_j with $\pi_j = i$ by the sequence $i_1, \ldots, i_{w(i)}$. In particular, the optimum value for the new instance is smaller or equal to the optimum value of the original one.

Conversely, we can use a permutation of the new instance to create a permutation of the same or even smaller cost for the original instance as follows. Let $(\pi_1, \ldots, \pi_r)$ a permutation for the new instance. We claim that this permutation can be altered such that all copies of an original city occur consecutively together which allows to extract a permutation to the original instance by replacing the grouped copies by the single original city. Assume that for some $1 < i \leq n$, the cities $i_1, \ldots, i_{w(i)}$ do not occur (in some arbitrary order) as one consecutive block in the sequence $(\pi_1, \ldots, \pi_r)$. Let $1 \leq x \leq w(i)$ be such that i_x occurs last, among all cities in $\{i_1, \ldots, i_{w(i)}\}$, in the sequence $(\pi_1, \ldots, \pi_r)$. Consider altering the sequence, by moving all cities in $\{i_1, \ldots, i_{w(i)}\} \setminus \{i_x\}$ to be visited right after i_x. All edges after i_x have the exact same cost, since neither the weight nor the cities have changed. All edges among the cities in $\{i_1, \ldots, i_{w(i)}\}$ are zero, so they add no cost at all. All edges before the block $\{i_1, \ldots, i_{w(i)}\}$ now are attached with equal or less weight than before, since the weight of the shifted cities is postponed. Further, triangle inequality implies that jumping over the gaps previously filled with cities in $\{i_1, \ldots, i_{w(i)}\}$ does not increase the tour cost. Repeating this procedure yields a permutation that can be translated to the original instance and has the same or smaller cost.

Overall, it follows that an α-approximate solution for metric W-TSP with bounded non-negative integer weights can be computed by creating a new instance of metric 1W-TSP, running the assumed

α-approximation on it, and then translating the resulting permutation to the original instance (this can be done in linear time by scanning the permutation in reverse and skipping duplicates). □

Combined with Theorem 3.1, this result gives the following.

Theorem 3.3. *Metric weighted TSP with polynomially bounded, non-zero, integer weights can be approximated within a ratio of at most 3.59 in polynomial time.*

4 1-WEIGHTED TSP{1,2}

We now consider the further restriction to distance values 1 and 2. For the classical TSP, this is one of the most studied restrictions, usually called {1,2}-TSP, as this problem can be seen as a generalization of the Hamiltonian cycle problem and is therefore still $\mathcal{NP}$-hard. Different approximation algorithms have been developed for the {1,2}-TSP and we investigate how to make use of those when investigating 1W-TSP with distances 1 and 2. We refer to this restriction by 1-weighted TSP{1,2}, 1W-TSP{1,2} for short.

A $(2-2/3n)$-approximation algorithm for a restriction to distances 1 and 2 on the related time dependent TSP has been presented in [6]. Although 1W-TSP is a special case of TDTSP1, this result can not be used to derive an equivalent approximation for 1W-TSP{1,2}, since edge-cost restriction for our problem does not translate to edge-cost restriction to 1 and 2 in the representation as TDTSP1; observe that the costs of 1 and 2 have to be multiplied by the weights, which, even with the restriction to all weights being 1, gives a range of time dependent distances between 1 and $2n$.

We first consider the case where the input allows for a TSP tour of cost n. First observe that for this case, an optimal tour for 1W-TSP has cost $\sum_{i=1}^{n} = n(n+1)/2$. Let k be the number of 2-edges introduced into the tour by an approximate solution. The tour has the highest possible costs if these edges are at the end of the tour. Compared to the optimal tour the cost increase by

$$n + (n-1) + \ldots + (n-(k-1)) = kn - k(k-1)/2 = k(n-(k-1)/2)$$

Let π be an $\alpha = (1+c)$-approximation, $c \geq 0$ for the {1,2}-TSP. Let $k \leq cn$ be the number of 2-edges in π. The resulting approximation ratio for 1W-TSP{1,2} is at most $1 + k(n-(k-1)/2)/(n(n+1)/2) \leq 1 + (2kn - (k^2-k)/2)/(n^2)$. Setting $k = cn$, we get $1 + (2cn^2 - (c^2n^2 - cn)/2)/(n^2) = 1 + (2c - c^2/2) + o(1)$.

Assume that we use the 7/6-approximation for {1,2}-TSP given in [19], then we have $c = 1/6$ and therefore a $1 + (2/6 - 1/72) + o(1) = 95/72 + o(1)$ approximation for the weight dependent TSP.

We can improve our results by considering the π also in reverse order. Formally, for $\pi = (\pi_1, \ldots, \pi_n)$ we also consider the tour $\pi' = (\pi_1, \pi_n, \ldots, \pi_2)$. One of these two tours has at least $k/2$ edges of cost 2 at positions $1, \ldots, n/2$ which gives, for the better of these tours, an addition to the optimum of at most

$$kn/2 + kn/4 - 2\sum_{i=1}^{k/2-1} i = kn/2 + kn/4 - k^2/4 + k/2$$

With $k = cn$ the approximation ratio is hence bounded by

$$\begin{aligned} & 1 + (cn^2/2 + cn^2/4 - (cn)^2/4 + cn/2)/(n^2/2) \\ = \;& 1 + 1.5c - c^2/2 + o(1) \end{aligned}$$

For $c = 1/6$, this gives a ratio of $89/72 + o(1)$.

We now extend these observations to the general case where the optimal solution can include edges of cost 2. Let π^* be an optimal solution for the classical {1,2}-TSP of cost $OPT = n + u$. π^* has exactly u edges of cost 2. An $\alpha = (1+c)$-approximation algorithm for the {1,2}-TSP produces a tour π of TSP-cost at most $(1+c)(n+u) = n + cn + (1+c)u$ which has at most $k = cn + (1+c)u$ edges of cost 2. Note $k \leq n$.

A lower bound on the value of an optimal solution for the 1W-TSP{1,2} is obtained by assuming that the u edges of the optimal TSP tour appear at the beginning of the weighted TSP tour. Hence, we can bound the value of an optimal solution of 1W-TSP{1,2} by

$$\sum_{i=1}^{n} i + \sum_{i=1}^{u} i = n(n+1)/2 + u(u+1)/2$$

We now estimate the weighted tour value of the approximate tour π and its reversal π' more precisely. Recall that both tours contain at most $k = cn + (1+c)u \leq n$ edges of cost 2. Each edge of cost 2 that adds an addition of cost $n \leq r \leq 1$ to π (addition compared to $n(n+1)/2$), adds a cost of $n - r + 1$ to π'. Summing up, if all costs of edges of length 2 in π cause an addition of R, then these edges produce an additional cost of $k(n+1) - R$ for π'. In the worst case, R is equal to $k(n+1)/2$, which results in a worst-case cost of $(n(n+1) + k(n+1))/2$ for the better of the two options.

Compared to the above bound on the optimal solution, this results in an approximation ratio of at most:

$$\begin{aligned} & \frac{n(n+1) + k(n+1)}{n(n+1) + u(u+1)} \\ = \;& \frac{n(n+1) + cn^2 + cn + (1+c)u + (1+c)un}{n(n+1) + u(u+1)} \\ \leq \;& 1 + c + (1+c)/2 \leq 1.5\alpha \end{aligned}$$

where the last step uses that $\frac{un}{(n(n+1)+u(u+1))}$ is monotonically increasing in u and attains its maximum for $u = n$. We summarize these results in the following theorem.

Theorem 4.1. *Using an α-approximation for {1,2}-TSP to compute a TSP tour π, π or its reverse tour π' is a 1.5α-approximation for the 1-weighted TSP{1,2}.*

Using the (7/6)-approximation for the {1,2}-TSP, we get a 1.75-approximation for the 1W-TSP{1,2}.

5 RANDOMIZED SEARCH HEURISTICS

In this section, we consider randomized search heuristics for W-TSP. We start by investigating variants of randomized local search and examine the use of popular mutation operators traditionally used for the classical TSP. Afterwards, we examine EAX as a state-of-the-art solver for the TSP and its adaptation to weighted TSP.

5.1 Problem instances

We consider a rich set of artificially generated metric TSP instances with different integer node weights. The instance generation approach is performed in two steps. First, $n \in \{25, 50, 100, 500, 1\,000\}$ nodes are placed in the Euclidean plane (bounded to $[0, 1\,000]^2$) utilizing different node placement generators. In this study we consider Random Uniform Euclidean (*rue*) placement, i.e., node

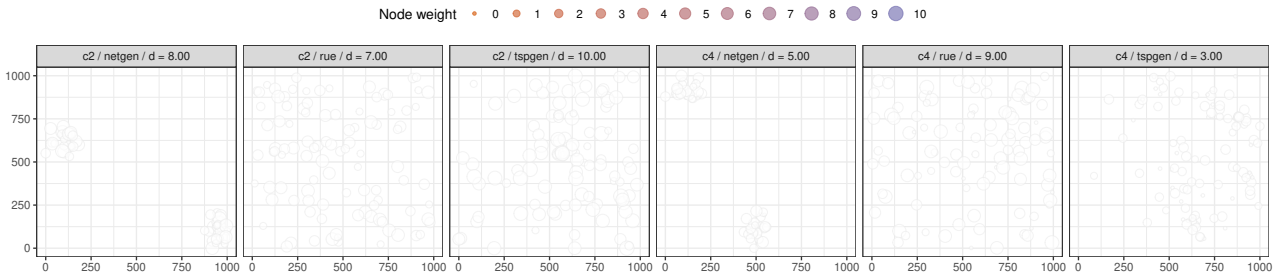


Figure 1: Examples of generated problem instances. Point size and color change with increasing node weight.

Algorithm 1: Randomized Local Search (RLS)

1 Choose a permutation π of the given n cities uniformly at random.
2 Produce π' from π by mutation.
3 If $\mathcal{W}(\pi') \leq \mathcal{W}(\pi)$, set $\pi := \pi'$.
4 If not termination condition, go to 2).

coordinates are sampled uniformly at random within the bounding box. Moreover, we consider so-called *netgen* placement with two distinct clusters of points sampled from a bivariate Gaussian distribution around the cluster centers. Further, we consider *tspgen* placement. Here, points are initially placed according to rue placement and subsequently altered in an iterative manner by a sequence of mutation operations [5]. The second step deals with the assignment of node weights. Here we consider three different configurations as described in the following. Note that $w_1 = 1$ across all cases for the fixed start node $p = 1$, whereas three different configurations are considered for w_i (with $2 \leq i \leq n$):

- C_1: $w_i = d$ with $d \in \{0.0, 0.1, \ldots, 1.0\}$,
- C_2: $w_i \in \{1, \ldots, d\}$ with $d \in \{2, \ldots, 10\}$ and
- C_3: $w_i \in \{0, \ldots, d\}$ with $d \in \{1, 2, \ldots, 10\}$.

To account for randomness in node placement and weight assignment we generate ten instances for each combination, which add up to 45 000 instances[1] (see Fig. 1 for examples).

5.2 Performance of the RLS variants

We first consider randomized local search (RLS) shown in Algorithm 1. It starts with a permutation π of the given n cities chosen uniformly at random. In each iteration a new permutation π' is produced from the current permutation π by a simple mutation. The new permutation π' replaces π if its weighted tour length is not larger than the one of π. We investigate popular mutation operators for permutation problems in this context, namely inversion, exchange, and jump operations [22]. Inversion operators usually achieve a high performance for the classical symmetric TSP as it only results in the update of the cost of two edges in the cost function. For weighted TSP the situation is different as weighted tours are not symmetric and the question arises whether inversion is still a good operator when considering weighted TSP. We run RLS with the three aforementioned mutation operators. Per instance, 30 independent runs are performed with a stopping condition of $1\,000 \cdot n$ function evaluations. The performance is measured as follows: let π be the final solution of algorithm A on instance I and let π^* be the best, i.e., shortest, tour found in all runs of all algorithms on I. We measure the performance as the relative deviation from π^*, i.e.,

$$\text{perf}(\pi) = \left(\frac{\mathcal{W}(\pi)}{\mathcal{W}(\pi^*)} - 1 \right) \cdot 100 \geq 0. \tag{1}$$

Note that this value is 0 if $\mathcal{W}(\pi) = \mathcal{W}(\pi^*)$. This measure allows to aggregate over instance sizes.

The results of our experiments comparing the three operators on the three instance classes C_1, C_2, and C_3 are shown in Table 1. Here we report the mean, standard deviation, the median and results of pairwise Wilcoxon-Mann-Whitney tests with Bonferroni p-value adjustment of the performance values defined in Eq. 1 split by class and d-value. It can be observed that the RLS variant using the inversion operation outperforms the other two variants for almost all settings. Comparing RLS using exchange operations with RLS using jump operations, we can see that jump operations are preferable over exchanges for the class C_1 whereas exchanges are preferable over jumps for the classes C_2 and C_3.

5.3 Performance of EAX

Next we investigate the adaptation of the evolutionary TSP solver EAX [17]. EAX is an evolutionary algorithm which uses a powerful *edge assembly crossover* operator to produce high-quality offspring individuals and a sophisticated population diversity mechanism. This algorithm has shown state-of-the-art performance in inexact TSP-solving in various studies [13, 14, 17]. We modified the algorithm to enable handling of node weights and consider two different fitness functions that guide the evolutionary search process: the classical TSP fitness function (ignoring node weights) and a fitness function based on the weighted TSP costs $\mathcal{W}(\pi)$. In the following we use the abbreviations EAX and W-EAX for brevity. Our main interest is the difference of tour lengths obtained by runs of EAX and W-EAX, respectively, depending on the structure of the weighted TSP instances under consideration.

We then performed ten independent runs on each instance with both fitness functions resulting in a total of 900 000 experiments which were strongly parallelized on a high performance computing cluster. EAX was run with a time-limit of five seconds for instances

[1] The total number of instances results from 30 different configurations (see details of C_1 to C_3), five instance sizes (n), ten replications due to node placement and ten replications for the weight-to-node assignments.

Table 1: Tabular values of mean (best values hightlighted in boldface), standard deviation (std), median and results of pairwise statistical tests (stat). Results are split by instance classes C_1, C_2, C_3 and the value of d. The notation in the stat columns reads as follows: X^+ indicates that the algorithm is significantly better with respect to the Wilcoxon Mann-Whitney test at a significance level of $\alpha = 0.05$ than algorithm X.

		RLS[Exchange] (1)				RLS[Inversion] (2)				RLS[Jump] (3)			
Class	d	mean	std	median	stat	mean	std	median	stat	mean	std	median	stat
C_1	0.0	104.06	76.33	67.01		**4.35**	3.46	3.45	$1^+, 3^+$	42.93	18.75	43.64	1^+
	0.1	65.07	33.61	56.57		**16.80**	11.96	14.87	$1^+, 3^+$	45.46	25.87	41.13	1^+
	0.2	64.11	33.34	55.08		**17.98**	12.70	16.07	$1^+, 3^+$	49.37	28.32	45.35	1^+
	0.3	64.02	33.42	55.59		**18.71**	13.09	16.82	$1^+, 3^+$	51.05	29.59	47.26	1^+
	0.4	64.14	33.59	56.07		**18.96**	13.29	16.93	$1^+, 3^+$	52.20	29.96	48.74	1^+
	0.5	63.63	33.31	54.75		**19.28**	13.64	17.30	$1^+, 3^+$	52.53	30.19	49.61	1^+
	0.6	63.35	33.37	54.87		**19.23**	13.58	17.27	$1^+, 3^+$	53.06	30.34	49.91	1^+
	0.7	63.51	33.42	54.84		**19.36**	13.65	17.24	$1^+, 3^+$	53.49	30.67	50.86	1^+
	0.8	63.57	33.76	54.33		**19.69**	13.78	17.75	$1^+, 3^+$	53.59	30.69	50.69	1^+
	0.9	63.56	33.51	54.66		**19.60**	13.81	17.47	$1^+, 3^+$	53.96	30.88	51.21	1^+
	1.0	63.80	33.49	55.59		**19.87**	13.91	17.89	$1^+, 3^+$	54.12	31.01	51.69	1^+
C_2	2.0	61.57	31.99	54.80		**21.90**	15.07	19.86	$1^+, 3^+$	62.10	36.99	57.16	
	3.0	59.51	29.71	53.82	3^+	**23.24**	15.91	21.60	$1^+, 3^+$	64.72	37.78	60.14	
	4.0	58.32	29.56	52.09	3^+	**23.88**	16.18	22.25	$1^+, 3^+$	66.73	38.96	62.06	
	5.0	57.19	28.54	52.56	3^+	**24.41**	16.63	22.86	$1^+, 3^+$	67.13	38.33	62.97	
	6.0	56.65	28.39	51.93	3^+	**24.73**	16.69	23.52	$1^+, 3^+$	67.95	39.00	63.89	
	7.0	55.71	28.24	51.30	3^+	**25.08**	16.92	23.86	$1^+, 3^+$	67.52	39.08	63.26	
	8.0	55.26	27.69	50.93	3^+	**25.09**	16.86	23.89	$1^+, 3^+$	68.20	38.69	64.59	
	9.0	55.07	27.59	50.13	3^+	**25.56**	17.07	24.41	$1^+, 3^+$	68.21	38.87	63.55	
	10.0	54.56	27.10	50.02	3^+	**25.18**	17.16	23.90	$1^+, 3^+$	68.40	38.82	64.13	
C_3	1.0	**39.04**	19.34	39.78	$2^+, 3^+$	41.42	26.42	43.21	3^+	62.55	32.41	62.50	
	2.0	44.24	21.58	43.80	3^+	**34.33**	21.70	35.35	$1^+, 3^+$	65.49	34.24	64.71	
	3.0	46.26	22.64	45.26	3^+	**32.80**	21.06	33.07	$1^+, 3^+$	67.80	34.84	66.98	
	4.0	47.58	23.46	46.25	3^+	**31.08**	19.97	31.53	$1^+, 3^+$	68.87	36.08	67.77	
	5.0	48.32	23.69	46.79	3^+	**30.02**	19.39	30.05	$1^+, 3^+$	69.28	36.56	67.93	
	6.0	48.55	24.43	46.12	3^+	**28.90**	19.06	28.40	$1^+, 3^+$	69.27	37.44	67.23	
	7.0	49.51	24.38	47.72	3^+	**28.70**	18.80	28.42	$1^+, 3^+$	69.94	37.59	68.83	
	8.0	49.93	24.34	48.29	3^+	**29.07**	19.02	28.85	$1^+, 3^+$	70.23	37.19	68.23	
	9.0	50.45	24.28	48.58	3^+	**28.96**	19.30	28.42	$1^+, 3^+$	70.60	37.48	68.79	
	10.0	50.09	24.68	48.16	3^+	**28.50**	18.73	28.17	$1^+, 3^+$	70.22	37.74	68.05	

with up to 100 nodes and three minutes for larger instances to keep the computational costs reasonable. These values may seem small at first glance, however, studies in [13] revealed that EAX is able to solve even large TSP instances – with thousands of nodes – to optimality within few seconds. Note that after completion of each run – regardless of the fitness function used internally as a driver – the final tour was evaluated by means of the weighted TSP fitness function. For evaluation we calculate the weighted tour length ratios, i.e., the weighted tour length obtained by EAX divided by the respective weighted tour length obtained by W-EAX for each instance and run. Note that values greater than 1 indicate an advantage of W-EAX over EAX. Fig. 2 shows the distributions of weighted tour length ratios separated by instance size, configuration and maximum weight d.

As expected we observed all ratios being greater than 1 with median values at about 1.15 across all combinations. Frequently, large outliers reached ratios up to 1.75 for instances with at least 100 nodes and even > 2.5 for smaller n. However, in general, no patterns can be identified with respect to configuration or maximal node weight. The sole exception is configuration C_1 and $n \in \{25, 50\}$ where we observe a slightly increasing trend in median ratios with increasing $d \in \{0.0, 0.1, \ldots, 1.0\}$ (see top-left boxplots in Fig. 2). This trend vanishes for $n \geq 100$ as a high number of nodes already imposes a large cumulative weight when considering only a part of any tour. In contrast, comparing node placement, we observe strong differences. While the ratios are lowest when the nodes are placed uniformly at random (rue), with values below 1.15 for large instances with at least 500 nodes, ratios become increasingly larger with increasing instance structure. For $n = 1\,000$ nodes, ratios on tspgen instances go up to about 1.5 while on netgen instances – with strongly segregated clusters – ratios reach values up to 1.8 with a median of about 1.2. Hence, more than 50% of the ratios are higher than the maximal ratio in case of rue placement.

Fig. 3 shows exemplary tours obtained by EAX, W-EAX and – for comparison – RLS with inversion mutation. The tours are shown for four instances of class C_2 (top row) and class C_3 (bottom row). We can make the following observation: since the variation operator of EAX was not modified, the resulting tours of both EAX and W-EAX are free of crossings. However, for the weighted TSP, optimal tours do not necessarily need to avoid crossings. As for instance shown in the fifth column of Fig. 3 (top row), RLS often finds solutions with many crossings, resulting in much shorter tours than produced by both EAX variants.

Additionally, for the weighted TSP – in particular in the presence of segregated cluster structures – it is often beneficial that long edges are included early in the permutation, as a later consideration would be associated with (the burden of) a huge amount of accumulated node weights. Once again, this is observable in the fifth column of Fig. 3: RLS places long edges early in the tour to quickly reach the top left cluster (top row) or the isolated node (bottom row), and then leave it just as quickly again. In case of netgen instances an even stronger effect of long inter-cluster edges

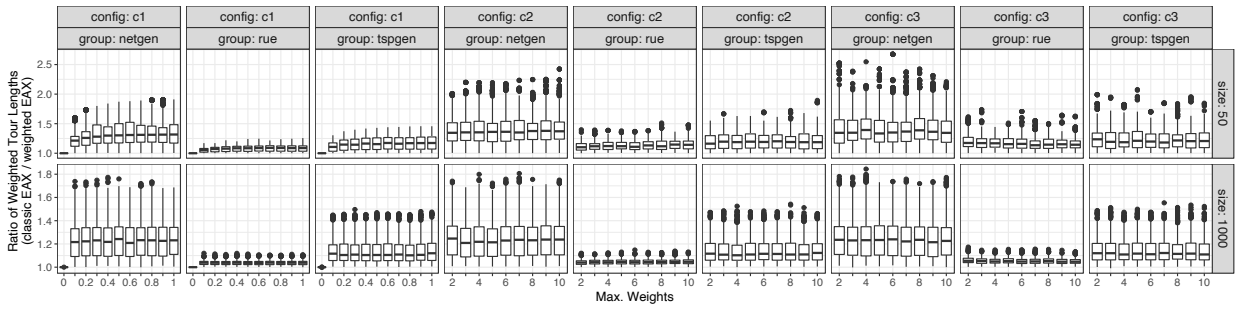


Figure 2: Ratios between the best tour lengths found by classical TSP and weighted TSP function. Each instance was optimized with EAX – which internally used the weighted or classical TSP fitness function, respectively – and all resulting tours have been assessed using the weighted fitness. We show results for $n \in \{50, 1000\}$ due to space limitations, but patterns for omitted data are similar.

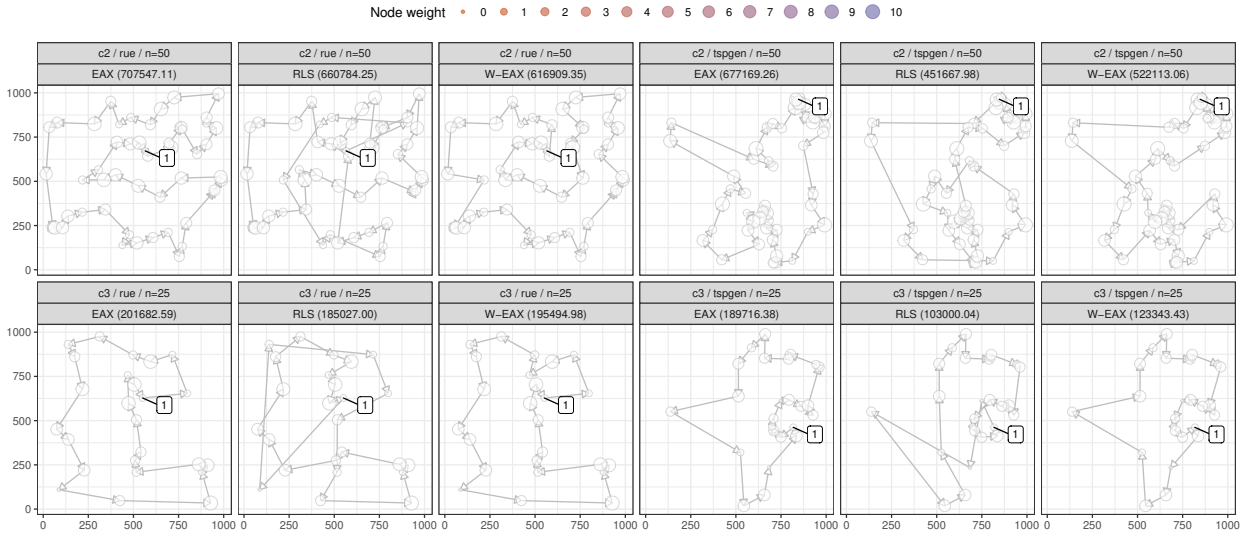


Figure 3: This plot shows weighted TSP tours determined by EAX, W-EAX and RLS[Inversion], respectively, for four selected, yet representative problem instances. The start node $\pi_1 = 1$ is highlighted and the direction of the tour is indicated by arrows.

can be expected. Here, W-EAX manages – due to the weighted TSP fitness function – to cumulate less weight in the respective clusters before they are left in order to reach another cluster. This explains why ratios increase with increasing cluster segregation.

6 CONCLUSIONS

Motivated by different complex variants of the traveling salesperson problem, we have introduced the node weight dependent TSP called W-TSP which captures aspects of important complex TSP variants such as the time dependent TSP or the traveling thief problem. We have pointed out the relation of W-TSP to the TSP and how the weights on the nodes impact the structure of the problem. Our insights provided the tools for designing approximation algorithms for the metric version of the problem. Furthermore, we have shown that approximation algorithms for the $\{1, 2\}$-TSP can be used as the basis for approximation algorithms for W-TSP when also considering the reverse tour. Our experimental studies show that, on almost all considered settings and a wide range of instances, inversion mutation is superior to exchange and jump operators when adopted by randomized local search. Furthermore, experimental investigations on the state-of-the-art TSP solver EAX show that the weights lead to significantly different results when comparing W-TSP to the classical TSP. For future work, it would be interesting to study other state-of-the-art heuristics for the TSP and how to adapt them to W-TSP. In order to systematically judge the performance of such approaches, it would be highly beneficial to have efficient exact solvers for W-TSP.

ACKNOWLEDGEMENTS

Frank Neumann has been support through a Humboldt Fellowship for Experienced Researchers by Alexander von Humboldt Foundation. Katrin Casel was funded by the Federal Ministry of Education and Research of Germany in the framework of KI-LAB-ITSE (project number 01IS19066).

REFERENCES

[1] Louis-Philippe Bigras, Michel Gamache, and Gilles Savard. 2008. The Time-Dependent Traveling Salesman Problem and Single Machine Scheduling Problems with Sequence Dependent Setup Times. *Discrete Optimization* 5, 4 (2008), 685 – 699.

[2] Avrim Blum, Prasad Chalasani, Don Coppersmith, William R. Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. 1994. The Minimum Latency Problem. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*. ACM, 163 – 171.

[3] Mohammad Reza Bonyadi, Zbigniew Michalewicz, and Luigi Barone. 2013. The Travelling Thief Problem: The First Step in the Transition from Theoretical Problems to Realistic Problems. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1037 – 1044.

[4] Mohammad Reza Bonyadi, Zbigniew Michalewicz, Markus Wagner, and Frank Neumann. 2019. Evolutionary Computation for Multicomponent Problems: Opportunities and Future Directions. In *Optimization in Industry, Present Practices and Future Scopes*, Shubhabrata Datta and J. Paulo Davim (Eds.). Springer, 13 – 30. https://doi.org/10.1007/978-3-030-01641-8_2

[5] Jakob Bossek, Pascal Kerschke, Aneta Neumann, Markus Wagner, Frank Neumann, and Heike Trautmann. 2019. Evolving Diverse TSP Instances by Means of Novel and Creative Mutation Operators. In *Proceedings Foundations of Genetic Algorithms (FOGA)*. ACM Press, 58 – 71. https://doi.org/10.1145/3299904.3340307

[6] Björn Brodén, Mikael Hammar, and Bengt J. Nilsson. 2004. Online and Offline Algorithms for the Time-Dependent TSP with Time Zones. *Algorithmica* 39, 4 (2004), 299 – 319.

[7] Kamalika Chaudhuri, Brighten Godfrey, Satish Rao, and Kunal Talwar. 2003. Paths, Trees, and Minimum Latency Tours. In *Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, 36 – 45.

[8] Nicos Christofides. 1976. *Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem.* Technical Report 388. Graduate School of Industrial Administration, Carnegie Mellon University.

[9] Gérard Cornuéjols, Jean Fonlupt, and Denis Naddef. 1985. The Traveling Salesman Problem on a Graph and Some Related Integerpolyhedra. *Mathematical Programming* 33, 1 (1985), 1 – 27.

[10] Mohamed El Yafrani and Belaïd Ahiod. 2016. Population-based vs. Single-solution Heuristics for the Travelling Thief Problem. In *Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 317 – 324.

[11] Hayden Faulkner, Sergey Polyakovskiy, Tom Schultz, and Markus Wagner. 2015. Approximate Approaches to the Traveling Thief Problem. In *Conference on Genetic and Evolutionary Computation (GECCO)*. ACM, 385 – 392.

[12] Raja Jothi and Balaji Raghavachari. 2007. Approximating the k-Traveling Repairman Problem with Repairtimes. *Journal of Discrete Algorithms* 5, 2 (2007), 293 – 303.

[13] Pascal Kerschke, Lars Kotthoff, Jakob Bossek, Holger H Hoos, and Heike Trautmann. 2018. Leveraging TSP Solver Complementarity through Machine Learning. *Evolutionary Computation* 26, 4 (2018), 597 – 620. https://doi.org/10.1162/evco_a_00215

[14] Lars Kotthoff, Pascal Kerschke, Holger H. Hoos, and Heike Trautmann. 2015. Improving the State of the Art in Inexact TSP Solving Using Per-Instance Algorithm Selection. In *Proceedings of the 9th International Conference on Learning and Intelligent Optimization (LION) (Lecture Notes in Computer Science (LNCS))*, Clarisse Dhaenens, Laetitia Jourdan, and Marie-Eléonore Marmion (Eds.), Vol. 8994. Springer, 202 – 217. https://doi.org/10.1007/978-3-319-19084-6_18

[15] Chryssi Malandraki and Mark S. Daskin. 1992. Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms. *Transportation Science* 26, 3 (1992), 185 – 200.

[16] Yi Mei, Xiaodong Li, and Xin Yao. 2016. On Investigation of Interdependence Between Sub-Problems of the Travelling Thief Problem. *Soft Computing* 20, 1 (2016), 157 – 172.

[17] Yuichi Nagata and Shigenobu Kobayashi. 2013. A Powerful Genetic Algorithm Using Edge Assembly Crossover for the Traveling Salesman Problem. *INFORMS Journal on Computing* 25, 2 (2013), 346 – 363. https://doi.org/10.1287/ijoc.1120.0506

[18] Frank Neumann, Sergey Polyakovskiy, Martin Skutella, Leen Stougie, and Junhua Wu. 2018. A Fully Polynomial Time Approximation Scheme for Packing While Traveling. In *4th International Symposium on Algorithmic Aspects of Cloud Computing (ALGOCLOUD), Revised Selected Papers (LNCS)*, Vol. 11409. Springer, 59 – 72. https://doi.org/10.1007/978-3-030-19759-9_5

[19] Christos H. Papadimitriou and Mihalis Yannakakis. 1993. The Traveling Salesman Problem with Distances One and Two. *Math. Oper. Res.* 18, 1 (1993), 1–11. https://doi.org/10.1287/moor.18.1.1

[20] Jean-Claude Picard and Maurice Queyranne. 1978. The Time-Dependent Traveling Salesman Problem and Its Application to the Tardiness Problem in One-Machine Scheduling. *Operations Research* 26, 1 (1978), 86 – 110.

[21] Sergey Polyakovskiy, Mohammad Reza Bonyadi, Markus Wagner, Zbigniew Michalewicz, and Frank Neumann. 2014. A Comprehensive Benchmark Set and Heuristics for the Traveling Thief Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 477 – 484. https://doi.org/10.1145/2576768.2598249

[22] Jens Scharnow, Karsten Tinnefeld, and Ingo Wegener. 2004. The analysis of evolutionary algorithms on sorting and shortest paths problems. *J. Math. Model. Algorithms* 3, 4 (2004), 349–366. https://doi.org/10.1007/s10852-005-2584-0

[23] Leonard J. Testa, Albert C. Esterline, Gerry V. Dozier, and Abdollah Homaifar. 2000. A Comparison of Operators for Solving Time dependent Traveling Salesman Problems Using Genetic Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. 995–1102.

[24] Markus Wagner, Marius Lindauer, Mustafa Mısır, Samadhi Nallaperuma, and Frank Hutter. 2017. A Case Study of Algorithm Selection for the Traveling Thief Problem. *Journal of Heuristics* 24, 3 (07 Apr 2017), 295 – 320. https://doi.org/10.1007/s10732-017-9328-y

[25] Junhua Wu, Sergey Polyakovskiy, and Frank Neumann. 2016. On the Impact of the Renting Rate for the Unconstrained Nonlinear Knapsack Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 413 – 419. https://doi.org/10.1145/2908812.2908862

[26] Junhua Wu, Sergey Polyakovskiy, Markus Wagner, and Frank Neumann. 2018. Evolutionary Computation Plus Dynamic Programming for the Bi-Objective Travelling Thief Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 777 – 784. https://doi.org/10.1145/3205455.3205488

[27] Junhua Wu, Markus Wagner, Sergey Polyakovskiy, and Frank Neumann. 2017. Exact Approaches for the Travelling Thief Problem. In *Proceedings of the 11th International Conference on Simulated Evolution and Learning (SEAL)*. Springer, 110 – 121.

[28] Mohamed El Yafrani and Belaïd Ahiod. 2018. Efficiently Solving the Traveling Thief Problem Using Hill Climbing and Simulated Annealing. *Information Sciences* 432 (2018), 231–244.