

Reducing the Size of Traveling Salesman Problem Instances by Fixing Edges

Thomas Fischer and Peter Merz

Distributed Algorithms Group
Department of Computer Science
University of Kaiserslautern, Germany
{fischer,pmerz}@informatik.uni-kl.de

Abstract. The Traveling Salesman Problem (TSP) is a well-known NP-hard combinatorial optimization problem, for which a large variety of evolutionary algorithms are known. However, these heuristics fail to find solutions for large instances due to time and memory constraints. Here, we discuss a set of edge fixing heuristics to transform large TSP problems into smaller problems, which can be solved easily with existing algorithms. We argue, that after expanding a reduced tour back to the original instance, the result is nearly as good as applying the used solver to the original problem instance, but requiring significantly less time to be achieved. We claim that with these reductions, very large TSP instances can be tackled with current state-of-the-art evolutionary local search heuristics.

1 Introduction

The Traveling Salesman Problem (TSP) is a widely studied combinatorial optimization problem, which is known to be NP-hard [1].

Let $G = (V, E, d)$ be an edge-weighted, directed graph, where V is the set of $n = |V|$ vertices, $E \subseteq V \times V$ the set of (directed) edges and $d : E \rightarrow \mathbb{R}^+$ a *distance function* assigning each edge $e \in E$ a distance $d(e)$. A *path* is a list (u_1, \dots, u_k) of vertices $u_i \in V$ ($i = 1, \dots, k$) holding $(u_i, u_{i+1}) \in E$ for $i = 1, \dots, k - 1$. A *Hamiltonian cycle* in G is a path $p = (u_1, \dots, u_k, u_1)$ in G , where $k = n$ and $\bigcup_{i=1}^k u_i = V$ (each vertex is visited exactly once except for u_1). The TSP's objective is to find a Hamiltonian cycle t for G that minimizes the cost function $C(t) = \sum_{i=1}^{k-1} d((u_i, u_{i+1})) + d((u_k, u_1))$ (weights of the edges in t added up). Depending on the distance function d , a TSP instance may be either *symmetric* (for all $u_1, u_2 \in V$ holds $d((u_1, u_2)) = d((u_2, u_1))$) or *asymmetric* (otherwise). Most applications and benchmark problems are *Euclidean*, i. e., the vertices V correspond to points in an Euclidean space (mostly 2-dimensional) and the distance function represents an Euclidean distance metric. The following discussion focuses on symmetric, Euclidean problem instances.

Different types of algorithms for the TSP are known, such as exact algorithms [2, 3] or local search algorithms [4]. Among the best performing algorithms are those utilizing Lin-Kernighan local search within an evolutionary framework such

as Iterated Lin-Kernighan [5] or memetic algorithms [6, 7]. Even exact algorithms like Branch & Cut rely on these heuristics. The heuristics used in Concorde [8] to find near-optimum solutions to large TSP instances is essentially a memetic algorithm using the Lin-Kernighan (LK) heuristics as local search and tour-merging [9] for recombination [10]. As the TSP is NP-hard, computation time is expected to grow exponentially with the instance size. E. g. for a TSP instance with 24 978 cities, even sophisticated heuristic algorithms such as Helsgaun’s LK (LK-H) [11] require several hours to find solutions within 1% distance to the optimum. For the same instance, an exact algorithm required 85 CPU years to prove a known tour’s optimality [12].

The problem of time consumption can be approached by distributing the computation among a set of computers using distributed evolutionary algorithms (DEA) [13, 14]. Another problem when solving extremely large TSP instances such as the World TSP [15] is an algorithm’s memory consumption, as data structures such as neighbor or candidate lists have to be maintained. We address this problem in this paper by proposing different *edge fixing heuristics*, which may reduce the problem to a size suitable for standard TSP solvers. In the general fixing scheme heuristics select edges of an existing tour for fixing; paths of fixed edges are merged into a single fixed edge reducing the instance size. Fixed edges are ‘tabu’ for the TSP solver, which is applied to the reduced instance in a second step. Finally, the optimized tour is expanded back to a valid solution for the original problem by releasing fixed edges and paths.

The remainder of this section discusses related work from Walshaw. In Sect. 2 problem reduction techniques based on fixing edges are discussed. A set of TSP instances is analyzed in Sect. 3 regarding the discussed fixing heuristics. Sect. 4 discusses the results when applying the fixing heuristics to an evolutionary local search. Sect. 5 summarizes our findings.

1.1 Related Work

Only limited research regarding the reduction of TSP instances in relation with evolutionary local search has been done. The primary related work to our concept is the *multilevel approach* by Walshaw [16], which has been applied to several graph problems including the TSP [17]. Basically, multilevel algorithms work as follows: In the first phase a given graph is recursively coarsened by matching and merging node pairs generating smaller graphs at each level. The coarsening stops with a minimum size graph, for which an optimal solution can easily be found. In the second phase, the recursion backtracks, uncoarsening each intermediate graph and finally resulting in a valid solution of the original problem. In each uncoarsening step, the current solution is refined by some optimization algorithm. It has been reported that this strategy results in better solutions compared to applying the optimization algorithm to the original graph only. When uncoarsening again, the optimization algorithm can improve the current level’s solution based on an already good solution found in the previous level. As the coarsening step defines the solution space of a recursion level, its strategy is decisive for the quality of the multilevel algorithm.

In [17] the multilevel approach has been applied to the TSP using a CLK algorithm [18] for optimization. Here, a multilevel variant ($\text{MLC}^{N/10}\text{LK}$) of CLK gains better results than the unmodified CLK, being nearly 4 times faster. The coarsening heuristics applied to the TSP’s graph matches node pairs by adding a fixed edge in between. In each step, nodes are selected and matched with their nearest neighbor, if feasible. Nodes involved in an (unsuccessful) matching may not be used in another matching at the same recursion level to prevent the generation of sub-tours. Recursion stops when only two nodes and one connecting edge are left.

2 Problem Reduction by Fixing Edges

To reduce a TSP instance’s size different approaches can be taken. Approaches can be either node-based or edge-based. At a different level, approaches can be based only on a TSP instance or using an existing solution, respectively.

A *node-based approach* may work as follows: Subsets of nodes can be merged into meta-nodes (cluster) thus generating a smaller TSP instance. Within a meta-node a cost-effective path connecting all nodes has to be found. The path’s end nodes will be connected to the edges connecting the meta-node to its neighbor nodes building a tour through all meta-nodes. Problems for this approach are (i) how to group nodes into meta-nodes (ii) how to define distances between meta-nodes (iii) which two nodes of a cluster will have outbound edges. In an *edge-based approach*, a sequence of edges can be merged into a meta-edge, called a fixed path. Subsequently, the inner edges and nodes are no longer visible and this meta-edge has to occur in every valid tour for this instance. Compared to the node-based approach, problems (ii) and (iii) do not apply, as the original node distances are still valid and a fixed path has exactly two nodes with outbound edges. So, the central problem is how to select edges merged into a meta-edge. Examples for both node-based and edge-based problem reductions are shown in Fig. 1.

Edges selected for merging into meta-edges may be chosen based on instance information only or on a tour’s structure. The former approach may select from an instance with n nodes any of the $\frac{n(n-1)}{2}$ edges for a merging step, the latter approach reuses only edges from a given tour (n edges). The *tour-based approach*’s advantage is a smaller search space and the reuse of an existing tour’s inherent knowledge. Additionally, this approach can easily be integrated into memetic algorithms. A disadvantage is that the restriction to tour edges will limit the fixing effect especially in early stages of a local search when the tour quality is not sufficient.

Walshaw’s multilevel TSP approach focuses on an edge-based approach considering the TSP instance only. In this paper, we will discuss edge-based approaches, too, but focus on the following tour-based edge fixing heuristics:

Minimum Spanning Tree (MST) Tour edges get fixed when they occur in a *minimum spanning tree* (MST) for the tour’s instance. This can be motivated

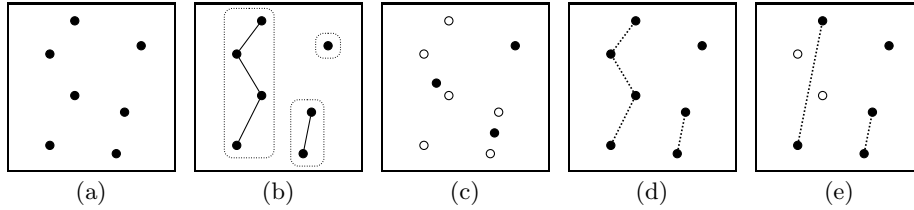


Fig. 1. Examples of node-based and edge-based problem reductions. Starting from the original problem instance (a), the node-based approach assigns node sets to clusters (marked by dashed boxes) and defines a spanning path within each cluster (b). Subsequently, in (c) only representatives of the clusters have to be considered (black nodes, here arbitrarily located at each cluster’s center), whereas the original nodes (white) can be ignored. For the edge-based approach, edges to be fixed have to be selected (dotted lines in (d)). Subsequently, paths can be merged to single edges and inner path nodes (white) may be ignored (e).

by the affinity between the TSP and the MST problem [19], as the latter one can be used to establish a lower bound for the TSP. However, global instance knowledge in form of an MST (complexity of $O(m + n \log n)$ for m edges using Fibonacci heaps) has to be available in advance.

Nearest Neighbor (NN) As already exploited by the *nearest neighbor* tour construction heuristics, edges between a node and its nearest neighbor are likely to occur in optimal tours thus being promising fixing candidates, too. Determining nearest neighbor lists may be computationally expensive (complexity of $O(n^2 \log n)$), but can be sped up e. g. by using *kd-trees* [20, 21].

Lighter than Median (<M) Edges that length is below the median over all edges’ lengths in a tour are selected, as it is beneficial to keep short edges by fixing them and leaving longer edges for further optimization. The most expensive operation of this approach is the necessary sorting of all tour edges (complexity of $O(n \log n)$). There may be tours that have very few different edge lengths resulting in a small number of edges that are strictly shorter than the median.

Close Pairs (CP) Here, a tour edge’s length is compared to the lengths of the two neighboring edges. The edge will be fixed if it is shorter than both neighboring edges and the edge’s nodes therefore form a *close pair*. This approach considers only local knowledge (edge and its two neighbor edges) allowing it to be applied even on large instances. It is expected to work well in graphs with both sparse and dense regions.

The actual number of edges selected by one of the above heuristics during a fixing step depends on the current tour and the problem instance. For the first two heuristics (MST and NN) it can be expected that more edges will be selected with better tours. For the lighter than median variant (<M) and the close pairs variant (CP) at most half of all edges may be selected. In the example in Fig. 2, solid lines represent tour edges and dotted lines represent edges of the minimal

spanning tree (MST). Out of 7 tour edges, 5 edges are MST edges, too, 4 edges connect nearest neighbors (NN), 3 edges connect close node pairs (CP) and 3 edges are lighter than the edge weight median ($<M$).

For asymmetric TSP (ATSP) instances edge fixation heuristics can be developed, too, but this approach has not been pursued here. Applying the fixations heuristics presented here to ATSP instances poses new questions such as determining the direction when the fixation heuristics is based on undirected knowledge (e. g. from an MST).

3 Analysis of TSP Instances

The tests in this section were performed to evaluate if the edge selection strategies discussed in this paper are applicable for edge-fixation heuristics. Probabilities of each selection strategy to select a tour edge and the probabilities if the selected edge is part of an optimal tour were evaluated. These criteria describe the quantity and quality, respectively, of a selection strategy.

For our analysis, five TSP instances have been selected: From the TSPLIB collection [22] instances `brd14051`, `d15112`, and `d18512` and from a collection of national TSPs [23] instances `it16862`, and `sw24978` were taken (numbers in the instances' names denote the problem size). These instances were selected, because they are among the largest instances an optimal solution is known for. The choice of instances was limited by the availability of optimal or at least high quality tours which were used to to evaluate the tours found in our experiments. Preferably, larger TSPLIB instances and benchmark instances from the DIMACS challenge [24] (E series instances) would have been included, too.

For each TSP instance 20 nearest-neighbor tours were constructed. Each of these tours was optimized to a local optimum by the local search heuristics 2-opt, 3-opt, Lin-Kernighan (LK-opt), and LK-Helsgaun (LK-H), respectively. For Helsgaun's LK parameter `MAX_TRIALS` was set to 100 (instead of number of cities). Totally, 500 tours were constructed to test the fixing heuristic when applied to tours of different quality levels.

Each heuristics' selection scheme was applied to the set of 500 tours. Average values over each set of 20 tours with the same setup were taken and summarized

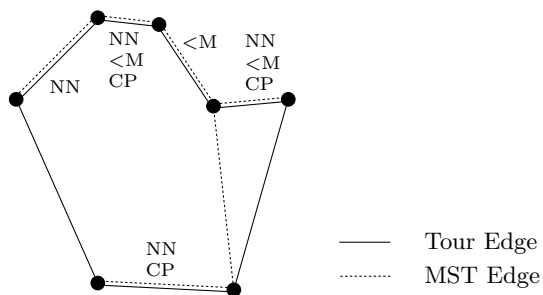


Fig. 2. Example tour and minimum spanning tree with different edge properties highlighted. Tour and MST edges are drawn in different line styles, properties *nearest neighbor* (NN), *lighter than median* ($<M$), and *close pairs* (CP) are shown as markings next to the edges.

Type	P(OPT)	P(MST)	P(OPT MST)	P(NN)	P(OPT NN)	P(<M)	P(OPT <M)	P(CP)	P(OPT CP)	
brd14051	NN	59.12	70.44	73.22	58.29	74.82	54.64	71.07	40.11	71.68
	2-opt	65.13	73.58	75.78	60.39	77.33	53.28	74.67	38.49	75.70
	3-opt	70.49	76.20	78.47	62.73	79.70	52.17	77.77	37.14	78.83
	LK-opt	77.93	77.53	83.07	62.99	83.99	52.16	82.78	35.03	83.55
	LK-H	92.44	76.59	93.93	61.18	94.08	48.61	93.45	31.75	93.88
optimal	100.00	75.33	100.00	59.43	100.00	47.83	100.00	30.34	100.00	
dl5112	NN	61.21	71.33	74.09	59.16	75.89	54.86	71.73	39.88	73.37
	2-opt	66.06	73.69	75.99	60.58	77.73	54.22	74.36	38.50	76.24
	3-opt	71.07	76.04	78.61	62.99	79.91	53.67	77.63	37.70	79.07
	LK-opt	77.50	77.01	82.63	63.12	83.71	52.38	82.03	35.83	83.09
	LK-H	92.11	76.15	93.51	61.33	93.83	50.65	93.05	32.59	93.67
optimal	100.00	74.70	100.00	59.61	100.00	49.79	100.00	31.25	100.00	
it16862	NN	61.35	72.20	74.44	60.06	76.04	52.60	70.59	32.56	73.59
	2-opt	66.72	74.66	76.64	61.90	78.10	53.82	74.34	30.56	76.29
	3-opt	71.87	76.94	79.33	64.34	80.37	54.63	79.06	29.08	79.34
	LK-opt	77.91	77.64	83.07	64.23	84.03	53.35	83.12	27.14	83.28
	LK-H	91.56	76.58	93.10	62.52	93.22	51.16	92.70	24.07	93.29
optimal	100.00	75.07	100.00	60.59	100.00	49.92	100.00	22.89	100.00	
dl8512	NN	60.68	71.58	73.99	59.19	75.71	55.22	72.11	39.14	73.05
	2-opt	65.88	74.18	76.01	60.77	77.62	53.83	75.06	37.94	76.13
	3-opt	70.80	76.68	78.40	63.05	79.74	53.23	77.94	37.15	78.88
	LK-opt	77.95	77.76	82.95	63.15	83.97	52.94	82.73	35.05	83.38
	LK-H	92.70	76.66	94.08	61.16	94.22	50.03	93.67	31.72	94.01
optimal	100.00	75.34	100.00	59.41	100.00	49.06	100.00	30.34	100.00	
sw24978	NN	65.39	76.00	75.24	65.38	75.78	42.56	74.30	24.96	74.83
	2-opt	68.18	75.47	76.95	64.14	77.67	41.14	75.22	25.04	76.94
	3-opt	72.26	76.91	79.09	65.45	79.67	41.82	77.55	24.30	79.11
	LK-opt	77.22	77.00	82.38	64.46	83.08	41.19	80.97	23.18	82.45
	LK-H	90.33	76.01	92.06	62.90	92.13	40.31	90.66	20.54	92.56
optimal	100.00	74.38	100.00	60.85	100.00	39.15	100.00	19.36	100.00	

Table 1. Probabilities (in percent) for edges in a tour to match certain criteria. The data is grouped by TSP instance and tour type. The eight left most columns contain the original and the conditional probabilities for the four edge fixation heuristics discussed in this paper.

in Tab. 1. Here, the first column sets the instance under consideration. The ‘Type’ column determines in which local optimum the tours are located. Column ‘P(OPT)’ shows for a local optimal tour the percentage of edges that also occur in the known global optimal tour. Columns ‘P(MST)’, ‘P(NN)’, ‘P(<M)’, and ‘P(CP)’ contain the probability that an edge from a local optimal tour matches the given property. Columns ‘P(OPT|MST)’, ‘P(OPT|NN)’, ‘P(OPT|<M)’, and ‘P(OPT|CP)’ contain conditional probabilities, that an edge in a local optimal tour is part of the global optimal tour, given that it matches the properties MST, NN, <M, or CP, respectively.

Column ‘P(OPT)’ shows the percentage of edges from a local optimal tour occurring in the global optimal tour. The better a tour construction and improvement heuristics works, the more edges the resulting tour has in common with the optimal tour. Whereas tours constructed by a nearest neighbor construction

heuristics share about 60–65% of all edges with optimal tours (depending on the instance), tours found by subsequently applied local search algorithms are better, ranging from 65–70% for 2-opt to more than 90% for LK-H-opt tours.

As each edge selection strategy has different criteria how to select edges, they differ in the number of optimal edges. The most edges get chosen by the MST strategy (column ‘P(MST)’) selecting about 70–80% of all tour edges. Other strategies select less edges: From NN with 60–65% down to <M and CP with 40–55% and 20–40% of all edges, respectively. Interestingly, the number of selected edges decreases for all instances and all selection strategies when applied to high quality tours such as ‘LK-H’ and optimal tours.

However, the quantity (number of edges selected for fixing) is not the only criterion to rate a selection strategy. Selected edges were checked whether they occur in the corresponding known optimal tour, too. When applying a fixing heuristics to a sub-optimal tour, a good heuristics should more likely select edges that occur in the optimal tour, too, rather than edges that do not occur in the optimal tour. Therefore we were interested in the probability that a sub-optimal tour’s edge selected by a edge selection strategy would actually be contained in an optimal tour (‘true positive’) rather than being a ‘false positive’.

Edge selection strategies tend to be more successful for better tours. For every selection strategy, the percentage of correctly selected edges (edges that occur in the optimal tour, too) increases with the tour quality. E. g. for nearest neighbor tours of instance `it16862` using the MST heuristics, only about 74.4% of all selected edges are optimal tour edges, too, but for LK-H optimal tours, the percentage of selected edges is much higher (93.1%). Comparing selection strategies, the nearest neighbor selection strategy (NN) has the best probability values for all combinations except for three cases, where the close pairs strategy (CP) is more likely to find the right edges for LK-H tours. Especially for lower quality tours (NN and 2-opt), selection strategies <M and CP have the lowest conditional probabilities, but this effect disappears with higher quality tours.

4 Experimental Setup and Results

For the experimental evaluation, the fixing heuristics have been integrated into a simple TSP solver written in Java. The solver works as follows: Each tour was reduced using one of the fixing heuristics and subsequently improved by an iterated local search (ILS) algorithm. In each iteration of the algorithm the current tour was perturbed and locally optimized by an LK implementation, which is able to handle fixed edges. For the perturbation a variable-strength *double-bridge move* (DBM) was used increasing the number of DBMs each two non-improving iterations. At the end of each iteration the new tour was compared to the previous tour and discarded if no improvement was found, otherwise kept for subsequent iterations. The iterations would stop after 2 non-improving iterations. Finally, the improved tour was expanded back to a solution for the original TSP instance. For comparison, all tours were optimized by the iterated local search algorithm without any reduction, too. This solver was not designed

to compete with state-of-the-art solvers, but merely to evaluate our fixation heuristics. Each parameter setup was tested by applying it to the tours described in Sec. 3; average values were used for the following discussion. Computation times are utilized CPU time on a 2.8 GHz Pentium 4 system with 1 GB memory running Linux.

Table 2 holds the results for the different setups applied to the ILS and is structured as follows: Rows are grouped by instance (`brd14051` to `sw24978`) and by starting tour for the ILS ('NN' to 'LK-H'). The instances are ordered by number of cities, the starting tours are ordered by descending tour length. Columns are grouped into blocks, where the first block summarizes an ILS setup without any fixation and the subsequent blocks summarize ILS setups with each fixation heuristics ('MST' to 'Close Pairs'). Each column block consists of four columns: Improvement found when applying the ILS, required CPU time until termination, size of the reduced instance (normalized number of cities), and fraction of edges that are fixed (tabu for any ILS operation).

For every instance and each fixing heuristics (including no fixing) holds that the better the starting tour is, the smaller the improvements found by the ILS are. Applying our TSP solver to nearest neighbor tours (Tab. 2, rows with start tour type 'NN') results in improvements of more than 20% for most setups (columns 'Impr. [%]'). For better starting tours, less improvement is achieved, down to improvements of 0% for starting tours coming from LK-H.

Each fixing ILS setup can be compared with the corresponding ILS setup without fixing regarding improvement on the given start tour and the required CPU time. The following observations can be drawn from Tab. 2:

- For non-fixing setups, the CPU time is always higher compared to fixing setups, as the effective problem size is larger for the non-fixing setup. However, time consumption does not directly map to better tour quality.
- The Close Pairs (CP) fixing heuristics yields in improvements as good as for non-fixing ILS, but requires significantly less time to reach these quality levels. E. g. for instance `brd14051` starting with 3-opt tours, both the non-fixing ILS and the CP fixing ILS improve the start tour by about 6.2%, but the CP-ILS requires only 31.9s, whereas the non-fixing ILS requires 44.8s.
- For the other fixing heuristics hold that they consume both less CPU time and result in lesser improvements. Although this makes comparing the different fixing strategies hard, improvements are still competitive while requiring significantly less CPU time compared to the non-fixing ILS.
- Among all fixation-based ILS setups, the MST heuristics results in both the smallest improvements and lowest running times compared to the other fixation heuristics. E. g. for instance `sw24978` starting with 2-opt tours, the MST heuristics results in an improvement of 10.1% consuming 21.8s, whereas all other fixation and non-fixation ILS setups find improvements of 11.4% and better consuming 41.0s and more.
- Comparing CPU time consumption versus possible improvement, the fixation heuristics can be put into three groups: Between the expensive, but good CP heuristics and the cheap, but not so good MST heuristics the re-

Instance	Start Tour Type	No Fixing				MST				Nearest Neighbor				Light Median \triangleleft				Close Pairs			
		Impr. [%]	CPU Time [s]	Free Edges [%]	Red. Size [%]	Impr. [%]	CPU Time [s]	Free Edges [%]	Red. Size [%]	Impr. [%]	CPU Time [s]	Free Edges [%]	Red. Size [%]	Impr. [%]	CPU Time [s]	Free Edges [%]	Red. Size [%]	Impr. [%]	CPU Time [s]	Free Edges [%]	Red. Size [%]
brd14051	NN	24.1	36.3	-	-	21.7	8.5	56.8	47.1	23.3	14.5	55.5	72.5	22.6	18.4	73.7	68.9	24.0	31.6	68.8	100.0
	2-opt	13.2	33.8	-	-	10.6	7.3	55.8	46.8	12.3	13.3	55.0	72.2	11.9	15.2	73.9	68.8	13.0	25.2	68.8	100.0
	3-opt	6.3	44.8	-	-	4.1	8.1	55.1	46.8	5.5	15.7	54.7	72.1	5.4	18.9	74.5	69.2	6.2	31.9	68.8	100.0
	LK-opt	0.1	19.6	-	-	0.0	4.0	55.0	49.0	0.0	7.7	56.0	75.5	0.1	8.8	74.4	68.1	0.1	13.8	69.2	100.0
	LK-H	0.0	11.4	-	-	0.0	2.9	54.4	45.6	0.0	5.5	55.1	73.5	0.0	6.4	75.6	69.1	0.0	9.4	69.6	100.0
d15112	NN	25.5	43.7	-	-	22.9	9.9	57.1	46.8	24.5	17.8	55.5	71.6	23.9	20.8	75.4	66.8	25.2	37.7	68.2	100.0
	2-opt	14.6	38.7	-	-	12.1	8.1	56.0	46.9	13.7	15.1	55.2	71.6	13.3	16.8	75.5	66.9	14.4	30.8	68.3	100.0
	3-opt	6.7	44.2	-	-	4.6	7.8	55.4	47.4	5.9	15.3	54.9	71.7	5.7	17.2	75.7	66.7	6.5	32.6	68.2	100.0
	LK-opt	0.1	22.1	-	-	0.0	4.3	55.2	49.6	0.0	8.2	56.2	75.3	0.0	8.7	75.8	66.5	0.1	15.1	68.6	100.0
	LK-H	0.0	12.4	-	-	0.0	3.2	54.3	46.6	0.0	5.8	55.2	73.5	0.0	6.5	76.4	65.9	0.0	10.0	68.9	100.0
it16862	NN	24.2	51.1	-	-	21.7	10.5	56.4	45.6	23.4	17.7	55.2	70.2	23.6	23.3	71.4	70.7	24.3	41.4	75.8	100.0
	2-opt	13.5	60.6	-	-	11.2	11.2	55.5	45.5	12.7	20.4	54.9	70.1	12.9	24.7	71.7	70.7	13.5	46.7	75.9	100.0
	3-opt	6.5	57.0	-	-	4.5	9.2	54.9	46.0	5.8	17.7	54.5	70.1	6.1	24.0	72.0	71.2	6.4	41.7	76.1	100.0
	LK-opt	0.8	35.1	-	-	0.4	5.6	55.0	48.5	0.7	11.8	55.7	73.3	0.9	14.6	71.6	70.2	0.7	25.9	76.5	100.0
	LK-H	0.0	13.6	-	-	0.0	3.3	54.3	46.0	0.0	6.1	55.0	71.9	0.0	7.6	71.5	70.2	0.0	11.3	77.2	100.0
d18512	NN	24.2	37.5	-	-	21.7	8.9	56.7	46.6	23.4	15.3	55.3	72.4	22.8	19.9	72.8	70.2	24.1	32.8	68.9	100.0
	2-opt	13.7	49.8	-	-	11.2	10.6	55.8	46.7	12.8	19.5	55.0	72.4	12.6	22.9	73.0	70.1	13.6	39.4	68.9	100.0
	3-opt	6.8	47.6	-	-	4.6	8.9	55.1	47.0	5.9	18.0	54.7	72.4	5.9	19.8	73.1	69.8	6.6	34.6	68.8	100.0
	LK-opt	0.1	25.0	-	-	0.0	5.4	55.0	48.9	0.0	10.6	56.1	75.7	0.0	11.6	73.0	69.2	0.0	18.4	69.4	100.0
	LK-H	0.0	14.4	-	-	0.0	3.7	54.3	45.4	0.0	7.1	55.1	73.6	0.0	8.0	73.5	69.3	0.0	11.8	69.7	100.0
sw24978	NN	20.8	100.5	-	-	17.8	16.1	56.7	43.1	19.6	29.7	54.9	66.0	20.1	48.6	77.4	77.4	20.7	89.0	79.7	100.0
	2-opt	12.6	144.3	-	-	10.1	21.8	56.0	45.2	11.7	41.0	54.9	67.4	12.1	65.2	77.2	78.3	12.6	119.1	79.6	100.0
	3-opt	6.0	109.5	-	-	4.1	14.7	55.2	46.1	5.2	29.9	54.8	68.1	5.6	48.2	77.0	78.4	5.9	88.5	79.6	100.0
	LK-opt	0.5	57.2	-	-	0.1	8.2	55.3	49.4	0.2	16.3	56.1	72.2	0.3	26.7	77.1	79.9	0.3	46.2	80.0	100.0
	LK-H	0.0	17.7	-	-	0.0	4.9	54.7	47.2	0.0	8.3	55.3	70.8	0.0	12.7	76.9	79.2	0.0	16.9	80.7	100.0

Table 2. Results for different setups of instance, start tour, and fixing heuristics. The data is grouped by TSP instance and tour type. The result columns can be grouped into five blocks (for different fixing heuristics) consisting of four columns each: Achieved improvement from starting tour, utilized CPU time, fraction of free edges after fixing and size of the reduced instance (based on number of nodes).

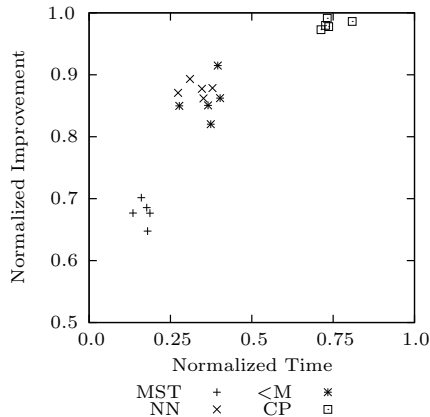


Fig. 3. For each of the five instances using 3-opt starting tours, improvements and CPU time of the ILS with fixation (MST, NN, <M, and CP) are normalized with the results from ILS runs without fixation.

maintaining two heuristics NN and <M can be located. These two “medium” heuristics show similar results both for found tour improvement and CPU time consumption.

The last observation can be visualized as in Fig. 3, which compares the improvements for each fixing heuristics applied to 3-opt tours. Both time and improvement are normalized for each of the five TSP instances by the values from the runs without fixing. The CP and the MST heuristics’ values are separated from the central cluster consisting of NN and <M results. As can be seen, the CP heuristics reaches improvements as good as the non-fixing ILS (normalized improvement close to 1.0), but requires only $\frac{3}{4}$ of the time. NN and <M heuristics find improvements of about 90% of those from the non-fixing ILS and still demand less than the half of the time. The MST heuristics reaches about 70% of the non-fixing ILS’s improvement, while consuming only a quarter of the CPU time.

For all fixing heuristics the size of the original instance has been compared to the corresponding reduced instances’ size (in percent, columns ‘Red. Size [%]’ in Tab. 2) and the number of free edges within the reduced instance (in percent, columns ‘Free Edges [%]’).

- For close pairs (CP) fixations holds that the reduced instance’s size equals always with the original instance’s size, as fixed edges can not have neighboring edges that are fixed, too, as this would contradict the selection criterion. Thus, no nodes are redundant.
- For all combinations of instance and start tour, the MST fixing heuristics is the most progressive one resulting in the smallest instances. Here, fixed instances have on average less than half the number of cities compared to the original instances. Within these reduced instances, more than half of the edges are free for the local search. E. g. for instance `d15112` and 3-opt tours, only 47.4% of the original nodes are left, whereas the other heuristics leave 54.9% (NN) to 75.7% (<M) (not considering CP).

- The nearest neighbor heuristics reduces all instance types for about the same level (to 70–75% of the original size) except for instance `sw24978`, where the fixed instance reach 66.0% to 72.2% of the original instances’ sizes.
- Over all instances and start tour type the nearest neighbor heuristics has a very stable percentage of free edges, ranging only between 54.2% and 56.2%.

5 Conclusion

In order to extend current memetic algorithms for the TSP to find close to optimum solutions for large TSP instances, we studied several edge-based problem reduction techniques that can easily be incorporated into an evolutionary local search framework. We have shown that fixing edges in TSP tours can considerably reduce the computation time of a TSP solver compared to applying the same solver to the unmodified problem instance. Still, the solutions found when using fixing heuristics are nearly as good as the solutions found without fixing. Therefore, edge fixing is a feasible approach to solve tours that are otherwise too large for solvers regarding memory or time consumption.

When selecting one of the proposed fixing heuristics, a trade-off between expected solution quality, computation time, or required preprocessing steps has to be made. E. g. the MST heuristics is expected to consume the least time, but requires building an MST in advance. The close pairs strategy can be used if no global knowledge is available, but here too few edges get fixed to decrease an instance’s size considerably. As a compromise regarding time and quality, either the nearest neighbor or the lighter than median heuristics can be applied.

Future work will focus on developing fixing heuristics for population-based EAs and for very large TSP instances. For EAs, the knowledge of which edges occur in parent tours can be used to select edges in offspring tours. For very large instances such as the World TSP, fixation heuristics may exploit geographical properties.

References

1. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, CA, USA (1979)
2. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: On the Solution of Traveling Salesman Problems. *Documenta Mathematica Extra Volume ICM III* (1998) 645–656
3. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: Implementing the Dantzig-Fulkerson-Johnson Algorithm for large Traveling Salesman Problems. *Mathematical Programming* **97** (2003) 91–153
4. Lin, S., Kernighan, B.W.: An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research* **21**(2) (1973) 498–516
5. Johnson, D.S.: Local optimization and the traveling salesman problem. In: *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*. Number 443 in *Lecture Notes in Computer Science*, Warwick University, England, Springer-Verlag (1990) 446–461

6. Moscato, P., Norman, M.G.: A Memetic Approach for the Traveling Salesman Problem Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems. In Valero, M., Onate, E., Jane, M., Larriba, J.L., Suarez, B., eds.: *Parallel Computing and Transputer Applications*, Amsterdam, IOS Press (1992) 177–186
7. Merz, P., Freisleben, B.: Memetic Algorithms for the Traveling Salesman Problem. *Complex Systems* **13**(4) (2001) 297–345
8. Applegate, D., Bixby, R., Chvátal, V., Cook, W.J.: Concorde TSP Solver. <http://www.tsp.gatech.edu/concorde/> (2005)
9. Cook, W., Seymour, P.: Tour Merging via Branch-Decomposition. *INFORMS Journal on Computing* **15**(3) (2003) 233–248
10. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: Finding Cuts in the TSP (a Preliminary Report). Technical Report 95-05, Rutgers University, Piscataway NJ (1995)
11. Helsgaun, K.: An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. *European Journal of Operational Research* **126**(1) (2000) 106–130
12. Cook, W.J.: Log of SW24978 Computation. <http://www.tsp.gatech.edu/world/swlog.html> (2004)
13. Arenas, M.G., Collet, P., Eiben, A.E., Jelasity, M., Merelo, J.J., Paechter, B., Preuß, M., Schoenauer, M.: A Framework for Distributed Evolutionary Algorithms. In Guervós, J.J.M., Adamidis, P., Beyer, H.G., Fernández-Villacañas, J.L., Schwefel, H.P., eds.: *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature, PPSN VII*. Volume 2439 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg (2002) 665–675
14. Fischer, T., Merz, P.: Embedding a Chained Lin-Kernighan Algorithm into a Distributed Algorithm. In: *MIC'2005 – 6th Metaheuristics International Conference*, Vienna, Austria (2005)
15. Cook, W.J.: World Traveling Salesman Problem. <http://www.tsp.gatech.edu/world/> (2005)
16. Walshaw, C.: Multilevel Refinement for Combinatorial Optimisation Problems. *Annals of Operations Research* **131** (2004) 325–372
17. Walshaw, C.: A Multilevel Approach to the Travelling Salesman Problem. *Operations Research* **50**(5) (2002) 862–877
18. Applegate, D., Cook, W.J., Rohe, A.: Chained Lin-Kernighan for Large Traveling Salesman Problems. *INFORMS Journal on Computing* **15**(1) (2003) 82–92
19. Kruskal, J.B.: On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society* **7** (1956) 48–50
20. Friedman, J.H., Baskett, F., Shustek, L.H.: An Algorithm for Finding Nearest Neighbors. *IEEE Transactions on Computers (TOC)* **C-24** (1975) 1000–1006
21. Sproull, R.F.: Refinements to Nearest-Neighbor Searching in k -Dimensional Trees. *Algorithmica* **6**(4) (1991) 579–589
22. Reinelt, G.: TSPLIB — a traveling salesman problem library. *ORSA Journal on Computing* **3**(4) (1991) 376–384 See also <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>.
23. Cook, W.J.: National Traveling Salesman Problems. <http://www.tsp.gatech.edu/world/countries.html> (2005)
24. Johnson, D.S., McGeoch, L.A. In: *Experimental Analysis of Heuristics for the STSP*. Kluwer Academic Publishers (2002) 369–443