

Effective Heuristics for Large Euclidean TSP Instances Based on Pseudo Backbones¹

C. Dong C. Ernst G. Jäger D. Richter P. Molitor

Computer Science Institute, University Halle, D-06120 Halle, Germany

Abstract

We present two approaches for the *Euclidean TSP* which compute high quality tours for large instances. Both approaches are based on pseudo backbones consisting of all common edges of good tours. The first approach starts with some pre-computed good tours. Using this approach we found record tours for seven VLSI instances. The second approach is window based and constructs from scratch very good tours of huge TSP instances, e. g., the World TSP.

Key words: Euclidean Traveling Salesman Problem, Pseudo Backbone, Problem Contraction, Iterative Approach, Window Based Approach.

1 The overall approach

Given a set of cities and the distances between each pair of them, the Traveling Salesman Problem (TSP) is the \mathcal{NP} -hard problem of finding a shortest cycle visiting each city exactly once. In this paper we consider *Euclidean TSP* whose cities are embedded either in the Euclidean plane using the Euclidean distance or a ball using the spherical grid of latitude and longitude. The *backbone* of a TSP instance consists of all edges, which are contained in *each* optimum tour of the instance, and is an important criterion for the hardness of a TSP instance. The larger the backbone of an instance, the simpler is the remaining

Email addresses: dong@informatik.uni-halle.de (C. Dong),
ernstc@informatik.uni-halle.de (C. Ernst),
jaegerg@informatik.uni-halle.de (G. Jäger),
richter@informatik.uni-halle.de (D. Richter),
molitor@informatik.uni-halle.de (P. Molitor).

¹ This work is supported by German Research Foundation (DFG) with grant number MO 645/7-3.

sub-instance. Unfortunately it is usually hard to compute the backbone of an instance. An interesting observation is that tours of an instance with good quality are likely to share many edges. We can presume that these edges are also contained in optimum tours and call them *pseudo backbone edges*. This basic observation is elaborated in detail in our approach. Assume that for a given TSP instance a set of pseudo backbone edges is computed. Our idea is to contract maximal paths of pseudo backbone edges to single edges which are kept fixed during the following process. By the contraction step, a new TSP instance with smaller size is created which can be attacked more effectively.

2 Using good starting tours for pseudo backbone computation

Let a TSP instance be given as a complete graph $G = (V, E)$ with $E = V \times V$. Our first approach undergoes the following five steps (see Fig. 1). The first step is to find a set Ω of good tours for G which are called *starting tours*. The second step is to collect the pseudo backbone edges, i. e., compute the set $B := \{e \in E; e \in \cap_{T \in \Omega} T\}$ of edges which are contained in each tour of Ω . Let V_B be the set of vertices which are endpoints of at least one edge of B . The third step is to construct all maximal paths consisting only of edges in B and contract each of these maximal paths to an edge, the endpoints of which are that of the path. We denote them by *p-edges* (path edges) and the set of all end points of the *p-edges* by V_p . The contraction step results in a new TSP instance $H = (W, F)$ with $H = (V \setminus V_B) \cup V_p$, $F = W \times W$, where the weight of the *p-edges* can be chosen arbitrarily. The fourth step is to find a good tour t' for the new TSP instance H subject to the condition that all *p-edges* must be in the tour. Finally, the fifth step is to obtain a tour t for the original TSP instance G by re-substituting the *p-edges* by the corresponding paths in the computed tour t' . The experimental results strongly demonstrate the effectivity of the approach: for seven VLSI instances with sizes 13584, 17845, 19402, 21215, 28924, 47608 and 52057 we could find better tours than the best tours known so far (see TSP homepage: <http://www.tsp.gatech.edu/>). The success of this approach strongly depends on having good starting tours generated by different methods – for the above mentioned results we used starting tours which had been constructed by different tolerance based algorithms presented in [3] (see [1] for more information on tolerances).

3 Iterative window based pseudo backbone computation

Our second approach computes tours of large Euclidean TSP instances from scratch, i. e., it does not require starting tours. In fact, computing multiple

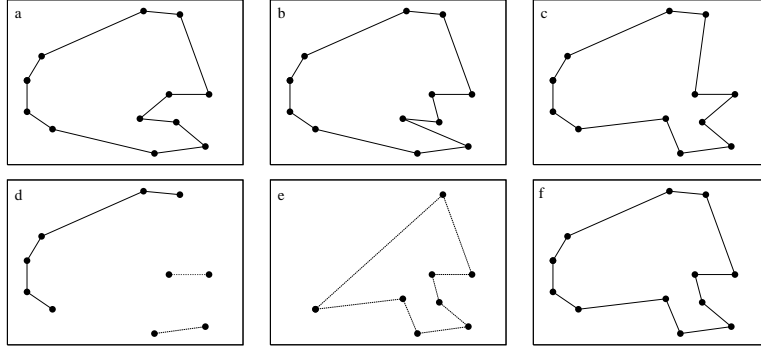


Figure 1. Illustration of the first approach. The instance has 12 points in the Euclidean plane. By the three starting tours given in (a), (b), and (c), we receive the pseudo backbone edges (d). From the maximal paths consisting only of pseudo backbone edges, only one has a length greater than 1. Only this path contributes to the size reduction. After contracting, we receive a new instance with 8 points which contains 3 p -edges (e). The three p -edges are fixed while searching tours for the new instance. In (e) an optimal tour t' for the new instance is shown. After re-substituting the p -edges by the corresponding paths, we receive a tour t for the original instance (f). For this instance, the final tour is optimal.

different good starting tours for the `World TSP` with 1,904,711 cities is hardly realizable in reasonable time. The basic idea of our window based approach consists of splitting the bounding box of the vertices of the TSP instance in non-disjoint windows by moving a window frame across the bounding box of the vertices of the TSP instance with a step size of half the width (height) of the window frame (see Fig. 2). Thus each vertex is contained in up to four windows. Each window defines a sub-instance for which a good tour is computed, e. g., by Helsgaun's LKH [2], independently of the neighboring sub-instances. Now, the approach is based on the assumption that an edge (u, w) , which is contained in the same four windows and in each of the four tours, has high probability to appear in an optimal tour of the original TSP instance – in some sense the four windows together reflect the surrounding area of (u, w) with respect to the four directions. Such edges are declared as pseudo backbone edges (see Fig. 3(a)-3(c)). After the contraction of the maximum paths of pseudo backbone edges, the approach is iterated with monotonically increasing window frame. We applied the described algorithm to the `World TSP` and required about 4.75 days for computing from scratch a tour of length 7,569,766,108 which is only at most 0.7161% greater than the length of an optimum tour. Currently, our approach is still dominated in some sense by LKH. By assigning the right values to the parameters, LKH computes a tour for the `World TSP` in an hour which is at most 0,6% greater than the length of an optimal tour, and by using stronger parameters, a tour at most 0.25% greater in two days. Note that the computation time of 1,500 seconds, Helsgaun states in [2], does not include the computation time of the starting tour (private communication, K. Helsgaun, 2009). However, note that till now we have used only default parameters for LKH without any parameter tuning. By detailed parameter tuning, the window frames of our approach can be chosen much larger which should lead to an improvement of the computed tours and running times.

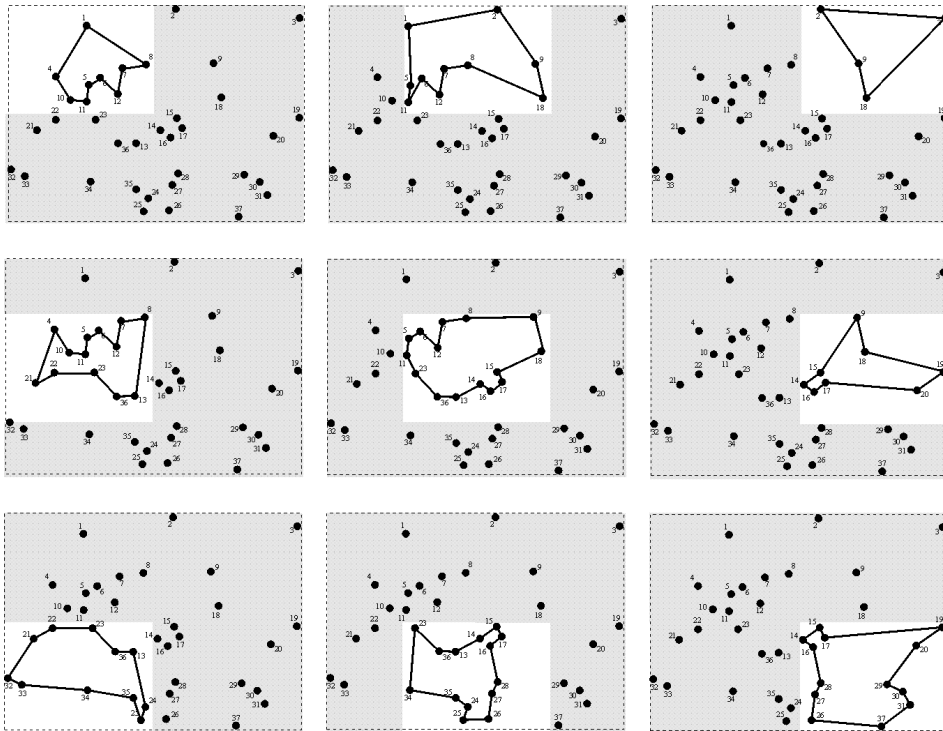
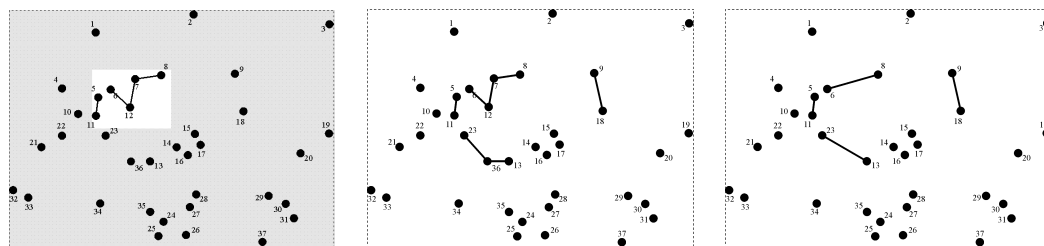


Figure 2. Illustration of the window based technique of splitting large TSP instances into sub-instances.



(a) Pseudo backbone edges found by the four top left-hand windows (b) Pseudo backbone edges found in the current iteration. (c) Constrained ETSP after contraction of the paths shown in (b).

Figure 3. Window based pseudo backbone computation and contraction.

References

- [1] B. Goldengorin, G. Jäger, and P. Molitor. Tolerances applied in combinatorial optimization. *J. Comput. Sci.* 2(9), 716-734, Science Publications, 2006.
- [2] K. Helsgaun. An Effective Implementation of K-opt Moves for the Lin-Kernighan TSP heuristic. *Writings on Computer Science* 109, Roskilde University, 2007.
- [3] D. Richter, B. Goldengorin, G. Jäger, and P. Molitor. Improving the Efficiency of Helsgaun's Lin-Kernighan Heuristic for the Symmetric TSP. *Proc. of the 4th Workshop on Combinatorial and Algorithmic Aspects of Networking (CAAN)*, Lecture Notes in Comput. Sci. 4852, 99-111, 2007.