# Improving the Efficiency of Helsgaun's Lin-Kernighan Heuristic for the Symmetric TSP

Dirk Richter[1], Boris Goldengorin[2,3,4], Gerold Jäger[5], and Paul Molitor[1]

[1] Computer Science Institute, University of Halle-Wittenberg,
D-06099 Halle (Saale), Germany
richterd@informatik.uni-halle.de, paul.molitor@informatik.uni-halle.de
[2] Faculty of Economic Sciences, University of Groningen,
9700 AV Groningen, The Netherlands
B.Goldengorin@rug.nl
[3] University of Economics and Business, Lviv Highway 51/2,
29016 Khmelnitsky, Ukraine
[4] Department of Applied Mathematics,
Khmelnitsky National University, Ukraine
[5] Department of Computer Science, Washington University
Campus Box 1045, One Brookings Drive
St. Louis, Missouri 63130-4899, USA
jaegerg@cse.wustl.edu

**Abstract.** Helsgaun has introduced and implemented the lower tolerances ($\alpha$-values) for an approximation of Held-Karp's 1-tree with the purpose to improve the Lin-Kernighan Heuristic (LKH) for the Symmetric TSP (STSP). The LKH appears to exceed the performance of all STSP heuristic algorithms proposed to date.

In this paper we improve Helsgaun's LKH based on an approximation of Zhang and Looks' backbones and an extension of double bridges further combined with implementation details by all of which we guide the search process instead of Helsgaun's $\alpha$-values. Our computational results are competitive and lead to improved solutions for some of the VLSI instances announced at the TSP homepage.

**Keywords:** Traveling Salesman Problem, Lin-Kernighan Heuristic, Tolerances, Backbones, Double Bridge Technique.

## 1 Introduction

The traveling salesman problem (TSP) is the problem of finding a Hamiltonian cycle with minimum costs of a graph. If the graph has $n$ nodes, a tour $T$ is a permutation $T = (x_1, x_2, \ldots, x_n)$ of the vector $(1, 2, \ldots, n)$ with corresponding costs $c(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n-1} c(x_i, x_{i+1}) + c(x_n, x_1)$. This paper focus on the symmetric case where all costs satisfy $c(x_i, x_j) = c(x_j, x_i)$.

Lin and Kernighan introduced a heuristic which is based on the exchange of $k$ tour edges, called $k$-swap or $k$-opt [20]. This local search algorithm still remains at the heart of the most successful approaches. In fact, Johnson and

J. Janssen and P. Prałat (Eds.): CAAN 2007, LNCS 4852, pp. 99–111, 2007.

McGeoch [17] describe the Lin-Kernighan (LK) algorithm as the world champion heuristic for the TSP from 1973 to 1989. Further, this was only conclusively superseded by chained or iterated versions of LK, originally proposed by Martin et al.[21,22]. For the TSP, multiple-run heuristics have long been the method of choice when very high-quality solutions are required. Lin and Kernighan [20] have suggested to use pseudo-random starting tours to permit repeated application of their local search procedure. Besides just taking the best of the tours that are produced, Lin and Kernighan propose to use the intersection of the edge sets of the tours as a means to guide further runs of their algorithm. Their idea is to modify the basic procedure so that it will not delete any edge that has appeared in each of the tours that have been found up to that point. They start this restricted search after a small number of tours have been found (they use between two and five tours in their tests). Variations of this idea have been explored recently by Helsgaun [16], Schilham [26], and Tamaki [28]. Considering STSP heuristics, Helsgaun's LKH [16] appears to exceed all further algorithms including the multiple runs of Chained Lin-Kernighan and some other high-end STSP heuristics introduced by Applegate et al. [1], Balas and Simonetti [2], Cook and Seymour [5], Gamboa et al. [6,7], Kahng and Reda [18], Schilham [26], Tamaki [28], and Walshaw [30].

Zhang and Looks [32] made an interpretation of a *backbone* for the STSP as an edge between two cities that appears in all optimal STSP tours. In fact they have measured edge appearance frequencies to estimate the probabilities of backbone variables since to find the backbones is not possible without solving the problem exactly. A theoretical study of backbones is started in Chrobak and Poljak [3] by proving that the intersection of edges from the optimal STSP and Minimum Spanning Tree (MST) solution has at least two common edges. Goldengorin et al. [10,11] have shown that all common edges in all optimal tours have strictly positive upper tolerances, but Libura [19] has indicated that it is $\mathcal{NP}$-hard to find out an upper tolerance for an edge in an optimal tour.

Van der Poort [24] has used the upper tolerance of an edge in MST for an approximation of the upper tolerance of the same edge in an optimal tour and Helsgaun [16] has used the lower tolerance ($\alpha$-value) for the same purpose. Goldengorin et al. [10,11] and Turkensteen et al. [29] have shown that the arcs with strictly positive upper tolerance in an optimal Assignment Problem (AP) solution are common arcs for all optimal AP solutions. Ghosh et al. [8], Goldengorin and Jäger [9], Goldengorin et al. [12,13], and Turkensteen et al. [29] have applied the largest (bottleneck) upper tolerance for an arc in an optimal solution of the relaxed AP to guide a search of either a high quality heuristic or an exact algorithm for the Asymmetric TSP. Experimentally Helsgaun has used $\alpha$-values (lower tolerances) for indicating the most likely edges in an optimal STSP solution.

We have used Helsgaun's implementation as a basis and incorporated backbone approximations and tolerances to guide the search process and $k$-swap-kicks to speed up the search.

In Section 2 we define the notion of tolerances [10,29]. The following sections discuss special aspects of TSP optimization that are suitable to enhance Helsgaun's TSP heuristic: the application of $k$-swap-kicks in Section 3, backbones in Section 4 and further implementation aspects in Section 5. In Section 6 we give experimental results which show the efficiency of the proposed methods. In particular, they allowed us to set world records for two well-known TSP instances [37]. The paper closes in Section 7 with a summary and suggestions for future work.

## 2 Tolerances

Tolerances are successfully used to guide the search process within different frameworks of heuristics for the Asymmetric [8,9,12], and Symmetric TSP [16]. A theoretical background of the tolerance based approach for solving different classes of combinatorial optimization problems is outlined in [10,11]. We distinguish between two types of tolerances: upper and lower tolerances. We introduce the concept of tolerances for an "optimal" tour having in our mind that the optimality will be further used with respect to either one of the TSP relaxations (for example, 1-Tree [16]) or a polynomially searchable neighborhood (for example, k-opt [14,15,23,25]), since finding an exact tolerance for a $\mathcal{NP}$-hard problem is also a $\mathcal{NP}$-hard problem.

Given an optimal tour $T$, we define for each edge $x \in T$ ($x \notin T$) the upper (lower) tolerance as the maximum increase $u_T(x)$ (decrease $l_T(x)$) of the edge length $c(x)$ preserving the optimality of $T$ under the assumption that the lengths of all other edges remain unchanged. Formally, for the edges $x, y$ and $\alpha \in \mathbb{R}$ let

$$c_{\alpha,x}(y) := \begin{cases} c(x) + \alpha, & \text{if } x = y \\ c(y), & \text{otherwise} \end{cases}$$

be a modification of the cost function which changes the costs for edge $x$ to $c(x) + \alpha$. Further let $\mathcal{T}_c$ be the set of all optimal tours. The tolerances with respect to an optimal tour $T$ are defined as follows:

$$u_T(x) := \sup\{\alpha \in \mathbb{R} \mid T \in \mathcal{T}_{c_{+\alpha,x}}\}, \quad \text{if } x \in T$$
$$l_T(x) := \sup\{\alpha \in \mathbb{R} \mid T \in \mathcal{T}_{c_{-\alpha,x}}\}, \quad \text{if } x \notin T$$

Let $T^+(x)$ be an optimal tour under the condition that it contains $x$, and $T^-(x)$ be an optimal tour under the condition that it does not contain $x$. Then the upper and lower tolerance of $x$ with respect to the optimal tour $T$ can be computed as follows (see [10]):

$$u_T(x) = c(T^-(x)) - c(T), \quad \text{if } x \in T \tag{1}$$

$$l_T(x) = c(T^+(x)) - c(T), \quad \text{if } x \notin T \tag{2}$$

## 3 *k*-Swap-Kicks

In Helsgaun's heuristic a greedy initial tour is constructed in each trial, where a trial is a repeated phase in which an initial tour is permanently improved by doing $k$-swaps until no more improving $k$-swaps can be found (Helsgaun considers $k \leq 5$). The resulting tours are called $k$-optimal. As the search space for $k$-swaps is restricted by a candidate system of all edges, in fact, Helsgaun's code only computes approximations of $k$-optimal tours. Constructing a new initial tour in each trial leads to the loss of $k$-optimality. Our idea is to rescue a part of $k$-optimality in the next trial instead of constructing a new initial tour. We modify the $k$-optimal tour from the last trial by one or more special $l$-swaps ($l > k$) and we choose the resulting tour as the new initial tour. In the literature for $k = 4$ this technique is known as *double bridge technique* [17,27]. Figure 1 shows an example of a double bridge move. If we would use only a simple double bridge move (a special 4-swap), the 5-swap search would end in the same local minimum as before. Thus we adopt this technique for special $l$-swaps with $l \geq 6$.

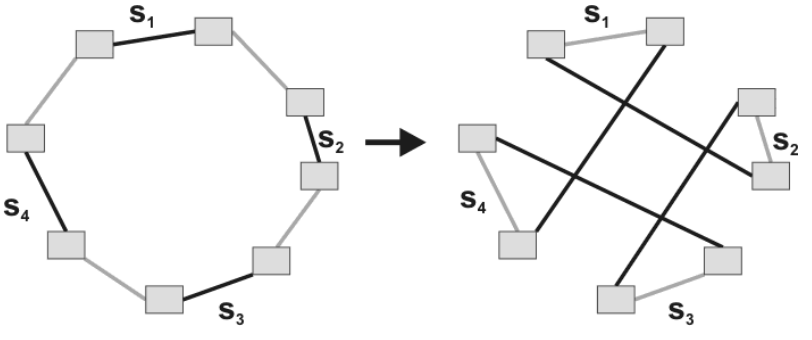


**Fig. 1.** Double bridge move

In our approach, the edges $s_1$, $s_2$, $s_3$, $s_4$ shown in Figure 1 are replaced by paths that are segments of the tour, where a *segment* is an ordered list of nodes. So a $k$-segmentation is a split of a tour $T$ into $k$ segments $s_i$, so that the concatenation in the order $s_1, s_2, \ldots, s_k$ gives the original tour $T$, i.e, $concat(s_1, s_2, \ldots, s_k) = T$. Given a $k$-segmentation $(s_1, s_2, \ldots, s_k)$ of a tour $T$, we call the $k$-swap which transforms $T$ into the tour $T' = concat(s_1, s_k, s_{k-1}, s_{k-2}, \ldots, s_2)$ a *k-swap-kick*. Clearly this is a natural extension of a double bridge.

## 4 Backbones

Helsgaun [16] uses $\alpha$-values to guide the search process of his algorithm which are lower tolerances to the minimum 1-tree. He shows that using his $\alpha$-values instead of costs leads to tours with much better quality.

In [25] we have introduced and experimented with tolerances for many problems related to the TSP and different from $\alpha$-values (e.g. relaxed assignment, assignment, 2-opt) with the purpose to improve Helsgaun's heuristic. Most of these tolerances give worse results in comparison to $\alpha$-values. In this section, we introduce our most promising approach, the *backbone* tolerance.

It might be possible that there is an edge $x$ which is contained in each optimal tour ($x \in \bigcap \mathcal{T}_c$). Edges occurring in each optimal tour are called *backbones* [4,31,32]. Identifying edges to be a backbone would therefore reduce the problem size and thus speed up a heuristic solving the TSP. Note that backbones are exactly the edges with a strictly positive upper tolerance w.r.t. an arbitrary chosen optimal tour [10].

In [4,31,32] the probability of being a backbone of an edge $x$ is approximated by the relative frequency of occurring in approximated $k$-optimal tours found during an initialization phase. In our context approximated $k$-optimal means $k$-optimal for the restricted search space, i.e. only for the edges in the candidate system.

We measure by means of this relative frequency the probability of being a backbone of an edge $x$ and call it a *backbone approximation*. In other words, the relative frequency of an edge will play the opposite role of its cost.

The main distinction between an exact and a heuristic algorithm is that an exact algorithm proves the optimality of an outputted solution on the whole set of feasible solutions and a heuristic makes a choice of the best solution among a small subset of feasible solutions. If this small subset contains an optimal solution, then the heuristic outputs an optimal solution, otherwise it outputs the best within that small subset. If we replace the optimal solution by the best solution in a small subset and treat it as an optimal one, then we are able to introduce the upper and lower tolerances w.r.t. the best solution for all edges of this small subset. If the small subset is defined for the set of all approximated tours found during an initialization phase, then we have arrived to the notions of *approximated backbone tolerances*. Using (1) and (2), these *approximated backbone tolerances* can be computed as follows:

For an edge $e$ which is contained in any best approximated tour found during the initialization phase, the *upper approximated backbone tolerance* of $e$ is defined as the difference of the optimum value of all approximated tours not containing $e$ minus the optimum value of *all* approximated tours.

For an edge $e$ which is not contained in a best approximated tour (but in at least one approximated tour), the *lower approximated backbone tolerance* of an edge $e$ is defined as the difference of the optimum value of all approximated tours containing $e$ minus the optimum value of *all* approximated tours.

Note that the approximated backbone tolerance is a measure of how likely an edge is in an optimal tour. Whereas backbone approximations use an average value over all tours, approximated backbone tolerances are dominated only by the best tours found during the initialization phase.

## 5 Implementation Aspects

Based on the ideas of $k$-swap-kicks and backbones, we have developed a new version of Helsgaun's heuristic.

In all experiments we use the same standard parameters for Helsgaun's heuristic, with two exceptions: we use 5 independent runs instead of 10 independent runs and the internal constant $maxdim = 15,000$ instead of $2,000$ (where $maxdim$ denotes the maximum dimension for which the costs of the edges are fully cached into a matrix in memory), as we have observed that increasing this internal constant considerably improves the general heuristic speed.

At the beginning of the algorithm a set of independent greedy initial tours is chosen, which are improved by one or more trials of Helsgaun's original heuristic, where a trial ends, when 5-swaps can find no further improvement. After this initialization phase we determine a new candidate system depending on backbone approximations. In this way backbone approximations are used to guide the search process instead of Helsgaun's $\alpha$-values. The decision to apply backbone approximations instead of approximated backbone tolerances is traced back to the fact that the experiments made so far show that approximated backbone tolerances give worse results than backbone approximations in average. Nevertheless, we believe that approximated backbone tolerances are the better approach, thus more sophisticated heuristics have to be found.

Furthermore, in the main phase of the algorithm an initial tour for the next trial is constructed by applying multiple $l$-swap-kicks ($l > k$) randomly to the approximated $k$-optimal tour from the last trial. Thus a local optimum can be left with rescuing a lot of $k$-optimality. Each such start with a new initial tour is called *step*. We choose – like in Helsgaun's original implementation – the number of trials as the number of nodes $n$.

We use two different implementations: one is tuned for speed, the other for tour quality. In the first implementation which we tuned for speed, at the end of the initialization phase the tour edges are sorted using a randomized quicksort to identify duplicates and to count the occurrence for computing the backbone approximations. In contrast, in the second implementation which we tuned for quality we use (double) hashing instead of quicksort as it saves memory and thus enables to handle larger problems and longer initialization phases (hashing behaved slower than quicksort in our experimental runs). Additionally in the second implementation, the independent initial tours are chosen randomly instead of greedily (this is more effort but leads to slightly better tours) and $k$-swap-kicks are used with tuned parameters, e.g. we use a better distribution function for the segments.

## 6 Experimental Results

The following experiments were executed on Intel Xeons 2.4 GHz with 1G RAM. In total we investigated about 4.5 years of running time for all these experiments. All times are given in the format "hours:minutes:seconds".

We tested the algorithms BB... T1 (the first implementation tuned for speed), BB... T2 (the second implementation tuned for quality), and LKH (the original version of Helsgaun). For example BB5P2T1 means that a backbone approximation is used after an initialization of cardinality 5% of the dimension (i.e. $\lceil 0.05 * n \rceil$ initial tours are constructed independently) and each step consists of 2 trials.

### 6.1 Comparison of Quality for the First Trials

First we compare two variants BB3P2T1 and BB5P2T1 with LKH considering tour quality, more exactly considering the following measure. As the main differences appear in the first trials, we consider only this area.

Let $P$ be the set of analyzed problems, $c^X_{j,p}(i)$ the costs of the tour found by heuristic or tolerance $X$ at Trial $i$ in Run $j$ for a problem $p \in P$. Further let $c_{best}(p)$ be the costs of the currently best known tour for problem $p \in P$ with dimension $n_p$ and $R$ the number of runs. Then we define:

$$avg.excess^X(i) = \frac{1}{|P|} \sum_{p \in P} \frac{1}{R} \sum_{j=1}^{R} \frac{c^X_{j,p}(\frac{i \cdot n_p}{100}) - c_{best}(p)}{c_{best}(p)} \tag{3}$$

As test instances we use the 33 smallest unsolved problems of the national and VLSI instances [34,37].

In Figure 2 the results are shown. We consider at the x-axis the number of trials in percentage up to 20 % of all trials.

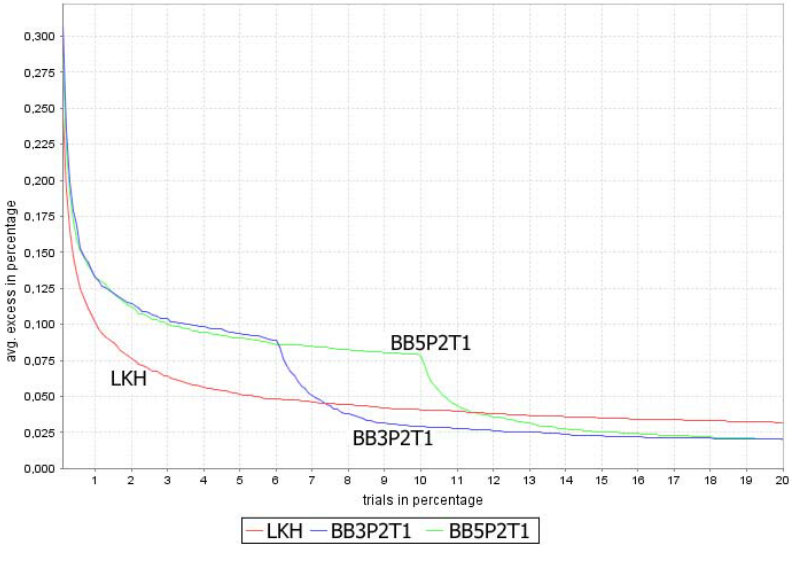


**Fig. 2.** Average quality of the national and VLSI instances for the first trials

We observe that the average difference to an optimal solution or to the best known lower bound is reduced by 21.73 % for BB3P2T1 and by 24.09 % for BB5P2T1 after *all* trials.

### 6.2 Improved Instances

During our experiments we have either improved or confirmed the same quality for many TSP instances from the TSP homepage [36]. Two of them xsc6880 and frh19289 were placed at the website [37] as currently best solutions. Note that despite the efforts of many researchers during recent three years we have not only found better tours, but also much faster in terms of normalized times [33]. In Table 1 the detailed results can be found. In the last column you find the normalized running times according to the DIMACS Implementation Challenge. The current overview of this competition can be found in [33].

**Table 1.** Results for the improved instances

| Problem | lb | Found by | Old ub | Algorithm | New ub | Time | Normalized |
|---|---|---|---|---|---|---|---|
| xsc6880 | 21,507 | Nguyen | 21,537 | BB3P1T1 | 21,535 | 1:28:47 | 6:08:52 |
| frh19289 | 55,163 | Helsgaun | 55,801 | BB5P1T1 | 55,799 | 49:02:58 | 125:03:37 |

### 6.3 Comparison of Time and Quality

For these experiments again we use the 33 smallest unsolved problems of the national and VLSI instances (because of too large times, some larger problems are only tested by the first implementation tuned for speed). Table 2 and Figure 3 show the results of these experiments. The exact values of average time and average excess can be found in the second and third column of Table 2, respectively. In Figure 3, the average computation time for 5 independent runs is plotted. Thus the more left a point is, the faster the corresponding algorithm is. Smaller excess means better tours in average (see (3)).

We observe that six parameter settings of our versions found faster *and* better tours in average than Helsgaun's version. The version BB1P3T1 is the fastest algorithm which gives nearly the same tour quality as LKH. The backbone version BB3P2T2 finds in average the best tours, but needs more time than LKH.

The $k$-swap-kicks were the main reason for the speed-up, as for each following trial less $k$-swaps are needed to find an approximated $k$-optimal tour. Also we do not need to construct a new initial tour, which additionally saves some time. The shorter the initialization phase is, the worse is the backbone approximation.

## 7 Summary and Future Research Directions

In this paper we have improved Helsgaun's version of the Lin-Kernighan Heuristic (LKH) which is the world champion heuristic for the Symmetric TSP (STSP) from 1998 to the current date applied to large instances including the World TSP

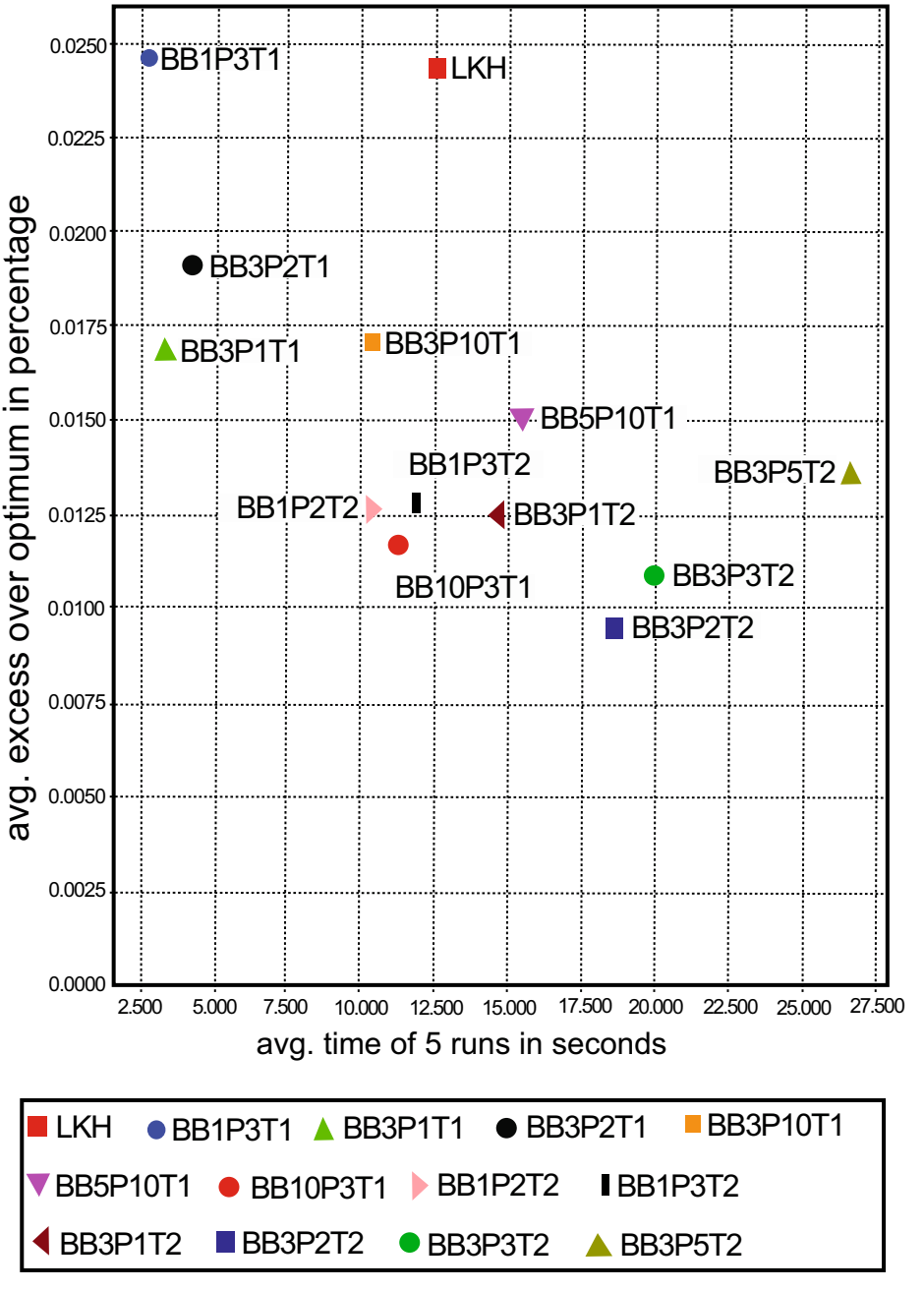


**Fig. 3.** Average time and average quality for the national and VLSI instances

**Table 2.** Average time and average quality for the national and VLSI instances

| Version | Avg. time in sec. | Avg. excess in % |
|---|---|---|
| LKH | 03:27:47 | 0.024493 |
| BB1P3T1 | **00:45:32** | 0.024716 |
| BB3P1T1 | 00:55:16 | 0.016972 |
| BB3P2T1 | 01:12:29 | 0.019171 |
| BB3P10T1 | 02:51:05 | 0.017121 |
| BB5P10T1 | 04:13:03 | 0.015167 |
| BB10P3T1 | 03:06:33 | 0.011782 |
| BB1P2T2 | 02:51:27 | 0.012631 |
| BB1P3T2 | 03:16:58 | 0.012869 |
| BB3P1T2 | 04:03:36 | 0.012502 |
| BB3P2T2 | 05:10:07 | **0.009563** |
| BB3P3T2 | 05:32:54 | 0.010899 |
| BB3P5T2 | 07:22:21 | 0.013636 |

[38] with 1,904,711 cities. Our improvements are based on a fundamental notion of a backbone edge, coined by Zhang and Looks [32]. Unfortunately to find a backbone edge has the same computational complexity as to find an optimal tour (see e.g., [3,10]). We have avoided this difficulty by using the notion of backbone approximation which can be efficiently computed, compared to the exact tolerance the computation of which for an optimal tour is also $\mathcal{NP}$-hard. We have used the backbone approximation to guide the search process instead of Helsgaun's $\alpha$-values (or exactly lower tolerances) computed for the corresponding 1-Tree relaxation of the STSP. Furthermore we have introduced *approximated backbone tolerances* which lead to slightly worse experimental results than backbone approximations. Nevertheless, we will investigate approximated backbone tolerances in more detail, as we believe that we can obtain even better results when applying this approach.

Another improvement is based on a generalization of double bridge move (a special 4-swap) which can be considered as a $k$-swap-kick for $k \geq 6$ and allows us to speed up the LKH. The above mentioned improvements are incorporated into two different implementations of LKH the first of which is tuned for speed at the initialization phase by a randomized quicksort for computing the backbone approximations. The second implementation is tuned for quality by using the double hashing which reduces the necessary memory and allows us to handle larger instances. Our computational experiments show that, for example, the first implementation leads to an essential quality improvement of outputted tours w.r.t. either the known lower bounds (for instances with unknown optimal tours) or optimal solutions by at least 21% compared to the LKH. Despite the efforts of many researchers during recent three years we have found not only better tours but also much faster in terms of normalized times.

An interesting direction of research is to apply the $k$-swap-kicks not randomly but guide them by using tolerances for some other promising data structures like stem and cycle including ejection chains for solving large scale STSP instances.

We believe that our notions of backbone approximations and approximated backbone tolerances can be applied for construction improvement type heuristics for other computationally difficult combinatorial optimization problems the first of which is the Capacitated Vehicle Routing Problem and its variations induced by distance-capacitated, time windows, pickup and delivery constraints.

Our source code is available at [35].

## Acknowledgement

The research of all authors was supported by a DFG grant MO 645/7-3, Germany. The research of the third author was additionally funded by the United States National Science Foundation grant IIS-053525.

## References

1. Applegate, D., Cook, W., Rohe, A.: Chained Lin-Kernighan for Large Traveling Salesman Problems. INFORMS J. Comput. 15(1), 82–92 (2003)
2. Balas, E., Simonetti, N.: Linear Time Dynamic Programming Algorithms for New Classes of Restricted TSPs: A Computational Study. INFORMS J. Comp. 13, 56–75 (2001)
3. Chrobak, M., Poljak, S.: On Common Edges in Optimal Solutions to the Traveling Salesman and Other Optimization Problems. Discrete Appl. Math. 20(2), 101–111 (1988)
4. Climer, S., Zhang, W.: Searching for Backbones and Fat: A Limit-Crossing Approach with Applications. In: AAAI-02, American Association for Artificial Intelligence. Proceedings of the 18th National Conference on Artificial Intelligence (2002), `www.aaai.org`
5. Cook, W., Seymour, P.: Tour Merging via Branch-Decomposition. INFORMS J. Comput. 15(3), 233–248 (2003)
6. Gamboa, D., Rego, C., Glover, F.: Data Structures and Ejection Chains for Solving Large Scale Traveling Salesman Problems. European Journal Oper. Res. 160(1), 154–171 (2005)
7. Gamboa, D., Rego, C., Glover, F.: Implementation Analysis of Efficient Heuristic Algorithms for the Traveling Salesman Problem. Comput. Oper. Res. 33(4), 1154–1172 (2006)
8. Ghosh, D., Goldengorin, B., Gutin, G., Jäger, G.: Improving the Performance of Greedy Heuristics for TSPs Using Tolerances. Communications in Dependability and Quality Management 10(1), 52–70 (2007)
9. Goldengorin, B., Jäger, G.: How to Make a Greedy Heuristic for the Asymmetric Traveling Salesman Problem Competitive. SOM (Systems, Organisations and Management) Research Report 05A11, University Groningen, The Netherlands (2005)
10. Goldengorin, B., Jäger, G., Molitor, P.: Some Basics on Tolerances. In: Cheng, S.-W., Poon, C.K. (eds.) AAIM 2006. LNCS, vol. 4041, pp. 194–206. Springer, Heidelberg (2006)
11. Goldengorin, B., Jäger, G., Molitor, P.: Tolerances Applied in Combinatorial Optimization. J. Comput. Sci. 2(9), 716–734 (2006)

12. Goldengorin, B., Jäger, G., Molitor, P.: Tolerance Based Contract-or-Patch Heuristic for the Asymmetric TSP. In: Erlebach, T. (ed.) CAAN 2006. LNCS, vol. 4235, pp. 86–97. Springer, Heidelberg (2006)
13. Goldengorin, B., Sierksma, G., Turkensteen, M.: Tolerance Based Algorithms for the ATSP. In: Hromkovič, J., Nagl, M., Westfechtel, B. (eds.) WG 2004. LNCS, vol. 3353, pp. 222–234. Springer, Heidelberg (2004)
14. Gutin, G.: Exponential Neighborhood Local Search for the Traveling Salesman Problem. Comput. Oper. Res. 26, 313–320 (1999)
15. Gutin, G., Glover, F.: Further Extension of the TSP Assign Neighborhood. Journal of Heuristics 11(5-6), 501–505 (2005)
16. Helsgaun, K.: An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. European Journal Oper. Res. 126(1), 106–130 (2000)
17. Johnson, D., McGeoch, L.: The Traveling Salesman Problem: A Case Study in Local Optimization. In: Aarts, E., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 215–310. John Wiley and Sons, Chichester (1997)
18. Kahng, A.B., Reda, S.: Match Twice and Stitch: A New TSP Tour Construction Heuristic. Oper. Res. Lett. 32(6), 499–509 (2004)
19. Libura, M.: Sensitivity Analysis for Minimum Hamiltonian Path and Traveling Salesman Problems. Discrete Appl. Math. 30, 197–211 (1991)
20. Lin, S., Kernighan, B.W.: An Effective Heuristic Algorithm for the Traveling Salesman Problem. Oper. Res. 21, 498–516 (1973)
21. Martin, O., Otto, S.W., Felten, E.W.: Large-Step Markov Chains for the Traveling Salesman Problem. Complex Systems 5(3), 299–326 (1991)
22. Martin, O., Otto, S.W., Felten, E.W.: Large-Step Markov Chains for the TSP Incorporating Local Search Heuristics. Oper. Res. Lett. 11, 219–224 (1992)
23. Orlin, J.B., Sharma, D.: Extended Neighborhood: Definition and Characterization. Math. Program., Ser. A 101(3), 537–559 (2004)
24. Van der Poort, E.S.: Aspects of Sensitivity Analysis for the Traveling Salesman Problem. PhD Thesis, Department of Econometrics and Operations Research, University of Groningen, The Netherlands (1997)
25. Richter, D.: Toleranzen in Helsgauns Lin-Kernighan-Heuristik für das TSP. Diploma Thesis, Martin-Luther-University Halle-Wittenberg, Germany (2006)
26. Schilham, R.M.F.: Commonalities in Local Search. PhD Thesis, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, The Netherlands (2001)
27. Stützle, T., Grün, A., Linke, S., Rüttger, M.: A Comparison of Nature Inspired Heuristics on the Traveling Salesman Problem. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) Parallel Problem Solving from Nature-PPSN VI. LNCS, vol. 1917, pp. 661–670. Springer, Heidelberg (2000)
28. Tamaki, H.: Alternating Cycles Contribution: A Tour Merging Strategy for the Traveling Salesman Problem. Research Report MPI-I-2003-1-007, Max-Planck-Institut für Informatik, Saarbrücken, Germany (2003)
29. Turkensteen, M., Ghosh, D., Goldengorin, B., Sierksma, G.: Tolerance-Based Branch and Bound Algorithms for the ATSP. European Journal Oper. Res., 1–14 (to appear, 2007)
30. Walshaw, C.: A Multilevel Approach to the Traveling Salesman Problem. Oper. Res. 50(5), 862–877 (2002)
31. Zhang, W.: Configuration Landscape Analysis and Backbone Guided Local Search: Part I: Satisfiability and Maximum Satisfiability. Artificial Intelligence 158(1), 1–26 (2004)

32. Zhang, W., Looks, M.: A Novel Local Search Algorithm for the Traveling Salesman Problem that Exploits Backbones. In: Kaelbling, L.P., Saffiotti, A. (eds.) IJCAI 2005. Proceedings of the 19th International Joint Conference on Artificial Intelligence, pp. 343–350 (2005)
33. DIMACS Implementation Challenge: www.research.att.com/~dsj/chtsp/
34. National Instances from the TSP Homepage: www.tsp.gatech.edu/world/summary.html
35. Source code of this paper. http://www.informatik.uni-halle.de/ti/forschung/toleranzen/quelltexte/index.en.php
36. TSP Homepage: www.tsp.gatech.edu/
37. VLSI Instances from the TSP Homepage: www.tsp.gatech.edu/vlsi/summary.html
38. World TSP from the TSP Homepage: www.tsp.gatech.edu/world/