

Improving the Efficiency of Helsgauns Lin-Kernighan Heuristic for the Symmetric TSP

Dirk Richter¹, Boris Goldengorin^{2,3}, Gerold Jäger¹, and Paul Molitor¹

¹ Computer Science Institute, University of Halle-Wittenberg,
D-06099 Halle (Saale), Germany
E-mail: richterd@informatik.uni-halle.de,
jaegerg@informatik.uni-halle.de
paul.molitor@informatik.uni-halle.de

² Faculty of Economic Sciences, University of Groningen,
9700 AV Groningen, The Netherlands
E-mail: B.Goldengorin@rug.nl

³ Department of Applied Mathematics,
Khmelnitsky National University, Ukraine

Abstract. Helsgaun has introduced and implemented the lower tolerances (α -values) for an approximation of Held-Karp's 1-tree with the purpose to improve the Lin-Kernighan's 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 the Helsgaun's LKH based on an approximation of Zhang and Looks' backbones by tolerances and an extension of double bridges further combined with implementation details by all of which we guide the search process instead of Helsgaun's α -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 $T = (x_1, x_2, \dots, x_n)$ of the vector $(1, 2, \dots, n)$ with corresponding costs $c(x_1, x_2, \dots, x_n) = \sum_{i=1}^{n-1} c(x_i, x_{i+1}) + c(x_n, x_1)$. This paper focus on symmetric problems where the distances satisfy $c(x_i, x_j) = c(x_j, x_i)$ (for the asymmetric case see for example [8, 9, 13, 10]).

Lin and Kernighan introduced a heuristic which is based on the exchange of k tour edges, called k -swap or k -opt [20].

The local search algorithm proposed by Lin and Kernighan [20] still remains at the heart of the most successful approaches. In fact, Johnson and 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 Further, this was only conclusively superseded by chained or iterated versions of LK, originally proposed by Martin et al.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 has been found up to that point. They start this restricted search after a small number of tours (they use between two and five tours in their tests) have been found. 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 an edge appearance frequencies to estimate the probabilities of backbone variables since to find the exact backbone frequencies are hard to come by 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. [12] have shown that all common edges in all optimal tours have strictly positive upper tolerances but Libura [19] has indicated that it is NP-hard to find out an upper tolerance for an edge in an optimal tour.

Van der Poort [24] has used an upper tolerance for an edge in MST for an approximation of the upper tolerance of the same edge in an optimal tour and Helsgaun [16] has used lower tolerance (α -value) for the same purpose. Goldengorin et al. [11, 12] and Turkensteen et al. [29] have shown that the strictly positive upper tolerances for the arcs in optimal Assignment Problem (AP) solution are common arcs for all optimal AP solutions, and independent on the chosen optimal AP solution. Ghosh et al. [8], Goldengorin et al. [10, 13], and Turkensteen et al. [29] have applied the largest (bottleneck) upper tolerance for an arc in an optimal solution of a relaxed Asymmetric TSP to guide a search of either a high quality heuristic or exact solutions. Experimentally Helsgaun have used α -values (lower tolerances) for indicating the most likely common edges in the STSP and MST (1-Tree) optimal solutions.

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 [11, 12, 29]. The following sections discuss special aspects of TSP optimization that are suitable to enhance Helsgaun’s TSP heuristic [16]: 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 [36]. The paper closes in Section 7 with conclusions 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, 13], and Symmetric TSP [16]. A theoretical background of tolerance based approach for solving different classes of combinatorial optimization problems in Goldengorin et al. [11, 12] is outlined. 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]), since finding an exact tolerance for a NP-hard problem is also a 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:

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

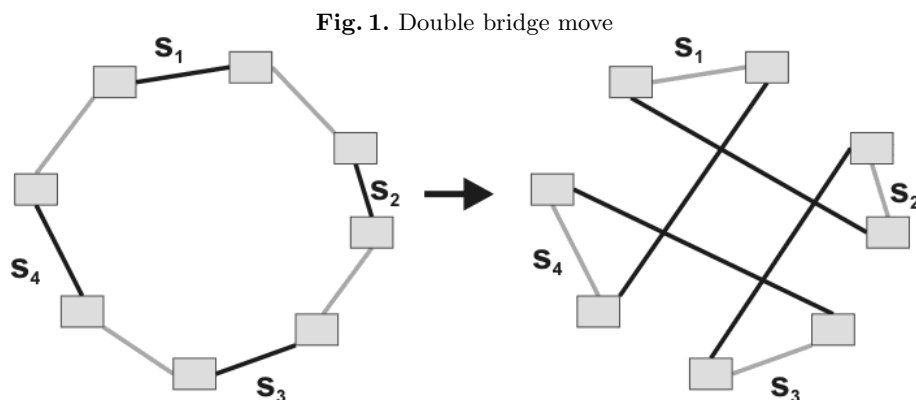
Let $T^+(x)$ be an optimal tour under the condition that it contains x , and $T^-(x)$ be an optimal tour under the condition that it does not contain x . Then the upper and lower tolerance of x with respect to the optimal tour T can be computed as follows (see [11, 12]):

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

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

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. While restricting the search space by k -swaps, 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. The k -optimal tour from the last trial is modified by one or more special $(k + 1)$ -swaps and the resulting tour is chosen as the new initial tour. In the literature for $k \leq 4$ this technique is known as *double bridge technique* [17, 27]. Figure 1 shows an example of a double



bridge move on a 3-optimal tour. If we would use only a simple double bridge move (a special 4-swap), the 5-swap search would probably end in the same local minimum as before. Thus we adopt this technique for special l -swaps with $l \geq 6$. In Figure 1 the edges s_1, s_2, s_3, s_4 can be interpreted as 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 parts (k segments s_i), so that the concatenation in the order s_1, s_2, \dots, s_k gives the original tour T , i.e., $concat(s_1, s_2, \dots, s_k) = T$. Given a k -segmentation (s_1, s_2, \dots, s_k) of a tour T , we call the k -swap which transforms T into the tour $T' = concat(s_1, s_k, s_{k-1}, s_{k-2}, \dots, s_2)$ a k -swap-kick. This is a natural extension of double bridges.

4 Backbones

Helsgaun [16] uses α -values to guide the search process of his algorithm which are lower tolerances to the minimum 1-tree. He shows that using his α -values

instead of costs output a tour with much better quality. Our experiments have shown that the additional use of upper tolerances in Helsgaun’s heuristic does not lead to a considerable extra improvement.

We have introduced and experimented with tolerances for many (see e.g. [8, 9, 13, 25], relaxed assignment and assignment, 2-opt, etc.) problems related to the TSP and different from α -values with the purpose to improve the Helsgaun’s heuristic. Most of these tolerances give worse results in comparison to α -values. In this section, we introduce our most promising approach based on 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 strictly positive upper tolerances [11, 12] w.r.t. an arbitrary chosen optimal tour.

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.

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 role of its cost in the TSP.

The main distinction between an exact and heuristic algorithms is that an exact algorithms 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 best 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, 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 tours, approximated backbone tolerances use only 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 version of Helsgaun’s heuristic.

In all experiments we use the same standard parameters for Helsgaun’s heuristic, with two exceptions: we use 5 runs instead of 10 runs and the internal constant $maxdim = 15,000$ instead of 2,000 (where $maxdim$ determines the maximal problem size 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.

A set of independent greedy initial tours is chosen, which are improved by one or more trials of Helsgaun’s heuristic (controlled by parameters). A trial ends, when 5-swaps can find no further improvement. To reach a significant speed up, k -swap-kicks with good parameters are used during and after the initialization phase. Then the next initial tour for the next trial is constructed applying multiple k -swap-kicks randomly so that a possible local optimum can be left with rescuing a lot of k -optimality. Each such approach starting with an independent initial tour is called step. Helsgaun’s adjustments of candidate sets are deactivated during the initialization, and tour edges are saved on the corresponding nodes.

After the initialization phase the backbone approximation is used to guide the search process instead of Helsgaun’s α -values. The decision to apply backbone approximation 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.

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, although 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

We included all ideas and implementation tricks from Section 5 into the original Helsgaun heuristic.

The 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 our experiments.

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 problem size, each step consists of 2 trials and $\lceil 0.05 * n \rceil$ initial tours are constructed independently, where n is the number of the nodes.

Comparison with LKH

We compare two backbone variants BB3P2T1 and BB5P2T1 with LKH considering tour quality, more exactly considering the following measure.

For each constructed initial tour a k -optimal approximation is estimated using α -values from Helsgaun. Let K be the set of analyzed problems, $c_{j,k}^X(i)$ the costs of the tour found by heuristic or tolerance X at Trial i in Run j for a problem $k \in K$. Further let $c_{best}(k)$ be the costs of the currently best known tour for problem $k \in K$ with size n_k listed in [36] and R the number of runs. Then we define:

$$avg.excess^X(i) = \frac{1}{|K|} \sum_{k \in K} \frac{1}{R} \sum_{j=1}^R \frac{c_{j,k}^X(i, \frac{n_k}{100}) - c_{best}(k)}{c_{best}(k)} \quad (3)$$

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 (note that we choose – like in Helsgaun’s original implementation – the number of trials as the number of nodes).

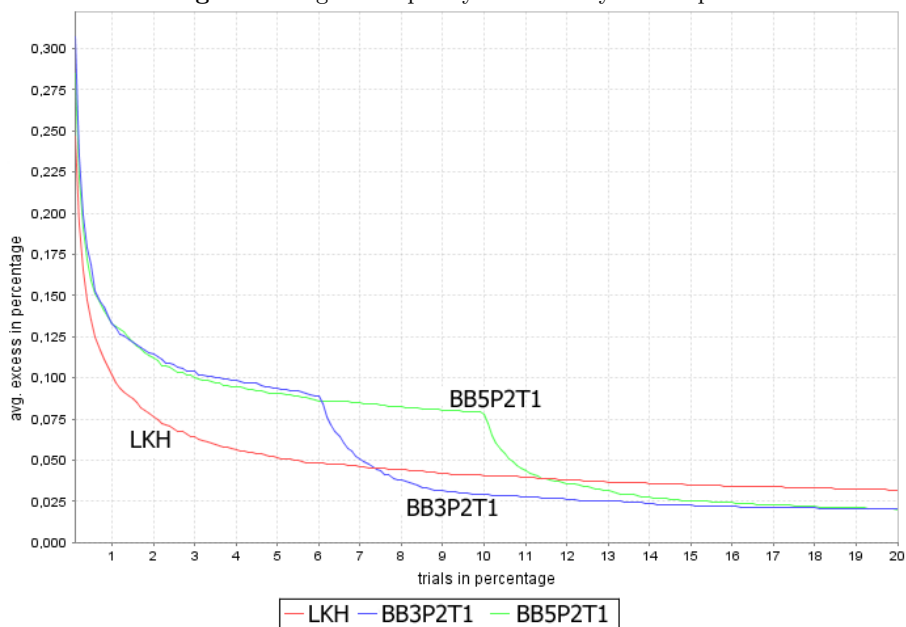
We observe that on average computing the candidate set by backbone approximations instead by α -values after the initialization phase leads to a considerable improvement, namely the average difference to an optimal solution or to the best known lower bound is reduced by 21.73 % for BB3P2T1 and by 24.09 % for BB5P2T1.”

Improved Instances

During our experiments we have either improved or confirmed the same quality for many TSP instances from the TSP homepage [34]. Two of them xsc6880 and frh19289 were placed at the website [36] as currently best solutions. Note that despite the efforts of many researches during recent three years we have found better tours 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 [33]. The current overview of this competition can be found in [36].

Fig. 2. Average tour quality of the analyzed samples



problem	found by	lb	old ub	new ub	algorithm	hh:mm:ss	normalized
xsc6880	Nguyen	21507	21537	21535	BB3P1T1	01:28:47	6:08:52
frh19289	Helsgaun	55163	55801	55799	BB5P1T1	49:02:58	125:03:37

Table 1. Tour costs for found improvements

National and VLSI Instances

We have also tested our implementation on the 33 smallest unsolved problems of the national and VLSI instances from [35, 36] (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 calculated for 5 runs is plotted. So the more left a point is, the faster the corresponding algorithm works. Smaller excess means better tours in average (see (3)).

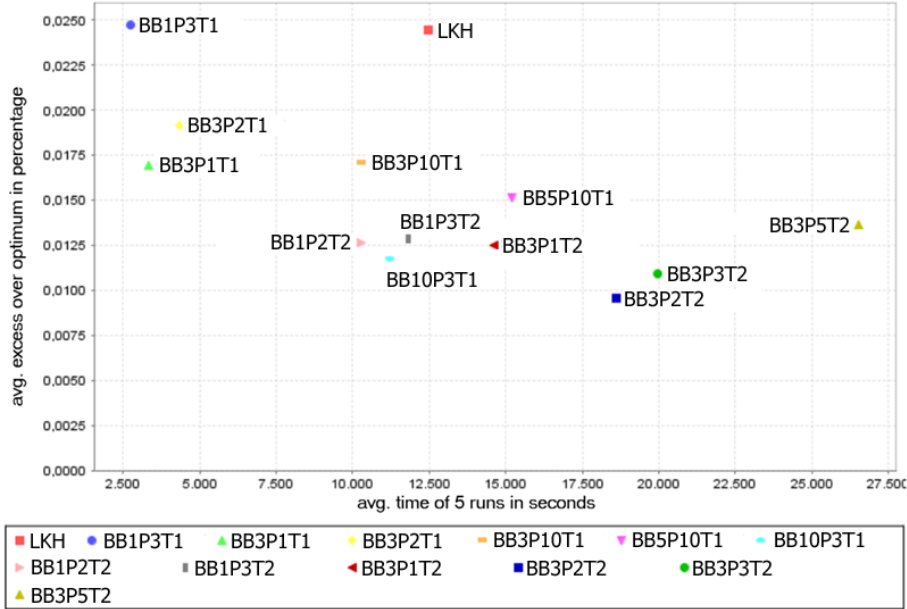
We observe that six parameter settings of our versions found faster *and* better tours in average than Helsgaun’s version.

As can be seen in Table 2, the version BB1P3T1 is the fastest algorithm which gives nearly the same tour quality as LKH. This version needs about 45 minutes in average. 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. But a longer initialization phase needs much more time and probably focus too much on some local optima. Also if the initialization phase is too long, there are not enough remaining trials for the backbone search. The backbone version BB3P2T2 finds in average the best tours, but needs more time than LKH.

variant	avg. time hh:mm:ss	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

Table 2. Results for the analyzed samples

Fig. 3. Results for the analyzed samples



7 Conclusions and Future Work

We have presented some new ideas that improve performance (runtime behavior and tour quality) for Helsgaun’s heuristic. Especially the backbone approximations lead to the best tour quality, while k -swap-kicks often only speed up Helsgaun’s heuristic. On large problems with more than 15,000 nodes, first experiments have shown that this situation probably changes. There k -swap-kicks also improve tour quality and using backbones seems to get useless. But we have not yet enough results to maintain in general that k -swap-kicks improve tour quality on large problems.

The segments for k -swap-kicks are currently chosen randomly with some restrictions on the length and distribution. Using tolerances to guide these k -swap-kicks seems to be interesting, for example we could consider the sum of the edge costs of the segments.

Furthermore, we will investigate approximated backbone tolerances in more detail, as we believe that we can obtain even better results when applying this approach than when applying backbone approximations.

Acknowledgement

The research of all authors was supported by a DFG grant SI 657/5, Germany.

References

1. D. Applegate, W. Cook, A. Rohe. *Chained Lin-Kernighan for Large Traveling Salesman Problems*. INFORMS J. Comput. 15(1), 82-92, 2003.
2. E. Balas, N. Simonetti. *Linear Time Dynamic Programming Algorithms for Some New Classes of Restricted TSPs: A Computational Study*. INFORMS J. Comp. 13, 56-75, 2001.
3. M. Chrobak, S. Poljak. *On Common Edges in Optimal Solutions to the Traveling Salesman and Other Optimization Problems*. Discrete Appl. Math. 20, 101-11, 1988.
4. S. Climer, W. Zhang: *Searching for Backbones and Fat: A Limit-Crossing Approach with Applications*. In Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-02), American Association for Artificial Intelligence (www.aaai.org), Washington University, 2002.
5. W. Cook, P. Seymour. *Tour Merging via Branch-Decomposition*. INFORMS J. Comput. 15(3), 233-248, 2003.
6. D. Gamboa, C. Rego, F. Glover. *Data Structures and Ejection Chains for Solving Large Scale Traveling Salesman Problems*. European Journal Oper. Res. 160(1), 154-71, 2005.
7. D. Gamboa, C. Rego, F. Glover. *Implementation Analysis of Efficient Heuristic Algorithms for the Traveling Salesman Problem*. Comput. Oper. Res. 33, 1154-1172, 2006.
8. D. Ghosh, B. Goldengorin, G. Gutin, G. Jäger: *Improving the Performance of Greedy Heuristics for TSPs Using Tolerances*. Communications in Dependability and Quality Management 10(1), 2007.
9. B. Goldengorin, G. Jäger: *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. B. Goldengorin, G. Sierksma, M. Turkensteen: *Tolerance Based Algorithms for the ATSP*. In: J. Hromkovic, M. Nagl and B. Westfechtel, 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG), Lecture Notes Comput. Sci. 3353, 222-234, 2004.
11. B. Goldengorin, G. Jäger, P. Molitor: *Some Basics on Tolerances*. In: S.-W. Cheng and C.K. Poon, The 2nd International Conference on Algorithmic Aspects in Information and Management (AAIM), Lecture Notes in Comput. Sci. 4041, 194-206, 2006.
12. B. Goldengorin, G. Jäger, P. Molitor: *Tolerances Applied in Combinatorial Optimization*. J. Comput. Sci. 2(9), 716-734, 2006.
13. B. Goldengorin, G. Jäger, P. Molitor: *Tolerance Based Contract-or-Patch Heuristic for the Asymmetric TSP*. In: Third Workshop on Combinatorial Algorithmic Aspects of Networking (CAAN), Lecture Notes in Comput. Sci. 4235, 86-97, 2006.
14. G. Gutin. Exponential neighborhood local search for the traveling salesman problem. Computers and Operations Research 26, 313-320, 1999.
15. G. Gutin, F. Glover. Further Extension of the TSP Assign Neighborhood. Journal of Heuristics, 11(5-6) 1572-9397, 2005.
16. K. Helsgaun: *An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic*. European Journal of Operational Research, 126(1), 106-130, 2000.
17. D. Johnson, L. McGeoch: *The Traveling Salesman Problem: A Case Study in Local Optimization*. In E.H.L. Aarts and J.K.Lenstra, Local Search in Combinatorial Optimization, 215-310. John Wiley and Sons, 1997.

18. A.B. Kahng, S. Reda. *Match Twice and Stitch: a New TSP Tour Construction Heuristic*. Oper. Res. Lett. 32, 499-509, 2004
19. M. Libura: *Sensitivity Analysis for Minimum Hamiltonian Path and Traveling Salesman Problems*. Discrete Appl. Math. 30, 197211, 1991.
20. S. Lin, W. Kernighan: *An Effective Heuristic Algorithm for the Traveling Salesman Problem*. Oper. Res. 21, 498-516, 1973.
21. O. Martin, S.W. Otto, E.W. Felten: *Large-Step Markov Chains for the Traveling Salesman Problem*. Complex Systems 5, 299-326, 1991.
22. O. Martin, S.W. Otto, E.W. Felten: *Large-Step Markov Chains for the TSP Incorporating Local Search Heuristics*. Oper. Res. Lett. 11, 219-224, 1992.
23. J. B. Orlin, D. Sharma. Extended neighborhood: Definition and characterization. Math. Program., Ser. A 101, 537-559, 2004.
24. E.S. Van der Poort. Aspects of Sensitivity Analysis for the Traveling Salesman Problem. PhD Thesis, Department of Econometrics and Operations Research, University of Groningen, The Netherlands, 1997.
25. D. Richter: *Toleranzen in Helsgauns Lin-Kernighan-Heuristik für das TSP*. Martin-Luther-University Halle-Wittenberg, Diploma Thesis, 2006.
26. R. M. F. Schilham. *Commonalities in Local Search*. PhD Thesis, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 2001.
27. T. Stützle, A. Grün, S. Linke, M. Rüttger: *A Comparison of Nature Inspired Heuristics on the Traveling Salesman Problem*. In Deb et al., editors, Proceedings of PPSN-VI, Sixth International Conference on Parallel Problem Solving from Nature, Lecture Notes in Comput. Sci. 1917, 661-670, 2000.
28. H. Tamaki: *Alternating Cycles Contribution: A Tour Merging Strategy for the Traveling Salesman Problem*. Research Report MPI-2003-1-007, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 2003.
29. M. Turkensteen, D. Ghosh, B. Goldengorin, G. Sierksma: *Tolerance-based Branch and Bound Algorithms for the ATSP*. European Journal Oper. Res., 1-14, 2007.
30. C. Walshaw: *A Multilevel Approach to the Traveling Salesman Problem*. Oper. Res. 50(5), 862-877, 2002.
31. W. Zhang: *Configuration Landscape Analysis and Backbone Guided Local Search: Part I: Satisfiability and Maximum Satisfiability*. Artificial Intelligence 158(1) (www.aaai.org), Washington University, pages 1-26, 2004.
32. W. Zhang, M. Looks: *A Novel Local Search Algorithm for the Traveling Salesman Problem that Exploits Backbones*. Department of Computer Science and Engineering of Washington University in Saint Louis, 2005.
33. DIMACS Implementation Challenge: www.research.att.com/dsj/chtsp/index.html.
34. TSP Homepage (hosted by David Applegate, Robert Bixby, Vasek Chvátal and William Cook.): www.tsp.gatech.edu/.
35. National Samples from the Research Institute of Discrete Mathematics, University Bonn (hosted by David Applegate, Robert Bixby, Vasek Chvátal and William Cook.): www.tsp.gatech.edu/world/summary.html.
36. VLSI Samples from the Research Institute of Discrete Mathematics, University Bonn (hosted by David Applegate, Robert Bixby, Vasek Chvátal and William Cook.): www.tsp.gatech.edu/vlsi/summary.html.