

# Effective Tour Searching for TSP by Contraction of Pseudo Backbone Edges

Changxing Dong, Gerold Jäger, Dirk Richter, Paul Molitor  
E-mail: {dong, jaegerg, richterd, molitor}@informatik.uni-halle.de

Computer Science Institute  
University of Halle-Wittenberg  
D-06120 Halle (Saale), Germany

**Abstract.** We introduce a reduction technique for the well-known TSP. The basic idea of the approach consists of transforming a TSP instance to another one with smaller dimension by contracting pseudo backbone edges computed in a preprocessing step, where pseudo backbone edges are edges which are likely to be in an optimal tour. A tour of the small instance can be re-transformed to a tour of the original instance. We experimentally investigated TSP benchmark instances by our reduction technique combined with the currently leading TSP heuristic of Helsgaun. The results strongly demonstrate the effectivity of this reduction technique: for the six VLSI instances *xvb13584*, *pjh17845*, *fnc19402*, *ido21215*, *boa28924*, and *flt47608* we could set world records, i.e., find better tours than the best tours known so far. The success of this approach is mainly due to the effective reduction of the problem dimension so that we can search the more important tour subspace more intensively.

## 1 Introduction

The traveling salesman problem (TSP) is a well known and intensively studied problem [1, 11, 16, 28] which plays a very important role in combinatorial optimization. It can be simply stated as follows. Given a set of cities and the distances between each pair of them, find a shortest cycle visiting each city exactly once. Formally, for a complete, undirected and weighted graph with  $n$  vertices, find a shortest Hamiltonian cycle. The dimension of the problem instance is denoted by  $n$ . If the distance between two cities does not depend on the direction, the problem is called *symmetric TSP* (STSP). In this paper we consider only the STSP. Although TSP is easy to understand, it is hard to solve, even  $\mathcal{NP}$ -hard. The hardness of a TSP instance depends on the graph structure, but also very strongly on the dimension  $n$ . The latter dependence comes from the immense search space, i.e., the number of tours is  $(n - 1)!/2$  for the STSP. Therefore it is very hard to effectively find good tours for very large problem instances. We distinguish two classes of algorithms for the STSP, namely heuristics and exact algorithms. For the exact algorithms the program package *Concorde* [1, 32], which combines techniques of linear programming and branch-and-cut, is the currently leading code. Concorde has exactly solved many benchmark instances, the largest one has even dimension 85,900 [2]. On the other hand, in the field of STSP heuristics Helsgaun's code [12, 13, 33] (LKH), which is an

effective implementation of the Lin-Kernighan heuristic [17], is one of the best packages. This code found the currently best tours for the most not exactly solved TSP benchmark instances [26, 27, 29, 30] including the famous *World TSP* instance [31].

As the search space for the STSP is huge, we can only traverse a tiny part of it in reasonable time. An interesting observation is that tours with good quality are likely to share many edges, which we call *pseudo backbone edges* [25]. In contrast, *backbone edges* are edges which are contained in each optimal tour [15, 24]. In this paper we try to improve TSP heuristics by exploiting this observation. Our main idea is to considerably reduce the dimension of the TSP instance by contraction of edges, which have high probability to appear in an optimal tour. Our approach works as follows. First, we find a number of tours with good quality. Second, we draw out from these tours the pseudo backbone edges. Third, we compute the maximal paths from the pseudo backbone edges, and contract each maximal path to an edge. In this way, we create a new TSP instance with smaller dimension. This reduced instance can be attacked more effectively and combined with each possible TSP heuristic. In our experimental investigation, we used Helsgaun’s implementation of the Lin-Kernighan-Heuristic [33]. This improved version contains many new efficient features such as general  $k$ -OPT moves, different types of partitioning, tour merging [3], iterative partial transcription [18] and backbone-guided search [23, 25].

The concept of edge fixing was already used by Lin, Kernighan [17]. Fischer and Merz [6] extended this idea to size reduction, but paid more attention to the different reduction heuristics and the enhancement to evolutionary algorithms than in our approach. Further related ideas are Cook and Seymour’s tour merging algorithm [3], which merges a given set of starting tours to get an improved tour, and compression in LKH, which is similar to contraction, but works for subproblems of partitions. All these approaches are in some sense a heuristic parallel to FPT kernelization (for an overview over the theory of parameterized complexity see [4, 7, 14, 19]), albeit TSP is not a problem of this class. One special concept which is applied during FPT kernelization is data reduction and iterative compression which reduce a hard instance to a smaller equivalent problem kernel [10]. Note that our approach is closely related to this kernelization technique.

Our experiments led to impressive results, e.g., we found record tours for six VLSI instances [30]: *xvb13584*, *pjh17845*, *fnc19402*, *ido21515*, *boa28924*, and *fht47608*, where some of the previous record tours had not been improved for several years.

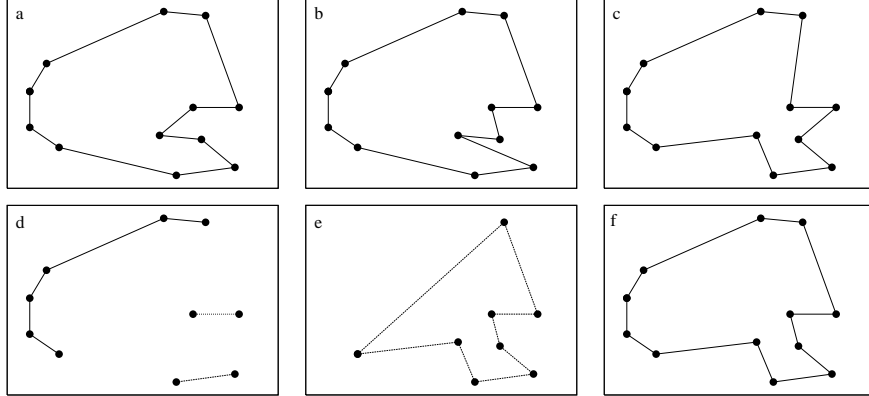
The paper is structured as follows. Our pseudo backbone contraction approach is described in Section 2 and the experimental results are presented in Section 3. Finally, we give conclusions and suggestions for future work in Section 4. Furthermore the development and an analysis of the starting tours are given in the appendix.

## 2 Pseudo backbone contraction

### 2.1 The phases of pseudo backbone contraction

Let a TSP instance be given as a complete graph  $G = (V, E)$ , where  $V$  is the set of vertices, and  $E$  the set of edges. Our approach undergoes the following phases:

1. Find a set  $\Omega$  of good tours for the given instance. We call these tours *starting tours*.



**Fig. 1.** Demonstration of the pseudo backbone edge contraction

2. Collect the pseudo backbone edges, i.e., the edges that appear in all starting tours. Formally, we have a set  $B := \{e \in E : e \in \cap_{T_i \in \Omega} T_i\}$ . Let  $V_B := \{v \in V : v \in e, e \in B\}$  and  $\bar{V}_B := V \setminus V_B$ .
3. Construct all maximal paths consisting only of pseudo backbone edges contained in  $B$ , where a path is called *maximal* with property  $\mathcal{P}$ , if it meets  $\mathcal{P}$  and cannot be extended by an edge without violating  $\mathcal{P}$ . Each maximal path is contracted to an edge, the end points of which are that of the path. We call such an edge a *cb-edge*. The cb-edge of a path with only one pseudo backbone edge is just the edge itself. We denote the set of all the end points of the cb-edges by  $V_C$ .
4. Construct a new TSP instance  $H = (W, F)$ , where  $W = \bar{V}_B \cup V_C, F = W \times W$ .
5. Find a good tour for the new TSP instance  $H$  subject to the condition that all cb-edges must be in the tour, i.e., the cb-edges are *fixed*. Note that the length of a cb-edge can be chosen arbitrarily, as it is fixed.
6. Obtain a tour for the original instance by re-transforming the tour of the new instance.

## 2.2 Illustration of the approach

Our approach is illustrated in Fig. 1 by a small TSP instance. This instance has 12 points in a two dimensional Euclidean plane, which means the distances between the points are given by the Euclidean metric. By the three starting tours in Fig. 1a, 1b, and 1c, we receive the pseudo backbone edges (Fig. 1d). From the maximal paths consisting only of pseudo backbone edges, only one has a length greater than 1. Only this path contributes to the dimension reduction. This pseudo backbone edge is contracted to a cb-edge. After contracting, we receive a new instance with 8 points (Fig. 1e). The three cb-edges are fixed while searching tours for the new instance. In Fig. 1e an optimal tour for the new instance is shown. After expanding the cb-edges in the tour of the new instance, we receive a tour for the original instance (Fig. 1f). For this instance the final tour is optimal.

### 2.3 Why contracting?

An alternative to this approach is fixing without backbone contraction. In this case all the pseudo backbone edges are forced to appear in every tour. Thus the search space is considerably cut, although the dimension of the problem is not reduced. In contrast, the main feature of contraction is the reduction of the problem dimension. This reduction has great influence to the effectivity of tour searching. The reason is that all edges incident to the vertices in  $V \setminus W$  do not appear in the new instance, whilst by only fixing many of them should be traversed – without any chance to find better tours. Note that also the time used for the edge length computations becomes smaller by the reduction of the problem dimension.

### 2.4 Solving the reduced TSP-instance

Local search is an improvement heuristic, which means it steadily improves the best tour found. Certainly we need an initial tour constructed by other heuristics, e.g., the nearest neighbor heuristic or a random heuristic. Local search transforms a tour to another one by exchanging  $k$  tour edges with  $k$  non-tour edges. This is called a  $k$ -OPT move. To improve a tour, the sum of the length of the  $k$  tour edges must be larger than the sum of the length of the  $k$  non-tour edges. A tour which cannot be improved by  $k$ -OPT moves for  $k \leq r$  is called  $r$ -optimal. Note that an  $n$ -optimal tour is optimal. As finding an  $r$ -optimal tour has complexity  $\mathcal{O}(n^r)$ , we are forced to constrain the search space such that we obtain good tours in tolerable time. LKH uses several parameters for this purpose. The most important ones beside the value of  $r$  are the maximal number  $s$  of edges incident to each vertex to be considered while searching tours and the number  $t$  of independent runs. In this context we define a tour as *approximated  $r$ -optimal*, if it is  $r$ -optimal for the given maximal number  $s$  of edges incident to each vertex. The larger  $r$ ,  $s$ , and  $t$  are, the larger is the search space. With large search space we can find better tours against cost of time. The parameter values that make the search space large are called strong parameter values, otherwise weak.

In our approach, LKH was used for searching both the starting tours of the original instances and tours for the reduced instances. To collect the starting tours, we could not afford strong parameter values because of the time limit. Therefore, the search for the starting tours was associated with relative weak parameter values and the search for the reduced instances with stronger ones. More exactly, for the reduced instances we used  $r = 8, 10$ ,  $s = 5, 6$  and  $t = 10, 20$ , while the standard parameter values are  $r = 5$ ,  $s = 5$ , and  $t = 10$ . It is just the reduction by contraction that makes the choice of strong parameter values possible. Note that LKH with standard parameter values finds optimal solutions frequently for small instances, e.g., most instances from TSPLIB. Since the reduced instances in this work have dimensions not larger than 18,242 and stronger searching parameter values are applied to them, we have not tried to solve them with the program package Concorde [32]. In addition, the reduced instances have fixed edges to be treated specifically which is not implemented in Concorde.

**Table 1.** TSP instances investigated and the best tour lengths found before and by us

	Instance	Src	Best	Ours		Instance	Src	Best	Ours
1	dsj1000	1	18659688	=	27	ei8246	2	206171	=
2	pr1002	1	259045	=	28	dga9698	3	27724	=
3	u1060	1	224094	=	29	kz9976	2	1061881	=
4	pcb1173	1	56892	=	30	xmc10150	3	28387	=
5	rl1304	1	252948	=	31	fi10639	2	520527	=
6	rl1323	1	270199	=	<b>32</b>	<b>xvb13584</b>	<b>3</b>	<b>37084</b>	<b>37083</b>
7	nrw1379	1	56638	=	33	brd14051	1	469385	469392
8	fi1400	1	20127	20188	34	xrb14233	3	45462	45464
9	u1432	1	152970	=	35	xia16928	3	52850	=
10	u1817	1	57201	=	<b>36</b>	<b>pjh17845</b>	<b>3</b>	<b>48094</b>	<b>48093</b>
11	rl1889	1	316536	=	37	frh19289	3	55798	=
12	u2152	1	64253	=	<b>38</b>	<b>fnc19402</b>	<b>3</b>	<b>59288</b>	<b>59287</b>
13	xqc2175	3	6830	=	<b>39</b>	<b>ido21215</b>	<b>3</b>	<b>63519</b>	<b>63518</b>
14	pr2392	1	378032	=	40	fma21553	3	66527	=
15	pcb3038	1	137694	=	41	lsb22777	3	60977	60983
16	pia3056	3	8258	=	42	xrh24104	3	69294	=
17	xqc3891	3	11995	=	43	irx28268	3	72607	=
18	bgb4355	3	12723	=	44	icx28698	3	78090	78095
19	rl5915	1	565530	=	<b>45</b>	<b>boa28924</b>	<b>3</b>	<b>79624</b>	<b>79623</b>
20	rl5934	1	556045	=	46	pbh30440	3	88313	88314
21	tz6117	2	394718	=	47	xib32892	3	96767	96780
22	xsc6880	3	21535	=	48	fry33203	3	97240	97242
23	bnd7168	3	21834	=	49	ics39603	3	106821	106826
24	lap7454	3	19535	=	<b>50</b>	<b>fht47608</b>	<b>3</b>	<b>125124</b>	<b>125119</b>
25	ym7663	2	238314	=	51	fna52057	3	147802	147818
26	ida8197	3	22338	=					

### 2.5 Starting tours

As starting tours we used tours received by previous experiments with many different variants of LKH. The main distinctions between these variants are that the candidate edges are chosen by different criteria, which are mostly tolerances (for an overview over the theory of tolerances see [8, 9]). These variants and an analysis of the starting tours used in our experiments, especially the ones which led to our world records, are given in the appendix. Note that the number of starting tours we used is different for each instance and depends on the size of the TSP instance.

## 3 Experimental results

We ran the programs on several computers to attack different instances simultaneously, where for solving the reduced instances we used LKH-2.0 and for computing the starting tours the older version LKH-1.3. The time limit for the reduced instances was set to two weeks. Mostly, we used the default parameter values of LKH, whereas we varied only the already mentioned parameters  $r, s, t$ . The main goal of

this work is to find tours as good as possible, therefore we pay less attention to the running time. Furthermore the running times strongly depend on the instances, i.e., smaller dimension does not necessarily mean less time. Unfortunately our running times cannot be compared with those of previous experiments listed at [28], as in most cases no times were given there. To get a feeling for the running times we also applied LKH to several large original TSP instances with the same parameter value  $r$  as applied to the reduced instances. After more than one month we could not find any better tours for the original instances.

All experimental information such as starting tours, parameter values of LKH, running times and machines are available at [34].

### 3.1 Investigated instances

All the TSP instances investigated in this work are shown in Table 1, where their dimensions can be seen in their names. These instances come from three different areas: TSPLIB instances [20, 27] (**Src 1**), national instances [29] (**Src 2**) and VLSI instances [30] (**Src 3**). In column **Best** the length of the best tour found so far is given for each instance. The length of our best tour can be seen in column **Ours**. If for an instance we found a tour with the same length as that in column **Best**, an equal sign “=” is used in column **Ours**. The six VLSI instances *xvb13584*, *pjh17845*, *fnc19402*, *ido21215*, *boa28924*, and *fmt47608*, for which we found new best tours, are highlighted in the table.

### 3.2 The effectivity of the pseudo backbone contraction

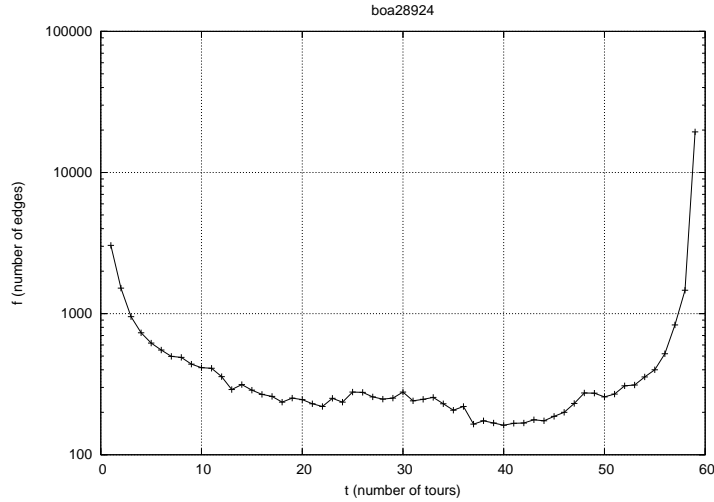
From the column **Ours** of Table 1 we observe the effectivity of the pseudo backbone contraction approach. For all but two (*fl1400*, *brd14051*) of the TSPLIB and national instances (**Src**=1, 2), we found tours that are equally good as that of the best known ones, most of them are optimally solved. For the VLSI instances we have all three cases: we found better, equally good or worse tours in comparison with the best tours known before this work. Actually for the instance *fmt47608* we found three tours that are better than the previously best known tour. The reasons that we could not find better tours for some instances can be described as follows. First, we gave a time limit of two weeks to the calculation. Second, we used mostly standard parameter values for all the instances, i.e., we did not do any tuning work except for parameters  $r$ ,  $s$ , and  $t$ . The third reason, which is the most important one, is that some of the pseudo backbone edges may be no real backbones, since the starting tours were found also by local search. And finally, the best tours of some instances may be optimal and it is not possible to improve them. Note that even for the case that we found only worse tours, the length differences between our best tours and the best ones known so far are usually very small. In summary, our approach based on pseudo backbone edge contraction gives satisfiable, in some cases excellent results.

### 3.3 Discussion

Now let’s consider some details about our approach. In Table 2 we give some data about the reduction for the investigated instances. The column **BEP**(%) gives the

**Table 2.** Reduction data of the investigated TSP instances

	Instance	BEP(%)	#Tours	NewD	Impr	Grade(%)
1	dsj1000	95.40	35	88	–	53.41
2	pr1002	98.20	35	34	–	55.88
3	u1060	90.09	35	143	–	74.13
4	pcb1173	94.63	35	118	–	54.24
5	rl1304	94.33	35	136	–	55.15
6	rl1323	95.16	35	120	–	54.17
7	nrv1379	88.18	35	301	–	54.49
8	fl1400	32.50	35	1002	-24	94.41
9	u1432	52.51	35	799	–	85.23
10	u1817	73.75	35	730	–	65.48
11	rl1889	92.80	35	245	–	55.92
12	u2152	72.96	35	826	–	70.58
13	xqc2175	79.91	66	733	0	59.75
14	pr2392	95.40	35	212	–	52.36
15	pcb3038	86.44	35	754	–	54.77
16	pia3056	75.26	66	1307	0	57.92
17	xqe3891	77.87	66	1447	0	59.57
18	bgb4355	73.69	66	1917	0	59.83
19	rl5915	91.46	35	905	14	55.91
20	rl5934	91.94	35	882	43	54.31
21	tz6117	77.90	66	2185	11	61.92
22	xsc6880	75.74	66	2850	0	58.60
23	bnd7168	70.69	64	3482	0	60.37
24	lap7454	78.84	65	2715	0	58.12
25	ym7663	85.03	66	2029	3	56.58
26	ida8197	77.50	66	3132	0	58.91
27	ei8246	84.00	67	2363	0	55.86
28	dga9698	77.41	67	3694	0	59.34
29	kz9976	84.13	66	2895	4	54.72
30	xmc10150	75.67	66	4146	2	59.58
31	fl10639	78.13	35	4099	21	56.79
<b>32</b>	<b>xvb13584</b>	<b>65.89</b>	<b>100</b>	<b>7577</b>	<b>1</b>	<b>61.17</b>
33	brd14051	76.63	35	5719	8	57.44
34	xrb14233	77.17	15	5637	2	57.65
35	xia16928	65.50	57	9392	3	62.20
<b>36</b>	<b>pjh17845</b>	<b>70.71</b>	<b>39</b>	<b>8887</b>	<b>1</b>	<b>58.83</b>
37	frh19289	82.91	8	5877	1	56.12
<b>38</b>	<b>fnc19402</b>	<b>69.79</b>	<b>59</b>	<b>9754</b>	<b>8</b>	<b>60.11</b>
<b>39</b>	<b>ido21215</b>	<b>64.65</b>	<b>58</b>	<b>12233</b>	<b>9</b>	<b>61.31</b>
40	fma21553	69.09	59	10965	11	60.77
41	lsb22777	80.95	6	7717	4	56.24
42	xrh24104	66.63	60	13073	6	61.54
43	irx28268	74.15	23	12398	13	58.94
44	icx28698	67.42	58	15200	11	61.52
<b>45</b>	<b>boa28924</b>	<b>67.13</b>	<b>59</b>	<b>15518</b>	<b>5</b>	<b>61.28</b>
46	pbh30440	81.46	9	10009	6	56.40
47	xib32892	65.55	54	18242	8	62.13
48	fry33203	79.66	8	11817	8	57.17
49	ics39603	84.26	3	11434	79	54.54
<b>50</b>	<b>fht47608</b>	<b>83.04</b>	<b>8</b>	<b>14568</b>	<b>30</b>	<b>55.43</b>
51	fna52057	82.00	5	16620	30	56.40



**Fig. 2.** Frequency distribution of starting tour edges for the TSP instance *boa28924*

percentage of pseudo backbone edges with respect to the dimension of the instance. The number of the starting tours is given in column **#Tours**. After the reduction we have an instance with smaller dimension, where the new dimension can be seen in column **NewD**. The column **Impr** gives the improvement by the reduction which is the length difference between the best starting tour and the best tour we found by pseudo backbone contraction. If the best tour in the set of starting tours is optimal, then we certainly could not improve it at all. In this case, a “-” symbol is shown in this column, if our approach finds also an optimal tour. Note that there is only one instance, for which we found only worse tours (*fl1400*). The last column **Grade** will be explained later.

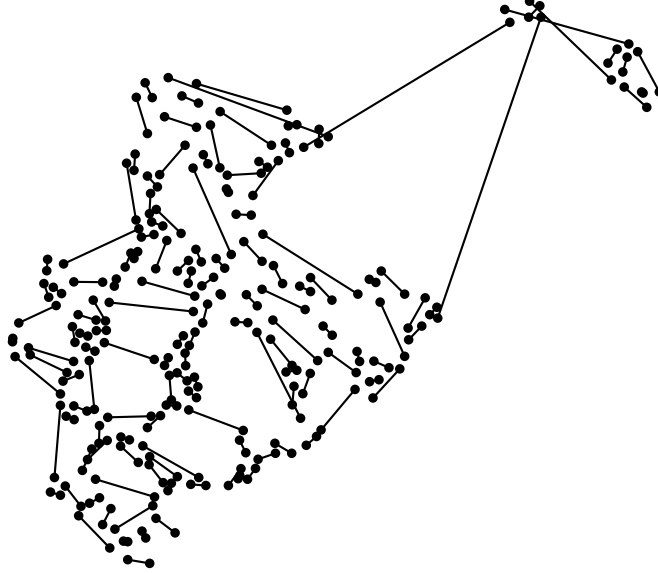
We analyzed the distribution of all edges in the starting tours for every instance. Some of these edges appear in all the starting tours, therefore they are just the pseudo backbone edges. Some of them may appear in only one starting tour. These edges have the smallest probability to keep in an optimal tour. Other edges appear between these two extreme cases. Fig. 2 shows the frequency distribution of all tour edges for the instance *boa28924*. From this figure we observe that about 200 edges appear in exactly 40 (different) of the 59 starting tours and about 20,000 edges are pseudo backbone edges, i.e., appear in each of the 59 starting tours. Note that the  $y$ -axis is in logarithmic scale. Interestingly, the run of this curve is typical for most of the investigated instances. From this curve it is easy to understand the high BEP values. This curve also shows a large frequency difference between that of the pseudo backbone edges ( $t = 59$  in the figure) and that of the almost pseudo backbone edges for  $t = 58$ . As in most instances BEP is large it suffices to reduce the original instances by contracting the pseudo backbone edges, whereas almost pseudo backbone edges have not to be considered. Fig. 3 shows the pseudo backbone edges of the instance *nrv1379*, for which we found an optimal tour. From this figure, we observe



**Fig. 3.** Tour segments for the TSP instance *nrv1379* showing only the backbone edges

that there are very long tour segments which can be contracted to corresponding cb-edges. This means that all vertices in the middle of the segments are not included in the reduced instance. Note that for this instance all these segments are contained in an optimal tour. Fig. 4 shows all cb-edges from Fig. 3. It can be seen that the original instance can be strongly reduced. The new instance obtained after the contraction of the pseudo backbone edges has a dimension of 301, which is much smaller than the original dimension 1379. Because of the smaller dimension, we can search the more “important” and “difficult” areas more intensively by choosing stronger parameters.

The number of starting tours ranges from 3 to 100 (**#Tours**). For the just discussed instance *nrv1379* we have 35 starting tours. It is interesting to point out that the number of starting tours is not the most important factor. For example, for the instance *ft47608*, for which we could find three new best tours in two weeks, we have only 8 starting tours. Instead, the quality of the starting tours and also the independence among the tours play an important role (see appendix). This can also be seen from the contrast of the BEP values between *ft47608* and *ei8246*. The



**Fig. 4.** Contracted pseudo backbone edges (cb-edges) of TSP instance *nrv1379*

former has slightly smaller BEP than the latter, although the latter has much more starting tours.

The length distribution of the tour segments of the pseudo backbone edges determines the new dimension of the reduced instance. Let  $b$  be the number of pseudo backbone edges and  $d$  the new dimension, then for  $b > n/2$ , which is the case for all but one investigated instances, we have:  $n - (b - 1) \leq d \leq 2(n - b) < n$ . The above equation follows from the following reasoning. If all the pseudo backbone edges lie in only one path, then by contracting this path to a cb-edge, we have  $b - 1$  vertices fewer in the new instance than in the original instance. On the other hand, we have exactly  $n - b$  non-pseudo backbone edges for constructing a tour from the  $b$  pseudo backbone edges. We have at most  $2(n - b)$  vertices in the reduced instance, if all these non-backbone edges are disjoint. At this point, we introduce the grade of reduction as  $\gamma = (n - b + 1)/d$  and calculate the grade of the contraction for each instance. They are listed in the last column **Grade** of Table 2. The larger this value is, the longer are the average paths consisting of only pseudo backbone edges. For most of the instances the grade of contraction has a value between 50% and 60%, but we also see a few large reduction grades. The instance *fl1400* has even a reduction grade of 94%, which indicates few but very long pseudo backbone edge paths for this instance.

## 4 Conclusions and current work

Our approach of effectively finding good tours by contracting pseudo backbone edges is justified by the excellent results. For all but one instances we could find improved tours with respect to the starting tours. Especially we found better tours than the best ones known so far for six VLSI instances with dimensions from 13,584 to 47,608, where some of the previous record tours had not been improved for several years.

A natural generalization of our idea would be to fix also edges, which do not appear in *all* starting tours, but in almost all of them. By this idea, it would be no problem, if among the starting tours also a bad one would appear.

Our current work concentrates on enhancing the central idea of our approach presented in this paper in order to attack even larger TSP instances with more than 100,000 vertices. In the present work we use the contraction of pseudo backbone edges only once, but for larger TSP instances the approach has to be extended. In [5] we suggest to apply the contraction idea in a iterative manner. This iterative approach which doesn't require starting tours is a dynamic and automatic process, i.e., the number of the contraction steps is dynamically determined during the run of the program depending on the hardness of the instance. In each iteration, the pseudo backbone edges are computed by a window based technique in which the TSP instance is tiled in non-disjoint sub-instances. We hope that a further development of this approach makes it possible in (near) future to successfully attack the *World TSP* instance or further large TSP instances.

## References

1. D.L Applegate, R.E. Bixby, V. Chvátal, W.J. Cook: *The Traveling Salesman Problem. A Computational Study*. Princeton University Press, 2006.
2. D.L Applegate, R.E. Bixby, V. Chvátal, W.J. Cook, D. Espinoza, M. Goycoolea, K. Helsgaun: *Certification of an optimal Tour through 85,900 cities*. Oper. Res. Lett. **37**(1), 11-15, 2009.
3. W. Cook, P. Seymour: *Tour Merging via Branch-Decomposition*. INFORMS J. Comput. **15**(3), 233-248, 2003.
4. R.G. Downey, M.R. Fellows: *Parameterized Complexity*. Springer, New York, 1999.
5. C. Ernst, C. Dong, G. Jäger, P. Molitor, D. Richter: *Finding Good Tours for Huge Euclidean TSP Instances by Iterative Backbone Contraction*. Submitted for Publication, 2009.
6. T. Fischer, P. Merz: *Reducing the Size of Traveling Salesman Problem Instances by Fixing Edges*. 7th European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP). Lecture Notes in Comput. Sci. **4446**, 72-83, 2007.
7. J. Flum, M. Grohe: *Parameterized Complexity Theory*. Springer, Berlin, 2006.
8. B. Goldengorin, G. Jäger, P. Molitor: *Some Basics on Tolerances*. In S.-W. Cheng and C.K. Poon (Eds.): 2nd International Conference on Algorithmic Aspects in Information and Management (AAIM). Lecture Notes in Comput. Sci. **4041**, 194-206, 2006.
9. B. Goldengorin, G. Jäger, P. Molitor: *Tolerances Applied in Combinatorial Optimization*. J. Comput. Sci. **2**(9), 716-734, 2006.
10. J. Guo, R. Niedermeier: *Invitation to data reduction and problem kernelization*. SIGACT News **38**(1), 31-45, 2007.
11. G. Gutin, A.P. Punnen (Eds.): *The Traveling Salesman Problem and Its Variations*. Kluwer, Dordrecht, 2002.

12. K. Helsgaun: *An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic*. European Journal Oper. Res. **126**(1), 106-130, 2000.
13. K. Helsgaun: *An Effective Implementation of K-opt Moves for the Lin-Kernighan TSP Heuristic*. Writings on Computer Science **109**, 2007.
14. F. Hüffner: *Algorithms and Experiments for Parameterized Approaches to Hard Graph Problems*. PhD Thesis, Friedrich-Schiller-University Jena, Germany, 2007.
15. P. Kilby, J.K. Slaney, T. Walsh: *The Backbone of the Travelling Salesperson*. L.P. Kaelbling, A. Saffiotti (Eds.), Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), 175-180, 2005.
16. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (Eds.): *The Traveling Salesman Problem - A Guided Tour of Combinatorial Optimization*. John Wiley & Sons, Chichester, 1985.
17. S. Lin, B.W. Kernighan: *An Effective Heuristic Algorithm for the Traveling Salesman Problem*. Oper. Res. **21**, 498-516, 1973.
18. A. Möbius, B. Freisleben, P. Merz, M. Schreiber: *Combinatorial Optimization by Iterative Partial Transcription*. Phys. Rev. E, **59**(4), 4667-4674, 1999.
19. R. Niedermeier: *Invitation to Fixed-Parameter Tractability*. Oxford University Press, 2006.
20. G. Reinelt. TSPLIB – a Traveling Salesman Problem Library. ORSA J. Comput. **3**, 376-384, 1991.
21. C.C. Ribeiro, R.F. Toso: *Experimental Analysis of Algorithms for Updating Minimum Spanning Trees on Graphs Subject to Changes on Edge Weights*. 6th Workshop on Experimental Algorithms (WEA). Lecture Notes in Comput. Sci. **4525**, 393-405, 2007.
22. D. Richter: *Toleranzen in Helsgauns Lin-Kernighan-Heuristik für das TSP*. Diploma Thesis, Martin-Luther-University Halle-Wittenberg, Germany, 2006.
23. D. Richter, B. Goldengorin, G. Jäger, P. Molitor: *Improving the Efficiency of Helsgaun's Lin-Kernighan Heuristic for the Symmetric TSP*. In J. Janssen and P. Pralat (Eds.): 4th Workshop on Combinatorial and Algorithmic Aspects of Networking (CAAN). Lecture Notes in Comput. Sci. **4852**, 99-111, 2007.
24. J.K. Slaney, T. Walsh: *The Backbones in Optimization and Approximation*. B. Nebel (Ed.), Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), 254-259, 2001.
25. W. Zhang, M. Looks: *A Novel Local Search Algorithm for the Traveling Salesman Problem that Exploits Backbones*. L.P. Kaelbling, A. Saffiotti (Eds.), Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), 343-350, 2005.
26. DIMACS Implementation Challenge: <http://www.research.att.com/~dsj/chtsp/>.
27. TSPLIB: <http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html>.
28. TSP Homepage: <http://www.tsp.gatech.edu/>.
29. National Instances from the TSP Homepage: <http://www.tsp.gatech.edu/world/summary.html>.
30. VLSI Instances from the TSP Homepage: <http://www.tsp.gatech.edu/vlsi/summary.html>.
31. World TSP from the TSP Homepage: <http://www.tsp.gatech.edu/world/>.
32. Source Code of [1] (Concorde): <http://www.tsp.gatech.edu/concorde/index.html>.
33. Source Code of [12] (LKH): <http://www.akira.ruc.dk/~keld/research/LKH/>.
34. Additional Information about Experiments of this Paper: <http://www.informatik.uni-halle.de/ti/forschung/toleranzen/kantenkontraktion>.

## Appendix

### 1 LKH-Variants leading to starting tours

In the following we describe the LKH variants, which had led to the starting tours used in our experiments in Section 3.

1. **LKH:** The original LKH implementation.
2. **UT:** [22] Helsgaun uses  $\alpha$ -values to guide the search process of his algorithm which are lower tolerances to a minimum 1-tree. They are defined as follows. For each edge *not in* the minimum 1-tree  $S$  define the *lower tolerance* as the maximal value, by which the length of  $e$  can be reduced, such that the optimality of  $S$  is preserved. For each edge *in* the minimum 1-tree, Helsgaun defines the  $\alpha$ -value as 0. Although having the same  $\alpha$ -value 0, these edges have different importance for the search process. A natural idea is to distinguish them by their upper tolerances which are defined as follows. For each edge *in* the minimum 1-tree  $S$  define the *upper tolerance* as the maximal value, by which the length of  $e$  can be increased, such that the optimality of  $S$  is preserved. As upper tolerances are hard to compute [21], we have only implemented an approximation of upper tolerances, which computationally has the same complexity as the computation of the lower tolerances.
3. **SK:** [23] The candidate system of LKH is not changed, but in the main phase of the algorithm an initial tour for the next trial is constructed by applying multiple  $l$ -swap-kicks ( $l > r$ ) randomly to the approximated  $r$ -optimal tour from the last trial.
4. **BB:** [23] Like SK, this variant uses  $l$ -swap-kicks, but the candidate system of LKH consists of pseudo backbones of an initialization phase. More exactly, for each edge in tours of this phase define the *backbone approximation* value as the relative frequency of appearing in approximated  $r$ -optimal tours. Edges with large value are good ones, in contrast to the  $\alpha$ -value, i.e., we have to multiply the value by  $-1$ .
5. **BBT:** [23] Like BB, this variant uses  $l$ -swap-kicks, where the candidate system of LKH consists of pseudo backbones of an initialization phase, but instead of the backbone approximation we use backbone tolerances which are defined 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 smallest value of all approximated tours not containing  $e$  minus the smallest 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 smallest value of all approximated tours containing  $e$  minus the smallest value of all approximated tours. Both tolerances can be compared by multiplying all upper tolerances by  $-1$ .
6. **2-V:** [22] Again the candidate system of LKH is changed by a new tolerance value for edges. The idea behind this tolerance is that an edge  $e = (u, v)$  should be a good one, if the length of most of the edges incident to  $u$  or  $v$  are larger than the length of  $e$ . In particular, consider the order of the length of  $e$  in comparison

with all edges which are also incident to the vertex  $u$ . If  $e$  belongs to the two smallest edges incident to  $u$ , it receives a non-positive first value, namely the negative difference to the third smallest edge incident to  $u$ . If  $e$  does not belong to the two smallest edges incident to  $u$ , it receives a non-negative first value, namely the difference to the second smallest edge incident to  $u$ . The second value is computed analogously for the edge  $e$  and the incident vertex  $v$ , and both values are added leading to the tolerance value of  $e$ .

7. **AP:** [22] Again the candidate system of LKH is changed by a new tolerance value. For each edge  $(u, v)$  search for a different vertex  $w$  such that  $c(u, w) + c(w, v)$  is minimum, i.e., we look for the cheapest alternative path from  $u$  to  $v$  (note that for Euclidean instances it holds  $c(u, v) \leq c(u, w) + c(w, v)$  because of the triangle inequality). Then we use  $c(u, w) + c(w, v) - c(u, v)$  as the tolerance value of  $(u, v)$ , where again we have to multiply the value by  $-1$ .

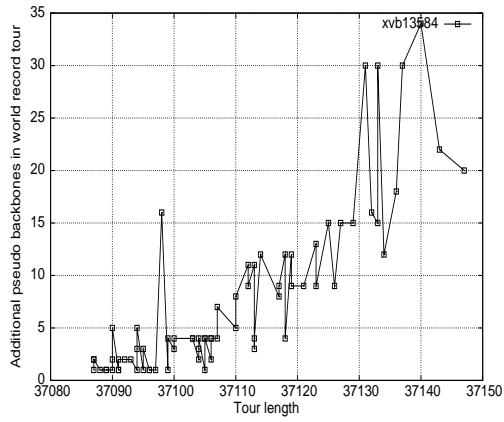
## 2 Analysis of starting tours

It is essential for our approach to use good starting tours, as in general, for each set of starting tours we receive another set of pseudo backbones and thus another reduced TSP instance. Using only one bad starting tour among many good ones would lead to less pseudo backbones and to a smaller reduction. On the other hand the analysis of starting tours is very time consuming, as in general, each variation of the set of starting tours would lead to a new run of LKH. Therefore in this section we only analyze the sets of starting tours which were really used for our experiments. We further restrict our analysis to the six instances *xvb13584*, *pjh17845*, *fnc19402*, *ido21215*, *boa28924*, and *fht47608* for which we could set world records. For these six instances, Table 3 shows the number of starting tours and the variants of LKH they come from. This distribution of starting tours was given by previous experiments from [22, 23]. In average, we had more starting tours for smaller instances than for larger ones. Furthermore, we chose more tours from good classes of starting tours, i.e., classes which resulted in good tours in shorter time, but we also included worse classes.

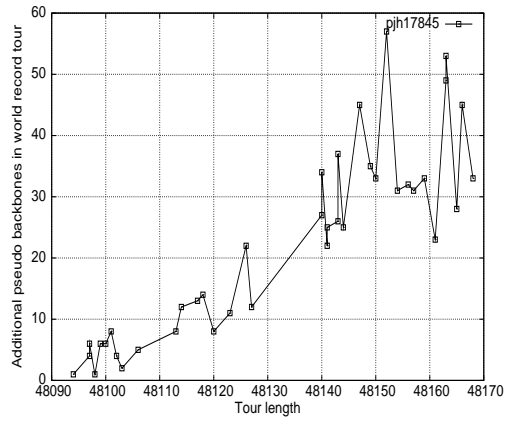
**Table 3.** Distribution of starting tours

Instance	# Tours	LKH	UT	SK	BB	BBT	2-V	AP
<i>xvb13584</i>	100	1	1	40	31	21	2	4
<i>pjh17845</i>	39	1	0	6	12	20	0	0
<i>fnc19402</i>	59	1	1	12	24	21	0	0
<i>ido21215</i>	58	1	0	12	24	21	0	0
<i>boa28924</i>	59	1	1	12	23	22	0	0
<i>fht47608</i>	8	1	1	6	0	0	0	0

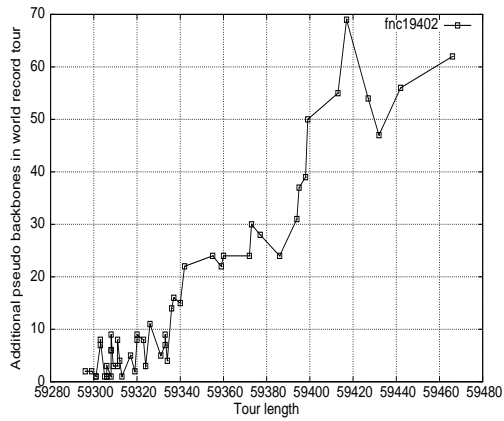
In general, removing some tours from the set of starting tours will lead to more pseudo backbones. This may have two consequences:



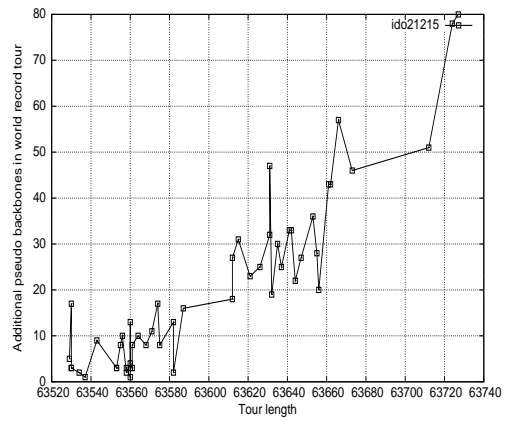
(a) Instance *xvb13584*



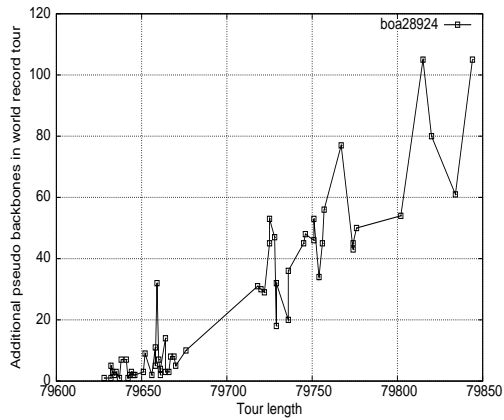
(b) Instance *pjh17845*



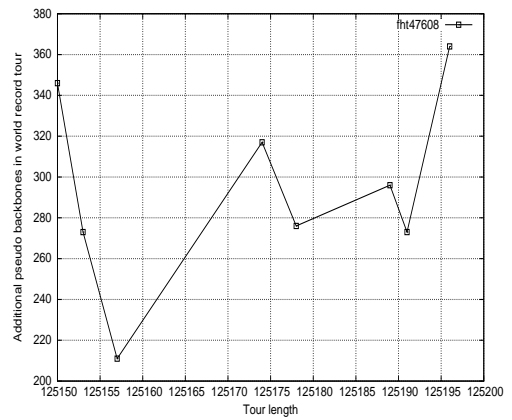
(c) Instance *fnc19402*



(d) Instance *ido21215*



(e) Instance *boa28924*



(f) Instance *fht47608*

**Fig. 5.** Dependence of additional pseudo backbones in record tour on tour length

**Table 4.** Distribution of all minimum cardinality subsets of starting tours, which still lead to the record tour

Instance	# Minimum Subsets	# Tours	# Backbone Tours	LKH	UT	SK	BB	BBT	2-V	AP
xvb13584	2	12	11	0	0	6	2	3	0	1
pjh17845	3	12	9	0	–	2	3.3	6.7	–	–
fnc19402	1	12	12	0	0	3	5	4	–	–
ido21215	15	15	7	0	–	3.3	6	5.7	–	–
boa28924	2	14	13	0	1	2.5	6.5	4	–	–
fht47608	1	8	8	1	1	6	–	–	–	–

- **Positive:** The new reduced TSP instance becomes smaller, and LKH can be applied more effectively.
- **Negative:** If one additional pseudo backbone is not present in the record tour, it is impossible to find again the record tour using the new reduced TSP instance.

Note that the negative consequence could be less important, as there could be further record tours or even better tours than our considered record tour, but we assume that it is essential to find the record tour again.

In the following we will investigate both consequences. To optimize the effectiveness of our algorithm, we would like to reduce the set of starting tours as much as possible without losing the possibility to find the record tour again. To find out the importance of a tour with respect to the set of starting tours we omit this tour from the set and compute the additional pseudo backbones which appear also in the record tour. If this value is large and all additional pseudo backbones are contained in the record tour, this tour can be omitted making our algorithm more effective. We only consider the case that one tour is omitted and suppose that tours with a large length are the less important ones. For that we consider for each of the six instances and for each used starting tour, how the number of additional pseudo backbones, which are contained in the record tour, depends on the tour length.

As expected, we observe in Fig. 5 that the larger the length of the omitted tours is, the larger is the number of additional pseudo backbones in the record tour.

The second point we consider is the influence of the class to the quality of the starting tours. It is interesting, whether we can omit a whole class without losing the possibility to find again the record tour. If we omit all tours from one class and this leads to an additional pseudo backbone edge *not* in the record tour, this class is necessary. For this purpose, we computed for each of the six instances *all* minimum cardinality subsets of the set of starting tours with the condition that each subset still leads to the record tour. In Table 4 the distribution of these subsets is shown. For each instance, the number in the first column is the number of minimum subsets, the number in the second column the minimum cardinality, i.e., the number of starting tours in each of these subsets, and the number in the third column the number of backbone starting tours, i.e., the tours, which appear in *all* minimum subsets. The numbers in the remaining columns are the average cardinality of the starting tours of each class over all subsets. Furthermore “–” means that this class

contains no starting tours and therefore also no starting tours in a subset of minimum cardinality.

We observe that except for *ido21215* only a small number of minimum subsets exists, and that the cardinality of the subsets is considerably smaller than that of the original sets of starting tours and contains many backbone starting tours. Furthermore all classes except 2-V are contained in at least one of the minimum cardinality subsets. This means that although only a few starting tours are really important, only one class can be omitted.

Thus two criteria of the sets of starting tours seem to be rather important: the good quality of tours and that they differ as much as possible, i.e., they come from different heuristics or at least (as in our case) different variations of the same heuristic. For example, if all starting tours come from the same class, e.g., from LKH with similar parameter values, we expect too many pseudo backbone edges and then too many good tours would be excluded from the search. In contrast, the number of starting tours is less important; about 15 starting tours should be enough for our purpose. We are convinced that an optimization of the set of starting tours with respect to both criteria would lead to better tours and maybe further world records.