SINTEF REPORT

SINTEF ICT

Address:	NO-7465 Trondheim NORWAY
Location:	Forskningsveien 1 0373 Oslo
Telephone:	+47 22 06 73 00
Fax:	+47 22 06 73 50

Enterprise No.: NO 948 007 029 MVA

TITLE

Extending the Lin-Kernighan algorithm to improve solutions to VRPs with Time Windows

AUTHOR(S)

Nina Holden, Geir Hasle

CLIENT(S)

SINTEF ICT

REPORT NO.	CLASSIFICATION	CLIENTS REF.	
SINTEF A13822	Open	Geir Hasle	
CLASS. THIS PAGE	ISBN	PROJECT NO.	NO. OF PAGES/APPENDICES
Open	978-82-14-04459-1	90A34901	12
ELECTRONIC FILE CODE		PROJECT MANAGER (NAME, SIGN.)	CHECKED BY (NAME, SIGN.)
Nina Holden TSPTW.doc		Geir Hasle	Oddvar Kloster
FILE CODE	DATE	APPROVED BY (NAME, POSITION, SIGN.)	
	2009-10-01	Martin Stølevik	

ABSTRACT

Helsgaun's implementation of the Lin-Kernighan algorithm (LKH) is an effective heuristic solver for the Traveling Salesman Problem (TSP). The report presents an extension of LKH to support time windows, i.e., to a solver for the TSPTW. The new solver is referred to as LKHTW. SINTEF's solver for rich Vehicle Routing Problems, Spider, has been integrated with LKHTW. Several alternative methods for combining Spider's VRP heuristics with optimization of each tour using LKHTW have been developed and investigated empirically on a set of test instances. The results show that small but significant improvements of the Spider solutions can be obtained in cases with few or very wide time windows, whereas only few and marginal improvements are obtained in cases with narrower time windows.

KEYWORDS	ENGLISH	NORWEGIAN
GROUP 1	transport; optimization	transport; optimering
GROUP 2	VRP; VRPTW; TSP; TSPTW; Rich VRP	VRP; VRPTW; TSP; TSPTW; rik VRP
SELECTED BY AUTHOR	time window; heuristic	tidsvindu; heuristikk
	Spider	Spider

SINTEF REPORT

SINTEF

SINTEF ICT

Address: NO-7465 Trondheim
NORWAY
Location: Forskningsveien 1
0373 Oslo
Telephone: +47 22 06 73 00
Fax: +47 22 06 73 50

Enterprise No.: NO 948 007 029 MVA

TITLE

Extending the Lin-Kernighan algorithm to improve solutions to VRPs with Time Windows

AUTHOR(S)

Nina Holden, Geir Hasle

CLIENT(S)

SINTEF ICT

REPORT NO.	CLASSIFICATION	CLIENTS REF.	
SINTEF A13822	Open	Geir Hasle	
CLASS. THIS PAGE	ISBN	PROJECT NO.	NO. OF PAGES/APPENDICES
Open	978-82-14-04459-1	90A34901	12
ELECTRONIC FILE CODE		PROJECT MANAGER (NAME, SIGN.)	CHECKED BY (NAME, SIGN.)
Nina Holden TSPTW.doc		Geir Hasle	Oddvar Kloster
FILE CODE	DATE	APPROVED BY (NAME, POSITION, SIGN.)	
	2009-10-01	Martin Stølevik	

ABSTRACT

Helsgaun's implementation of the Lin-Kernighan algorithm (LKH) is an effective heuristic solver for the Traveling Salesman Problem (TSP). The report presents an extension of LKH to support time windows, i.e., to a solver for the TSPTW. The new solver is referred to as LKHTW. SINTEF's solver for rich Vehicle Routing Problems, Spider, has been integrated with LKHTW. Several alternative methods for combining Spider's VRP heuristics with optimization of each tour using LKHTW have been developed and investigated empirically on a set of test instances. The results show that small but significant improvements of the Spider solutions can be obtained in cases with few or very wide time windows, whereas only few and marginal improvements are obtained in cases with narrower time windows.

KEYWORDS	ENGLISH	NORWEGIAN
GROUP 1	transport; optimization	transport; optimering
GROUP 2	VRP; VRPTW; TSP; TSPTW; Rich VRP	VRP; VRPTW; TSP; TSPTW; rik VRP
SELECTED BY AUTHOR	time window; heuristic	tidsvindu; heuristikk
	Spider	Spider

TABLE OF CONTENTS

1 Background

Helsgaun's implementation of the Lin-Kernighan algorithm (LKH) is an effective heuristic solver for the Traveling Salesman Problem (TSP).

SINTEF's VRP solver Spider has a flexible and generic rich model that supports a variety of industrial cases and VRP variants in the literature. The algorithm is basically a combination of Iterated Local Search and Variable Neighborhood Search, utilizing a large repertoire of constructors, local search operators and diversifiers that have been designed to accommodate the rich VRP model. For details, see [1] and [2].

The research described in this report had two main objectives:

- to investigate the quality of the individual tours of VRP solutions from Spider
- to extend LKH to a solver for the TSPTW

2 The Lin-Kernighan algorithm with time windows

The basis for the TSPTW solver presented here is Helsgaun's implementation 2.0.3 of the Lin-Kernighan algorithm (LKH), see [3], [4] and [5] for code and documentation. To simplify the extension, some features of the LKH algorithm were removed. A test to check time window feasibility was added before the each suggested move. The removed features do not have a big impact on performance according to a limited computational study on TSP instances with few cities. Many of the features can be relatively easily rewritten to accommodate time window constraints in an efficient way.

Here follows a description of the Lin-Kernighan algorithm LKH the way it works with the selected input parameters. More details can be found in [4] and [5]. With other input arguments the algorithm may work slightly differently.

1. A number of independent runs (default: 100) are done, where we search for a solution with lowest possible cost in each run. After each run, we try to merge the solution from the run with the currently best solution, see [5] for details.
2. Each run consists of a certain number (default: the number of nodes in the problem) of trials.
3. In the beginning of each trial we choose an initial tour. We start out in a random node n_1, then we pick n_2, n_3 etc, such that n_1-n_2-n_3-etc becomes our initial tour. When we choose a new node n_i, we give priority to candidate edges (see 6.), edges that belongs to the currently best tour and edges with alpha value equal to 0 (see 6.).
4. After we have chosen our initial tour, we try to improve the tour by the Lin-Kernighan heuristics. When we have improved the initial tour, we try to improve this tour by merging it with the currently best tour.
5. The Lin-Kernighan heuristics alternates between doing sequential and non-sequential moves, as long as improvements can be done. A sequential move consists of several submoves of length $\leq k$, where each submove is sequential, feasible (i.e. performing the move gives us one closed tour containing all nodes once) and the accumulated gain along the move is positive, but where the cost reduction obtained by doing the move is not necessarily positive. See [5] for details.
6. Candidate edges: Each node has a number of other nodes associated with it through so called candidate edges. An edge is a candidate edge if it is likely that the edge is part of the final tour. For each edge we calculate the minimum length of a 1-tree containing the specified node, minus the length of the network's minimum 1-tree, and this is the edge's alpha value. We look at an edge's alpha value to determine whether it is a candidate edge or not. We also adjust the candidate set when we have improved a tour, such that an edge

is more likely to become a candidate edge if it belongs to the two currently best tours. The candidate set is reset between each run, but saved between each trial.

Relative to LKH, the changes in LKHTW are as follows:

1. The solution from each run is not merged with the currently best solution. There are fewer runs because of more determinism/less randomness. Otherwise this part of the algorithm is unchanged.
2. Unchanged, except that fewer trials are normally performed for the same reason as described in 1.
3. As the initial tour needs to be feasible (i.e. satisfy the time windows), this part of the algorithm had to be changed. The assumption is that the sequence of nodes in the input file is feasible. This sequence determines the initial tour in each trial.
4. Unchanged, except that the tour returned from the Lin-Kernighan method is not merged with the currently best tour.
5. The LKHTW algorithm does only sequential moves. Before we perform each sequential move, there is a test checking whether the move satisfies the time windows. If the move satisfies the time windows, it is performed, otherwise it is rejected, and we continue to search for feasible sequential moves.
6. Unchanged.

The possible and necessary input parameters for LKHTW are different from the input parameters to LKH:

- All node distances must be Euclidean, and the problem must be 2D.
- The problem file, containing all city locations in LKH, must also contain the time window constraints in LKHTW. The file consists of 6 columns: The first column describes the number of the node, the second and third columns describe the *x* and *y* coordinates, the fourth and fifth columns describe the beginning and end of the feasible time window, whereas the last column describes the service time.

Some of the input parameter choices not possible in LKHTW are easy to include by making small changes in the code.

3 Integrating LKHTW and Spider

The Spider code is modified such that it is possible to improve a solution by calling LKHTW. An option “lkhtw” has been added to the menu, and when choosing this option, each tour in the currently best Spider solution is sent to LKHTW. Each tour in a Spider solution is a TSP with time windows, and can therefore be improved by LKHTW. The solution obtained by Spider is used as a starting solution in LKHTW. After LKHTW has tried to improve the tours, the solution from LKHTW is transported back to Spider, such that we can continue to improve this solution with Spider.

The program only supports VRPs where

1. all orders are either pickup orders or delivery orders
2. each time constraint is a single time interval
3. the city locations are given by *x* and *y* coordinates, and all distances are 2D Euclidean distances
4. the time it takes to travel between two places has the same numeric value as the distance between the places
5. there are no waiting costs – the cost function only consists of the distance traveled, plus an additional cost for each tour
6. there is only one possible location for each task

It is possible to allow multiple time windows, non-Euclidean distances and variable speed with relatively small changes in the code and algorithm, i.e. it is relatively easy to remove the requirements 2, 3 and 4.

4 Computational experiments

The input parameters to Spider were:
-ins –rel –two 0 –rar 100 1 1 1 –rem 1 –cro 10 –nex 10 1 1 5 0.

The input parameters to LKH that did not take default value were:

- PATCHING_C = 3. The maximum number of disjoint cycles to be patched in an attempt to find a feasible and gainful move.
- PATCHING_A = 2. The maximum number of disjoint alternating cycles to be used for patching.
- RUNS = 3. The number of runs (see 2.1) were set to 3.
- MAX_TRIALS=10. The number of trials per run (see 2.2) were set to 10.

Four different combined optimization methods were developed and investigated:

- *opt1*. The input solution was optimized for 8 times 300 seconds with Spider (=40 minutes altogether), using Spider's *tiop* function to interrupt the optimization each 5 minutes.
- *opt2*. Same as *opt1*, except that the currently best solution from Spider is optimized with LKHTW after the 40 minutes of optimization with Spider.
- *opt3*. Same as *opt1*, except that currently best solution from Spider is optimized every 5 minutes with LKHTW. Every 5 minutes the solution is sent from Spider to LKHTW, LKHTW improves the solution, and Spider continues to optimize the solution that was found by LKHTW. The optimization with LKHTW comes in addition to the optimization with Spider, so the total optimization time is above 40 minutes.
- *opt4*. Same as *opt3*, except that LKHTW improves the solution from Spider after every 100 seconds of optimization with Spider, instead of after 300 seconds (5 minutes).

Three different sets of problem instances were investigated:

1. One set of 30 data files where all the cities have time windows, and the time windows are predominantly narrow. The 30 instances are the first instance in each of the six problem classes in Gehring and Homberger's well known VRPTW benchmark over five different sets corresponding to number of customers (200, 400, 600, 800, 1000). The average number of orders per tour in the best known solution was between 10 and 56.

2. One set of 30 data files, where none of the cities had time windows. Except for the time windows the problem data (the number of cities, the number and capacity of the vehicles, the location of the cities etc.) were identical to the problem data described in 1.

3. One set of 27 data files, where the time windows were wider than for the files described in 1, or where only a certain percentage of the cities had time windows. Five of the instances from Set 1: RC1_8, R1_6, C2_4, RC2_2 and RC2_4 were selected. For each of these instances, derived instances with six different time window characteristics were generated (except for R1_6, where only three variants were generated). Time window width varied between 3, 5 and 7 on a scale from 1 to 7 (7 meaning wide time windows), and the number of cities that had time window constraints varied between 25%, 50% and 75%. The time

window width was 1 in the cases where only some of the cities had time windows, and all the cities had time constraints in the cases where the time windows were wide.

5 Results

In Appendix 1, there are three tables describing the result for each of the three problem data sets, see Table 5.1, Table 5.2 and Table 5.3. The different columns in the tables have this meaning:

- *Case*. The name of the corresponding Gehring and Homberger case. If the name has only 2 numbers in it, it means that the cities do not have any time constraints. The second number in each name gives the number of cities divided by 100, for example are there 400 cities in the case with name C2_4_1.
- *Orders with TW*. The percentage of the orders that have time window constraints.
- *TW width*. Gives the width of the time windows on a scale from 1 to 7, with 7 meaning wide time windows and 1 meaning narrow time windows.
- *Best known solution*. The best known solution to the given case.
- *opt1, opt2, opt3, opt4*. Gives the optimization method used, see “4 Computational experiments” for description.
- *Distance*. Gives the distance corresponding to the value of the objective function after the optimization has been performed. Includes only the traveling distance, not the cost associated with the number of tours.
- *Avg orders per tour*. The average number of orders per tour in the solution.
- *Relative improvement.* Gives the relative improvement of the cost function when using *optx* compared to *opt1* (x = 2, 3, and 4).
- *Time*. Gives the CPU time it took to perform the optimization, in seconds.
- *LKHTW improvement*. Gives the number of different starting tours that was given to LKHTW during the optimization, and the number of times LKHTW managed to improve the initial tour it was given.

Data set 1: Problems with narrow time windows
Table 5.1 shows that LKHTW does not manage to improve solutions from Spider in cases where the time windows are narrow and all the cities have time constraints.

Data set 2: Problems without time windows
Table 5.2 illustrates that LKHTW can be used to improve problems without any time windows on the cities. On average, the improvement is 0.21%.

- *opt2* gave a better result than *opt1* in 25 of the 30 runs. The solution was on average 0.21% better when using *opt2* compared to *opt1*, and the maximum improvement was 0.838%.
- *opt3* gave better results than *opt1* in 20 of 30 cases, gave a worse result in 9 cases, and gave the same result in one case. The average improvement was 0.25%, with an estimated standard deviation of 0.016. Assuming the relative difference of the solutions with *opt3* and *opt1* has a normal distribution, [-0.0026, 0.0076] is a 90% confidence interval for the relative improvement.
- There was no significant difference between the results for random, clustered and random/clustered problems.
- The improvement by using *opt2* instead of *opt1* was larger for problems with many cities. The average improvement was -0.028%, 0.22%, 0.25%, 0.27% and 0.30% for problems with 200, 400, 600, 800, and 1000 cities, respectively.
- The improvement by using *opt2* instead of *opt1* was larger for problems that had long and few tours, compared to problems with more and shorter tours. There are 15 problems where the average number of cities per tour in the optimal solution is between 10 and 12,

there are 5 problems where the same number is between 33 and 37 cities, and 10 problems where the number is between 50 and 56 cities. In these cases the average improvement was 0.029%, 0.39%, and 0.40%, respectively.

For the *opt3* runs, the number of cases where LKHTW was able to improve the tour it got from Spider, was observed. LKHTW improved the tours more often when the number of cities was large, and when the tours were long. The fraction of the times LKHTW managed to improve the Spider solution was 9/21 (43%), 16/34 (47%), 23/33 (70%), 22/33 (67%), and 19/30 (63%) for problems with 200, 400, 600, 800 and 1000 cities respectively. The similar fractions were 34/73 (47%), 17/24 (70.8%) and 38/54 (70.4%) for problems with 10-12, 33-37 and 50-56 cities per tour in average, respectively. See table 5.2 for details.

Data set 3: Problems with wide or few time windows

LKHTW may improve solutions from Spider in cases where few of the cities have time constraints, or where the time windows are wide:

- LKHTW improves the solutions from Spider more if the time windows are wide. The travelling distance when using *opt2* was averagely 0.019% less than when using *opt1* for the problems with the widest time windows (width 7). For problem data with width 5 and width 3 the average improvement was 0.0088% and 0.0006% respectively. LKHTW improved the solution from Spider in 3/4, 3/4 and 1/4 cases for time window widths 7, 5 and 3 respectively. When using *opt3* LKHTW improved the solution from Spider in 15/22, 13/23 and 4/19 cases for time window widths of 7, 5, and 3, respectively.
- LKHTW improves the solution from Spider more if not all the orders have time constraints. The solutions from *opt2* were averagely 0.21% less for the data sets where 25% of the cities had time window constraints. The similar number for data sets where 50% and 75% of the cities had time constraints was 0.00098% and 0.00344%. LKHTW improved the solution from Spider in 5/5, 1/5, and 1/5 of the cases for the problem sets where respectively 25%, 50%, and 75% of the orders had time windows. When comparing *opt1* and *opt3*, the similar numbers were 21/34, 8/37 and 2/34 respectively.

See Table 5.3 for detailed results.

6 Conclusion

LKHTW is a generalized version of the Lin-Kernighan Heuristic for TSP that can be used to improve solutions of TSP/VRP problems with time windows. Four new methods that combine the heuristics of SINTEF's VRP solver Spider with the LKHTW for optimization of individual routes have been developed. Experimental investigation show that LKHTW can improve the results from Spider in cases where the time windows are wide, or in cases where only some of the cities have time constraints.

7 References

[1] Hasle G., O. Kloster: ***Industrial Vehicle Routing Problems.*** Chapter in Hasle G., K-A Lie, E. Quak (eds): ***Geometric Modelling, Numerical Simulation, and Optimization.*** ISBN 978-3-540-68782-5, Springer 2007.

[2] Hasle G., O. Kloster: ***Vehicle Routing in Practice***. Chapter in ***Buchholz P., A. Kuhn (eds): Optimization of Logistics Systems – Methods and Experiences***. Praxiswissen, Dortmund, Germany, 2008. ISBN 978-3-89957-068-7.

[3] Helsgaun K.: ***Helsgaun's implementation of Iterated Lin-Kernighan***, http://akira.ruc.dk/~keld/research/LKH/

[4] Helsgaun K.: ***An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. Datalogiske skrifter*** **(*Writings on Computer Science*)** no. 81. Roskilde University, 1999.

[5] Helsgaun K.: ***An Effective Implementation of K-opt Moves for the Lin-Kernighan TSP Heuristic. Datalogiske skrifter*** **(*Writings on Computer Science*)** no. 109 (2006). Roskilde University, 2007.

Appendix 1. Detailed experimental results

Table 5.1. Result of the runs with data set 1.

Case	Best known solution		Opt1			Opt2			Opt3			
	Number of tours	Distance	Distance	Number of tours	Avg orders per tour	Distance	Relative improvement (%)	Time (s)	Distance	Relative improvement (%)	Time (s)	LKHTW improvement
C1_2_1	20	2704,57	2704.568	20	10	2704.568	0	2405	2704.568	0	2439	0/1
C1_4_1	40	7152,02	7152.057	40	10	7152.057	0	2410	7152.057	0	2468	0/1
C1_6_1	60	14095,64	14095.644	60	10	14095.644	0	2415	14095.644	0	2518	0/1
C1_8_1	80	25030,36	25190.822	80	10	25190.822	0	2422	25190.822	0	2552	0/2
C110_1	100	42478,95	42479.055	100	10	42479.055	0	2426	42483.123	-0.0096	2569	0/3
C2_2_1	6	1931,44	1929.394	7	28.6	1929.394	0	2402	1929.394	0	2412	0/2
C2_4_1	12	4116,05	4160.807	13	30.8	4160.807	0	2405	4160.327	0.0115	2436	0/3
C2_6_1	18	7774,10	7833.604	19	31.6	7833.604	0	2407	7832.245	0.0173	2444	0/3
C2_8_1	24	11654,81	11886.948	26	30.8	11886.948	0	2411	11847.131	0.3350	2454	0/3
C210_1	30	16879,24	17318.923	33	30.3	17318.923	0	2415	17521.434	-1.1693	2468	0/2
R1_2_1	19	5024,65	3493.205			3493.205	0		3493.205	0		0/4
R1_4_1	38	11084,00	7678.351			7678.351	0		7768.076	-1.1686	2425	0/7
R1_6_1	59	21131,09	22191.810	64	9.38	22191.810	0	2413	22080.283	0.5026	2439	0/5
R1_8_1	79	39612,2	38474.128	83	9.64	38474.128	0	2411	38576.059	-0.2649	2454	0/3
R110_1	100	53904,23	56945.883	102	9.8	56945.883	0	2412	57121.130	-0.3077	2473	0/3
R2_2_1	4	4483,16	2822.215	13	15.4	2822.215	0	2401	2822.215	0	2411	0/3
R2_4_1	8	9213,68	7702.763			7702.763	0	2405	7768.076	-0.8479	2425	0/7
R2_6_1	11	18291,18	15738.952	32	18.8	15738.952	0	2408	15681.992	-0.3619	2437	0/7
R2_8_1	15	23274,22	26301.950	36	22.2	26301.950	0	2414	26572.211	-1.0275	2453	0/8
R210_1	19	42467,87	39982.418			39982.418	0	2418	40449.027	-1.1670	2469	0/6
RC1_2_1	18	3602,80	3590.080	20	10	3590.080	0		3569.178	0.5821		0/6
RC1_4_1	36	8630,94	8954.024	39	10.3	8954.024	0	2404	8985.991	-0.3570	2435	0/5
RC1_6_1	55	17317,13	18121.453	58	10.3	18121.453	0	2407	18096.562	0.1374	2452	0/4
RC1_8_1	72	35102,79	32536.597	76	10.5	32536.597	0	2411	32628.959	-0.2839	2480	0/4
RC110_1	90	47143,90	49844.908	95	10.5	49844.908	0	2417	49822.969	0.0440	2490	0/4
RC2_2_1	6	3099,53	2822.215	10	20	2822.215	0	2401	2822.215	0	2411	0/3
RC2_4_1	11	6688,31	6198.853	17	23.5	6198.853	0	2404	6279.458	-1.3003	2425	1/8
RC2_6_1	15	13163,03	12491.805	28	21.4	12491.805	0	2408	12548.448	-0.4534	2442	0/8
RC2_8_1	19	20520,49	20062.227	28	28.6	20062.227	0	2413	20315.440	-1.2621	2456	2/7
RC210_1	21	29754,06	30553.460	27	37	30553.460	0	2419	30651.390	-0.3205	2481	0/7
Average							**0**			**-0.2649**		

Table 5.2. Result of the runs with data set 2.

Case	Opt1			Opt2			Opt3			
	Distance	**Number of tours**	**Avg orders per tour**	**Distance**	**Relative improve-ment (%)**	**Time (s)**	**Distance**	**Relative improve-ment (%)**	**Time (s)**	**LKHTW improve-ment**
C1_2	2584.132	18	11.1	2583.563	0.0220	2411	2582.336	0.0695	2493	0/3
C1_4	6814.015	37	10.8	6810.379	0.0534	2416	6837.298	-0.3417	2519	2/6
C1_6	13953.718	56	10.7	13942.566	0.0799	2429	13923.584	0.2160	2630	2/3
C1_8	24383.616	72	11.1	24372.371	0.0461	2441	24257.181	0.5185	2714	2/4
C110	40279.442	90	11.1	40262.988	0.0409	2443	40180.915	0.2446	2730	1/3
C2_2	1494.266	6	33.3	1494.266	0	2406	1485.004	0.6198	2453	2/3
C2_4	3293.729	11	36.4	3276.265	0.5302	2418	3208.438	2.5895	2496	4/7
C2_6	6696.678	17	35.3	6670.680	0.3882	2416	6605.315	1.3643	2570	4/4
C2_8	10063.562	22	36.4	9989.517	0.7358	2434	10086.242	-0.2254	2600	4/6
C210	15350.952	28	35.7	15304.798	0.3007	2434	15332.498	0.1202	2964	3/4
R1_2	2930.257	18	11.1	2930.257	0	2404	2916.610	0.4657	2460	1/4
R1_4	7355.765	37	10.8	7355.765	0	2415	7467.987	-1.5256	2505	3/6
R1_6	16213.517	55	10.9	16213.517	0	2420	16249.421	-0.2214	2584	4/5
R1_8	29018.181	73	11	29016.349	0.0063	2433	29017.068	0.0038	2631	3/5
R110	44383.062	92	10.9	44358.310	0.0558	2434	44409.677	-0.0600	2658	3/7
R2_2	1626.348	4	50	1626.348	0	2405	1626.348	0	2456	2/3
R2_4	3464.082	8	50	3462.992	0.0315	2407	3406.073	1.6746	2485	3/4
R2_6	6638.056	12	50	6627.388	0.1607	2420	6587.559	0.7607	2509	4/7
R2_8	11057.492	15	53.3	10995.036	0.5648	2422	11034.284	0.2099	2683	4/5
R210	16665.491	19	52.6	16551.261	0.6854	2425	16524.609	0.8454	2896	5/7
RC1_2	2858.039	19	10.5	2856.916	0.0393	2406	2847.586	0.3657	2456	2/5
RC1_4	7463.856	37	10.8	7462.210	0.0221	2416	7466.348	-0.0334	2515	1/6
RC1_6	15398.261	56	10.7	15395.315	0.0191	2429	15306.199	0.5979	2624	3/6
RC1_8	27826.810	74	10.8	27822.552	0.0153	2442	27946.018	-0.4284	2687	5/7
RC110	43245.758	91	11	43233.826	0.0276	2448	43664.373	-0.9680	2766	2/3
RC2_2	1522.735	4	50	1521.073	0.1091	2404	1626.348	-6.8044	2456	2/3
RC2_4	3210.806	8	50	3189.128	0.6751	2431	3193.324	0.5445	2543	3/5
RC2_6	6197.925	12	50	6145.991	0.8379	2423	6036.489	2.6047	2536	6/8
RC2_8	10096.730	15	53.3	10073.173	0.2333	2415	9915.143	1.7985	2591	4/6
RC210	15162.124	18	55.6	15060.667	0.6692	2511	14785.030	2.4871	3034	5/6
Average					**0.2117**			**0.2498**		

Table 5.3. Result of experiments with data set 3.

Case	Orders with TW	TW width	Best know solution		Opt1				Opt2			Opt3				Opt4			
			Number of tours	Distance	Distance	Time (s)	Number of tours	Avg orders per tour	Distance	Relative improve-ment (%)	Time (s)	Distance	Relative improve-ment (%)	Time (s)	LKHTW improve-ment	Distance	Relative improve-ment (%)	Time (s)	LKHTW improve-ment
RC1_810	100%	7	72	31766,56	29869.08	2403	75	10.7	29868.96	0.00043		30176.76	-1.030	2535	1/5	30106.035	-0.7933	2760	1/5
C2_410	100%	7	11	4115,46	3782.494	2401	13	30.8	3782.274	0.0058	2407	3750.384	0.848905	2413	3/3	3834.72	-1.38073	2432	4/8
RC2_210	100%	7	4	2015,60	2011.006	2401	6	33.3	2011.006	0		2011.651	-0.03207	2406	3/6	2011.006	0	2418	2/7
RC2_410	100%	7	8	4311,59	4480.97	2401	9	44.4	4477.898	0.069	2401	4455.565	0.566953	2444	8/8	4543.750	-1.40104	2494	6/14
RC1_8_8	100%	5	72	33188,75	30297.3	2401	75	10.7	30295.22	0.0069		30420.12	-0.40536	2526	4/7	30390.253	-0.30679	2803	2/8
C2_4_8	100%	5	12	3787,08	3960.268	2401	14	28.6	3959.293	0.025		3914.328	1.160023		3/4	3943.854	0.414467	2435	3/9
RC2_2_8	100%	5	4	2293,35	2207.706	2400	6	33.3	2207.706	0	2400	2196.441	0.510258	2414	1/5	2200.797	0.312949	2437	0/6
RC2_4_8	100%	5	8	4848,87	4948.691	2401	11	36.4	4948.506	0.0037		4919.085	0.598249	2443	5/7	4821.6468	2.567221	2511	6/18
RC1_8_6	100%	3	72	34849,96	31624.18	2403	76	10.5	31623.42	0.0024		31556.1	0.215263	2498	0/5	31546.169	0.246666	2694	3/14
C2_4_6	100%	3	12	3875,94	3928.153	2401	13	30.8	3928.153	0		3893.692	0.877288	2413	1/3	3988.476	-1.53565	2435	3/9
RC2_2_6	100%	3	4	2975,13	2508.735	2401	7	28.6	2508.735	0	2402	2517.651	-0.3554	2415	0/5	2504.776	0.157809	2452	1/5
RC2_4_6	100%	3	8	5863,56	5530.885	2401	13	30.8	5530.885	0		5470.788	1.086571	2425	3/6	5463.144	1.224777	2472	3/13
RC1_8_4	25%	1	72	28363,65	28481.53	2402	75	10.7	28473.27	0.029		28503.46	-0.07699	2615	2/5	28736.436	-0.89497	3017	2/3
R1_6_4	25%	1	54	15947,03	16776.15	2401	57	10.5	16775.25	0.0054		16797.27	-0.1259	2533	3/5	16592.912	1.092241	2802	8/12
C2_4_4	25%	1	11	3865,45	3830.122	2401	14	28.6	3808.118	0.57		3810.054	0.52396	2413	5/8	3850.574	-0.53397	2437	5/11
RC2_2_4	25%	1	4	2043,05	1890.548	2400	7	28.6	1890.476	0.0038		1884.256	0.332814	2407	3/4	1878.330	0.646268	2417	2/10
RC2_4_4	25%	1	8	3635,04	3803.872	2401	11	36.4	3787.682	0.43		3777.044	0.705281	2441	8/12	3797.131	0.177214	2532	4/8
RC1_8_3	50%	1	72	30608,16	29721.37	2402	76	10.5	29719.92	0.0049		29904.21	-0.61516	2557	3/6	29842.179	-0.40647	2894	6/10
R1_6_3	50%	1	54	17216,16	18124.88	2401	56	10.7	18124.88	0		18102.11	0.125656	2465	2/7	18041.737	0.458745	2633	3/9
C2_4_3	50%	1	11	4109,88	3951.172	2401	14	28.6	3951.172	0		3991.529	-1.02139		0/5	3991.529	0.263719	2439	3/11
RC2_2_3	50%	1	4	2043,05	2252.373	2401	8	25	2252.373	0		2251.127	0.055319	2407	1/5	2245.374	0.310739	2420	0/5
RC2_4_	50%	1	8	4958,74	4762.827	2401	14	28.6	4762.827	0		4728.368	0.723499	2428	2/14	4757.419	0.113546	2480	2/7

3																			
RC1_8_2	75%	1	72	33361,67	30784.62	2402	77	10.4	30784.62	0		30725.64	0.191573	2500	1/6	30934.64	-0.48734	2704	2/8
R1_6_2	75%	1	54	19147,38	19727.74	2402	59	10.2	19724.35	0.017		19720.48	0.036796	2453	0/4	19722.639	0.025857	2453	0/7
C2_4_2	75%	1	12	3929,89	4057.531	4802	14	28.6	4057.531	0		4027.316	0.744665	4824	0/11	4040.104	0.429498	4865	0/13
RC2_2_2	75%	1	5	2825,24	2495.908	2401	8	25	2495.908	0	2402	2495.908	0	2436	0/6	2500.853	-0.19812	2421	0/5
RC2_4_2	75%	1	9	6355,59	5509.967	2391*	15	26.7	5509.967	0		5592.915	-1.50542	2427	1/7	5573.353	-0.9689	2478	2/15
Average										**0.043**			**0.153**				**-0.017**		

*One of the Spider runs was interrupted 10 seconds too early.