A parallel multi-neighborhood cooperative Tabu search for CVRP

Jianyong Jin	Molde University College, Specialized University in Logistics
Arne Løkketangen	Molde University College, Specialized University in Logistics
Teodor Gabriel Crainic	Université du Québec à Montréal, CIRRELT

Outline

- Motivation
- Description of the algorithm
- Computational results
- Some observations
- Future work

Motivation (1)

- Parallel algorithms often use several (or many) processes which work simultaneously on available processors for solving a given problem instance.
- Parallel algorithms can both **speed up** the search and **improve** the **robustness** and the **quality** of the solutions obtained (Crainic, 2008).
- Parallel computing platforms are increasingly accessible. Parallel algorithms can use such computing resources in a more efficient way.

Motivation (2)

- Vehicle routing problem (VRP) is a classical operations research problem.
- It also holds a central place in distribution or transportation management.
- As the classical version of VRP, the capacitated vehicle routing problem (CVRP) remains difficult to solve.

Motivation (3)

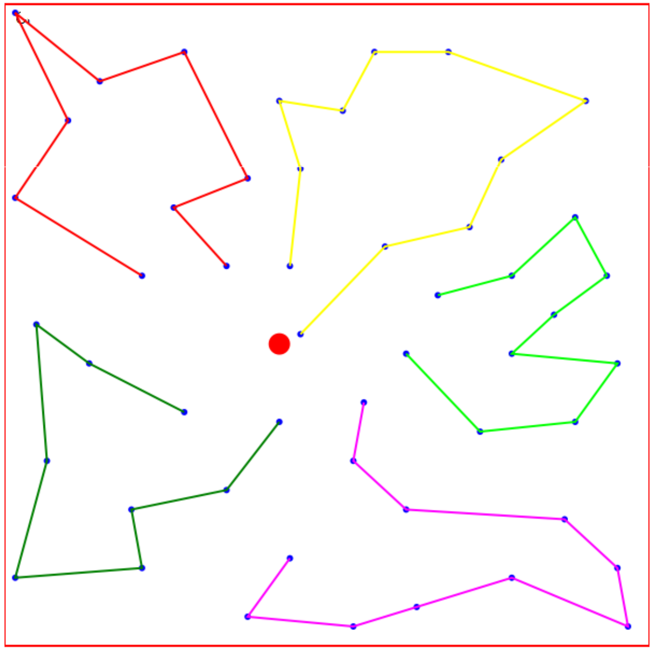
- Among the latest metaheuristic algorithms for VRP, some use multiple neighborhoods. In these methods, multiple neighborhoods are used in serial fashion, one after another following a fixed or randomized sequence.

- The objective of this paper is to explore the strategies of utilizing multiple neighborhoods in a parallel setting and compare their effectiveness.

Description of the algorithm (1)

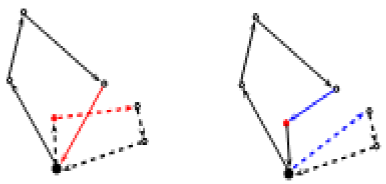
Solving the CVRP is to determine a set of vehicle routes

- Start and end at the depot,
- Each customer is visited exactly once,
- The total demand of any routes does not exceed vehicle capacity,
- The length(duration) of any routes does not exceed a upper bound,
- The total cost of all routes is minimized.

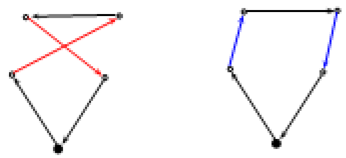


Description of the algorithm (2)

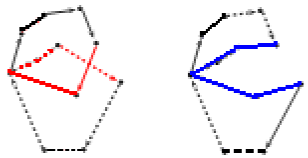
Four neighborhoods are used:



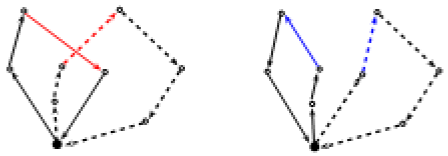
Reinsertion: move a node to another position.



2-opt: remove two edges, add two new ones.



Exchange: swap two nodes from two routes.



2-opt*: swap the head/tail parts of two routes.

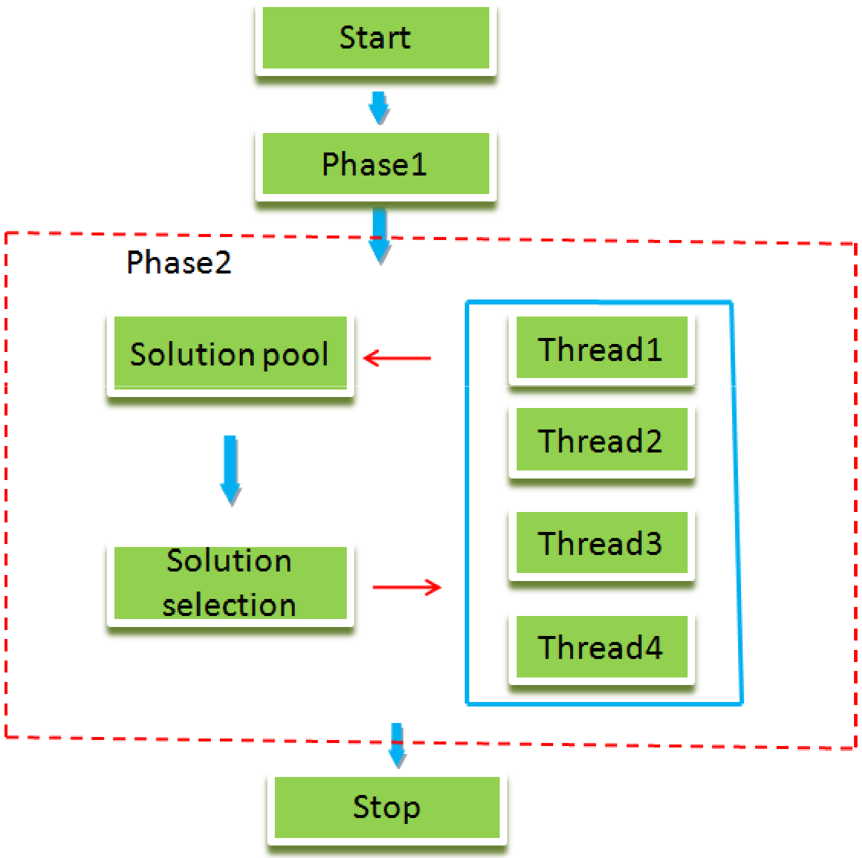
Description of the algorithm (3)

- The selected neighborhoods are applied in Granular Tabu search (Toth & Vigo, 2003) setting to develop several TS threads. In GTS, most long edges are not considered when generating neighbors. When we consider to move a node, we only allow to move it to a position next to one of its nearest neighbors (or the depot).

Description of the algorithm (4)

- The TS threads are run in parallel. A solution pool is used to support the cooperation among them. Each thread runs for a certain time, stops to exchange solutions with the pool and resumes search again. All threads restart from the same solution.

Description of the algorithm (5)



Phase 1 aims to create a feasible starting solution.

Phase 2 aims to improve the starting solution with four parallel threads using different neighborhoods.

Description of the algorithm (6)

	Neighborhoods used	Role
Thread 1	Reinsertion, 2-opt*.	Main improving thread.
Thread 2	2-opt*.	Assistant improving thread.
Thread 3	Exchange	Assistant improving thread.
Thread 4	Shaking procedure + Improving procedure (Thread 1+ Exchange)	Diversifying the search.
Solution pool		Keep and sort the solutions from each threads, select new starting solutions for the TS threads.

Computational results

Results for the benchmarks of Golden et al. (1998)

Problem	Previous best known	Li et al. (2005)	Psinger and Röpke (2007)	Mester and Bräysy (2007)	Kytöjoki et al. (2007)	Nagata and Bräysy (2009)	Groër et al. (2010) 4pc	Groër et al. (2010) 129pc	PMNTS aver.	PMNTS aver. time (min)	PMNTS best
1(240)	5623.47	5666.42	5650.91	5627.54	5867.84	5626.81	5644.44	5623.47	5627.54	16.54	5623.47
2(320)	8431.66	8469.32	8469.32	8447.92	8476.26	8431.66	8447.92	8435.00	8447.15	29.72	**8419.50**
3(400)	11036.22	11145.80	11047.01	11036.22	11043.41	11036.22	11036.22	11036.22	11080.60	58.44	**11030.80**
4(480)	13592.88	13758.08	13635.31	13624.52	13631.72	13592.88	13624.52	13624.52	13666.84	87.03	13615.20
5(200)	6460.98	6478.09	6466.68	6460.98	6460.98	6460.98	6460.98	6460.98	6464.40	10.47	6460.98
6(280)	8404.26	8539.61	8416.13	8412.88	8415.67	8404.26	8412.90	8412.90	8468.10	23.96	**8403.25**
7(360)	10156.58	10289.72	10181.75	10195.56	10297.66	10156.58	10195.59	10195.59	10209.24	62.16	10184.40
8(440)	11643.90	11920.52	11713.62	11663.55	11872.64	11691.06	11680.31	11649.89	11725.82	86.68	11671.00
9(255)	579.71	588.25	585.14	583.39	620.67	580.42	583.37	579.71	583.15	18.70	581.73
10(323)	737.28	749.49	748.89	741.56	784.77	738.49	742.43	737.28	739.97	38.65	738.50
11(399)	913.35	925.91	922.70	918.45	986.80	914.72	917.91	913.35	917.50	58.96	914.98
12(483)	1102.76	1128.03	1119.06	1107.19	1209.02	1106.76	1117.05	1102.76	1112.44	72.84	1109.93
13(252)	857.19	865.20	864.68	859.11	925.81	857.19	858.89	857.19	864.45	18.82	861.92
14(320)	1080.55	1097.78	1095.40	1081.31	1155.19	1080.55	1081.24	1080.55	1084.59	27.94	1082.52
15(396)	1338.00	1361.41	1359.94	1345.23	1461.49	1342.53	1346.45	1338.00	1353.07	38.07	1351.13
16(480)	1613.66	1635.58	1639.11	1622.69	1742.86	1620.85	1624.42	1613.66	1632.88	55.78	1629.78
17(240)	707.76	711.74	708.90	707.79	726.01	707.76	707.79	707.76	708.46	15.83	707.83
18(300)	995.13	1010.32	1002.42	998.73	1077.53	995.13	998.66	995.13	1002.53	30.81	1000.27
19(360)	1365.60	1382.59	1374.24	1366.86	1444.51	1365.97	1369.34	1365.60	1368.22	36.39	1367.31
20(420)	1818.25	1850.92	1830.80	1820.09	1938.12	1820.02	1824.98	1818.25	1830.10	49.62	1827.39
Aver. deviation %		1.33	0.47	0.26	4.76	0.11	0.36	0.04	0.53		0.28
Time min		1.13	10.80	24.40	0.02	355.9	5.00	5.00		41.87	

3 new best solutions. Average deviation 0.28%.

Computational results

Results for the benchmarks of Li et al. (2005)

Problem	Previous best known	Li et al. (2005)	Psinger and Röpke (2007)	Mester and Bräysy (2007)	Kytöjoki et al. (2007)	Dorronsoro et al. (2007)	Groër et al. (2010) 4pc	Groër et al. (2010) 129pc	PMNTS aver.	PMNTS aver. time (min)	PMNTS best
21(560)	16212.74	16602.99	16224.81	16212.74	16221.22	16212.83	16212.83	16212.83	16247.82	98.63	16220.00
22(600)	14584.42	14651.27	14631.08	14597.18	14654.87	14652.28	14631.73	14584.42	14618.83	121.69	14598.70
23(640)	18801.12	18838.62	18837.49	18801.12	18810.72	18801.13	18801.13	18801.13	18883.80	139.82	18829.80
24(720)	21389.33	21616.25	21522.48	21389.33	21401.41	21389.43	21390.63	21389.43	21427.93	88.08	21399.00
25(760)	16763.72	17146.41	16902.16	17095.27	17358.18	17340.41	17089.62	16763.72	16826.62	176.35	16781.70
26(800)	23971.74	24009.74	24014.09	23971.74	23996.86	23977.73	23977.73	23977.73	24127.10	103.54	23986.10
27(840)	17433.69	17823.40	17613.22	17488.74	18233.93	18326.92	17589.05	17433.69	17522.93	101.37	**17432.30**
28(880)	26565.92	26606.11	26791.72	26565.92	26592.05	26566.04	26567.23	26566.03	26609.50	136.17	26574.40
29(960)	29154.34	29181.21	29405.60	29160.33	29166.32	29154.34	29155.54	29154.34	29190.08	188.64	29162.70
30(1040)	31742.51	31976.73	31968.33	31742.51	31805.28	31743.84	31743.84	31742.64	31772.95	252.06	31753.40
31(1120)	34330.84	35369.17	34770.34	34330.84	34352.48	34330.94	34333.37	34330.94	34384.17	246.23	34340.50
32(1200)	36919.24	37421.44	37377.35	36928.70	37025.37	37423.94	37285.90	37185.85	37305.33	272.59	37204.80
Aver. deviation %		1.18	0.68	0.20	0.80	0.87	0.35	0.06	0.35		0.12
Time min		3.20	48.80	104.30	0.10	1830.00	5.00	5.00		160.43	

1 new best solution. Average deviation 0.12%.

Some observations

Observation 1: both using multiple neighborhoods in serial and parallel fashion have advantages.

Problem	Previous best known	Serial variant Obj	Serial variant Time (min)	SNP variant Obj	SNP variant Time (min)	PMNTS Obj	PMNTS Time (min)	Constraints tightness Capacity %	Constraints tightness Tour length %
1(240)	5623.47	5636.77	26.15	5667.90	13.49	5627.54	16.54	97.0	96.1
4(480)	13592.88	13790.27	100.37	13894.88	63.08	13666.84	87.03	96.0	85.2
5(200)	6460.98	6489.97	21.41	6473.53	9.12	6464.40	10.47	88.9	71.8
8(440)	11643.90	11797.88	88.74	11774.16	45.38	11725.82	86.68	97.8	97.1
9(255)	579.71	586.51	29.70	585.27	17.88	583.15	18.70	95.9	N/A
12(483)	1102.76	1120.62	105.93	1114.21	64.93	1112.44	72.84	98.4	N/A
13(252)	857.19	872.00	17.26	869.95	12.32	864.45	18.82	96.7	N/A
16(480)	1613.66	1647.00	60.81	1638.92	40.90	1632.88	55.48	96.7	N/A
17(240)	707.76	709.80	14.81	714.68	11.40	708.46	15.83	98.2	N/A
20(420)	1818.25	1839.13	42.44	1871.28	33.84	1830.10	49.62	99.5	N/A
Aver. deviation %		1.15		1.33		0.56			
Time min			50.76		31.23		43.20		

Some observations

Observation 2: 2-opt* neighborhood is effective for instances with loose constraints to obtain right route structure.

Example Instance: Golden_Benchmark_5

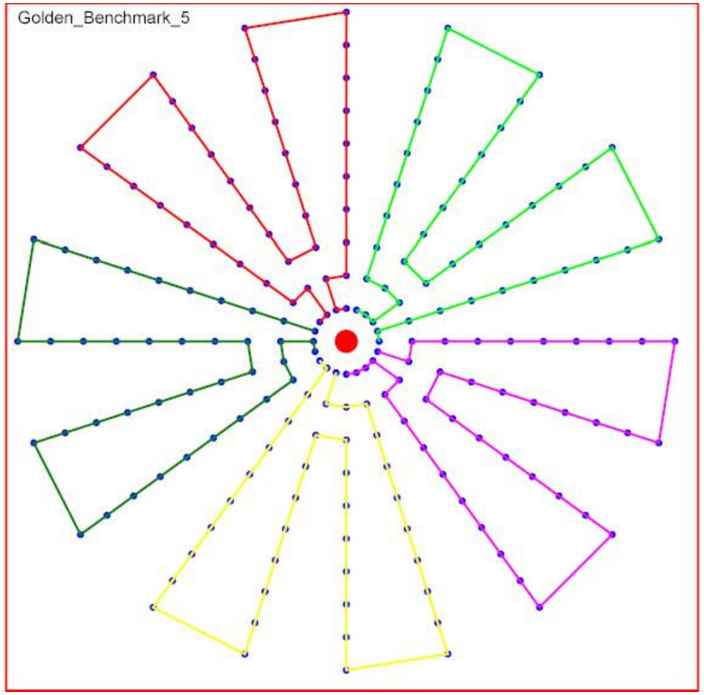
Step	IniSolObj	Reinsertion	2-opt*	Exchange	BestObj	Improve
1	7239.91	6909.00	6885.87	7027.84	6885.87	354.04
2	6885.87	6847.59	6682.38	6866.22	6682.38	203.49
3	6682.38	6664.59	6496.06	6680.95	6496.06	186.32
4	6496.06	6466.68	6496.06	6501.67	6466.68	29.38
5	6466.68	6460.98	6466.68	6466.68	6460.98	5.7
Improvement		35.08	743.85	0		778.93
Contribution(%)		4.5%	95.5%	0		100%

Some observations

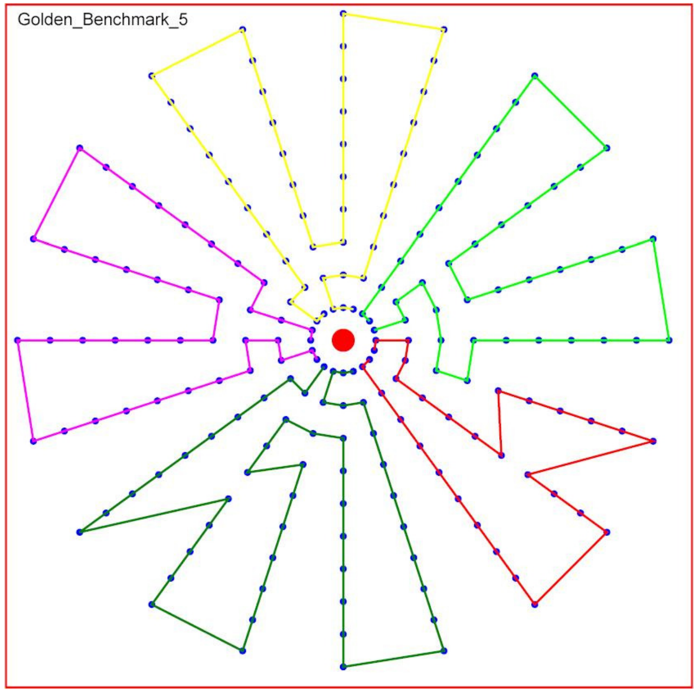
Observation 2: Example Instance: Golden_Benchmark_5

Average route length/route length constraint= 71.8%

Average route load/vehicle capacity = 88.9%



A high quality solution



A low quality solution

Some observations

Observation 3: Exchange neighborhood is effective for instances with overlapping routes

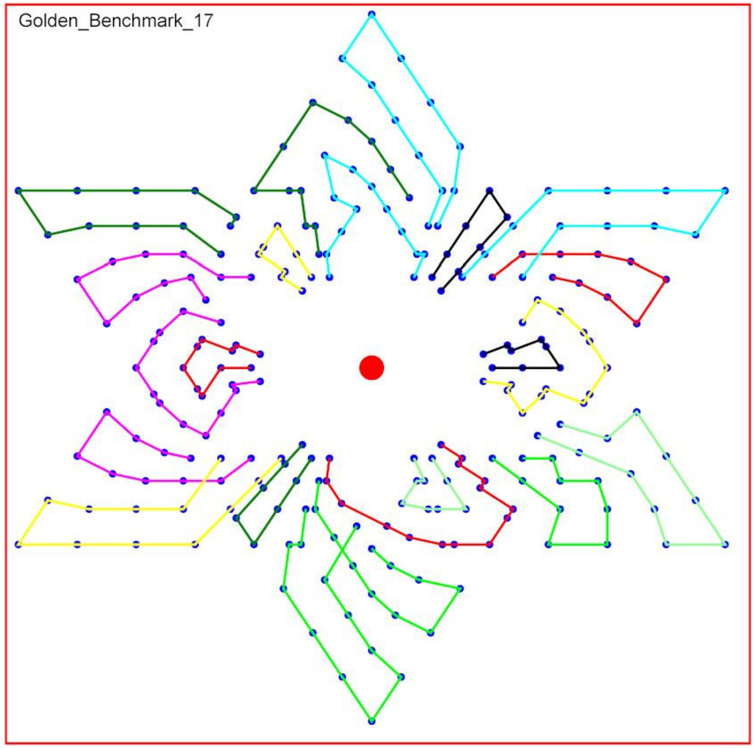
Example Instance: Golden_Benchmark_17

Step	IniSolObj	Reinsertion	2-opt*	Exchange	BestObj	Improve
1	789.414	713.021	733.136	759.092	713.021	76.393
2	713.021	712.438	712.957	712.600	712.438	0.583
3	712.438	710.857	712.438	712.438	710.857	1.581
4	710.857	710.857	710.857	710.722	710.722	0.135
5	710.722	710.722	710.722	710.534	710.534	0.188
6	710.534	710.167	710.534	710.534	710.167	0.367
7	710.167	709.955	710.167	709.702	709.702	0.465
8	709.702	709.252	709.256	709.691	709.252	0.45
Improvement		79.374	0	0.788		80.162
Contribution(%)		99.0%	0	1.0%		100%

Some observations

Observation 3:

Example Instance: Golden_Benchmark_17



Some observations

Observation 4: Reinsertion neighborhood is often more effective than exchange or 2-opt*.

Instance	Solution value	Solution path	Frequency of neighborhood used		
			Reinsertion	2-opt*	Exchange
Golden1	5646.89	R/O/R/R/R/E	4	1	1
Golden4	13839.10	R/R/O/E/E/O/R/R	4	2	2
Golden5	6460.98	O/O/O/E/E/O/E/O /E/R	1	5	4
Golden8	11768.4	R/E/O/R/R/R/R	5	1	1
Golden9	584.44	R/E/E/R/R/R/O/E	4	1	3
Golden12	1116.48	R/R/R/E/R/E/R/E /R/R/E/O/E	7	1	5
Golden13	868.23	O/R/O/R/R/O/R/E /E/R/E	5	3	3
Golden16	1629.38	R/E/E/O/R/O/R/E /R/O/E/E/O/R/R/E	6	4	6
Golden17	708.81	R/R/R/E/R/R/R/O/E	6	1	2
Golden20	1867.58	R/E/E/R/R/E/O/R /E/R/R/O/R/R/R/E	9	2	5

Here, R represents reinsertion, E stands for exchange and O represents 2-opt*.

Some observations

Observation 5: In the setting of parallel multiple-neighborhood cooperation, one neighborhood can either contribute by **improving the solutions more efficiently** than the others or by **generating intermediate solutions** that enable other neighborhoods to find good solutions later.

Step	IniSolObj	Thread1	Thread2	Thread3	BestObj	Improve
1	7239.91	6934.63	**6860.00**	7126.31	6860.00	379,91
2	6860.00	6854.39	**6606.15**	6860.00	6606.15	253.85
3	6606.15	6590.60	**6547.60**	6601.92	6547.60	58.55
4	6547.60	6508.26	**6496.77**	6536.05	6496.77	50.83
5	6496.77	**6491.16**	6496.77	6496.77	6491.16	5.61
6	6536.05	**6508.26**	6516.23	6532.27	6491.16	0.00
7	6516.23	**6483.79**	6508.35	6492.21	6483.79	7.37
8	6508.35	**6472.38**	6475.19	6484.28	6472.38	11.41
9	6484.28	**6466.68**	6466.68	6484.28	6466.68	5.7
10	6532.27	6508.26	**6498.79**	6507.20	6466.68	0.00
11	6498.79	**6483.79**	6486.60	6489.40	6466.68	0.00
12	6489.40	**6483.79**	6486.59	6489.40	6466.68	0.00
13	6486.59	**6483.79**	6486.59	6484.89	6466.68	0.00
14	6484.99	**6460.98**	6479.38	6484.99	6460.98	5.8
Improvement		35.79 (4.6%)	743.14 (95.4%)	0 .00		778.93

Future work

- Explore effective guiding mechanism in parallel setting to further improve the performance.

- Apply parallel multi-neighborhood search framework for rich vehicle routing problems.

Thanks for your attention!

References

- Li, F., B.L. Golden and E.A. Wasil (2005). Very large-scale vehicle routing: New test problems, algorithms and results. Computers & Operations Research 32, 1165–1179.
- Kytöjoki J., Nuortio T., Bräysy O., Gendreau M. (2007), An efficient variable neighborhood search heuristic for very large scale vehicle routing problems, Computers & Operations Research 34, 2743-2757
- Mester, D., and O. Braysy (2007). Active-guided evolution strategies for large-scale capacitated vehicle routing problems. Computers & Operations Research, 34, 2964-2975.
- Pisinger D., Ropke S. (2007). A general heuristic for vehicle routing problems. Computers & Operations Research, 34, 2403-2435.
- Dorronsoro, B., Arias, D., Luna, F., Nebro,A.J., and Alba, E. (2007). A grid-based hybrid cellular genetic algorithm for very large scale instances of the CVRP. High Performance Computing & Simulation Conference (HPCS), *IEEE Press*, *Piscataway*, *NJ*, 759-765.
- Groer, C., Golden, B.L., & Wasil, E.A. (2010). A parallel algorithm for the Vehicle routing problems. INFORMS Journal on Computing, Forthcoming.

References

- Golden, B.L., Wasil, E.A. , Kelly, J.P. and Chao, I.-M. (1998), The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In: T. Crainic and G. Laporte, Editors, Fleet management and logistics, Kluwer, Boston, pp. 33–56.
- Crainic, T.G. (2008), Parallel solution methods for vehicle routing problems. In B. Golden, S. Raghavan, and E. Wasil, editors, The Vehicle Routing Problem: Latest Advances and New Challenges, pages 171–198. Springer, New York.
- Intel Threading Building Blocks: http://www.threadingbuildingblocks.org/.
- Toth, P. and Vigo, D. (2003), The granular tabu search and its application to the vehicle routing problem, INFORMS Journal on Computing 15, 333–346.
- Nagata, Y. and Bräysy O. (2009), Edge Assembly Crossover for the Capacitated Vehicle Routing Problem, Networks 54,205-215.