Analysis on the Implementation of Efficient Heuristic Algorithms for the Traveling Salesman Problem

Cesar Rego[*] *Dorabela Gamboa*[†] *Fred Glover*[*]

[*] Hearin Center for Enterprise Science
School of Business Administration, University of Mississippi, University, MS 38677, USA
Email: {crego, fglover}@bus.olemiss.edu

[†] Universidade Portucalense, Departamento de Informatica
Rua Dr. Antonio Bernardino de Almeida, 541-619, 4200 Porto, Portugal
Email: dgamboa@upt.pt

1 Introduction

The traveling salesman problem (TSP) has been frequently used as a testbed for the study of new local search techniques developed for general circuit-based permutation problems. In addition, tests performed on challenging TSP instances can provide a basis for analyzing the performance characteristics of global search metaheuristic techniques.

To identify the current state-of-the-art, and to better understand the heuristic components that prove most effective for several classes of heuristic algorithms applied to the TSP, the 8th DIMACS Implementation Challenge on the Traveling Salesman Problem [3] was recently organized by Johnson, McGeogh, Glover and Rego [3]. In this talk we describe findings from this Implementation Challenge as well as our own experience with different algorithm implementations. The outcomes give important insights about how to improve the performance of algorithms for several other circuit-based permutation problems.

Some of the most efficient local search algorithms for the TSP are based on variable depth neighborhood search methods such as the standard Lin-Kernighan procedure and the Stem-and-Cycle ejection chain method. Critical aspects of implementing these algorithms efficiently and effectively derive from considering appropriate candidate lists for the potentially available moves, and from taking advantage of special data structures - especially for large TSP problems. We describe how these elements affect the performance of a Stem-and-Cycle ejection chain method, and additionally show the impact of several other algorithmic components, including the use of different types of starting solutions and of incorporating complementary neighborhood structures. Computational findings are reported for TSP instances containing up to 10,000,000 nodes.

2 Algorithm Description

We consider a local search algorithm based on the stem-and-cycle ejection chain method proposed in Glover [4]. The current implementation of the stem-and-cycle algorithm is a slight variant of the P-SEC algorithm described in Rego [8].

The algorithm can be briefly described as follows. Starting from an initial tour the algorithm attempts to improve the current solution iteratively by means of a subpath ejection chain method, which generates moves coordinated by a reference structure called a Stem-and-Cycle (S&C). The stem-and-cycle procedure is a specialized variable depth neighborhood approach that generates a dynamic alternating paths (as opposed to static alternating paths produced, for example, by the classical Lin-Kernighan approach). The generation of moves throughout the ejection chain process is based on a set of rules and legitimacy restrictions determining the set of edges allowed to be used in subsequent steps of constructing an ejection chain. Implementation improvements in the basic algorithm strategy described in Rego [8] make the current version more efficient and effective for solving very large scale problems.

Maintaining the fundamental rules of our algorithm unchanged, we made th following modifications. The first was to introduce the two-level tree data structure described in Fredman et al. [7], and used in some of the most efficient Lin-Kernighan implementations reported in this Challenge (e.g. those of Johnson and McGeoch [6]; Appelgate and Cook [1]; and Helsgaun [5]). In our algorithm the two-level tree data structure was appropriately adapted to replace the less effective doubly-linked lists previously used to represent both the TSP tours and the S&C reference structures. Another modification consisted of replacing a two-dimension array-based data structure by an n-vector to control "legitimacy restrictions" during the ejection chain construction, substantially reducing the storage space required by the earlier version. We also incorporated the possibility for the algorithm to directly access external neighbor lists in the format used by the Concorde system, thus providing additional options to the nearest neighbor list used in the P-SEC algorithm.

The algorithm is implemented as a local search improvement method in the sense that no meta-strategy is used to guide the method beyond local optimality. Also, the method always stops after N iterations of the "re-routing" strategy fail to improve the best solution found so far. (Re-routing consists of starting an S&C ejection chain from a different route node.) This makes our implementation of the S&C algorithm simpler than Lin-Kernighan implementations that make use of additional supplementary techniques such as a "don't look bits" strategy, caching distances, and other implementation tricks.

3 Computational Results

We provide results for the algorithm on the testbed provided in this 8th DIMACS Challenge, starting from both randomly generated solutions and solutions provided by Boruvka's construction algorithm (available in the Concorde TSP library [2]). Random solutions were generated by the systematic generator used in Rego [8] using k=2 and k=5 to provide starting solutions for two runs. Runs were carried out on two different computer systems: a SGI Origin 2000 with six 300 MHz R12000 processors and 24 GB of memory; a Sun Enterprise 3500 with two 400MHz Ultra Sparc processors and 1 GB of memory. (See http://www.mcsr.olemiss.edu/ for additional detail on the characteristics of these machines.) Although these are multiprocessor machines, the Stem-and-Cycle method is implemented as a serial algorithm and no compiler directives for implicit parallelism are used. All runs were performed with a fixed set of algorithm parameters. We use N re-routing iterations of ejection chains of 50 levels (component steps, or "depth"), and 12 quadrant neighbors concatenated with Delaunay triangulations provided by Concorde.

Tables 1,2 and tables 3, 4 report computational results for runs starting from randomly generated solutions and from Boruvka's solution, respectively. Table 3 and 4 provide the running times for the greedy multi-fragment algorithm (provided in Challenge) as a benchmark for estimating the relative scale factors between the different computers.

Problem	Random	S&C (Avg.)	S&C (Avg. Time)	Time (2 runs)	S&C (Min)
E1k.0	521694603	23681042	7.9	15.9	23637160
E1k.1	514379324.5	23201321	15	30.1	23194506
E1k.2	517152497	23380958	7.7	15.4	23346072
E1k.3	525281474	23464054	9.8	19.6	23416795
E1k.4	514060156.5	22958157	10.8	21.7	22943720
E1k.5	519852359	23472354.5	12.7	25.4	23409539
E1k.6	518159018.5	23599948.5	12.4	24.9	23591091
E1k.7	527124504	23202149	9.2	18.3	23199734
E1k.8	526322721.5	23275070	8.3	16.6	23253544
E1k.9	513093242	23622114	11.3	22.7	23607066
E3k.0	1668883498	41102756	49.6	99.2	41050091
E3k.1	1662790584	40770583	50	100	40750076
E3k.2	1668664491	40838290.5	43.1	86.1	40817289
E3k.3	1653275959	41083719	45.4	90.8	40942117
E3k.4	1647546591	41216940	51.2	102.5	41202985
E10k.0	5259263306	72789355	231.2	462.5	72775634
E10k.1	5236649542	73006951	274.3	548.6	72942007
E10k.2	5201586798	72821782.5	273.2	546.3	72769264
E31k.0	16505505990	128958561	2396.8	4793.7	128953570
E31k.1	16459825804	129183522	1858.7	3717.4	129120478
E100k.0	52099256093	228802953	20207.4	40414.8	228707625
E100k.1	52209047730	228636297.5	20413.6	40827.3	228620444
C1k.0	581133653	11602773	11.5	23	11579311
C1k.1	464985345.5	11524849.5	12.5	25.1	11521270
C1k.2	507982808.5	11167859.5	9.1	18.3	11120458
C1k.3	590544403	12097826	12.5	25.1	12072045
C1k.4	520555535.5	11738201.5	10.5	21.1	11691878
C1k.5	429975174.5	11581640	9.1	18.2	11578446
C1k.6	409787059.5	10310730	10.4	20.9	10287025
C1k.7	417079225	10845483.5	9.5	19.1	10823231
C1k.8	557754576.5	11820166	8.5	17.1	11814075
C1k.9	497728307	11163641	7.7	15.5	11157814
C3k.0	1619686549	19532839.5	51.1	102.2	19494864
C3k.1	1588135536	19500411.5	52	104	19484897
C3k.2	1804857299	19957862.5	46.9	93.9	19941757
C3k.3	1711149072	19582162.5	43.1	86.2	19533585
C3k.4	1585846028	19277692.5	51.6	103.2	19216048
C10k.0	5208876976	34010829.5	235.9	471.8	33927763
C10k.1	4831342165	34052098.5	303.9	607.9	33868168
C10k.2	5014977933	33970352.5	289.8	579.6	33955588
C31k.0	16630431372	61494569.5	1853.9	3707.9	61488450
C31k.1	16771377474	61057173.5	2529.5	5059.1	60948010
C100k.0	51368917901	108098607.5	23106.3	46212.7	107889475
C100k.1	51011734350	108960771	27010.2	54020.5	108655651

Table 1: Results for 2 runs with random starting solutions on a SGI Origin 2000

Problem	Random	S&C (Avg.)	S&C (Avg. Time)	Time (2 runs)	S&C (Min)
dsj1000	549651895.5	19122509.5	10.6	21.2	19019807
pr1002	6540958.5	262288	10.8	21.6	261461
si1032	369605.5	92809.5	9.2	18.5	92790
u1060	6779662	226814	9.8	19.6	226101
vm1084	8731347.5	243049	12	24.1	241870
pcb1173	1413171.5	57700	13.7	27.4	57578
d1291	1697854.5	51604.5	13	26.1	51296
rl1304	9388666.5	258038.5	13.2	26.4	257033
rl1323	9778250.5	274580.5	11.3	22.7	274129
nrw1379	1426603.5	57161	15.6	31.3	57142
fl1400	1670691.5	20382.5	15.4	30.8	20278
u1432	3945470.5	154686	11.7	23.4	154552
fl1577	1344956.5	22746.5	13.7	27.5	22469
d1655	2153481.5	63409	16.8	33.7	63043
vm1748	14995257.5	341276.5	21.7	43.5	341257
u1817	2098150.5	58200	17	34.1	58118
rl1889	14829168	324059.5	17.5	35.1	322662
d2103	3249132.5	81885	26.2	52.4	81775
u2152	2527479	65494.5	19.4	38.9	65330
u2319	5995150	235591	18.8	37.7	235503
pr2392	15383836	383622	35.9	71.9	382485
pcb3038	5389270.5	139546	39	78	139480
fl3795	3602574.5	29938	46.6	93.3	29762
fnl4461	8300682.5	184635	70	140	184632
rl5915	42424987.5	579639	100.6	201.2	577790
rl5934	42215704	568679	119.1	238.2	566939
pla7397	1048603378	23698659	110.5	221.1	23616702
rl11849	87094133.5	941425	273.5	547	940301
usa13509	2155922743	20247562	408.1	816.2	20237299
brd14051	41861099	475063	496.4	992.9	474668
d15112	134442327	1593027	686.7	1373.4	1592865
d18512	59497804	653263	670.9	1341.8	652935
pla33810	9391839243	66854192.5	3994	7988.1	66845961
pla85900	28899078605	144286780	23793.1	47586.3	144264466

Table 2: Results for 2 runs with random starting solutions on a SGI Origin 2000

4 Concluding Observations

The ability of the basic sem-and-cycle approach to obtain high quality outcomes without recourse to the usual array of supplementary strategies and auxiliary neighborhoods used to make other methods competitive suggests the opportunity for additional enhancement. A natural change in this direction is simply to utilize more effectively designed candidate lists. In addition, several types of choice and trial solution options in the stem-and-cycle approach remain unexplored, including those arising from more general doubly-routed reference structures. Such alternatives afford interesting avenues for future investigation.

References

[1] D. Appelgate and W. Cook. Chained Lin-Kernighan for large traveling salesman problems. Technical report, Rice University, 2000. http://www.caam.rice.edu/keck/reports/chained_lk.ps.

[2] V. Chvatal D. Applegate, R. Bixby and W. Cook. Concorde: A code for solving traveling salesman problems. Technical report, Rice University, 2001.

[3] F. Glover D. Johnson, L. McGeogh and C. Rego. 8th DIMACS implementation challenge: The traveling salesman problem. Technical report, AT&T Labs, 2000. http://www.research.att.com/ dsj/chtsp/.

[4] F. Glover. New ejection chain and alternating path methods for traveling salesman problems. *Computer Sc. Oper. Res.*, pages 449–509, 1992.

[5] K. Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European J. Oper. Res.*, 1:106–130, 2000.

Problem	Boruvka	Boruvka (Time)	S&C	S&C (Time)	Memory (KB)
E1k.0	27129957	0.02	23593028	7.67	1856
E1k.1	26403263	0.02	23202389	9.63	1856
E1k.2	26424371	0.02	23384381	6.55	1856
E1k.3	26898061	0.02	23350759	6.38	1856
E1k.4	27009839	0.02	22967349	6.14	1856
E1k.5	26887302	0.02	23685793	5.58	1856
E1k.6	26825154	0.02	23710021	8.91	1856
E1k.7	26784161	0.02	23071355	5.75	1856
E1k.8	26434126	0.02	23266814	6.21	1856
E1k.9	27332571	0.02	23643665	10.26	1856
E3k.0	46548291	0.07	41115718	36.72	2888
E3k.1	46806588	0.07	40721473	48.24	2888
E3k.2	47141456	0.07	40782942	42.93	2888
E3k.3	46727005	0.07	41102465	37.38	2888
E3k.4	46754041	0.07	41396131	27.19	2888
E10k.0	82593002	0.22	72820471	181.65	6152
E10k.1	83006035	0.22	73168954	158.71	6152
E10k.2	82442492	0.22	72830493	192.62	6152
E31k.0	145524942	0.67	128902337	1382.16	16464
E31k.1	145340274	0.67	129179100	1479.53	16464
E100k.0	257268861	3.28	228819453	11371.9	49064
E100k.1	257598310	3.35	228680045	14143.07	49064
E316k.0	457407214	23.03	406751036	213825.59	152136
C1k.0	12843082	0.02	11608611	8.25	1856
C1k.1	13085035	0.02	11535963	7.73	1856
C1k.2	12795326	0.02	11182460	6.94	1856
C1k.3	14104258	0.02	12106981	8.11	1856
C1k.4	13207613	0.02	11692112	7.41	1856
C1k.5	13772293	0.02	11573209	9.93	1856
C1k.6	12760292	0.02	10334450	6.5	1856
C1k.7	12832158	0.02	10919364	5.87	1856
C1k.8	13974356	0.02	11846827	6.24	1856
C1k.9	13600810	0.02	11131986	7.01	1856
C3k.0	23020094	0.06	19633705	32.81	2888
C3k.1	23715019	0.06	19552645	30.31	2888
C3k.2	23513505	0.06	19965417	43.46	2888
C3k.3	23677592	0.07	20009915	38.28	2888
C3k.4	23064775	0.07	19245564	34.75	2888
C10k.0	39359535	0.22	34214912	200.62	6152
C10k.1	39668166	0.21	34171195	170.46	6152
C10k.2	40472631	0.22	34032594	258.33	6152
C31k.0	71956908	0.95	61483311	1879.4	16456
C31k.1	71407495	0.7	61063452	1906.26	16456
C100k.0	126394731	4.68	107743460	13035.13	49064
C100k.1	126792540	3.67	108880057	13485.18	49064
C316k.0	224475505	22.39	192699604	272859.56	152136

Table 3: Results for 1 run of the Stem-and-Cycle Algorithm on a Sun Enterprise 3500

Problem	Boruvka	Boruvka (Time)	S&C	S&C (Time)	Memory (KB)
dsj1000	22506843	0.02	18907610	10.49	1856
pr1002	299508	0.02	264541	8.14	1856
si1032	94318	0.11	92976	3.18	6040
u1060	261281	0.02	227715	6.76	1880
vm1084	293646	0.02	243862	6.11	1896
pcb1173	66832	0.02	57523	11.8	1936
d1291	59838	0.02	52155	6.05	1992
rl1304	284913	0.02	259570	11.57	2000
rl1323	313708	0.02	276727	9.88	2008
nrw1379	64900	0.02	57184	11.95	2032
fl1400	25010	0.03	20727	8.39	2048
u1432	183757	0.02	155027	9.24	2064
fl1577	26686	0.03	23075	8.1	2128
d1655	72842	0.03	63367	10.32	2168
vm1748	397359	0.03	340080	21.39	2216
u1817	64423	0.03	58052	10.92	2248
rl1889	365592	0.03	325051	15.71	2280
d2103	91147	0.03	81404	13.56	2384
u2152	76572	0.04	65402	13.01	2408
u2319	269736	0.04	235520	18.97	2488
pr2392	455906	0.04	385442	19.69	2520
pcb3038	161013	0.05	139347	28.29	2824
fl3795	34119	0.07	29104	45.36	3192
fnl4461	207506	0.08	184713	60.88	3504
rl5915	628608	0.11	575445	61.88	4200
rl5934	629855	0.11	566914	89.25	4208
pla7397	27717652	0.14	23617268	113.41	4912
rl11849	1043912	0.22	940448	231.2	7032
usa13509	23672435	0.26	20263933	253.4	7824
brd14051	541344	0.32	475771	306.9	8080
d15112	1803983	0.31	1594095	324.08	8592
d18512	737244	0.41	653586	456.33	10208
pla33810	77836523	0.72	66969596	1854.26	17504
pla85900	166304836	2.23	143996338	7118.59	42344

Table 4: Results for 1 run of the Stem-and-Cycle Algorithm on a Sun Enterprise 3500

[6] D. S. Johnson and L. A. McGeoch. *Local Search in Combinatorial Optimization*, chapter The Traveling Salesman Problem: A Case Study in Local Optimization, pages 215–310. John Wiley and Sons, Ltd., 1997.

[7] L. A. McGeoch M. L. Fredman, D. S. Johnson and G. Ostheimer. Data structures for traveling salesmen. *J. Algorithms*, 3:432–479, 1995.

[8] C. Rego. Relaxed tours and path ejections for the traveling salesman problem. *European J. Oper. Res.*, pages 522–538, 1998.