PAPER • OPEN ACCESS

New heuristic algorithm for traveling salesman problem

To cite this article: M L Shahab 2019 *J. Phys.: Conf. Ser.* **1218** 012038

View the article online for updates and enhancements.

New heuristic algorithm for traveling salesman problem

M L Shahab

Department of Mathematics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

E-mail: `luthfishahab@matematika.its.ac.id`

Abstract. Traveling salesman problem (TSP) is a basis for many bigger problems. If we can find an efficient method (that produce a good result in a short time) to solve the TSP, then we will also be able to solve many other problems. In this research, we proposed a new heuristic algorithm for TSP. We used 80 problems from TSPLIB to test the proposed heuristic algorithm. The proposed heuristic algorithm can find the best-known distance for 36 different TSPs. The average of all Goodness Value is 99.50%.

1. Introduction

A salesman wants to visit several different cities. He can starts his journey from any city, visit each other's cities one time, and then return to the first visited city. He wants to find the order of the city so that the traveled distance for the whole city is as small as possible. This kind of problem is often referred as the Traveling Salesman Problem (TSP).

TSP can be applied in many fields, including logistics (school bus routing, postal deliveries, meals on wheels, inspections), genome sequencing, scan chains, drilling problems, data clustering, etc [1]. TSP is a basis for many bigger problems. For example, in the Capacitated Vehicle Routing Problem (CVRP), if the customers for each vehicle have been properly divided, then the rest is to search the shortest travel route from the depot, visit each destination once, and then return to the depot (it is a TSP) [2, 3].

There are many algorithms that have been used to solve TSP. Exact algorithms can be used to solve TSP with small number of cities (less than 20 cities). The simplest algorithm is to try all possible routes and find the route with the shortest distance. The running time for this approach is $O(n!)$, the factorial of the number of cities .

The first exact algorithm that uses dynamic programming is Held-Karp algorithm [4]. Its running time is $O(n^2 2^n)$. Although at first glance this may appear to be a weak time bound, it is significantly less than the n factorial time it would take to enumerate all tours. In fact, this analysis of Held and Karp holds a place of honor in the TSP literature: it has the best time complexity for any known algorithm capable of solving all instances of the TSP [1].

Heuristic methods like cutting planes and branch and bound [5], can only optimally solve small problems whereas the heuristic methods, such as 2-opt, 3-opt [6], simulated annealing [7], tabu search, etc are good for large problems [8]. But many of these methods are difficult to apply. In this research, we proposed new simple heuristic algorithm for TSP.

IOP Conf. Series: Journal of Physics: Conf. Series **1218** (2019) 012038 doi:10.1088/1742-6596/1218/1/012038

2. Traveling Salesman Problem

Suppose that there are n cities in TSP, symbolized by $\mathbf{1}, \mathbf{2}, \ldots, \mathbf{n}$ (bold number), and the distance between $\mathbf{i}$ and $\mathbf{j}$ is stated as d_{ij}. In general, d_{ij} is calculated by

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{1}$$

where (x_i, y_i) is coordinate of $\mathbf{i}$ and (x_j, y_j) is coordinate of $\mathbf{j}$.

The solution (route) of TSP is $p_1 p_2 \ldots p_n$, which is a permutation of $\mathbf{1}, \mathbf{2}, \ldots, \mathbf{n}$. Distance from the solution is calculated by

$$D = \sum_{k=1}^{n} d_{p_k p_{k+1}} \tag{2}$$

where $p_{n+1} = p_1$. The purpose of TSP is to search for best $p_1 p_2 \ldots p_n$ so that the value of D is as small as possible. The following is an example of TSP and its solution.

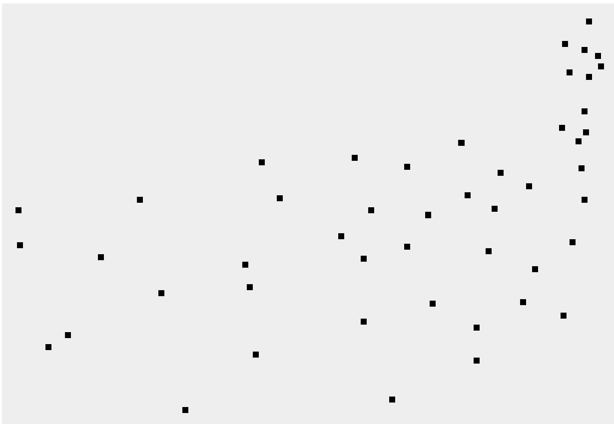

Figure 1. Example of TSP

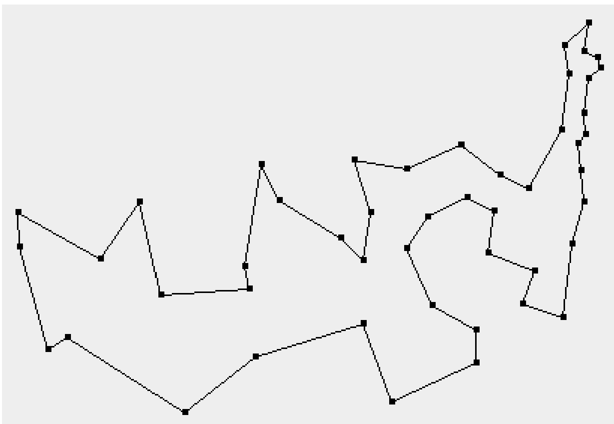

Figure 2. Example of TSP solution

IOP Conf. Series: Journal of Physics: Conf. Series **1218** (2019) 012038 doi:10.1088/1742-6596/1218/1/012038

3. TSPLIB

TSPLIB is a benchmark for TSP [9]. TSPLIB can be accessed online through http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html. There are around 110 different TSPs in the TSPLIB. The TSP is stored in different forms. The most common form is EUC_2D. Other forms found are ATT, GEO, LOWER_DIAG_ROW, UPPER_DIAG_ROW, UPPER_ROW, and FULL_MATRIX.

In [2], it can be seen how to get d_{ij}. But we feel that the explanation in [2] is difficult for readers to understand. Therefore, we provide an explanation of this.

3.1. EUC_2D

In TSP with EUC_2D form, there are n rows of $\{i, x_i, y_i\}$, where n is the number of cities and $i \in \{1, 2, \ldots, n\}$. d_{ij} is calculated by

$$d_{ij} = \left\lfloor \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + 0.5 \right\rfloor \tag{3}$$

where $\lfloor x \rfloor$ is floor function.

3.2. GEO

In TSP with GEO form, there are n rows of $\{i, x_i, y_i\}$, where n is the number of cities and $i \in \{1, 2, \ldots, n\}$. d_{ij} is calculated by

$$\begin{aligned}
latitude_i &= \frac{\pi(\lfloor x_i \rfloor + \frac{5}{3}(x_i - \lfloor x_i \rfloor))}{180} \\
longitude_i &= \frac{\pi(\lfloor y_i \rfloor + \frac{5}{3}(y_i - \lfloor y_i \rfloor))}{180} \\
latitude_j &= \frac{\pi(\lfloor x_j \rfloor + \frac{5}{3}(x_j - \lfloor x_j \rfloor))}{180} \\
longitude_j &= \frac{\pi(\lfloor y_j \rfloor + \frac{5}{3}(y_j - \lfloor y_j \rfloor))}{180} \\
q_1 &= \arccos(longitude_i - longitude_j) \\
q_2 &= \arccos(latitude_i - latitude_j) \\
q_3 &= \arccos(latitude_i + latitude_j) \\
d_{ij} &= \left\lfloor R \arccos\left(\frac{(1+q1)q2 - (1-q1)q3}{2}\right) + 1 \right\rfloor
\end{aligned} \tag{4}$$

where R is 6378.388.

3.3. ATT

In TSP with ATT form, there are n rows of $\{i, x_i, y_i\}$, where n is the number of cities and $i \in \{1, 2, \ldots, n\}$. d_{ij} is calculated by

$$d_{ij} = \left\lceil \sqrt{\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{10}} \right\rceil \tag{5}$$

where $\lceil x \rceil$ is ceil function.

IOP Conf. Series: Journal of Physics: Conf. Series **1218** (2019) 012038 doi:10.1088/1742-6596/1218/1/012038

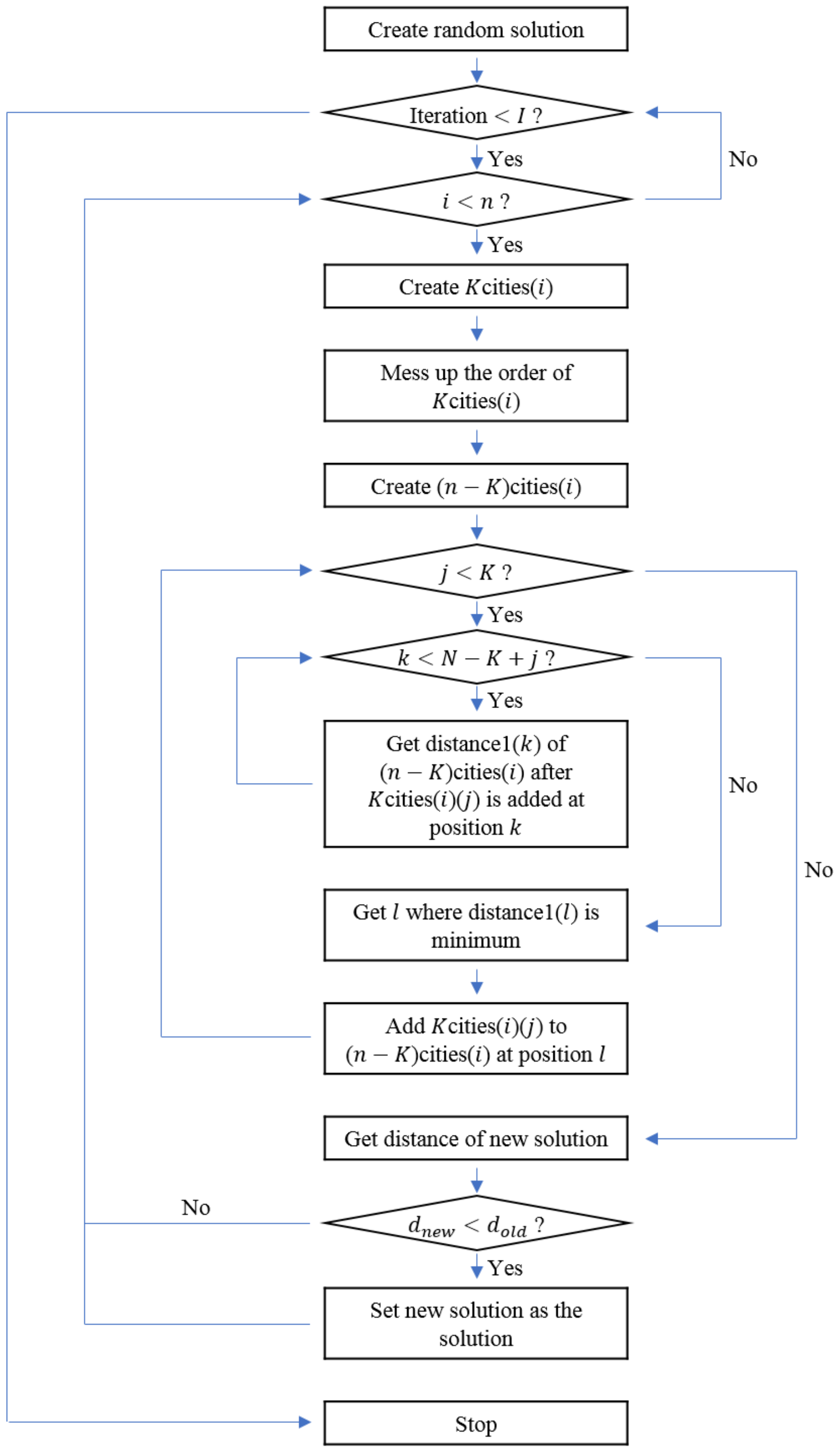

Figure 3. Proposed Heuristic Algorithm

IOP Conf. Series: Journal of Physics: Conf. Series **1218** (2019) 012038 doi:10.1088/1742-6596/1218/1/012038

4. Proposed Heuristic Algorithm

Suppose we will solve a TSP consisting of n cities. First, we make a random solution from it, for example $p_1p_2 \dots p_n$. From the random solution, we make Kcities(i) and $(n-K)$cities(i). Kcities(i) is $p_ip_{i+1} \dots p_{i+K-1}$ and $(n-K)$cities(i) is $p_1p_2 \dots p_{i-1}p_{i+K} \dots p_n$. Then we randomize the order of cities in Kcities(i), suppose that the new order is $q_iq_{i+1} \dots q_{i+K-1}$.

After we get $(n-K)$cities(i) and Kcities(i) with the new arrangement, we add cities in Kcities(i) to $(n-K)$cities(i). These cities are added according to the order in Kcities(i). The first city to be added is q_i.

If q_i is added to $(n-K)$cities(i) without changing the order of $(n-K)$cities(i), then q_i can be placed in $n-K$ different positions,

- $p_1q_ip_2 \dots p_{i-1}p_{i+K} \dots p_n$,
- $p_1p_2q_i \dots p_{i-1}p_{i+K} \dots p_n$,
- $\dots$,
- $p_1p_2 \dots q_ip_{i-1}p_{i+K} \dots p_n$,
- $p_1p_2 \dots p_{i-1}q_ip_{i+K} \dots p_n$,
- $p_1p_2 \dots p_{i-1}p_{i+K}q_i \dots p_n$,
- $\dots$,
- $p_1p_2 \dots p_{i-1}p_{i+K} \dots q_ip_n$, and
- $p_1p_2 \dots p_{i-1}p_{i+K} \dots p_nq_i$.

Next, we calculate the distance from all possible solutions and find the solution that has the smallest distance. If the smallest distance is owned by $p_1q_ip_2 \dots p_{i-1}p_{i+K} \dots p_n$, then we set $p_1q_ip_2 \dots p_{i-1}p_{i+K} \dots p_n$ as new $(n-K)$cities(i). Next, we set $q_{i+1} \dots q_{i+K-1}$ as new Kcities(i).

Next, we add q_{i+1} to the new $(n-K)$cities(i) in a similar way. And so on, up to Kcities(i) runs out and $(n-K)$cities(i) contains permutation from all cities.

After we get $(n-K)$cities(i) perfectly, we compare the distance from $(n-K)$cities(i) with the distance from the initial solution $p_1p_2 \dots p_n$. If the distance from $(n-K)$cities(i) is less than the distance from $p_1p_2 \dots p_n$, then we set $(n-K)$cities(i) as new $p_1p_2 \dots p_n$. If not, then $p_1p_2 \dots p_n$ does not change.

After we get the new $p_1p_2 \dots p_n$, we make new Kcities(i) and new $(n-K)$cities(i) and do the same thing as before. This is done for $i = 1, 2, \dots, n$.

If the steps have been carried out up to $i = n$, then the proposed heuristic algorithm has run one iteration. We can repeat the steps up to the wanted iteration limit. Steps from the proposed heuristic algorithm can be seen in Figure 3.

5. Results

In this research, we took 80 problems from TSPLIB. The number of cities varies, the smallest is 14 and the biggest is 1060. To simplify and speed up the computation time, we turn all existing forms into FULL_MATRIX form.

In this research, we used $K = \sqrt{n}$ and $I = 100$, where n is the number of cities in the TSP. For each problem, we run 100 times. From the obtained results, we take the best found distance and then show it. We also show the average computation time for one run. This test is done using the Java programming language, we use Netbeans IDE. The computer has an Intel I5 Processor and 4GB RAM.

In Table A1, we show the results of proposed heuristic algorithm. The first column is the problem number. The second column is the name of the TSP. The third column is the number of cities of TSP. The fourth column is best found distance by proposed heuristic algorithm. The fifth column is best known distance. The value of the best known distance is obtained from http://elib.zib.de/pub/mp-testdata/tsp/tsplib/stsp-sol.html. The sixth column is Goodness Value of best known distance. This value is calculated by

$$\left(1 - \frac{D - D_b}{D_b}\right) 100\% \tag{6}$$

where D is best found distance and D_b is best known distance. The seventh column is average computation time for one run. The computation time is in second.

From 80 tested TSPs, proposed heuristic algorithm can find the best known distance for 36 different TSPs. The average of all Goodness Value is 99.50%. Next, we show the convergence rate of u1060 in Figure 4. We show the distance after 1 iteration of proposed heuristic algorithm, after 2 iterations, and so on, up to 100 iterations.

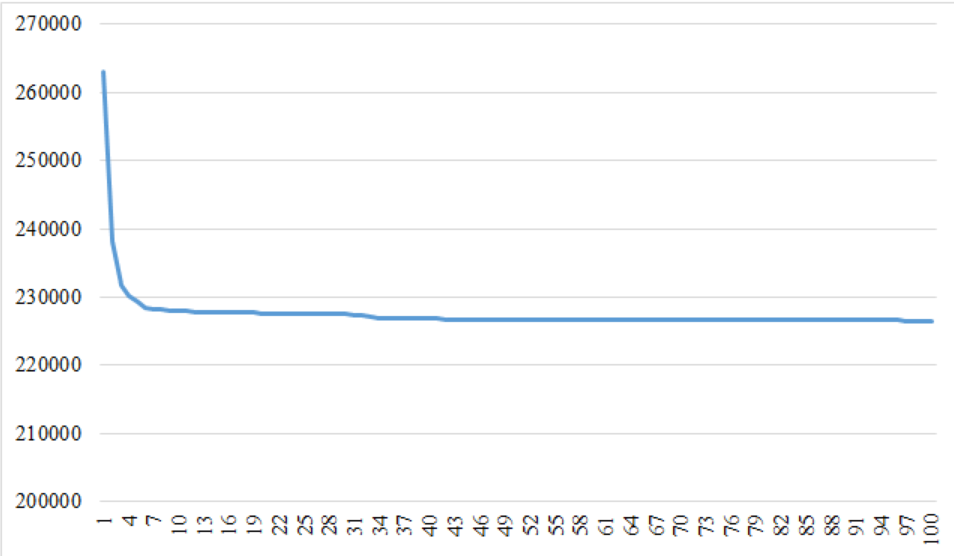

Figure 4. Convergence Rate of u1060

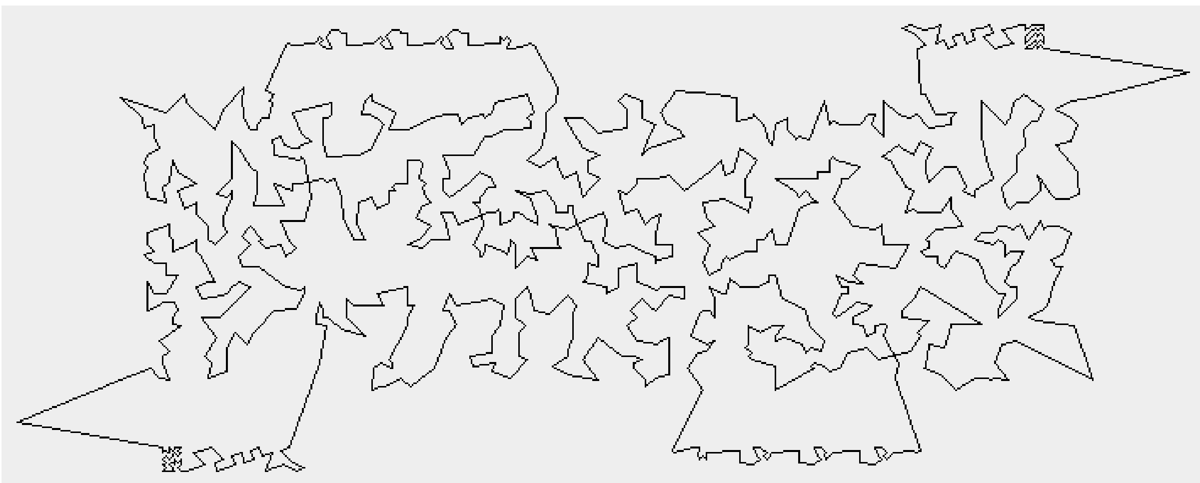

Figure 5. Example of u1060 Solution

IOP Conf. Series: Journal of Physics: Conf. Series **1218** (2019) 012038 doi:10.1088/1742-6596/1218/1/012038

We also show example of solution from proposed heuristic algorithm for u1060 in Figure 5. The distance for the solution is 228432. We can see the solution in Figure 6. It can be seen from the figure that the solution can still be improved by using 2-opt in 4 existing places.

6. Conclusions and Future Work

Proposed heuristic algorithm has a good ability to find solutions from TSP with a fairly short computation time.

Because proposed heuristic algorithm has a fairly short computation time, other researchers can use it to find initial values for other algorithm (such as genetic algorithm or other population-based meta-heuristics).

In this research, we choose K and I just by using assumption. The results of proposed heuristic algorithm depend on the value of K. Different problems have their own K values that are most suitable for them. Other researchers can research further about how to choose K that can provide better results.

References

[1] Applegate D L, Bixby R E, Chvatal V and Cook W J 2006 *The Traveling Salesman Problem: A Computational Study* (Princeton University Press)
[2] Taillard É 1993 *Networks* **23**(8) 661−673
[3] Shahab M L, Utomo D B and Irawan M I 2016 *Journal of Theoretical & Applied Information Technology* **87**(3) 461−468
[4] Held M and Karp R M 1962 *Journal of the Society for Industrial and Applied Mathematics* **10**(1) 196−210
[5] Padberg M and Rinaldi G 1987 *Operations Research Letters* **6**(1) 1−7
[6] Lin S and Kernighan B W 1973 *Operations Research* **21**(2) 498−516
[7] Kirkpatrick S, Gelatt C D and Vecchi M P 1983 *Science* **220**(4598) 671−680
[8] Bonyadi M R, Azghadi M R and Shah-Hosseini H 2008 *Traveling Salesman Problem* ed Greco F (InTech)
[9] Reinelt G 1991 *ORSA Journal on Computing* **3**(4) 376−384

Table A1. Results of Proposed Heuristic Algorithm

No	Name	Number of Cities	Best Found Distance	Best Known Distance	Goodness Value (in %)	Computation Time (in second)
1	burma14	14	3323	3323	100	0.0017
2	ulysses16	16	6859	6859	100	0.0017
3	gr17	17	2085	2085	100	0.0019
4	gr21	21	2707	2707	100	0.0025
5	ulysses22	22	7013	7013	100	0.0028
6	gr24	24	1272	1272	100	0.0032
7	fri26	26	937	937	100	0.0043
8	bayg29	29	1610	1610	100	0.0054
9	bays29	29	2020	2020	100	0.0051
10	dantzig42	42	699	699	100	0.0123
11	swiss42	42	1273	1273	100	0.0127
12	att48	48	10628	10628	100	0.0158
13	gr48	48	5046	5046	100	0.0161
14	hk48	48	11461	11461	100	0.0160
15	eil51	51	426	426	100	0.0200
16	berlin52	52	7542	7542	100	0.0210
17	brazil58	58	25395	25395	100	0.0258
18	st70	70	675	675	100	0.0413
19	eil76	76	542	538	99.26	0.0487
20	pr76	76	108159	108159	100	0.0493
21	gr96	96	55209	55209	100	0.0845
22	rat99	99	1212	1211	99.92	0.0925
23	kroA100	100	21282	21282	100	0.1029
24	kroB100	100	22199	22141	99.74	0.1027
25	kroC100	100	20749	20749	100	0.1078
26	kroD100	100	21294	21294	100	0.1108
27	kroE100	100	22068	22068	100	0.1080
28	rd100	100	7910	7910	100	0.1065
29	eil101	101	631	629	99.68	0.1104
30	lin105	105	14379	14379	100	0.1165
31	pr107	107	44303	44303	100	0.1225
32	gr120	120	6942	6942	100	0.1581
33	pr124	124	59030	59030	100	0.1835
34	bier127	127	118682	118282	99.66	0.1938
35	ch130	130	6148	6110	99.38	0.2046
36	pr136	136	96874	96772	99.89	0.2240
37	gr137	137	69980	69853	99.82	0.2248
38	pr144	144	58537	58537	100	0.2666
39	ch150	150	6553	6528	99.62	0.2943
40	kroA150	150	26584	26524	99.77	0.2960

continued

No	Name	Number of Cities	Best Found Distance	Best Known Distance	Goodness Value (in %)	Computation Time (in second)
41	kroB150	150	26132	26130	99.99	0.2954
42	pr152	152	73682	73682	100	0.3024
43	u159	159	42080	42080	100	0.3232
44	si175	175	21411	21407	99.98	0.4253
45	brg180	180	2060	1950	94.36	0.4640
46	rat195	195	2337	2323	99.40	0.5318
47	d198	198	15780	15780	100	0.5820
48	kroA200	200	29429	29368	99.79	0.5936
49	kroB200	200	29445	29437	99.97	0.5956
50	gr202	202	40363	40160	99.49	0.6004
51	ts225	225	127737	126643	99.14	0.8044
52	tsp225	225	3974	3919	98.60	0.8042
53	pr226	226	80369	80369	100	0.8000
54	gr229	229	135205	134602	99.55	0.8410
55	gil262	262	2396	2378	99.24	1.2276
56	pr264	264	49135	49135	100	1.2175
57	a280	280	2579	2579	100	1.3525
58	pr299	299	48266	48191	99.84	1.6927
59	lin318	318	42488	42029	98.91	1.8873
60	rd400	400	15468	15281	98.78	4.0263
61	fl417	417	11862	11861	99.99	4.3900
62	gr431	431	173545	171414	98.76	4.4100
63	pr439	439	108380	107217	98.92	4.9081
64	pcb442	442	51385	50778	98.80	5.2047
65	d493	493	35394	35002	98.88	6.7210
66	att532	532	27908	27686	99.20	8.4737
67	ali535	535	204738	202310	98.80	8.7127
68	si535	535	48553	48450	99.79	7.6809
69	pa561	561	2810	2763	98.30	9.3705
70	u574	574	37479	36905	98.44	9.7361
71	rat575	575	6909	6773	97.99	9.7940
72	p654	654	34653	34643	99.97	13.6819
73	d657	657	49651	48912	98.49	13.6994
74	gr666	666	298248	294358	98.68	14.3932
75	u724	724	42561	41910	98.45	16.8452
76	rat783	783	8972	8806	98.11	22.8752
77	dsj1000	1000	18934002	18659688	98.53	51.3816
78	pr1002	1002	263217	259045	98.39	47.4997
79	si1032	1032	93157	92650	99.45	54.4449
80	u1060	1060	227915	224094	98.29	57.7659