Benchmarking Algorithms for Dynamic Travelling Salesman Problems

Lishan Kang[1,2] **Aimin Zhou**[2] **Bob McKay**[3] **Yan Li**[4,1] **Zhuo Kang**[4]
[1]Department of Computer Science & Technology China University of Geosciences, Wuhan,P.R.China 430074
[2]State Key Lab of Software Engineering,Wuhan University, Wuhan, P.R.China 430072
[3]School of IT & EE University of New South Wales @ ADFA, Canberra, Australia 2600
[4]Computer Center Wuhan University Wuhan, P.R.China 430072
kang@whu.edu.cn, amzhou@hotmail.com, b.mckay@adfa.edu.au, llyyan2000@21cn.com, kang_zh@21cn.com

Abstract- Dynamic optimisation problems are becoming increasingly important; meanwhile, progress in optimisation techniques and in computational resources are permitting the development of effective systems for dynamic optimisation, resulting in a need for objective methods to evaluate and compare different techniques. The search for effective techniques may be seen as a multi-objective problem, trading off time complexity against effectiveness; hence benchmarks must be able to compare techniques across the Pareto front, not merely at a single point. We propose benchmarks for the Dynamic Travelling Salesman Problem, adapted from the CHN-144 benchmark of 144 Chinese cities for the static Travelling Salesman Problem. We provide an example of the use of the benchmark, and illustrate the information that can be gleaned from analysis of the algorithm performance on the benchmarks.

I. INTRODUCTION

With the evolution of computing environments from centralized computing through distributed computing to mobile computing, the confluence of Web services, peer-to-peer systems and grid computing provide the foundation for Internet distributed computing and mobile computing – allowing applications to scale from proximate Ad-hoc networks to planetary-scale distributed systems[3]. Recently, new features such as wire and wireless mixture, smartness, macro and micro-mobility, ultra-scalability, interoperability and invisibility have been added to the world of computing and communications [6,7]. Key research challenges they commonly face are the optimization of dynamic networking, arising from network planning and design, load-balance routing and traffic management [3,14,15]. They lead to a very important theoretical mathematical model: the Dynamic Travelling Salesman Problem (D-TSP).

A wide variety of algorithms have been proposed for the dynamic TSP, such as ant algorithms[9,10,11,12], competitive algorithms(on-line algorithms)[13] and dynamic inver-over evolutionary algorithms[8], and hence there arises a need to evaluate and compare them. This comparison, however, is not straightforward. In common with other optimisation problems, the search for good algorithms is a multi-objective problem, trading off algorithm performance against time complexity. However, unlike static optimisation problems, in dynamic optimisation problems the performance/complexity trade-off is not discretionary. In static optimisation problems, we can always wait longer for a solution. In dynamic optimisation problems, the trade-off between performance and time complexity is determined by the problem difficulty and the relationship between the rate of change and the available computing resources. So it is crucial that there be accurate evaluation of the performance of an algorithm at a given combination of problem complexity, and required computing resources relative to the rate of change. This in turn implies a need for benchmarks which permit a fine-grained study of the performance requirements of dynamic optimisation. In this paper, we propose a family of such benchmarks for D-TSP, and demonstrate the insights which may be obtained by their application.

The paper is organized as follows: we formally define the dynamic TSP in section 2, discuss the evaluation of solvers in section 3, and propose benchmarks in section 4. Section 5 discusses some experiments with the proposed benchmarks, while section 6 gives our conclusions.

II. DEFINING THE DYNAMIC TSP

The well-known travelling salesman problem (TSP) may be formally described as follows:
Given n cities $\{c_1, c_2, \dots, c_n\}$ and a cost(distance) matrix:

$$D = \{d_{ij}\}_{n\times n} \quad (1)$$

where d_{ij} is the cost(distance) from c_i to c_j, find a permutation $\pi = (\pi_1, \pi_2, \dots, \pi_n)$, such that:

$$\sum_{i=1}^{n} d_{\pi_i,\pi_{i+1}} = \min \quad (2)$$

where $\pi_{n+1} = \pi_1$.

Definition 1: A dynamic TSP(D-TSP) is a TSP determined by the dynamic cost (distance) matrix as follows:

$$D(t) = \{d_{ij}(t)\}_{n(t)\times n(t)} \quad (3)$$

where $d_{ij}(t)$ is the cost from city(node) c_i to city c_j, t is the real world time. This means that the number of cities $n(t)$ and the cost matrix are time-dependent.

But first, let us generalize the above to optimisation problems in general. An optimisation problem is defined over a space S of solutions; for any $s\in S$, there is a corresponding objective value $f(s)$. In a dynamic optimisation problem, the objective value is also a time-dependent function, $f(t,s)$.

Definition 2: A dynamic optimisation problem DP is an optimisation problem determined by a dynamic objective function $f(t,s)$, where the objective is to minimize $f(t,s)$ for all values of t.

However, this definition is just a theoretical model for mobile ad-hoc networking and similar applications. In practice, we need another approach.

For any continuous-time dynamic optimisation problem DP, there is a family of related discrete-time dynamic problems D*P with differing time windows. Can the D*P problems act as tracers for the corresponding DP problem – in other words, are the solutions always close to the solutions of the corresponding problem DP?

Formally, we may define a D*P as follows:

Let t_k, k = 0, 1, 2, … , m, t_0=0 and t_m= T be a sequence of discrete real world time sampling points.

Definition 3: D*P is a series of optimisation problems determined by the objective functions $f(t_k,s)$

k = 0, 1, 2, …. , m-1, with time windows [t_k, t_{k+1}], where $\{t_k\}_{i=0}^{m}$ is a sequence of real world time sampling points.

In particular, for the travelling salesman problem:

Definition 4: D*-TSP is a series of TSP determined by the cost matrix:

$$D(t_k) = \{d_{ij}(t_k)\}_{n(t_k)\times n(t_k)} \tag{4}$$

k = 0, 1, 2, …. , m-1, with time windows [t_k, t_{k+1}], where $\{t_k\}_{i=0}^{m}$ is a sequence of real world time sampling points.

From a practical perspective, there is an additional constraint: the objective function $f(t_k,s)$ should change relatively slowly with t_k; if the objective function changes too quickly for algorithms to track the solution, then there is no advantage in considering it as a dynamic optimisation problem. It is best treated as a sequence of independent static optimisation problems.

D-TSP and D*-TSP are certainly NP-Hard problems (since any static TSP may be straightforwardly converted into a D-TSP or D*-TSP with a time-invariant cost matrix).

There are many difficult open questions in the field of discrete-time dynamic optimization problems, because implicitly they involve a two-objective optimization problem. That is, D*P solvers should be designed as solutions of a two-objective optimization problem. One objective is to minimize the objective $f(t_k,s)$ at each time interval – in particular, for dynamic TSP, to minimize the length of the tour $\pi = (\pi_1, \pi_2, \dots, \pi_n(t_k))$:

$$d(\pi(t_k)) = \sum_{i=1}^{n(t_k)} d_{\pi_i,\pi_{i+1}}(t_k) \tag{5}$$

where $\pi_{n(t_k)+1} = \pi_1$.

The other is to minimize the size of the time interval :

$$s_k = t - t_k \tag{6}$$

where $t \geq t_k$.

The two-objective optimization problem has a set of Pareto optimal solvers. Different applications and users impose different biases that lead to different solvers being chosen. That is primarily why it is difficult to evaluate an algorithm for a dynamic optimisation problem. A second difficulty for evaluating an algorithm arises because the exact solution of a dynamic optimisation problem is usually unknown – there is limited value in designing an algorithm for problems with a known solution. This is a general issue for dynamic optimisation, but it arises particularly for dynamic TSP because of its NP-hardness. In the subsequent discussion, we consider only the dynamic TSP.

III. DESIGNING AND EVALUATING SOLVERS FOR D*-TSP

There are two methods for measuring the performance of D*-TSP solvers:

(1) Online Performance:

Denote the error

$$e(t_k)=d(\pi(t_k))-d(\pi^*(t_k)) \tag{7}$$

where $d(\pi^*(t_k))$ is the length of tour $\pi^*(t_k)$, which is the exact solution of the D*-TSP, and $\pi(t_k)$ is the approximate tour got by the D*-TSP solver A in time interval [t_k , t_{k+1}].

The performance of a D*-TSP solver A is defined as:

$$e(A) = \frac{1}{T}\sum_{k=0}^{m-1} e(t_k)(t_{k+1} - t_k) \tag{8}$$

(2) Offline Performance:

Denote the error

$$\bar{e}(t_k) = d(\pi(t_k)) - d(\bar{\pi}(t_k)) \tag{9}$$

where $\bar{\pi}(t_k))$ is the best tour got by solver A without time restriction (we assume that algorithm A is good enough to solve the static TSP. The assumption is useful in practice for large scale TSP). Then the offline performance of D*-TSP solver A is defined as:

$$\bar{e}(A) = \frac{1}{T}\sum_{k=0}^{m-1} \bar{e}(t_k)(t_{k+1} - t_k) \tag{10}$$

IV. BENCHMARKS FOR D-TSP

If D-TSP is implicitly a multi-objective optimization problem, then benchmarking of algorithms implicitly requires evaluation not just at one point, but along the Pareto front. Hence we require not just one benchmark, but a whole series of benchmarks trading off problem complexity against computational requirements.

We propose a family of benchmarks based on the well-known CHN144 benchmark for static TSP, which uses the positions of 144 Chinese cities. The positions of the 144 cities:$(x_i, y_i), i = 1,2,...,144$ as specified in [4] are appended to this paper. In satellite communications, as described in the DARPA Airborne Communications Node Project [1], the network routing configuration problem naturally gives rise to a D-TSP, in which the land-based communications nodes are fixed, but the satellite-based nodes are dynamic, their orbits being described by the equation:

$$(X_k(t) - X_k)^2 + (Y_k(t) - Y_k)^2 = R_k^2 \quad (11)$$

k=1, 2,..., M(t), where $(X_k(t), X_k(t)), k = 1,2,...,M(t)$ are the positions of the satellites. The positions are dependent on the real time t, as is the number of satellites $M(t)$. In this case, both the D-TSP and D*-TSP are symmetric, that is $d_{ij}(t) = d_{ji}(t)$. The parameters of the problems are m, $M(t_i)$ for t_0 (= 0), t_1, t_2, ..., t_m (= T), R_k, and (X_k, Y_k) for $k = 1,2,...,M(t_i)$.

In general, the above problems are highly dynamic, and possibly too difficult as a benchmark for the current state of the art in dynamic optimisation. However if $M(t_i)=M$ $=constant$ and the time windows $[t_i, t_{i+1}]=\Delta t$ are equal, we obtain a simpler version of the problem highly suited to benchmarking the current state of dynamic TSP. In the proposed benchmark, CHN144+M, there are the original CHN144 cities, plus M satellites. The simplest case is just one satellite node; we take the center of the orbit as (X_1, Y_1) = (2531,1906) and the radius is R_1 = 2905 (see Fig.1).

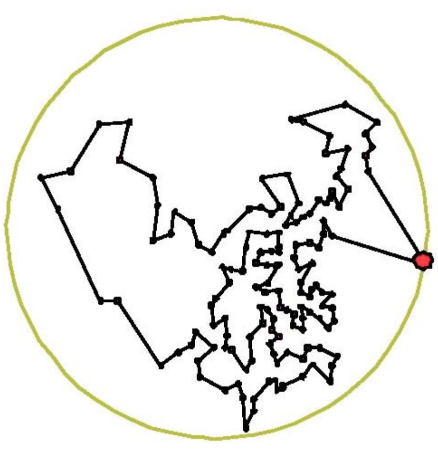

Figure 1: CHN144+1 Benchmark

More complex (and more realistic) CHN144+M problems can be constructed in 3D space, using city locations (x_i, y_i, z_i) satisfying:

$$(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2 = r^2 \quad (12)$$

where $(x_0, y_0, z_0)$is the center of the earth and r is the radius. The orbits of the satellites become:

$$(X_k(t) - X_k)^2 + (Y_k(t) - Y_k)^2 + (Z_k(t) - Z_k)^2 = R_k^2 \quad (13)$$

One can adjust the parameters to meet requirements for different dynamics, such as the frequency, severity and predictability of changes [2]. In addition, there are other advantages of the CHN144+M:

1. The best route ever known for CHN144 is given in the appendix A, and the length of the shortest route is 30353.8609965236, which can be used as a lower bound of CHN144+M for evaluating the global searching ability of the algorithms.

2. The dynamic behavior of the system of CHN144+M can be visualized easily. This is useful for examing the efficiency of the algorithms, as can be seen in figure 1, which is a screen-shot from a dynamic visualisation of CHN144+1.

V. EXPERIMENTS WITH CHN144+1

In this section, we perform some experiments using the CHN144+1 problem. Our aim with these experiments is to understand the behaviour of the the Dynamic Inver-Over Evolutionary Algorithm (DIOEA) [8] in an increasingly dynamic environment. DIOEA is used in all experiments. The environment for the experiments is an Intel P4 1.4GHz CPU with 256M RAM. We measure the offline error $\bar{e}$, together with:

Maximum error:

$$e_m = \max_{k=0,\cdots,m} \{\bar{e}(t_k)\} \quad (14)$$

Minimum error:

$$e_r = \min_{k=0,\cdots,m} \{\bar{e}(t_k)\} \quad (15)$$

Average error:

$$e_a = \frac{1}{m+1} \sum_{t=0}^{m} (\bar{e}(t_k)) \quad (16)$$

A. Test 1: 40 Sampling Points in a Cycle

We start with a relatively easy test for DIOEA, in which the satellite orbits relatively slowly, the time for a single revolution (T_1) ranging between 8.0s and 40.0s. The optimisation algorithm is updated 40 times per revolution (i.e Δt ranges from 0.2s to 1.0s). Figure 2 shows the error curve for T_1 = 8.0s, and Table 1 shows the experimental results.

TABLE 1: ERROR FOR 40 SAMPLING POINTS (LESS DYNAMIC)

T_1	e_m	e_r	e_a	$\bar{e}$
8.0s	$3.22*10^4$	0	$6.33*10^2$	$6.33*10^2$
16.0s	$4.37*10^4$	0	$5.89*10^2$	$5.89*10^2$
24.0s	$3.67*10^4$	0	$6.83*10^2$	$6.83*10^2$
32.0s	$3.55*10^4$	0	$5.48*10^2$	$5.48*10^2$
40.0s	$3.74*10^4$	0	$6.02*10^2$	$6.02*10^2$

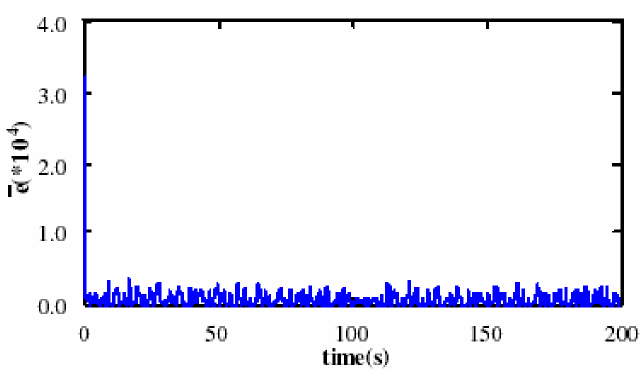

Figure 2 Error Curve for T_1 = 8:0*s*(*less dynamic*)

B. Test 2: 20 Sampling Points in a Cycle

We now sample an increasingly dynamic environment, with increased change between sample points. The optimisation is carried out 20 times per revolution, with T_1 ranging from 4.0s to 20.0s (ie Δt again ranges from 0.2s to 1.0s). Figure 3 shows the error curve for T_1 = 8.0s, and Table 2 shows the experimental results.

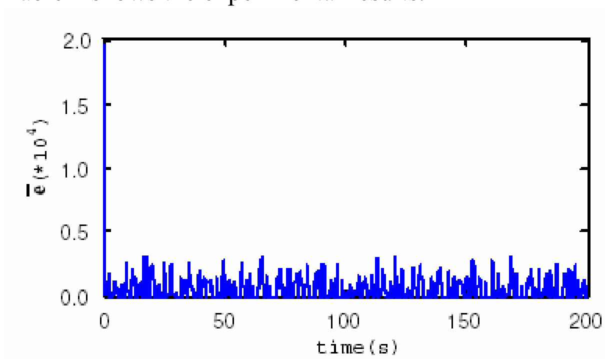

Figure 3: Error Curve for T1 = 8:0s(dynamic)

TABLE 2: ERRORS FOR 20 SAMPLING POINTS (DYNAMIC)

T_1	e_m	e_r	e_a	$\bar{e}$
4.0s	$1.87*10^4$	0	$5.07*10^2$	$5.07*10^2$
8.0s	$1.96*10^4$	0	$5.80*10^2$	$5.80*10^2$
12.0s	$1.41*10^4$	0	$6.39*10^2$	$6.39*10^2$
16.0s	$1.11*10^4$	0	$5.75*10^2$	$5.75*10^2$
20.0s	$1.56*10^4$	0	$5.88*10^2$	$5.88*10^2$

C. Test 3: 10 Sampling Points in a Cycle

Finally, we sample a highly dynamic environment, in which there are only 10 sample points per revolution, and T_1 ranges from 2.0s to 10.0s (ie Δt again ranges from 0.2s to 1.0s). Figure 4 shows the error curve for T_1 = 8.0s, and Table 3 shows the experimental results.

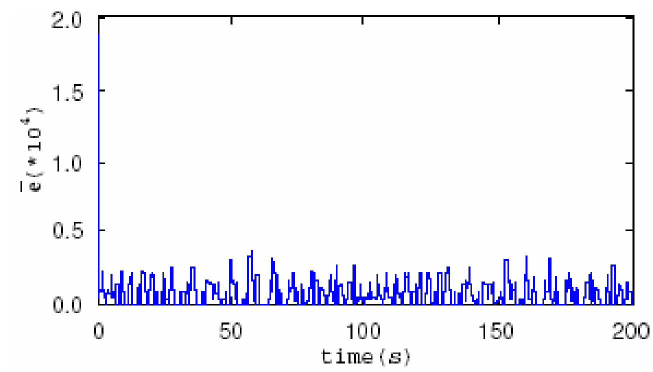

Figure 4: Error Curve for T_1 = 8:0*s*(*highly dynamic*)

TABLE 3: ERRORS FOR 10 SAMPLING POINTS (HIGHLY DYNAMIC)

T_1	e_m	e_r	e_a	$\bar{e}$
2.0s	$1.43*10^4$	0	$6.10*10^2$	$6.10*10^2$
4.0s	$1.32*10^4$	0	$5.20*10^2$	$5.20*10^2$
6.0s	$1.38*10^4$	0	$5.75*10^2$	$5.75*10^2$
8.0s	$1.87*10^4$	0	$6.77*10^2$	$6.77*10^2$
10.0s	$1.51*10^4$	0	$7.08*10^2$	$7.08*10^2$

D. Analysis

In each of the sets of experiments, we run the algorithm 10 times. All results are similar, the maximum errors are very large, but the minimum and average errors are quite acceptable, with the average relative errors being about 2%. The maximum errors all come from the beginnings of the runs, at time 0. That is, MIOEA has some difficulty in finding the optimum for the initial static TSP. However, once found, it can track the moving optimum in a dynamic TSP quite well. Paradoxically, the dynamic objective provides sufficient disturbance to move MIOEA away from the initial local optimum found, and track the global optimum. MIOEA actually performs better on the dynamic TSP than on a static TSP of equivalent complexity.

Equally important, MIOEA behaves relatively well as the frequency and severity of change increases, with a relatively smooth decline in performance.

VI. CONCLUSIONS

In this paper, we discussed the difficulties in providing benchmarks for dynamic optimisation problems, pointing out the implicitly multi-objective nature of the problem, and the concomitant requirement for a family of benchmarks able to compare algorithms along the Pareto front. We proposed one such family for the Dynamic Travelling Salesman Problem, the CHN144+M family of benchmarks, a family of problems of tunable difficulty. Taking one subfamily of the benchmarks, the CHN144+1 problems, we investigated the performance of a current algorithm, MIOEA, varying the frequency and severity of change. Using the benchmarks, we were able to gain an understanding of the behaviour of MIOEA, both of its overall behaviour, and of its performance degradation with frequency and severity of change.

ACKNOWLEDGEMENTS

This work was supported by National Natural Science Foundation of China (No.40275034 and No.60133010).

REFERENCES

[1] Airborne Communications Node at DARPA. www.darpa.mil/ato/programs/acn.htm.

[2] J.Branke.Evolutionary Optimization in Dynamic Environments.Norwell:Kluwer Academic Publishers,2002.

[3] D.W.Corne, M.J.Oates and G.D.Smith eds. Telecommunications Optimization: Heuristic and Adaptive Techniques. Chichester:John Wiley&Sons LTD,2000.

[4] L.S.Kang, Y.Xie,S.Y.You,etc. Nonnumerical Parallel Algorithms:Simulated Annealing Algorithm. Being:Science Press,1997.

[5] M.Milenkovic, S.H.Robinson, R.C.Knauerbase, etc. Toward Internet Distributed Computing. Computer,36(5),pp.38 46,2003.

[6] D.Saba and A.Mukberje. Pervasive Computing:A Paradigm for the 21th Century. Computer,36(3),pp.25 31,2003.

[7] Y.C.Tseng, C.C.Shen and W.T.Chen. Integrating Mobile IP with Ad Hoc Networks. Computer,36(5),pp.48 55,2003.

[8] A.M.Zhou, L.S.Kang and Z.Y.Yan. Solving Dynamic TSP with Evolutionary Approach in Real Time. Proceedings of the Congress on Evolutionary Computation, Canberra, Austrilia, 8-12, December 2003, IEEE Press, 951 – 957,2003

[9] C.J.Eyckelhof. Ant System for Dynamic Problem, the TSP Case – Ant Canght in a Traffic Jam. Master's thesis, Unversity of Twente, the Netherlands, August, 2001. (http://www.eyckelhof.nl)

[10] M.Guntsch, J.Branke, M.Middendorf and H.Schemck. ACO Strategies for Dynamic TSP. In M.Dorrigo et al.,editor, Abstract Proceedings of ANTS'2000, 59-62, 2000.

[11] M.Guntsch and M.Middendorf. Pheromone Modification Strategies for Ant Algorithms Applied to Dynamic TSP. in Applications of Evolutionary Computing, Lecture Notes in Computer Science, Vol.2037, 213-220, 2001

[12] M.Guntsch, M.Middendorf and H.Schemck. An Ant Colony Optimization Approach to Dynamic TSP. Proceedings of the GECCO 2001. San Francisco, USA, 7-11 July, 2001, Morgan Kaufmann, 860-867, 2001.

[13] G.Ausiello, E.Feuestein, S.Leonardi, L.Stougie, and M.Talamo. Algorithms for the On-line Traveling Salesman. Algorithmica, Vol. 29, No.4, 560-581, 2001.

[14] W.Power, P.Jaillet and A.Odoni. Stochastic and Dynamic Networks and Routing. Network Rougting, M.Ball, T.Maganti, C. Monma and G.Namhauser(eds.). North-Holland, Amsterdam, 1995.

[15] H.N.Psaraftis. Dynamic Vehicle Routing Problems. In Vehicle Routing: Methods and Studies, B.L.Golen and A.A.Assad(eds.), Elsevier Science Publishers, 223-248, 1988.

APPENDIX A: LOCATIONS OF CHN144

TABLE 4: LOCATIONS OF CHN144 CITIES

index	x	y	index	x	y	index	x	y	index	x	y
1	3639	1315	37	4634	654	73	4061	2370	109	3394	2643
2	4177	2244	38	4153	426	74	4207	2533	110	3402	2912
3	3712	1399	39	4784	279	75	4029	2498	111	3360	2792
4	3569	1438	40	2846	1951	76	4201	2397	112	3101	2721
5	3757	1187	41	2831	2099	77	4139	2615	113	3402	2510
6	3493	1696	42	3007	1970	78	3766	2364	114	3439	3201
7	3904	1289	43	3054	1710	79	3777	2095	115	3792	3156
8	3488	1535	44	3086	1516	80	3780	2212	116	3468	3018
9	3791	1339	45	1828	1210	81	3896	2443	117	3526	3263
10	3506	1221	46	2562	1756	82	3888	2261	118	3142	3421
11	3374	1750	47	2716	1924	83	3594	2900	119	3356	3212
12	3376	1306	48	2061	1277	84	3796	2499	120	3012	3394
13	3237	1764	49	2291	1403	85	3678	2463	121	3130	2973
14	3326	1556	50	2751	1559	86	3676	2578	122	3044	3081
15	3188	1881	51	2788	1491	87	3478	2705	123	2935	3240
16	3089	1251	52	2012	1552	88	3789	2620	124	2765	3321
17	3258	911	53	1779	1626	89	4029	2838	125	3140	3550
18	3814	261	54	2381	1676	90	3810	2969	126	3053	3739
19	3238	1229	55	682	825	91	3862	2839	127	2545	2357
20	3646	234	56	1478	267	92	3928	3029	128	2769	2492
21	3583	864	57	1777	892	93	4167	3206	129	2284	2803
22	4172	1125	58	518	1251	94	4263	2931	130	2611	2275
23	4089	1387	59	278	890	95	4186	3037	131	2348	2652
24	4297	1218	60	1064	284	96	3486	1755	132	2577	2574
25	4020	1142	61	1332	695	97	3492	1901	133	2860	2862
26	4196	1044	62	3715	1678	98	3322	1916	134	2778	2826
27	4116	1187	63	3688	1818	99	3334	2107	135	2592	2820
28	4095	626	64	4016	1715	100	3479	2198	136	2801	2700
29	4312	790	65	4181	1574	101	3429	1908	137	2126	2896
30	4252	882	66	3896	1656	102	3587	2417	138	2401	3164
31	4403	1022	67	4087	1546	103	3318	2408	139	2370	2975
32	4685	830	68	3929	1892	104	3176	2150	140	1890	3033
33	4386	570	69	3918	2179	105	3507	2376	141	1304	2312
34	4361	73	70	4062	2220	106	3296	2217	142	1084	2313
35	4720	557	71	3751	1945	107	3229	2367	143	3538	3298
36	4643	404	72	3972	2136	108	3264	2551	144	3470	3304

The best route ever known for CHN144 is(124 123 120 126 125 118 119 114 144 143 117 115 92 93 95 94 89 91 90 83 116 110 111 87 109 113 103 107 108 112 121 122 133 134 136 128 132 127 130 41 47 40 42 104 106 99 100 105 102 85 86 88 84 78 81 75 77 74 76 73 2 70 72 69 82 80 79 68 71 63 62 66 64 65 67 23 24 31 32 37 35 36 39 34 20 18 38 28 33 29 30 26 22 27 25 7 9 5 21 17 16 19 12 10 1 3 4 8 14 11 6 96 97 101 98 15 13 43 44 51 50 46 54 49 48 52 53 45 57 61 56 60 55 59 58 142 141 140 137 129 131 135 139 138) and the length is 30353.8609965236.