Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 854218, 8 pages
http://dx.doi.org/10.1155/2015/854218

Research Article

Solving Large-Scale TSP Using a Fast Wedging Insertion Partitioning Approach

Zuoyong Xiang,[1] Zhenyu Chen,[2] Xingyu Gao,[3] Xinjun Wang,[1] Fangchun Di,[2] Lixin Li,[2] Guangyi Liu,[2] and Yi Zhang[4]

[1]*School of Science, Central South University of Forestry and Technology, Changsha, Hunan 410004, China*
[2]*China Electric Power Research Institute, Beijing 100192, China*
[3]*Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China*
[4]*State Grid Fujian Electric Power Research Institute, Fuzhou, Fujian 350007, China*

Correspondence should be addressed to Zhenyu Chen; czy9907@gmail.com and Xingyu Gao; gxy9910@gmail.com

Received 10 November 2014; Accepted 18 May 2015

Academic Editor: P. Balasubramaniam

A new partitioning method, called Wedging Insertion, is proposed for solving large-scale symmetric Traveling Salesman Problem (TSP). The idea of our proposed algorithm is to cut a TSP tour into four segments by nodes' coordinate (not by rectangle, such as Strip, FRP, and Karp). Each node is located in one of their segments, which excludes four particular nodes, and each segment does not twist with other segments. After the partitioning process, this algorithm utilizes traditional construction method, that is, the insertion method, for each segment to improve the quality of tour, and then connects the starting node and the ending node of each segment to obtain the complete tour. In order to test the performance of our proposed algorithm, we conduct the experiments on various TSPLIB instances. The experimental results show that our proposed algorithm in this paper is more efficient for solving large-scale TSPs. Specifically, our approach is able to obviously reduce the time complexity for running the algorithm; meanwhile, it will lose only about 10% of the algorithm's performance.

1. Introduction

In the field of combinatorial optimization, the Traveling Salesman Problem (TSP) is probably the most famous and extensively studied problem, which aims to find the shortest Hamiltonian Cycle in a graph. This problem is NP-hard [1–19]: that is to say, no polynomial time algorithm is known for solving this problem at present.

The algorithms for solving the TSP can be categorized into two main paradigms: exact algorithms and heuristic algorithms. The exact algorithms are guaranteed to find the optimal solution in an exponential number of steps. The well-known exact algorithms are branch and cut algorithms [20], with which large TSP instances have been solved [21]. The major limitation of these algorithms is that they are quite complex and have heavy requirement of computing time [22]. For such reason, it is very difficult to find optimal solution for the TSP, especially for problems with a very large number of cities. Heuristic algorithms attempt to solve the Traveling Salesman Problem focusing on tour construction methods and tour improvement methods. Tour construction methods build up a solution step by step, while tour improvement methods start with an initial tour then try to transform it into a shortest tour.

In the past decades, a major technological breakthrough was obtained with the introduction of metaheuristics, such as simulated annealing, tabu search, genetic algorithms, ant colony optimization, particle swarm optimization, variable neighborhood search, and neural networks. These mentioned algorithms have the possibility to find their ways out of local optimization [23, 24].

Since there are so many algorithms for the solution, we have to evaluate the quality of each algorithm. A general way is to compare the obtained routine using the algorithm with

the optimal solution, through which we could judge the quality for the obtained solution; that is, the closer the better to the optimal solution. However, we do not know the optimal solution for the TSP instances in many cases, especially for the very large number of cities in the era of big data nowadays, with a variety of real-world applications ranging from multimedia [25–35] to mobile sensing [36–49]. Therefore, there will be an alternative and commonly used method: for each specific instance, we use a certain method to obtain the lower bound for the optimal solution, and then we evaluate the quality of the algorithm based on the lower bound, which is called Held and Karp bound (HK bound) [2–4].

There are lots of applications for the TSP problems in the real-world, also along with several variations of TSP. What we concern about is called Symmetric TSP (STSP), which is the type of most widely used TSP. In this kind of TSP, the distances between any two nodes are symmetric and equivalent: that is, $d(i, j) = d(j, i)$. Another one is Asymmetric TSP (ATSP), where $d(i, j) \neq d(j, i)$, while we would not discuss this kind of problem in this paper. One thing that should be mentioned is that we focus on geometric TSP in two-dimensional space, because at present the most widely applied one is this kind of TSP with geometric information, and a considerable portion of the algorithms are designed according to this kind of problem; accordingly our proposed algorithm in this paper is such kind of TSP.

The rest of this paper is organized as follows. In Section 2, we briefly introduce the construction methods. Section 3 briefly reviews several types of partitioning methods. In Section 4, we detail the first stage of our method. For the second stage the proposed algorithm is described in Section 5. Section 6 shows the experimental results, and we finally make the conclusion in Section 7.

2. Construction Method

There are a variety of solutions for TSP, which include several kinds of construction methods. The construction method is starting from a city, adding the unvisited cities into the tour constantly, and the process will be ended until all the cities have been inserted.

Insertion method is a typical kind of construction methods. Firstly, it selects a city randomly from all the cities and finds the nearest city. Furthermore, it inserts the unvisited city into the constructed sequence according to different rules. Finally, the method stops until all the cities have been inserted into the sequence. Insertion methods can be divided into these four categories by selected different rules: Nearest Insertion (NI), Cheapest Insertion (CI), Farthest Insertion (FI), and Random Insertion (RI). For the NI, it selects the nearest city for the next city to be inserted; for the CI, it chooses the next city with the shortest added distance after inserting; for the FI, it selects the farthest distance city from all nearest cities as the next one; for the RI, it chooses the inserted city randomly. Among these insertion methods, the performances of FI and RI are better than others, which averagely exceed about 15% compared to the HK bound, while the performances of NI and CI are a little worse with around 20%.

Spacefilling Curve [50] is another heuristics construction method for TSP. It firstly sorts the nodes in the order of their appearance along the spacefilling curve and then visits these nodes in the plane to make the short tour. The time complexity of this method is $O(N \log N)$, and the space complexity is $O(N)$.

3. Partitioning Method

When the scale of TSP becomes larger, in order to improve the running time or solve the memory limitation issue, it is necessary to utilize partitioning methods for solving large-scale TSP. The partitioning method firstly divides the cities to several segments. Secondly, it finds the shortest tour for each segment. Finally, the partitioning method connects each segment and obtains the final tour. There are two kinds of partitioning methods: one is Geometric Partitioning and the other is Tour-Based Partitioning [16–18].

Karp's Partitioning Heuristic (Karp) is proposed in [10]. For the construction of K-d tree, even though both of the subsets of cities contain the median city by the cut, through the median cities, we start to recursively divide the cities into parts using cuts from horizontal and vertical direction. When any set of the partition has no more than C cities, which is a setting parameter, the entire of recursion process will be stopped. By employing the DP (dynamic programming) approach proposed by Bellman [11], derived from the sets of cities, we cope with the subissues by optimal way in the final partition. At last, by utilizing the shared medians, we recursively fix the solutions along with fixed C, which is time-consuming parameter.

In Strip, firstly we partition the rectangle of minimal enclosure into $\sqrt{N/3}$ equal-width vertical strips, secondly for each strip order the cities by top-bottom way, and thirdly we obtain the tour leveraging the process running the strip from the leftmost to the rightmost, alternately traveling around by up-and-down way, with finally the edge being from the last city in rightmost strip to the first in leftmost strip. Obviously, we can see that Strip's tours could be as much as $\Omega(\sqrt{N})$ times optimum in the worst case. For equivalently distributed node ones in the unit of square, the anticipant Strip's tour length could be no more than $0.93\sqrt{N}$, and for these instances no more than $0.71\sqrt{N}$ over Held-Karp bound, which indicates that for these instances Strip's anticipant excess should be less than 31% [6–8].

Fast Recursive Partitioning (FRP) has been proposed in [9]. This begins with the rectangle of minimal enclosure; next recursively partition each rectangle, which includes more than 15 cities, into two rectangles with half of numerous cities approximately. If the rectangle of parent one is longer than it is wide, in the rectangle we could find the median 0-coordinate of cities and make a vertical partition at the x value; likewise, we could find the median y-coordinate and make a horizontal partition at the y value. Call the final rectangles, all including 15 or less cities, buckets. For all the buckets, we can construct nearest neighbor tours, which are then fixed along to make an overall tour.

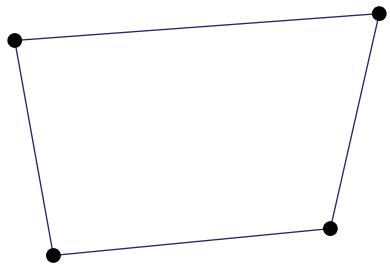


Figure 1: The tour of four nodes.

4. The First Stage—Partitioning

The first stage of the algorithm is to partition the cities by the rule of avoiding any cross for all the partitions.

4.1. The Simple Instance. As shown in Figure 1, we suppose the recent TSP instance includes only 4 cities, and any of the 3 cities are not in a straight line. The shortest way is always A-B-C-D-A, no matter how the distances between 2 cities change.

Now, we add one more city based on the 4 cities (we suppose that the added city would not replace any one from the mentioned A, B, C, and D cities; thus, it will not be any corner for the new bitmap). The order in tour for the added city will be related to its position. If it is above the line connecting between the upper left corner (city A) and the upper right corner (city B), there would be no doubt that the order is A-E-B-C-D-A; otherwise, it forms crosses. Similarly, if the city is on the right side of the line connecting between the upper right corner (city B) and the bottom right corner (city C), the order should be A-B-E-C-D-A. If the city happens to be below the line connecting between the bottom left corner (city D) and the bottom right corner (city C), the order will be A-B-C-E-D-A. If it is by the left side of the line connecting between the bottom left corner and the upper right corner, the order will be A-B-C-D-E-A. The above-mentioned situations are under the conditions that added 4 cities are all outside the convex. While the node could possibly fall in the convex, which obviously will make things a little bit complicated, we could confirm its position by a simple search and then insert it in the available tour order to make the added distance be the shortest of the whole circle. Figure 2 shows all the five possible situations.

While there are more than 5 cities, let us suppose there would be hundreds of or thousands of cities (we also suppose that the added city would not replace the position of any one from the mentioned A, B, C, and D cities), insertion proceeding would became complicated. However, we will always solve all the problems with the same method: putting all the added cities to the tour, while they are always located the two corners of the 4 ones, no matter how they would be inserted.

Let us explain it by another way: actually the whole tour is divided into four segments by four corner nodes. According to the clockwise direction, the first segment of the 4 segments starts from the upper left corner city and finally reaches the upper right corner city; the 2nd segment begins from the upper right corner city and ends at the bottom right corner city; the 3rd segment starts from the bottom right corner city and reaches the bottom left corner city finally; the 4th segment begins from the bottom left corner city and at last arrives to the starting city.

The conclusion works for any instance of TSP. It means any kind of TSP tour is combined by 4 segments, and we could get the final TSP circle by connecting the beginning node and the ending node of the four segments.

4.2. Wedging Insertion Partitioning Algorithm. The details of the proposed algorithm for wedging insertion partitioning are summarized as follows.

Step 1. Get the bitmap closest rectangle.

Step 2. Search for corner nodes.

Step 3. Judge all nodes which is inside node or outside node.

Step 4. Sort inside nodes and outsides.

Step 5. Obtain the four segments.

Step 6. Sort all preinsert nodes then insert them in turn.

Step 7. Determine the optimal segment and its position.

Let us suppose the bitmap to be solved including n cities. In the first step, obviously the time complexity is $O(n)$. In the second step, the time complexity is $O(n)$ for finding the corner nodes. In the third step, judging all nodes which is outside nodes, the time complexity is $O(n)$. In the fourth step, the time complexity is $O(n \lg n)$ if we run for fast-speed making sequence for the outside nodes and inside nodes, and the time complexity is $O(n \lg n)$. In the fifth step, the time complexity is the same as $O(n \lg n)$ if we run FI for making arranging for the inside nodes and outside nodes, respectively. In the 6th step and 7th step, first we need to find out the inserting position for each inside node, and the time complexity is $O(n^2 \lg n)$. After analyzing process above, we can get the time complexity which is $O(n^2 \lg n)$ for the whole algorithm. Due to the reason that it takes extra stage space only for the making sequence for the inside nodes and the outside nodes, we can get the result that the space complexity is $O(n)$ for the algorithm.

5. The Second Stage—Construction

During the first step of construction, the purpose for using the wedging insertion method is to reduce as much as possible crosses both between their segments and inside each segment; hence, we get the four segments and obtain the loop finally. However, it is obvious that the tour we have is not yet the optimal one. As we can see, too much peaks are formed inside the segments. Fortunately, there is no cross between the segments, we just need to reduce the peaks of each segments and then we could obtain an optimal result.

The reason for forming peaks is that we utilize the wedging-shape insertion method. During the process of wedging-shape insertion method, we only consider the relationships

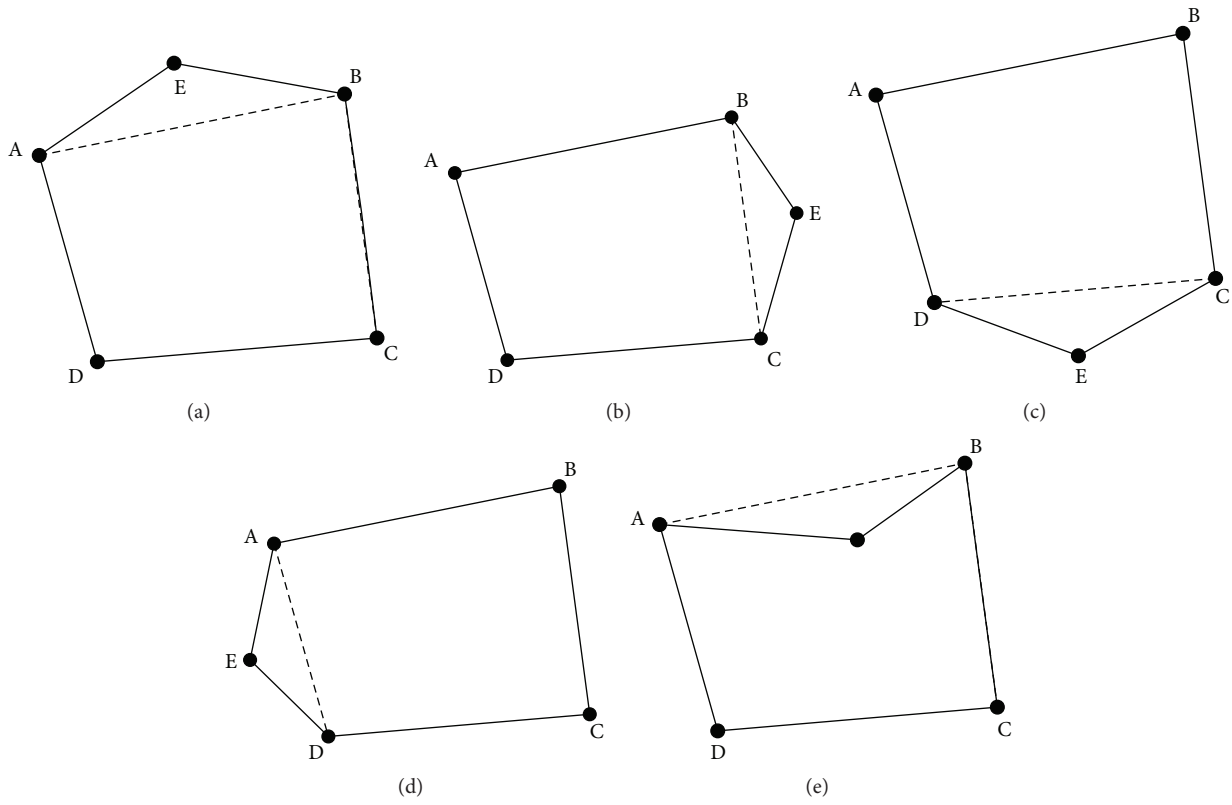


Figure 2: The tour of five nodes.

between the coordinates but without thinking of distance factors.

Therefore, we are inspired that we just need to change the sequence orders for the inside nodes of each segment; then the length could be optimized, so we only need to reconstruct each segment (the starting node and the ending node kept the same but only change other nodes' sequence position). We could use those abovementioned methods, while constructing each segment, different from that, and previously we search for the shortest tour, but now we look for the shortest route among the confirmed nodes' sequence, whose starting and ending nodes are known in advance.

Since the solution is the shortest route for each segment instead of the whole routine, we need to reform the known construction algorithms. For Inserting method as an example, since the starting node and the ending node are determined, we only need to select the nodes to insert among the starting node and the ending node by different methods (e.g., the Nearest Inserting method and the Farthest Inserting method) during the inserting process, until all the nodes in the segment have been inserted. Once the reconstruction for each segment is completed, we can achieve the final tour after we connect the starting node and the ending node of each segment.

6. Experiments

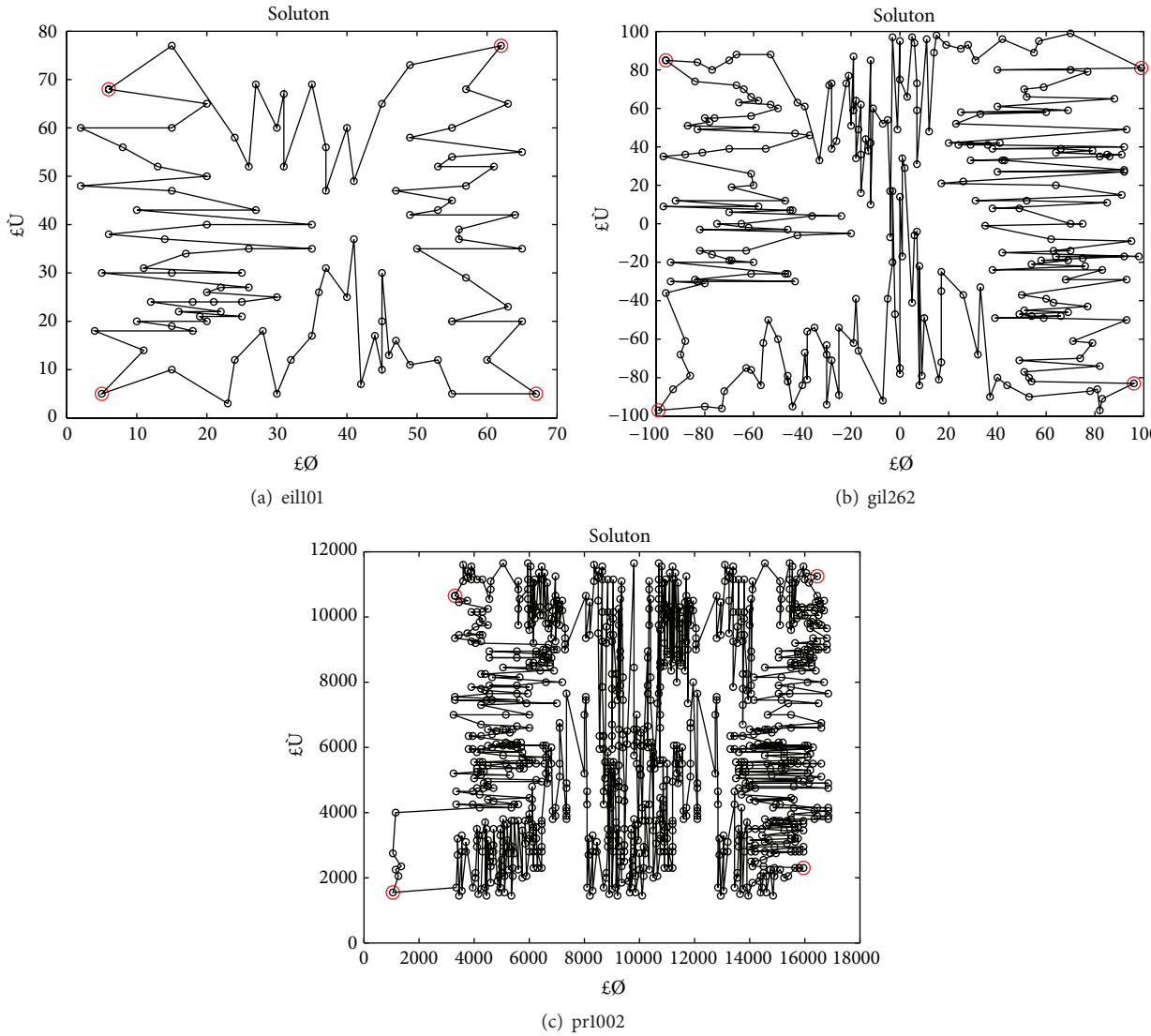
In this section, we present our experimental testbed and setup, and evaluation of our proposed algorithm and compared methods.

6.1. Experimental Testbed and Setup. We conduct the experiments which are carried out by MATLAB. The hardware platform is ASUS portable computer equipped with Intel Core Duo 2 CPU 2.2 G and Memory 2 G. The total number for testing instances is 12 from TSPLIB with more than 1000 cities. In the testing, we record the results of two time: one is the total time, including the time of loading data and calculating the distance matrix; another one is the practical running time.

6.2. Evaluation. First of all, we evaluate the partitioning tour for three instances, which is shown in Figure 3. From the results, we can draw the observation as follows. We obtain four segments in testing instances, and each segment does not cross with others and itself.

Furthermore, as shown in Table 1, we evaluate the running time of our proposed wedging insertion algorithm in comparison to four baselines: FRP, Strip, Spacefilling Curve, and Karp. We choose 12 instances from TSPLIB to test these algorithms, including more than 2000 cities. From the results, we can see that our proposed algorithm obtain the advantage on computational cost.

Besides, we evaluate the performance of our proposed wedging insertion algorithm with different insertion rules, that is, Wedging Insertion-Nearest Insertion (WI-NI), Wedging Insertion-Cheapest Insertion (WI-CI), Wedging Insertion-Farthest Insertion (WI-FI), and Wedging Insertion-Random Insertion (WI-RI). From the results in Table 2, we can observe that the maximum running result of our proposed



(a) eil101

(b) gil262

(c) pr1002

Figure 3: Partitioning tour for three instances.

Table 1: Running time (seconds) of Wedging Insertion, FRP, Strip, Spacefilling Curve, and Karp (15).

Instance	Wedging Insertion	FRP	Strip	Spacefilling Curve	Karp (15)
u2319	0.6	2.1	1.5	0.9	6.8
pr2392	0.6	2.2	1.8	0.8	6.9
pcb3038	1.3	2.6	2.6	1.5	20.5
fnl4461	1.3	5.2	2.8	1.7	21.34
rl5915	3.6	12.3	5.7	4.7	35.7
rl5934	3.1	13.2	6.2	4.8	37.3
pla7397	5.3	19.1	7.2	6.8	25.4
brd14051	10.7	36.2	16.7	13.5	279
d15112	18	56	22	20	85
d18512	19	62	23	24	220
pla33810	100	454	162	150	108
pla85900	708	2862	920	903	5002

Table 2: Running results for WI-NI, WI-CI, WI-FI, and WI-RI (average percent excess over the HK bound).

Instance	WI-NI	WI-CI	WI-FI	WI-RI
fl1400	19	17	15	16
fl1577	31	23	26	28
d2103	20	10	27	28
u2152	26	20	24	25
u2319	14	16	7	8
pr2392	38	34	29	30
pcb3038	27	22	21	20
fl3795	25	21	30	30
fnl4461	26	21	17	19
rl5915	33	27	28	28
rl5934	42	35	33	34

Table 3: Running results for Strip, FRP, Karp (15), and Karp (20) (average percent excess over the HK bound).

Cities	1000	3162	10 K	31 K	100 K
Strip (vertical)	61	36	73	91	73
FRP	68	56	61	76	66
Karp (15)	42	35	79	68	56
Karp (20)	33	26	64	54	56

algorithms with different insertion rules exceeds 42% than the HK Bound, and the minimum one exceeds 7% than the HK bound, while most of them are around 20%.

Finally, we evaluate the performance of the compared algorithms (Strip, FRP, and Karp) with Average Percent Excess over the HK Bound, as shown in Table 3.

From the results, we can draw the conclusion that, although there are differences for the running ranges of the algorithms, the performance of our proposed wedging insertion partitioning algorithm with four construction rules outperforms the compared algorithms for the same instance.

7. Conclusions

In this paper, we propose a novel partitioning approach to solve large-scale Traveling Salesman Problem (TSP). According to our experimental results, we can conclude that wedging insertion partitioning method is a kind of effective partitioning methods, it enables us to reduce and save the running time effectively when combining with other insertion construction methods, while it will lose only around 10% of the algorithm's performance. In the near further, our work can try to exploit parallel techniques [51–53] to accelerate the algorithm efficiency and further enhance the performance by fusing with other algorithms, such as 2-Opt, 3-Opt, LKH local search, or other construction algorithms.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by National Natural Science Foundation of China under Grant no. 11447034, National High-Tech Research and Development 863 Plan of China under Grant no. 2014AA051901, and Science and Technology Program of State Grid Corporation of China under Grant no. DZB17201400108.

References

[1] D. S. Johnson and L. A. McGeoch, "Experimental analysis of heuristics for the STSP," in *The Traveling Salesman Problem and Its Variations*, pp. 369–443, Berlin, Germany, Springer, 2007.

[2] M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees," *Operations Research*, vol. 18, pp. 1138–1162, 1970.

[3] K. H. Hansen and J. Krarup, "Improvements of the Held-Karp algorithm for the symmetric traveling-salesman problem," *Mathematical Programming*, vol. 7, no. 1, pp. 87–96, 1974.

[4] M. X. Goemans and D. J. Bertsimas, "Probabilistic analysis of the Held and Karp lower bound for the euclidean traveling salesman problem," *Mathematics of Operations Research*, vol. 16, no. 1, pp. 72–89, 1991.

[5] G. Gutin and A. P. Punnen, *The Traveling Salesman Problem and Its Variations*, vol. 12, Springer Science & Business Media, 2002.

[6] J. Beardwood, J. H. Halton, and J. M. Hammersley, "The shortest path through many points," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 55, no. 4, pp. 299–327, 1959.

[7] R. M. Karp and J. M. Steele, "Probabilistic analysis of heuristics," in *The Traveling Salesman Problem*, chapter 6, pp. 181–205, John Wiley & Sons, 1985.

[8] D. S. Johnson, L. A. McGeoch, and E. E. Rothberg, "Asymptotic experimental analysis for the held-karp traveling salesman bound," in *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 341–350, Atlanta, Ga, USA, January 1996.

[9] J. J. Bentley, "Fast algorithms for geometric traveling salesman problems," *ORSA Journal on Computing*, vol. 4, no. 4, pp. 387–411, 1992.

[10] R. M. Karp, "Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane," *Mathematics of Operations Research*, vol. 2, no. 3, pp. 209–224, 1977.

[11] R. Bellman, "Dynamic programming treatment of the travelling salesman problem," *Journal of the ACM*, vol. 9, no. 1, pp. 61–63, 1962.

[12] T. A. Feo and M. G. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[13] K. Holmqvist, A. Migdalas, and P. M. Pardalos, "Parallel continuous non-convex optimization," in *Parallel Computing in Optimization*, Appl. Optim., pp. 471–527, Springer, Berlin, Germany, 1997.

[14] K. Holmqvist, A. Migdalas, and P. M. Pardalos, "Parallelized heuristics for combinatorial search," in *Parallel Computing in Optimization*, pp. 269–294, Springer, 1997.

[15] M. G. C. Resende and C. C. Ribeiro, "Greedy randomized adaptive search procedures," in *Handbook of Metaheuristics*, pp. 219–249, Kluwer Academic, 2003.

[16] D. Applegate and W. Cook, "Solving large-scale matching problems," in *Network Flows and Matching: First DIMACS*

Implementation Challenge, pp. 557–576, American Mathematical Society, Boston, Mass, USA, 1993.

[17] J. R. Allwright and D. B. Carpenter, "A distributed implementation of simulated annealing for the travelling salesman problem," *Parallel Computing. Theory and Applications*, vol. 10, no. 3, pp. 335–338, 1989.

[18] M. G. A. Verhoeven, E. H. L. Aarts, and P. C. J. Swinkels, "A parallel 2-opt algorithm for the traveling salesman problem," *Future Generation Computer Systems*, vol. 11, no. 2, pp. 175–182, 1995.

[19] E. L. Lawler, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley-Interscience Series in Discrete Mathematics, Wiley-Interscience, 1985.

[20] G. Laporte, "The traveling salesman problem: an overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231–247, 1992.

[21] D. Applegate, W. Cook, and A. Rohe, "Chained lin-kernighan for large traveling salesman problems," *INFORMS Journal on Computing*, vol. 15, no. 1, pp. 82–92, 2003.

[22] K. Helsgaun, "An effective implementation of the Lin-Kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[23] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: a case study in local optimization," in *Local Search in Combinatorial Optimization*, vol. 1, pp. 215–310, John Wiley & Sons, 1997.

[24] F. Glover, M. Laguna, and R. Marti, "Scatter search and path relinking: advances and applications," in *Handbook of Metaheuristics*, pp. 1–35, Springer, 2003.

[25] X. Gao, Y. Chen, J. Liu, and J. Zhou, "Personalized 3D caricature generation based on PCA subspace," in *Advances in Multimedia Information Processing—PCM 2008*, vol. 5353 of *Lecture Notes in Computer Science*, pp. 713–720, Springer, Berlin, Germany, 2008.

[26] Z. Chen, J. Zhou, X. Gao, L. Li, and J. Liu, "A novel method for pencil drawing generation in non-photo-realistic rendering," in *Advances in Multimedia Information Processing—PCM 2008*, pp. 931–934, Springer, 2008.

[27] J. Liu, Y. Chen, C. Miao, X. Gao, J. Xie, and W. Gao, "Automatic 3D caricature generation by learning in enlarged manifold space," in *Proceedings of the 1st ACM SIGGRAPH Conference and Exhibition in Asia (SIGGRAPH Asia '08)*, ACM, Singapore, December 2008.

[28] J. Liu, Y. Chen, J. Xie, X. Gao, and W. Gao, "Semi-supervised learning of caricature pattern from manifold regularization," in *Proceedings of the 15th International Multimedia Modeling Conference (MMM '09)*, pp. 413–424, Sophia-Antipolis, France, January 2009.

[29] J. Xie, Y. Chen, J. Liu, C. Miao, and X. Gao, "Interactive 3D caricature generation based on double sampling," in *Proceedings of the 17th ACM International Conference on Multimedia (MM '09)*, pp. 745–748, ACM, Beijing, China, October 2009.

[30] X. Gao, Y. Chen, J. Zhou, J. Liu, C. Miao, and Z. Chen, "Genetic sampling in eigenspace for 3d caricature synthesis," in *Proceedings of the 7th International Conference on Information, Communications and Signal Processing (ICICS '09)*, pp. 1–4, IEEE, December 2009.

[31] J. Liu, Y. Chen, C. Miao et al., "Semi-supervised learning in reconstructed manifold space for 3D caricature generation," *Computer Graphics Forum*, vol. 28, no. 8, pp. 2104–2116, 2009.

[32] H. Zhang, Y. Liu, and Z. Ma, "Fusing inherent and external knowledge with nonlinear learning for cross-media retrieval," *Neurocomputing*, vol. 119, pp. 10–16, 2013.

[33] X. Gao, J. Cao, Z. Jin, X. Li, and J. Li, "GeSoDeck: a geo-social event detection and tracking system," in *Proceedings of the 21st ACM International Conference on Multimedia (MM '13)*, pp. 471–472, Catalunya, Spain, October 2013.

[34] X. Gao, J. Cao, Q. He, and J. Li, "A novel method for geographical social event detection in social media," in *Proceedings of the 5th International Conference on Internet Multimedia Computing and Service (ICIMCS '13)*, pp. 305–308, ACM, August 2013.

[35] X. Gao, S. C. H. Hoi, Y. Zhang, J. Wan, and J. Li, "Soml: sparse online metric learning with application to image retrieval," in *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, AAAI, Quebec City, Canada, July 2014.

[36] Z. Chen, Y. Chen, S. Wang, and Z. Zhao, "A supervised learning based semantic location extraction method using mobile phone data," in *Proceedings of the IEEE International Conference on Computer Science and Automation Engineering (CSAE '12)*, vol. 3, pp. 548–551, IEEE, Zhangjiajie, China, May 2012.

[37] Z. Zhao, Y. Chen, S. Wang, and Z. Chen, "Fallalarm: smart phone based fall detecting and positioning system," in *Proceedings of the 3rd International Conference on Ambient Systems, Networks and Technologies (ANT '12), the 9th International Conference on Mobile Web Information Systems (MobiWIS '12)*, pp. 617–624, Niagara Falls, Canada, August 2012.

[38] Y. Chen, Z. Chen, J. Liu, D. H. Hu, and Q. Yang, "Surrounding context and episode awareness using dynamic bluetooth data," in *Proceedings of the 14th International Conference on Ubiquitous Computing (UbiComp '12)*, pp. 629–630, September 2012.

[39] Y. Chen, Z. Zhao, S. Wang, and Z. Chen, "Extreme learning machine-based device displacement free activity recognition model," *Soft Computing*, vol. 16, no. 9, pp. 1617–1625, 2012.

[40] Z. Chen, "Mining individual behavior pattern based on significant locations and spatial trajectories," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops '12)*, pp. 540–541, Lugano, Switzerland, March 2012.

[41] Z. Chen, S. Wang, Y. Chen, Z. Zhao, and M. Lin, "InferLoc: calibration free based location inference for temporal and spatial fine-granularity magnitude," in *Proceedings of the 15th IEEE International Conference on Computational Science and Engineering (CSE '12)*, pp. 453–460, Nicosia, Cyprus, December 2012.

[42] S. Wang, Y. Chen, and Z. Chen, "Recognizing transportation mode on mobile phone using probability fusion of extreme learning machines," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 21, supplement 2, pp. 13–22, 2013.

[43] C.-W. You, N. D. Lane, F. Chen et al., "CarSafe app: alerting drowsy and distracted drivers using dual cameras on smartphones," in *Proceedings of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '13)*, pp. 13–26, ACM, Taipei, Taiwan, June 2013.

[44] Z. Chen, M. Lin, F. Chen et al., "Unobtrusive sleep monitoring using smartphones," in *Proceedings of the 7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops (PervasiveHealth '13)*, pp. 145–152, May 2013.

[45] Z. Chen, S. Wang, Z. Shen, Y. Chen, and Z. Zhao, "Online sequential ELM based transfer learning for transportation mode recognition," in *Proceedings of the 6th IEEE International*

Conference on Cybernetics and Intelligent Systems (CIS '13), pp. 78–83, IEEE, Manila, Philippines, November 2013.

[46] Z. Chen, Y. Chen, S. Wang, J. Liu, X. Gao, and A. T. Campbell, "Inferring social contextual behavior from bluetooth traces," in *Proceedings of the ACM Conference on Ubiquitous Computing (UbiComp '13)*, pp. 267–270, Zurich, Switzerland, September 2013.

[47] Z. Zhao, Z. Chen, Y. Chen, S. Wang, and H. Wang, "A class incremental extreme learning machine for activity recognition," *Cognitive Computation*, vol. 6, no. 3, pp. 423–431, 2014.

[48] Z. Chen, Y. Chen, L. Hu et al., "ContextSense: unobtrusive discovery of incremental social context using dynamic bluetooth data," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '14)*, pp. 23–26, ACM, Seattle, Wash, USA, September 2014.

[49] L. Hu, Y. Chen, S. Wang, and Z. Chen, "b-COELM: a fast, lightweight and accurate activity recognition model for mini-wearable devices," *Pervasive and Mobile Computing*, vol. 15, pp. 200–214, 2014.

[50] L. K. Platzman and I. Bartholdi, "Spacefilling curves and the planar travelling salesman problem," *Journal of the ACM*, vol. 36, no. 4, pp. 719–737, 1989.

[51] Y. Zhang, C. Yan, F. Dai, and Y. Ma, "Efficient parallel framework for H.264/AVC deblocking filter on many-core platform," *IEEE Transactions on Multimedia*, vol. 14, no. 3, pp. 510–524, 2012.

[52] C. Yan, Y. Zhang, J. Xu et al., "A highly parallel framework for HEVC coding unit partitioning tree decision on many-core processors," *IEEE Signal Processing Letters*, vol. 21, no. 5, pp. 573–576, 2014.

[53] C. Yan, Y. Zhang, J. Xu et al., "Efficient parallel framework for HEVC motion estimation on many-core processors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 12, pp. 2077–2089, 2014.