Heuristic Approaches for Minimum Weighted Latency Problem

Ziqi Wei[1, a] and Mike H.MacGregor[1, b]

[1]Department of Computing Science, University of Alberta, Edmonton, AlbertaT6G 2E8,Canada.

[a]ziqi.wei@ualberta.ca, [b]mike.macgregor@ualberta.ca

Abstract: This paper addresses the MWLP (Minimum Weighted Latency Problem). The problem has previously been proven NP-hard, which makes finding the exact solution impractical when the graph's scale is large. This paper explores the feasibility of using heuristics to solve MWLP. Five classic heuristic algorithms are tested in different situations. Different heuristics' effectiveness and efficiency are compared and analyzed.

Keywords: Minimum weighted latency problem, Heuristic algorithm, Ant colony optimization.

1. Introduction

MWLP was firstly proposed by Kousoupias[1] as GSP (Graph Searching Problem) in 1996. After that, the problem was renamed to MWLP by Wu[2] in 2000. A DP (dynamic programming) algorithm was given in the same paper as well. MWLP was proved to be NP-hard by Sitters[3] in 2002. In 2008, Liu[4] did research on TSP-WOCT (Traveling Sales-man Problem-Weighted Order Completion Times), which is a variant of MWLP. In some extreme situation, TSP-WOCT solves purely the same problem as MWLP does. In Liu's paper, he tested some of the classic heuristic algorithms including Local Search, TS (Tabu Search), GA (Genetic Algorithm) and ILS (Iterated Local Search).

The goal of MWLP is to minimize the sum of the latencies of the vertices multiplied by their weights. MWLP is formulated as follow: Given n vertices $v_1,...,v_n$ of graph G. MWLP asks for a tour T, which starts with a fixed vertex r and visiting all vertices of G. The sum of all vertices' arrival times $d_T(r, v)$ multiplied by their weights w(v) is minimum. The arrival time is defined as the distance traveled from r to v in T (first time visited). The weight w(v) is given in advance. It is presented as:

$$\min_{T} \sum_{v \in G} w(v) d_T(r, v) \tag{1}$$

Heuristic algorithms are designed by experience. They are able to give out feasible solutions for complex problems such as combined optimization or NP-hard problem with acceptable cost (time or space). Recent years, heuristics attracted more attentions in mathematical optimization and artificial intelligence area. Some classic heuristics such as GA, SA (Simulated Annealing), PSO (Particle Swarm Optimization), TS and ACO (Ant Colony Optimization) are widely studied. Nevertheless, because of the well-accepted "No Free Lunch Theorem", an algorithm's effectiveness and efficiency are unpredictable[5]. When facing a new problem, experiments are still needed to test a heuristic's effectiveness and efficiency. In this paper, we tested the five classic heuristics mentioned above for MWLP.

2. Experiment Settings

The heuristics we tested start searching from an initial solution gotten by a greedy algorithm which sorts the vertices by their weights in descending order. The manipulations to change the current solution in all the heuristics, such as cross-over in GA or intensification in TS, are 3-OPT. Different heuristics' own algorithm structures and parameters are set as below:

As for GA, the algorithm has a population of 100. The selection ratio and cross-over ratio are both set as 0.9, and the mutation probability is 0.05. The SA algorithm's initial temperature is set to 1000 degree. Its final temperature is 0.001 degree. The temperature drops to its 90% after every iteration. In our experiment, PSO evolves 1000 times with a population of 100. Considering the TS, the tabu

length is set to be $\left\lfloor \sqrt{(VertexNumber)^{2/2}} \right\rfloor$. 100 threads are searching in the searching space at the same time. ACO's ant population is 50. The pheromone concentration volatilizes to its 0.9 after each iteration.

3. Experiment Results and Analysis

The experiments were operated in Matlab and tested on a small-scale server. The server is equipped with two Inter(R) Xeon(R) E5-2670 v3 @ 2.30GHz CPUs and 64Gb memory.

The test data was obtained by randomly generating vertices in a 200×200 area. The distances between the vertices obey the triangle inequality, so the optimal solution has to be a Hamiltonian tour. The vertices' weights are allocated randomly. We tested MWLP instances with 20, 50 and 100 vertices, which are the representations of small-scale, mid-scale and big-scale problems. DP algorithm proposed by Wu is conducted as a control group for the small-scale instance. We did not run DP for the mid-scale and big-scale problems as it is very slow. In every problem instance, all five kinds of algorithms are tested by doing 30, 50 and 100 iterations respectively. The results are shown in Figure 1-3.

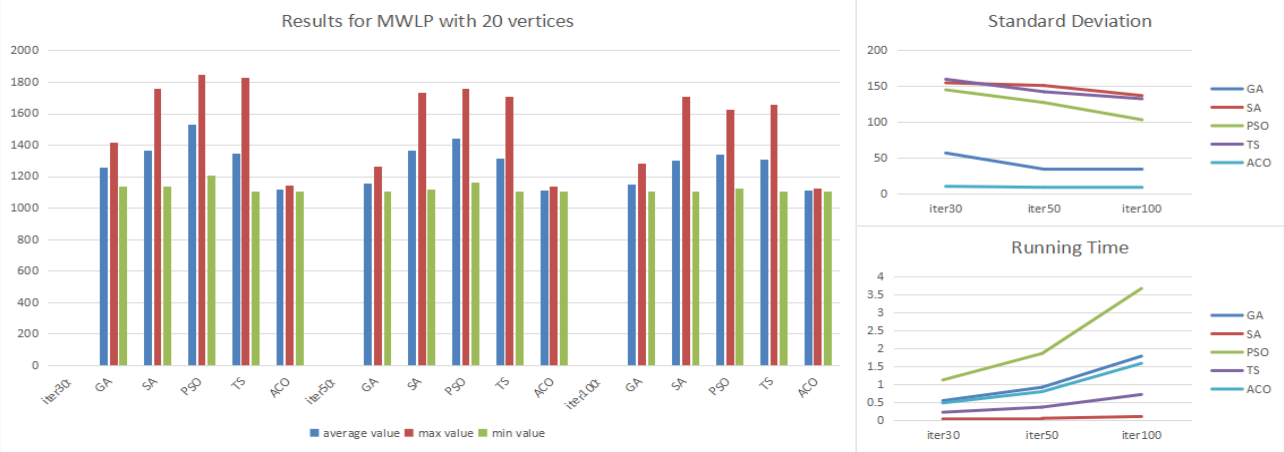

Fig. 1 Experiment Results for 20 Vertices Instance

The figures illustrate the average value, max value, min value, standard deviation and running time of different cases. We executed 100 experiments for every vertex and iteration number instance to avoid the contingency. The "average value" shows the average objective function values gotten by the 100 experiments. The "max value" and "min value" represents the maximal (worst cases) and minimum (best cases) objective function values. The "Running Time" illustrates the average execution time for one experiment and its unit is second.

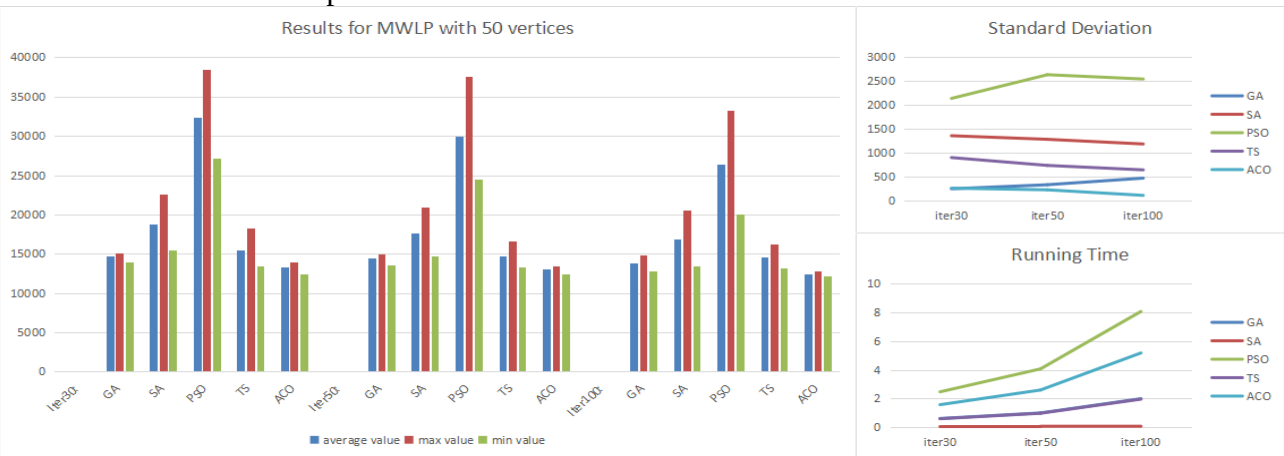

Fig. 2 Experiment Results for 50 Vertices Instance

We take the results in Figure 1 as an example to explain here. It shows the experiment results of MWLP with 20 vertices. DP was conducted for this instance, so we knew the global optima. It was

found in 9 cases out of the 15 cases (5 heuristics with 3 iteration numbers for each). However, even the global optima was found in more than half cases; PSO did not get it for even once. For the cases with 100 iterations, GA, SA, TS and ACO found the global optima. Out of the 100 experiments, they had 18, 1, 3 and 22 hits(found the global optima) respectively. Regarding the cases with 50 iterations, GA, TS and ACO found the global optima. Out of the 100 experiments, they found it 8, 2 and 17 times respectively. As for the case with 30 iterations, TS and ACO found the global optima and their hits number were 2 and 5 respectively. In all the three different iteration number cases, ACO got the minimal standard deviation value. In the aspect of the average value, max value and min value, ACO is still the dominant algorithm, while GA ranks the second.

As for the efficiency, it took the server 5.354e+05s to find the global optima by running DP method. On the contrary, ACO took only 1.584s for just one experiment when it iterated 100 times. SA consumed the shortest time in the five heuristics, which was only 9.899e-02 seconds. GA consumed almost the same time as ACO did. Comparing with DP method, all the five heuristic algorithms consumed affordable time. PSO consumed the longest time in this situation, which was almost 4 seconds for one experiment. Even though, it is still acceptable for most real-time applications.

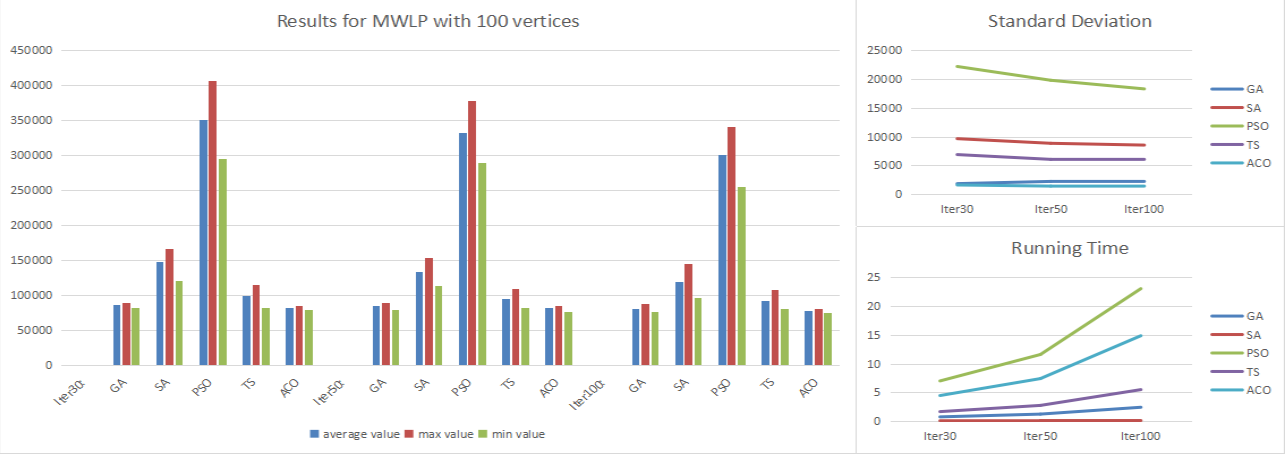

Fig. 3 Experiment Results for 100 Vertices Instance

Figure 1 and Figure 3 show the results for mid-scale and large-scale MWLP problems. Their experiment results are very similar to the small-scale instance. Based on these results, we can say that ACO performs the best within all the five algorithms and GA ranks the second. PSO is the last heuristic algorithm to choose for solving MWLP.

4.Theory Analysis

In this section, we will give a short analysis of the heuristics' efficiency and effectiveness.

4.1 Efficiency Analysis

We already know that ACO and GA are the two most effective heuristics. We wonder what ACO and GA's running-time patterns look like. Figure 4 illustrates the time consumed by ACO and GA when the vertex numbers are 12, 16, 20, 30, 50 and 100. Different lines represent the instances with different iteration numbers.

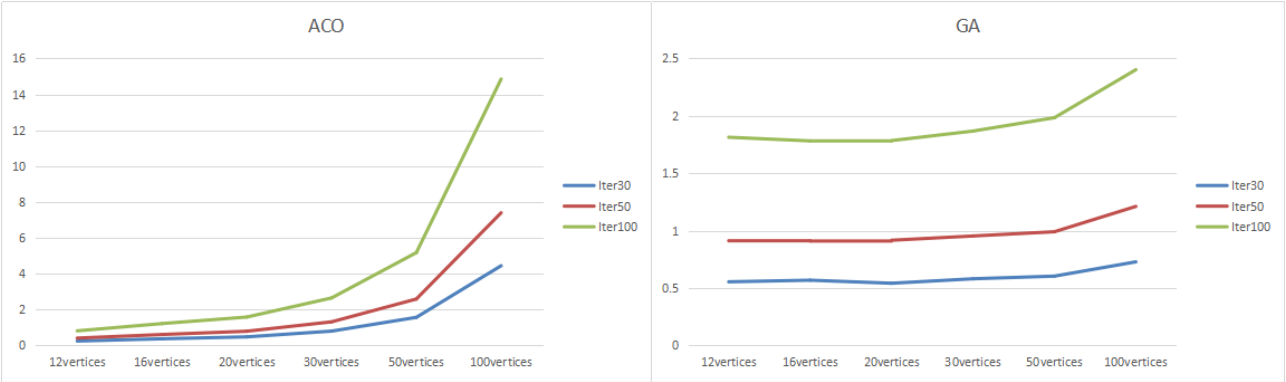

Fig. 4 Time Consumed by ACO and GA

For the same heuristic, instances with different iteration numbers' running-time curves are similar to each other. We choose the instances with 100 iterations as an example. Linear fittings are done for both algorithms' running-time curves. The results are shown in Figure 5. The R-squared values are only 0.1262 and 0.1516 respectively, which means both heuristics' running-time curves are non-linear.

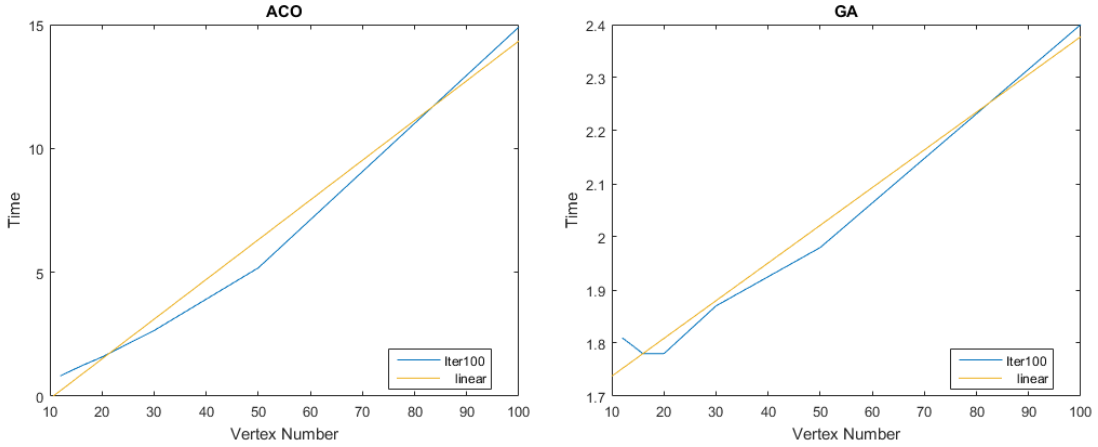

Fig. 5 Linear Fitting

4.2 Effectiveness Analysis

ACO got the best result for all the three problem scale instances. The reason may hide in its inner structure. When solving MWLP, the vertices with higher weights and the edges with shorter distances tend to be chosen earlier. In this aspect, the optimization objective is to find a balance between vertex weight and distance. That means a global thought of MWLP is important.

The ACO algorithm we used in experiment updates its pheromone data globally. After every iteration, the algorithm uses ten best global results to update its pheromone table. That means all the vertices are treated as a whole when updating and choosing. This mechanism is much different from the other algorithms. Take GA as an example. It does cross-over and mutation to find a new result. However, both manipulations update the result by changing only three vertices' positions in every iteration. It deals with only local vertices comparing to ACO.

To make this conclusion more credible, we designed an ACO who updates its pheromone table during every step but not globally. We call the new ACO AC_Local. It is tested for the mid-scale problem instance with 100 iterations. The results are shown in Figure 6. It can be seen that when the pheromone updating method changes, ACO's effectiveness drops.

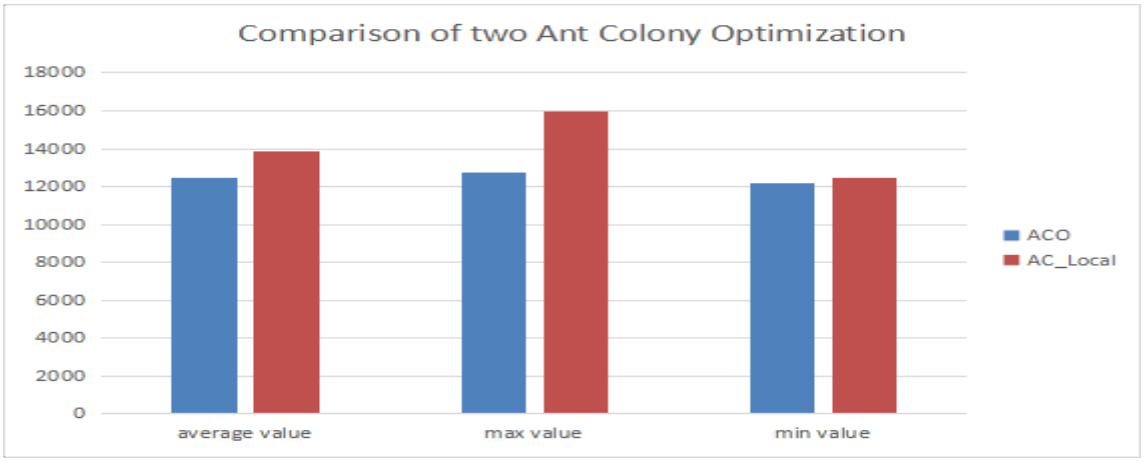

Fig. 6 Comparison Result for two ACOs

5. Conclusions and Future Work

In this paper, we used a variety of heuristic algorithms to solve WMWP. The experiment results show that the heuristic algorithms (GA, SA, PSO, TS and ACO) can give a feasible solution for this problem. Considering both effectiveness and efficiency, ACO is proved to be the most appropriate heuristic algorithm for MWLP. GA is the second best. PSO ranks the last.

All the heuristics ran in this paper are programmed in sequential. It is hard to code GA, SA and TS in parallel because of their own algorithm structure limits. However, PSO and ACO can be coded in parallel without many modifications. It is a way for the future work to improve ACO and PSO's efficiency, especially in the large-scale MWLP circumstances.

6. Acknowledgment

Ziqi Wei is supported by a grant from the China Scholarship Council.

7. References

[1] E. Koutsoupias, C. Papadimitriou, M. Yannakakis, Searching a fixed graph, Automata, Languages and Programming 1099 (1996) 280 – 289.

[2] B. Y. Wu, Polynomial time algorithms for some minimum latency problems, In-formation Processing Letters 75 (5) (2000) 225 – 229.

[3] R. Sitters, The minimum latency problem is np-hard for weighted trees, Lecture Notes in Computer Science 2337 (2002) 230 – 239.

[4] Y.S. Liu, Traveling saleman problem with order delivery, Master's thesis, National Chiao Tung University (2008).

[5] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67 – 82.