Time Bounds on Polynomial-space Exact Algorithms for TSP in Degree-5 and Degree-6 Graphs

Norhazwani Md Yunos Aleksandar Shurbevski Hiroshi Nagamochi

Department of Applied Mathematics and Physics, Kyoto University
{wanie, shurbevski, nag}@amp.i.kyoto-u.ac.jp

Abstract

This paper gives a correction to the time bounds on algorithms for TSP in degree-5 and degree-6 graphs. A polynomial-space algorithm has been designed by N. Md Yunos, A. Shurbevski and H. Nagamochi for edge-weighted undirected graphs with maximum degree 5 and degree 6, respectively in "A Polynomial-Space Exact Algorithm for TSP in Degree-5 Graphs," Technical Reports 2015 [2015-002], Department of Applied Mathematics and Physics, Kyoto University, 2015 and "A Polynomial-Space Exact Algorithm for TSP in Degree-6 Graphs," Technical Reports 2015 [2015-002], Department of Applied Mathematics and Physics, Kyoto University, 2015. These algorithms are branch-and-reduce algorithms that, given a degree-bounded graph with a set of forced edges, correctly deliver a minimum cost tour passing through all forced edges, and a correct set of branching vectors are derived to determine a best set of values to vertex weights in an analysis by the measure-and-conquer method, where one constraint on vertex weights was mistakenly unsatisfied with the weights claimed in these reports. After properly including the constraint, we have chosen the weight $w(\mathrm{f}i)$ of a degree-i forced vertex (a vertex to which a forced edge is incident) and the weight $w(\mathrm{u}i)$ of a degree-i unforced vertex as: $w(\mathrm{u5}) = 1$, $w(\mathrm{f5}) = 0.491764$, $w(\mathrm{u4}) = 0.700651$, $w(\mathrm{f4}) = 0.347458$, $w(\mathrm{u3}) = 0.322196$ and $w(\mathrm{f3}) = 0.183471$ for degree-5 graphs, which lead to a correct bound $O^*(2.4723^n)$ on the running time of the algorithm for an n-vertexed degree-5 graph, and $w(\mathrm{u6}) = 1$, $w(\mathrm{f6}) = 0.502801$, $w(\mathrm{u5}) = 0.815641$, $w(\mathrm{f5}) = 0.421871$, $w(\mathrm{u4}) = 0.580698$, $w(\mathrm{f4}) = 0.311647$, $w(\mathrm{u3}) = 0.262796$, and $w(\mathrm{f3}) = 0.149646$ for degree-6 graphs, which lead to a correct bound $O^*(3.0335^n)$ on the running time of the algorithm for an n-vertexed degree-6 graph.

Keywords: Traveling Salesman Problem, Exact Exponential Algorithm, Branch-and-reduce, Measure-and-conquer.

1 Introduction

Md Yunos et al. [4] recently presented a polynomial-space exact algorithm for TSP in degree-5 graphs, claiming an $O^*(2.4531^n)$ running time. Following this, Md Yunos et al. [5] also

[1] Technical Report 2015-004, October 29, 2015.

presented a polynomial-space exact algorithm for TSP in degree-6 graphs, with a claimed running time of $O^*(2.7467^n)$. Both of these time bounds are inaccurate due to a technical error at numerical calculations in a final stage of the analysis of their time complexities. The aim of this note is to present correct time bounds on TSP in degree-5 graphs and TSP in degree-6 graphs, while the correctness and the theoretical analysis of these algorithms remain the same as in the technical report of TPS in degree-5 graphs [4] and the technical report of TSP in degree-6 graphs [5].

The algorithms presented in both technical reports of TSP in degree-5 and degree-6 graphs [4, 5] are branch-and-reduce algorithms. The behavior of the algorithms is defined by respective sets of branching rules. It is common to illustrate the behavior of a branching algorithm as a search tree. The search tree is obtained by assigning the input instance of a problem as a root node, and recursively assigning children to a node for each smaller instance obtained by applying the branching rules. For a single node of the search tree, the algorithm takes time polynomial in the size of the node instance, which in turn, is smaller than or equal to the original instance size. Thus we can conclude that the running time of the branching algorithm is proportional up to a polynomial factor to the number of nodes of the search tree.

Let I be a given instance with size μ, and let I' and I'' be instances obtained from I by a branching operation. We use $T(\mu)$ to denote the maximum number of nodes in the search tree of an input of size μ when we execute the branching algorithm. Let a and b be the amounts of decrease in size of instances I' and I'', respectively; these values directly determine the performance of the algorithm [3]. Then we call (a, b) the *branching vector* of the branching rule, and this implies the linear recurrence:

$$T(\mu) \leq T(\mu - a) + T(\mu - b). \tag{1}$$

To evaluate the performance of this branching vector, a standard method for linear recurrence relations can be used. In fact, it is known that $T(\mu)$ is of the form $O(\tau^\mu)$, where τ is the unique positive real root of the function $f(x) = 1 - (x^{-a} + x^{-b})$ [3]. The value τ is called the *branching factor* (of a given branching vector), and the running time of the algorithm decreases with the value of this branching factor.

Md Yunos et al. [4, 5], following the approach of Eppstein [2], and Xiao and Nagamochi [6], in fact solve a slightly more general problem, named the *forced* TSP. An instance of the forced TSP is defined as the ordered pair (G, F) of a graph G and a subset F of edges of G, and it asks for a shortest Hamiltonian cycle which includes all edges in F. Edges in F are themselves called *forced* edges. With such a request in mind, it is obvious that vertices already incident to two forced edges can be excluded from further consideration [6, 4, 5]. Therefore, vertices with exactly one forced edge incident upon them are called forced vertices, and vertices with no incident forced edges are called unforced. With this regard, vertices can be distinguished and categorized by whether they are forced or unforced, and the total number of edges incident upon them. Henceforth, let fi (resp., ui) denote a forced (resp., unforced) vertex of degree i.

To effectively analyze the running time of the algorithms, Md Yunos et al. [4, 5] used the measure-and-conquer method [3]. The measure-and-conquer method allows for the running

time of an algorithm to be evaluated via a more general instance *measure*, as opposed to the straightforward instance size. Thereby, each type of vertex is assigned a different weight, and the measure of an instance becomes the sum of all vertex weights in that instance. Eppstein [1] showed how to analyze the running time of recursive branching algorithms by solving a quasiconvex optimization problem. Then all generated branching vectors will be the constraints in a suitable quasiconvex program, which can be easily solved to best values to vertex weights by a numerical calculation.

The branching rules of the algorithm due to Md Yunos et al. [4, 5] for TSP in degree-5 and degree-6 graphs are structured in such a way that the algorithms branch on edges incident to degree-k vertices according to a priority given by the branching rules. Following this technique, an input degree-k graph will eventually be reduced to an instance of a degree-$(k-1)$ graph. Then an existing algorithm for TSP in degree-$(k-1)$ graphs can be used as a black-box procedure to complete the algorithm for degree-k graphs. In the case of TSP in degree-5 graphs, Md Yunos et al. [4] used an algorithm for TSP in degree-4 graphs by Xiao and Nagamochi [6], whereas the algorithm for TSP in degree-6 graphs [5] uses the already established algorithm for TSP in degree 5 graphs [4]. Xiao and Nagamochi [7] have shown how to leverage the results obtained by a measure-and-conquer analysis and this call to an algorithm for TSP in degree-$(k-1)$ graphs. The only information necessary concerning the algorithm for TSP in degree-$(k-1)$ graphs is the vertex weights chosen for different types of vertices. Let w_t be the weight of vertex of type t, chosen in the analysis of the algorithm TSP in degree-k graphs, and let $\hat{w}_t$ be the weight of vertex of type t, chosen in the analysis of the algorithm TSP in degree-$(k-1)$ graphs. Let

$$\kappa = \max\{\frac{\hat{w}_t}{w_t} \mid t \text{ is a vertex type u}i \text{ or f}i, i = 3, 4, \ldots, k-1\}.$$

According to Xiao and Nagamochi [7, Lemma 3], if a call is made to the algorithm of running time $O^*(\tau_{k-1}^n)$ for an instance of size n, then it holds

$$T(\mu) \leq O^*(\tau_{k-1}^{\kappa}). \tag{2}$$

Finally, Eq. (2) will generate another constraint in the quasiconvex program used to determine the running time of an algorithm.

2 Problem Discussion

The algorithms reported in the technical report for TSP in degree-5 graphs and degree-6 graphs by Md Yunos et al. [4, 5] correctly deliver optimal tours, and a correct set of constraints for determining vertex weights to derive time bounds has been obtained. However

Md Yunos et al. [4] chose a vertex weight function for TSP in degree-5 graphs as follows:

$$\omega(v) = \begin{cases} 1 & \text{for a u5-vertex } v \\ 0.470059 & \text{for an f5-vertex } v \\ 0.607542 & \text{for a u4-vertex } v \\ 0.313373 & \text{for an f4-vertex } v \\ 0.276915 & \text{for a u3-vertex } v \\ 0.156687 & \text{for an f3-vertex } v \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

While for TSP in degree-6 graphs, Md Yunos et al. [5] chose a vertex weight function as follows:

$$\omega(v) = \begin{cases} 1 & \text{for a u6-vertex } v \\ 0.532091 & \text{for an f6-vertex } v \\ 0.458479 & \text{for a u5-vertex } v \\ 0.220838 & \text{for an f5-vertex } v \\ 0.333150 & \text{for a u4-vertex } v \\ 0.147225 & \text{for an f4-vertex } v \\ 0.155400 & \text{for a u3-vertex } v \\ 0.073612 & \text{for an f3-vertex } v \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

These vertex weight functions did not satisfy one of the constraint set, the switching condition. We then recompute a correct set of vertex weights for both degree-5 and degree-6 graphs as in the next section.

3 Result

Based on the new vertex weights, the following theorems from the technical report [4, 5] are now restated with the following correct time bounds.

Theorem 1 *The TSP in an n-vertex graph G with maximum degree 5 can be solved in $O^*(2.4723^n)$ time and polynomial space, by setting weights of each vertex as follows*

$$\omega(v) = \begin{cases} 1 & \textit{for a u5-vertex } v \\ 0.491764 & \textit{for an f5-vertex } v \\ 0.700651 & \textit{for a u4-vertex } v \\ 0.347458 & \textit{for an f4-vertex } v \\ 0.322196 & \textit{for a u3-vertex } v \\ 0.183471 & \textit{for an f3-vertex } v \\ 0 & \textit{otherwise.} \end{cases} \tag{5}$$

Theorem 2 *The TSP in an n-vertex graph G with maximum degree 6 can be solved in $O^*(3.0335^n)$ time and polynomial space, by setting weights of each vertex as follows*

$$\omega(v) = \begin{cases} 1 & \textit{for a u6-vertex } v \\ 0.502801 & \textit{for an f6-vertexv} \\ 0.815641 & \textit{for a u5-vertex } v \\ 0.421871 & \textit{for an f5-vertex } v \\ 0.580698 & \textit{for a u4-vertex } v \\ 0.311647 & \textit{for an f4-vertex } v \\ 0.262796 & \textit{for a u3-vertex } v \\ 0.149646 & \textit{for an f3-vertex } v \\ 0 & \textit{otherwise.} \end{cases} \tag{6}$$

Analytical analysis of the behavior of the branch-and-reduce algorithms to complete the proofs of Theorem 1 and Theorem 2 can be found in the concerned previous technical reports [4, 5].

References

[1] Eppstein, D.: Quasiconvex Analysis of Multivariate Recurrence Equations for Backtracking Algorithms. In: ACM Transactions on Algorithms, Vol. 2, No. 4, pp. 492-509, 2006.

[2] Eppstein, D.: The Traveling Salesman Problem for Cubic Graphs. In: Journal of Graph Algorithms and Application, Vol. 11, No. 1, pp. 61-81, 2007.

[3] Fomin, F. V. and Kratsch, D.: Exact Exponential Algorithms. Berlin Heidelberg: Springer, 2010.

[4] Md Yunos, N., Shurbevski, A. and Nagamochi, H.: A Polynomial-Space Exact Algorithm for TSP in Degree-5 Graphs. In: Technical Reports 2015 [2015-002], Department of Applied Mathematics and Physics, Kyoto University, 2015. Available at: `http://www.amp.i.kyoto-u.ac.jp/tecrep/ps_file/2015/2015-002.pdf`.

[5] Md Yunos, N., Shurbevski, A. and Nagamochi, H.: A Polynomial-Space Exact Algorithm for TSP in Degree-6 Graphs. In: Technical Reports 2015 [2015-003], Department of Applied Mathematics and Physics, Kyoto University, 2015. Available at: `http://www.amp.i.kyoto-u.ac.jp/tecrep/ps_file/2015/2015-003.pdf`.

[6] Xiao, M. and Nagamochi, H.: An Improved Exact Algorithm for TSP in Graphs of Maximum Degree 4. In: Theory of Computing Systems, DOI 10.1007/s00224-015-9612-x, pp. 1-32, 2015.

[7] Xiao, M. and Nagamochi, H.: Exact Algorithms for Maximum Independent Set. In: ISAAC 2013. LNCS 8283, pp. 328-338, 2013.