Journal of Computer and System Sciences 77 (2011) 1054–1057

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

A note on approximating the min–max vertex disjoint paths on directed acyclic graphs

Bang Ye Wu

Dept. of Computer Science and Information Engineering, National Chung Cheng University, ChiaYi, Taiwan 621, ROC

ARTICLE INFO

Article history:
Received 31 May 2010
Received in revised form 13 August 2010
Accepted 20 August 2010
Available online 25 August 2010

Keywords:
Approximation algorithm
Vertex disjoint paths
Rounding
FPTAS
Directed acyclic graph

ABSTRACT

This paper shows that the FPTAS for the min–max disjoint paths problem on directed acyclic graphs by Yu et al. (2010) [7] can be improved by a rounding and searching technique.

1. Introduction

Let $G = (V, E, w)$ be a directed acyclic graph (DAG), in which w is a non-negative integer weight on edges. For given $s, t \in V$ and fixed $k > 1$, the min–max k vertex disjoint paths (min–max k-VDP) problem looks for a set of k paths from s to t such that the paths are pairwise vertex-disjoint except for the endpoints s and t, and the maximum length of the paths is minimized, in which the length of a path is the total weight of the edges in the path. Yu et al. in [7] improved Li's result in [5] and gave a more efficient pseudo-polynomial time exact algorithm for the problem. They also improved the result in [2] and gave a fully polynomial time approximation scheme (FPTAS). Their FPTAS finds a $(1+\varepsilon)$-approximation in $O(n^{2k}(1/\varepsilon)^{k-1}\log^{k-1} W)$ time and $O(n^{2k-1}(1/\varepsilon)^{k-1}\log^{k-1} W)$ space, where $n = |V|$ and W is the maximum length of the optimal solution. This note shows that the rounding and searching techniques developed in [1,3,6] can be applied to this problem and improve the time and the space complexities to $O(n^{2k}(1/\varepsilon)^{k-1})$ and $O(n^{2k-1}(1/\varepsilon)^{k-1})$, respectively.

Using Li's approach, Yu et al. reduced the min–max k-VDP problem to a min–max multi-weighted path problem on an auxiliary DAG, and they obtained the improvement by giving a more efficient algorithm for the latter problem. Their FPTAS is also based on the reduction to the multi-weight problem. Since the instance of the multi-weight problem is a DAG with several weight functions on edges, so, in general, there is no obvious way to efficiently apply the rounding method. However, using their exact algorithm as a testing algorithm and applying the rounding scheme on the original graph, we can have a simpler and more efficient FPTAS. To make this note self contained, we detail the approach in the following.

E-mail address: bangye@cs.ccu.edu.tw.

0022-0000/$ – see front matter
doi:10.1016/j.jcss.2010.08.012

2. The efficient FPTAS

Let $\mathrm{OPT}(G)$ be the maximum weight of the paths in an optimal solution of the min–max k-VDP problem with input G. The next result comes from [7].

Lemma 1. *(See [7].) For a DAG $G=(V,E,w)$ and $\lambda \geqslant 0$, whether $\mathrm{OPT}(G) \leqslant \lambda$ can be determined in $O(n^{k+1}(\lambda+1)^{k-1})$ time and $O(n^k(\lambda+1)^{k-1})$ space. In the case that $\mathrm{OPT}(G) \leqslant \lambda$, an optimal solution can be constructed with the same time and space complexities.*

Remark 2. The time and space complexities appeared in [7] are $O(n^{k+1}\lambda^{k-1})$ and $O(n^k\lambda^{k-1})$, respectively. It is the same for $\lambda > 0$ but should be corrected as in Lemma 1 to include the case of $\lambda = 0$.

Note that by setting all edge weights to zero and $\lambda = 0$, Lemma 1 implies an algorithm for determining if there exists a feasible solution, and the algorithm runs in $O(n^{k+1})$ time and $O(n^k)$ space.

Let $w_1 < w_2 < \cdots < w_p$, $p \leqslant |E|$, be the distinct edge weights in G. Let $E_{\leqslant t} = \{e \in E \mid w(e) \leqslant t\}$ and $G_{\leqslant t} = (V, E_{\leqslant t})$. By binary search we can find the minimum $w^* \in \{w_i \mid 1 \leqslant i \leqslant p\}$ such that the graph $G_{\leqslant w^*}$ is feasible. By the definition of w^*, any optimal solution uses at least an edge of weight w^*, and therefore we have $w^* \leqslant \mathrm{OPT}(G)$. Furthermore, since a simple path has at most $n-1$ edges, we have $\mathrm{OPT}(G) \leqslant (n-1)w^*$. Note that if $w^* = 0$, we find the optimal 0; and if $G_{\leqslant w_p}$ is infeasible, there is no feasible solution. In both cases we solve the problem in $O(n^{k+1})$ time. In the following, we assume that both the cases are excluded.

Lemma 3. *In $O(n^{k+1}\log n)$ time and $O(n^k)$ space, we can determine an upper bound U_0 and a lower bound L_0 of $\mathrm{OPT}(G)$ such that $U_0 \leqslant nL_0$ and $L_0 > 0$.*

For any $r \geqslant 1$, let $w_{/r}(e) = \lfloor w(e)/r \rfloor$ and $G_{/r} = (V, E, w_{/r})$. The following algorithm is basically the same as the one in [1]. Note that the condition $\mathrm{OPT}(G_{/r}) \leqslant n/\delta$ in Step 3 of the test procedure can be computed by the exact algorithm mentioned in Lemma 1. To make the analysis easier, we slightly change the condition of the while loop (Step 2 in the algorithm Rounding-and-Searching).

Procedure Test(M, δ)
/* test if $\mathrm{OPT}(G) > M$ or $\mathrm{OPT}(G) \leqslant (1+\delta)M$ */
1: $r \leftarrow M\delta/n$;
2: construct $G_{/r}$;
3: if $\mathrm{OPT}(G_{/r}) \leqslant n/\delta$ then return "Yes";
4: else return "No".

Lemma 4. *If the test procedure returns "Yes", $\mathrm{OPT}(G) \leqslant (1+\delta)M$; and otherwise $\mathrm{OPT}(G) > M$. The time and space complexities are $O(n^{2k}\delta^{1-k})$ and $O(n^{2k-1}\delta^{1-k})$, respectively.*

Proof. First we show the correctness. Let $\mathcal{Q} = \{Q_i \mid 1 \leqslant i \leqslant k\}$ be a set of optimal paths of the problem, i.e., $\mathrm{OPT}(G) = \max_i\{w(Q_i)\}$, in which $w(Q_i)$ is the weight of path Q_i. If $\mathrm{OPT}(G_{/r}) \leqslant n/\delta$, there exist k paths P_i, $1 \leqslant i \leqslant k$, such that $\max_i\{w_{/r}(P_i)\} \leqslant n/\delta$. Since $rw_{/r}(e) \leqslant w(e) < r(w_{/r}(e)+1)$ for any $e \in E$ and any simple path has at most $n-1$ edges, $\max_i\{w(P_i)\} \leqslant rn/\delta + r(n-1)$. By $r = M\delta/n$, we have that $\max_i\{w(P_i)\} \leqslant M(1+\delta)$, which completes the proof of Yes-part. On the other hand, if $\mathrm{OPT}(G_{/r}) > n/\delta$, we have $\mathrm{OPT}(G) > rn/\delta = M$.

The time and the space complexities of the test procedure are dominated by Step 3, and the results directly come from Lemma 1 with $\lambda = n/\delta$. □

Algorithm Rounding-and-Searching
1: Set $U = U_0$ and $L = L_0$;
2: while $U > 4L$ do
3: $\quad\delta \leftarrow \sqrt{U/L} - 1$;
4: $\quad M \leftarrow \sqrt{UL/(1+\delta)}$;
5: $\quad$if Test(M, δ) returns "Yes" then
6: $\quad\quad U \leftarrow M(1+\delta)$;
7: $\quad$else
8: $\quad\quad L \leftarrow M$;
9: end while;
10: $r \leftarrow L\varepsilon/n$;
11: find and output $\mathrm{OPT}(G_{/r})$ by the exact algorithm mentioned in Lemma 1.

Lemma 5. *The algorithm* ROUNDING-AND-SEARCHING *computes a* $(1+\varepsilon)$*-approximation solution.*

Proof. The algorithm outputs $\mathrm{OPT}(G_{/r})$ with $r = L\varepsilon/n$ and $\mathrm{OPT}(G) \geqslant L$. Let P_i, $1 \leqslant i \leqslant k$, be the optimal paths with respect to $G_{/r}$. Then we have $\max_i\{w_{/r}(P_i)\} = \mathrm{OPT}(G_{/r})$. Similar to the proof of Lemma 4, we have $\max_i\{w(P_i)\} \leqslant r \cdot \mathrm{OPT}(G_{/r}) + r(n-1)$. Since $\mathrm{OPT}(G_{/r}) \leqslant \mathrm{OPT}(G)/r$ and $r = L\varepsilon/n$, we obtain

$$\max_i\{w(P_i)\} \leqslant \mathrm{OPT}(G) + L\varepsilon(n-1)/n < (1+\varepsilon)\,\mathrm{OPT}(G). \qquad \square$$

Lemma 6. *The algorithm* ROUNDING-AND-SEARCHING *takes* $O(n^{2k}(1/\varepsilon)^{k-1})$ *time and* $O(n^{2k-1}(1/\varepsilon)^{k-1})$ *space.*

Proof. The complexities for computing initial L_0 and U_0 are given in Lemma 3. Let U_i, L_i, δ_i and M_i, $0 \leqslant i \leqslant q$, be the parameters used in the i-th iteration of the while loop. When it leaves the loop and goto Step 10, $U_{q+1} \leqslant 4L_{q+1}$. Since the optimal is at most U_{q+1} and k is fixed, the time complexity of Step 11 is

$$O\left(n^{k+1}\left(U_{q+1}/(L_{q+1}\varepsilon/n)\right)^{k-1}\right) = O\left(n^{2k}(1/\varepsilon)^{k-1}\right).$$

Similarly the step takes $O(n^{2k-1}(1/\varepsilon)^{k-1})$ space. The space complexity can be easily analyzed since it is the maximum one in the iterations and Step 11. Since $4 < U_i/L_i \leqslant n$, $(1/\delta_i) < 1$, and the total space complexity is $O(n^{2k-1}(1/\varepsilon)^{k-1})$.

To show the total time complexity is $O(n^{2k}(1/\varepsilon)^{k-1})$, it is sufficient to show that

$$\sum_{i=0}^{q}(1/\delta_i)^{k-1} = O(1).$$

Since $1/\delta_i < 1$, we have $(1/\delta_i)^{k-1} \leqslant 1/\delta_i$ for any $k \geqslant 2$. Similar to the proof in [1] (seeing Appendix A), we can show that $\sum_{i=0}^{q}(1/\delta_i) = O(1)$, and so is $\sum_{i=0}^{q}(1/\delta_i)^{k-1}$. $\square$

By Lemmas 5 and 6, we summarize the result in the following theorem.

Theorem 7. *For any* $\varepsilon > 0$ *and fixed* $k > 1$*, a* $(1+\varepsilon)$*-approximation of the min–max k-VDP on a DAG can be found in* $O(n^{2k}(1/\varepsilon)^{k-1})$ *time and* $O(n^{2k-1}(1/\varepsilon)^{k-1})$ *space.*

Yu et al. [7] also improved the algorithm in [4] and obtained the following result: for an undirected planar graph G and nodes s, t adjacent to the unbounded face of G, there is an algorithm finding the min–max 2-VDP in $O(n^3W)$ time and $O(n^2W)$ space, in which W is the optimal weight. By the same approach as above, we can have the following FPTAS.

Corollary 8. *Given an undirected planar graph and two vertices adjacent to the unbounded face, for any* $\varepsilon > 0$*, the min–max* 2*-VDP can be* $(1+\varepsilon)$*-approximated in* $O(n^4/\varepsilon)$ *time and* $O(n^3/\varepsilon)$ *space.*

Acknowledgments

The author would like to thank the anonymous referees for their helpful comments. This work was supported in part by NSC 97-2221-E-194-064-MY3 and NSC 98-2221-E-194-027-MY3 from the National Science Council, Taiwan.

Appendix A

Proof of $\sum_{i=0}^{q}(1/\delta_i) = O(1)$**.** (*The proof is nothing but mimics the one in* [1].) At each iteration, we set either $U_{i+1} = M_i(1+\delta_i)$ or $L_{i+1} = M_i$. Since $\delta_i + 1 = \sqrt{U_i/L_i}$ and $M_i = \sqrt{U_iL_i/(1+\delta_i)} = U_i^{1/4}L_i^{3/4}$, for both cases we have

$$U_{i+1}/L_{i+1} = (U_i/L_i)^{3/4}. \tag{A.1}$$

By definition,

$$1/\delta_i = \frac{1}{\sqrt{U_i/L_i} - 1} = \frac{\sqrt{L_i/U_i}}{1 - \sqrt{L_i/U_i}} < 2\sqrt{L_i/U_i}$$

since $U_i > 4L_i$. Hence,

$$\sum_{i=0}^{q}(1/\delta_i) = O\left(\sum_{i=0}^{q}\sqrt{L_i/U_i}\right).$$

By (A.1), $\sqrt{L_i/U_i} = (L_q/U_q)^{(1/2)(4/3)^{q-i}}$, and then

$$\sum_{i=0}^{q}\sqrt{L_i/U_i} = \sum_{j=0}^{q}(L_q/U_q)^{(1/2)(4/3)^j} < \sum_{j=0}^{q}2^{-(4/3)^j}$$

since $U_q > 4L_q$. Using $2^{-(4/3)^{j+1}} = (2^{-1}\cdot 2^{-1/3})^{(4/3)^j} \leqslant (2^{-1/3})\cdot 2^{-(4/3)^j}$ repeatedly, we have $2^{-(4/3)^j} \leqslant (2^{-1/3})^j \cdot (2^{-1})$. Therefore

$$\sum_{j=0}^{q}2^{-(4/3)^j} < 2^{-1}\sum_{j=0}^{q}\left(2^{-1/3}\right)^j < \frac{2^{-1}}{1-2^{-1/3}} < 2.43. \qquad \square$$

References

[1] F. Ergun, R. Sinha, L. Zhang, An improved FPTAS for restricted shortest path, Inform. Process. Lett. 83 (2002) 287–291.

[2] R. Fleischer, Q. Ge, J. Li, H. Zhu, Efficient algorithms for k-disjoint paths problems on dags, in: Proceedings of the 3rd International Conference on Algorithmic Aspects in Information and Management, 2007, pp. 134–143.

[3] R. Hassin, Approximation schemes for the restricted shortest path problem, Math. Oper. Res. 17 (1) (1992) 36–42.

[4] H. van der Holst, J.C. de Pina, Length-bounded disjoint paths in planar graphs, Discrete Appl. Math. 120 (1–3) (2002) 251–261.

[5] C.-L. Li, S.T. McCormick, D. Simchi-Levi, The complexity of finding two disjoint paths with min–max objective function, Discrete Appl. Math. 26 (1) (1990) 105–115.

[6] D.H. Lorenz, D. Raz, A simple efficient approximation scheme for the restricted shortest path problem, Oper. Res. Lett. 28 (2001) 213–219.

[7] C.-C. Yu, C.-H. Lin, B.-F. Wang, Improved algorithms for finding length-bounded two vertex-disjoint paths in a planar graph and minmax k vertex-disjoint paths in a directed acyclic graph, J. Comput. System Sci. 76 (8) (2010) 697–708.