MINMAX STRONGLY CONNECTED SUBGRAPHS WITH NODE PENALTIES

ABRAHAM P. PUNNEN

Received 10 October 2002 and in revised form 25 July 2003

We propose an $O(\min\{m + n\log n, m\log^* n\})$ to find a minmax strongly connected spanning subgraph of a digraph with n nodes and m arcs. A generalization of this problem called the minmax strongly connected subgraph problem with node penalties is also considered. An $O(m\log n)$ algorithm is proposed to solve this general problem. We also discuss ways to improve the average complexity of this algorithm.

1. Introduction

Let G be a directed graph with node set $V(G)$ and arc set $E(G)$ such that $|V(G)| = n$ and $|E(G)| = m$. For each arc $e \in E(G)$, a cost c_e is prescribed. Let F be the family of all strongly connected spanning subgraphs of G. Then the *minsum strongly connected spanning subgraph problem* (Minsum SSP) is to find an $S \in F$ such that $\sum_{e\in S} c_e$ is minimized. A closely related problem, the *minmax strongly connected spanning subgraph problem* (Minmax SSP) is to find an $S \in F$ such that $\max\{c_e : e \in S\}$ is minimized.

The problem Minsum SSP has been studied by Frederickson and Ja'Ja' [4] and they proposed a 2-approximation algorithm to solve it. Since Hamiltonian cycle problem on a digraph is a special case of Minsum SSP, it is clearly NP-hard. Interestingly, we show that the Minmax SSP; however, can be solved in $O(\min\{m + n\log n, m\log^* n\})$ time, where $\log^* n$ is the iterative logarithm of n. Our algorithm is closely related to the 2-approximation algorithm of [4] for the Minsum SSP. Note that the optimal objective function value of Minmax SSP gives a lower bound on the optimal objective function value of the bottleneck traveling salesman problem (BTSP) [6] on a directed graph. Thus our algorithm can be used to compute lower bounds in enumerative algorithms for BTSP.

We also consider a generalization of the Minmax SSP, called the *minmax strongly connected subgraph problem with node penalties* (Minmax SNP). Let w_i be a prescribed weight for node $i \in V(G)$. For any digraph S, its node set is denoted by $V(S)$ and its arc set is denoted by $E(S)$. Let $F(\text{SNP})$ be the family of all strongly connected subgraphs of G.

Journal of Applied Mathematics and Decision Sciences 2005:2 (2005) 107–111
DOI: 10.1155/JAMDS.2005.107

Then the Minmax SNP can be defined as

$$\begin{aligned} &\text{Minimize } \max\{\max\{c_e : e \in S\}, \max\{w_i : i \notin V(S)\}\} \\ &\text{subject to } S \in F(\text{SNP}). \end{aligned} \tag{1.1}$$

If $w_i > \max\{c_e : e \in E(G)\}$ for all $i \in V(G)$, Minmax SNP reduces to Minmax SSP. Let A be a subset of $V(G)$. In the definition of Minmax SSP, if we replace "strongly connected spanning subgraphs" with "strongly connected subgraphs that spans A," we get the "Steiner version" of Minmax SSP. If $w_i > \max\{c_e : e \in E(G)\}$ for all $i \in A$ and $w_i < \min\{c_e : e \in E(G)\}$, Minmax SNP reduces to this Steiner version of Minmax SSP. Although Minmax SNP generalizes Minmax SSP and some of its variations, the algorithm we discussed for Minmax SSP does not seem to extend to this general problem. We propose an $O(m \log n)$ algorithm to solve Minmax SNP which is based on binary search over $\{c_e : e \in E(G)\}$ while keeping track of the contribution, if any, of $w_i's$ to the optimal objective function value. Operations that improve the average complexity of this algorithm are also discussed.

Minmax SNP can be used to obtain lower bounds to the optimal objective function value of the bottleneck prize collecting traveling salesman problem [2, 6].

2. The Minmax SSP

We first discuss our $O(\min\{m + n \log n, m \log^* n\})$ algorithm.

Algorithm 2.1. *(1) If G is not strongly connected, STOP. The problem is infeasible.*

(2) Fix a root node (say r) and find a bottleneck out-arborescence (arcs are directed away from r) rooted at r, say S_1^0.

(3) Find a bottleneck in-arborescence (arcs are directed towards r) rooted at r, say S_2^0.

(4) Output $S^0 = S_1^0 \cup S_2^0$.

Theorem 2.2. *Algorithm 2.1 solves* Minmax SSP *in $O(\min\{m + n \log n, m \log^* n\})$ time.*

Proof. Let $S^0 = S_1^0 \cup S_2^0$ be the output of Algorithm 2.1 and let S^* be an optimal solution to Minmax SSP. Clearly, S^0 is a strongly connected spanning subgraph of G and

$$\max\{c_e : e \in E(S^*)\} \leq \max\{c_e : e \in E(S^0)\}. \tag{2.1}$$

If possible, let

$$\max\{c_e : e \in E(S^*)\} < \max\{c_e : e \in E(S^0)\}. \tag{2.2}$$

Since S^* is strongly connected, there exist a spanning in-arborescence S_1^* and a spanning out-arborescence S_2^* of S^* rooted at r. Let $S^{**} = S_1^* \cup S_2^*$. Since S^{**} is also a strongly

connected spanning subgraph of G, from the optimality of S^*, we have

$$\max\{c_e : e \in E(S^*)\} = \max\{c_e : e \in E(S^{**})\}. \tag{2.3}$$

Thus in view of (2.2) and (2.3), either

$$\max\{c_e : e \in E(S_1^*)\} < \max\{c_e : e \in E(S_1^0)\} \tag{2.4}$$

or

$$\max\{c_e : e \in E(S_2^*)\} < \max\{c_e : e \in E(S_2^0)\}. \tag{2.5}$$

If (2.4) happens, we have a contradiction to the fact that S_1^0 is an optimal solution to the bottleneck out-arborescence problem rooted at r. If (2.5) happens, it contradicts the fact that S_2^0 is an optimal solution to the bottleneck in-arborescence problem rooted at r. This establishes the validity of the algorithm.

We now discuss the complexity. Camerini [3] showed that the bottleneck out-arborescence problem rooted at r can be solved in $O(m + n\log n)$ time. Gabow and Tarjan [5] showed that this problem can be solved in $O(m\log^* n)$ time. Thus the bottleneck out-arborescence problem can be solved in $O(\min\{m + n\log n, m\log^* n\})$ time. These algorithms can be easily modified to solve the corresponding bottleneck in-arborescence problem. Since strong connectivity of G can be tested in $O(m)$ time, the result follows. □

3. The Minmax SNP

We now develop an algorithm to solve the Minmax SNP. For any real number k, define the digraph $G(k)$ with $V(G(k)) = V(G)$ and $E(G(k)) = \{e \in E(G) : c_e \le k\}$. Let $Q(k) = \{i \in V(G) : w_i \ge k\}$. Let $G_1^k, G_2^k, \ldots, G_p^k$ be the strongly connected components of $G(k)$. If $G(k)$ is strongly connected, then $p = 1$.

Let $Z(G, C, W)$ be the optimal objective function value of the Minmax SNP on G with arc cost vector C and node weight vector W. Let C_i^k and W_i^k, respectively, be the restriction of C and W to the subgraph G_i^k of G. Let $Q(k) = \{i \in V(G) \mid w_i \ge k\}$.

Lemma 3.1. *If* $Q(k) \subseteq V(G_r^k)$ *for* $1 \le r \le p$, *then* $Z(G, C, W) = \max\{Z(G_r^k, C_r^k, W_r^k), \max\{w_i : i \in V(G) \setminus V(G_r^k)\}\}$.

Lemma 3.2. *If there does not exist* r *such that* $Q(k) \subseteq V(G_r^k)$ *and* $p > 1$, *then* $Z(G, C, W) > k$.

Lemma 3.3. *If* $\min\{w_i : i \in V(G)\} > \max\{e_e : e \in E(G)\}$, *then* Minmax SNP *is equivalent to* Minmax SSP.

The proofs of Lemmas 3.1, 3.2, and 3.3 are simple and we omit them. Using Lemmas 3.1, 3.2, and 3.3, Minmax SNP can be solved using binary search over $\{c_e : e \in E(G)\}$ while keeping track of the possibility of one of the w_i values becoming the optimal objective function value. The major difference between this algorithm and the standard threshold algorithm for bottleneck problems is the fact that we need to account for the node weights while performing binary search over the arc costs. A formal description of the algorithm is given below.

begin
$L = \min\{c_e : e \in E(G)\}; U = \max\{c_e : e \in E(G)\}$
If $\min\{w_i : i \in V(G)\} > U$ then solve the Minmax-SSP on G and output the solution.
STOP.
If $\max\{w_i : i \in V(G)\} < L$ then output a single node with maximum w_i value. STOP.
$\tilde{G} = G$, $\bar{w} = -\infty$
While $L < U$ do
begin
$R = \{c_e : L \le c_e \le U \text{ and } e \in G\}$
$k =$ median of R
Construct $G(k)$
$Q = \{i \in V(G) : w_i \ge k\}$
If there exists a strongly connected component G_r^k of $G(k)$ that contains Q then
$G = G_r^k$
$\bar{w} = \max\{\bar{w}, \max\{w_i : i \in V(G) \setminus V(G_r^k)\}\}$
If $\bar{w} \ge k$ then output G_r^k.
{* The objective function value of this solution is $\bar{w}$ *}
$U = k$
else
$L = k$
endif
endwhile
{* The optimal objective function value is L *}
Construct $\tilde{G}(L)$
{* All nodes i with $w_i > L$ will be in the same strongly connected component, say $\tilde{G}_r^L$, of $\tilde{G}(L)$ *}
Output $\tilde{G}_r^k$
end.

Algorithm 3.1.

Theorem 3.4. *Algorithm 3.1 solves* Minmax SNP *in* $O(m \log n)$ *time.*

Proof. The optimal objective function value of Minmax SNP is either an arc cost or a node weight. The algorithm performs binary search over the arc costs while keeping track of the possibility that a node weight becomes the objective function value. In view of Lemmas 3.1, 3.2, and 3.3, the validity of the algorithm is self-explanatory. The dominating complexity in each iteration of the algorithm is the median computation and testing strong connectivity of a digraph, and this can be done in $O(m)$ time [1]. Since k is chosen as the median of R, the algorithm terminates in $O(\log m) = O(\log n)$ iterations and the proof follows. □

In the above algorithm, if there is no r such that $Q \subseteq V(G_r^k)$, we update the lower bound L but considers the whole G in the next iteration. However, in this case, the size of the digraph for the next iteration may be reduced by collapsing the strongly connected components of $G(k)$ in G into single nodes and updating node weights and arc costs appropriately. However, this does not seem to reduce the worst-case complexity of the algorithm since the collapsed digraph may still have close to m arcs. In cases where the number of arcs in this collapsed digraph is at most a constant fraction of the number of arcs in G and this property holds through out the iterations, the complexity of the algorithm with this modification reduces to $O(m)$.

4. Conclusion

The Minsum SSP is known to be an NP-hard problem. However, unlike the Minsum SSP, we show that its minmax version, the Minmax SSP, can be solved efficiently in $O(\min\{m + n\log n, m\log^* n\})$. A generalization of the Minmax SSP is also considered and an $O(m\log n)$ algorithm is proposed to solve it. Our algorithms can be used to compute good lower bounds for the bottleneck traveling salesman problem and some of its generalizations.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley Series in Computer Science and Information Processing, Addison-Wesley, Massachusetts, 1975.

[2] E. Balas, *The prize collecting traveling salesman problem and its applications*, The Traveling Salesman Problem and Its Variations (G. Gutin and A. Punnen, eds.), Comb. Optim., vol. 12, chapter 14, Kluwer Academic, Dordrecht, 2002, pp. 663–695.

[3] P. M. Camerini, *The min-max spanning tree problem and some extensions*, Information Processing Lett. **7** (1978), no. 1, 10–14.

[4] G. N. Frederickson and J. Ja'Ja', *Approximation algorithms for several graph augmentation problems*, SIAM J. Comput. **10** (1981), no. 2, 270–283.

[5] H. N. Gabow and R. E. Tarjan, *Algorithms for two bottleneck optimization problems*, J. Algorithms **9** (1988), no. 3, 411–417.

[6] S. N. Kabadi and A. P. Punnen, *The bottleneck TSP*, The Traveling Salesman Problem and Its Variations (G. Gutin and A. Punnen, eds.), Comb. Optim., vol. 12, chapter 15, Kluwer Academic, Dordrecht, 2002, pp. 697–735.

Abraham P. Punnen: Department of Mathematics, Simon Fraser University, 13450 102 Avenue Surrey, BC, Canada, V3T 5X3

E-mail address: punnen@unbsj.ca