6th International Conference on
Combinatorics, Cryptography, Computer Science and Computing
November. 17-18, 2021

A strongly polynomial-time algorithm to minimum st-cut interdiction problems on planar networks

Javad Tayyebi[1], Hamid Bigdeli[2]
[1]Department of Industrial Engineering, Faculty of Industrial and Computer Engineering,
Birjand University of Technology, Birjand, I.R. Iran.
[2]Institute for the Study of War, Army Command and Staff University,
Tehran, I.R. Iran
Javadtayyebi@birjandut.ac.ir; Hamidbigdeli92@gmail.com

ABSTRACT

A popular class of combinatorial optimization problems is minimum st-cut problems in which a decision maker, called the attacker, wants to cut off any link joining two prescribed nodes s and t in a weighted network. For this purpose, he chooses a st-cut with minimum weight. An extension of this problem is that there is another decision maker, called the defender, which would like prevent that the attacker achieves his goal. So, he increases the edge weights under some budget and bound constraints. In this paper, this problem is considered in the case that the underlying network is planar. A strongly polynomial-time algorithm is proposed to solve the problem.

KEYWORDS: st-cut, interdiction problem, Stackelberg game, polynomial-time algorithm

1 INTRODUCTION

Combinatorial optimization problems are a wide class of optimization problems in which, among many objects with common structure, one is searched with minimum cost (maximum utility). A minimum st-cut problem is a well-known member of this class. In this problem, any object is an st-cut in the network, namely, the minimal set of edges whose removal separates two prescribed nodes s and t. Our aim of solving the minimum st-cut problem is to find an st-cut which the sum of its weights is minimized. The problem is defined on an undirected graph $G(V, A)$ where V is the set of n nodes, and A is the set of m edges. It admits the following mathematical programming formulation [2].

$$\begin{aligned} &\min z = \sum_{(i,j)\in A} w_{ij}x_{ij}, \\ &\pi_i - \pi_j + x_{ij} \geq 0 \quad \forall (i,j) \in A, \\ &\pi_t - \pi_s \geq 1, \\ &\pi_i \in \{0,1\} \quad \forall i \in V, \\ &x_{ij} \in \{0,1\} \quad \forall (i,j) \in A, \end{aligned} \tag{1}$$

in which x_{ij} is a zero-one variable which takes one if and only if (i,j) belongs to the desired st-cut; w_{ij} is the nonnegative weight associated to (i,j), and π_i is a zero-one variable that takes one if and only if i belongs to the connected component containing t.

Fortunately, we can replace the last two constraints to the nonnegativity constraints $\pi_i \geq 0, x_{ij} \geq 0$ due to the unimodularity property of the coefficient matrix. This implies that the problem belongs to the prominent class of linear programming problems. So, it can be solved by the available approaches in the linear programming class, such as the well-known simplex method. As another approach, one can solve the corresponding duality problem to obtain an optimal solution of problem (1) by the complementary slackness

theorem [3]. This approach is a common way deriving several polynomial-time algorithms for solving the problem because the dual problem is the well-known maximum flow problem [2].

A typical application of problem (1) is the situation in which there is an attacker that wants to cut off all paths between two vital points s and t. For this end, he chooses a minimum st-cut for incurring the minimum possible cost. Now, suppose that there exists another decision maker, called the defender. Two decision makers have conflicting goals. On the other word, the defender would like to prevent that the attacker achieves his goal. For this purpose, he is capable of increasing the edge weights under a budget constraint as well as some bound constraints. This problem is a game which has a hierarchy structure. At first, the defender chooses some edges and increases their weights. Then, the attacker observes the defender's action and selects an st-cut. In the context of game theory, such the problems are referred to as Stackelberg game due to hierarchy structure [4]. On the other hand, the problem can be regarded as a zero-sum game since the payoff of the attacker is equal to the loss of the defender. One can formulate the problem as follows:

$$
\begin{aligned}
&\max z \\
&\sum_{(i,j)\in A} r_{ij} d_{ij} \le R, \\
&0 \le d_{ij} \le \bar{d}_{ij} \quad \forall (i,j) \in A, \\
&\mathrm{z} = \min \{ \sum_{(i,j)\in A} (w_{ij} + d_{ij}) x_{ij} \qquad (2) \\
&\quad \pi_i - \pi_j + x_{ij} \ge 0 \quad \forall (i,j) \in A \\
&\quad \pi_t - \pi_s \ge 1 \\
&\quad x_{ij} \in \{0,1\} \quad \forall (i,j) \in A \}.
\end{aligned}
$$

This formulation contains two levels. The first level is the defender's level which has a continuous variable d_{ij}. This variable is the amount of increasing the weight of (i,j). So, the modified weight of (i,j) is $w_{ij} + d_{ij}$. The first level contains two types of constraints: one budget constraint as well as bound constraints. In this level, R is the total amount of available budget; r_{ij} is the cost of increasing the weight of (i,j) by one unit, and $\bar{d}_{ij}$ is an upper bound on this increment. The second level is the attacker's level that is the same minimum st-cut problem with respect to the edge weights $w_{ij} + d_{ij}$.

This problem is studied in [1], and a (weakly) polynomial-time algorithm is presented to solve the problem in the general case. In this paper, we consider a special case of the problem in which the underlying graph is planar. We propose a strongly polynomial-time algorithm for solving the problem.

2 PRELIMINARIES

In this section, we mention some notions of cuts and planar graphs used throughout the paper.
For any subset N of V with $s \in N$ and $t \in V \backslash N$, a set C of edges that have one endpoint in N and the other in $V \backslash N$ is referred to as an st-cut, denoted by $C = [N, \bar{N}] = [N, V \backslash N]$. The weight of an st-cut $[N, \bar{N}]$ with respect to a weight vector $\boldsymbol{w}$, denoted by $\boldsymbol{w}[N, \bar{N}]$, is the total weight of its edges, i.e., $\boldsymbol{w}[N, \bar{N}] = \sum_{(i,j)\in C} w_{ij}$.

Definition 1. A graph is called planar if it can be drawn in a two-dimensional plane so that no two edges cross. A face of a planar graph is a region bounded by edges that satisfies the condition that any two points in the region can be connected by a continuous curve that meets no nodes and edges.

An important property of planar networks is that finding a minimum st-cut is equivalent to finding a shortest $st-$path (a shortest path from s to t) in an auxiliary network. This new network is called the dual of the original network and can be constructed from the original network in linear time [2].

3 PROPOSED ALGORITHM

In this section, we formally state our proposed algorithm to solve problem (2) on planar networks in strongly polynomial time. Let us begin our discussion with a key result.

Lemma 1. If C is a minimum st-cut with respect to the initial weights w_{ij}, then there is an optimal solution of problem (2) for which C also remains a minimum st-cut.

Proof. The proof is easy to be proved since one can convert any optimal solution to another optimal solution in which C is also a minimum st-cut.

From Theorem 1, since there is at least a minimum st-cut C with respect to the initial edge lengths w_{ij}, it follows that we must increase its weight so that it remains minimum st-cut with respect to $w_{ij} + d_{ij}$ for any feasible strategy d_{ij} of the attacker. It means that if the weight of any st-cut C' becomes equal to that of C, it is necessary that we also increase the weight of C' together C. Based on this observation, the general procedure of our proposed algorithm is in the following way. At each iteration, it finds all minimum st-cuts and increases their weights as long as possible with minimum cost. This procedure is repeated until that either the budget constraint is satisfied in the equality form or there exists an st-cut, which the weight of all its edges is on their upper bound. In the first case, there is not any additional budget to be paid for increasing the weight of minimum st-cuts. So, the algorithm terminates because increasing the objective function of the attacker is not possible. In the second case, the weight of an st-cut cannot be increased, and consequently, increasing the weight of other minimum st-cuts is unnecessary.

To increase the weight of all minimum st-cuts, it is sufficient to increase the length of one edge in each minimum st-cut. For this purpose, we mention that st-cuts and st-paths are complementary structures. On the other word, any st-path and any st-cut have at least one common edge. So, we can use the notion of shortest st-paths to find a set of edges with minimum cost for increasing weight of all minimum st-cuts.

The algorithm maintains a set S containing all edges belonging to minimum st-cuts. At each iteration, if a new minimum st-cut is found, then its edges are added to S. According to Lemma 1, any element of S is not removed during the running of the algorithm, but it is possible that some edges are added to it at an iteration. The algorithm begins with running the Dijkstra algorithm on the dual network to find a minimum st-cut in the original network. It finds all shortest paths by using distance labels. Then, it finds all minimum st-cuts in the original network because any edge a shortest path in the dual network corresponds to exactly one edge of a minimum st-cut in the original network. So, all such the edges are added to S. Algorithm 1 provides a pseudocode that explains how to determine the set S for an arbitrary weight vector.

Algorithm 1

Input: *A planar graph $G(V, A)$ with two specified nodes s and t, edge weights $w_{ij} + d_{ij}$.*

Output*: A set S containing edges belonging to all minimum st-cuts.*

Construct the dual network $\bar{G}(\bar{V}, \bar{A})$.

Apply the Dijkstra algorithm to compute the distance label $dist(i)$ for every $i \in \bar{V}$ in $\bar{G}$.

Set $A' = \emptyset$.

for $(i,j) \in \bar{A}$ ***do***
if $dist(i) + c_{ij} + d_{ij} = dist(j)$ ***then***
Add edge corresponding to (i,j) *in the original network to* A'.
Use a traversal algorithm to find the set A_s *of all edges being accessible from* s *in* $G(V, A')$.
Find the set A_t *of all edges which* t *is accessible from them in* $G(V, A')$.
Set $S = A_s \cap A_t$.

To increase the weight of all st-cuts, a length vector $\boldsymbol{c}$ is defined as

$$c_{ij} = \begin{cases} r_{ij} & (i,j) \in S \wedge d_{ij} < d_{ij}^u, \\ +\infty & (i,j) \in S \wedge d_{ij} = d_{ij}^u, \ \forall (i,j) \in A. \\ 0 & (i,j) \in A \backslash S, \end{cases} \quad (3)$$

Our proposed algorithm searches a shortest st-path (a st-path P with minimum length) with respect to $\boldsymbol{c}$. The edges in the shortest st-path P are optimal candidates to increase the weight of all minimum st-cuts with the minimum possible cost. Notice that the length of P cannot be zero because S contains at least a minimum st-cut whose edges have positive weights by Definition (3). Moreover, if the length of P is $+\infty$, then it means that there is an st-cut C which $d_{ij} = d_{ij}^u$ for every $(i,j) \in C$. This implies that the algorithm terminates abruptly because the weight of this minimum st-cut cannot be increased more.

Now, consider the case that the length of P is finite. The algorithm increases the weight of all edges of $P \cap S$ by a positive value α. This causes that weight of all minimum st-cuts grows by α. The value of α is computed based on three following properties:

1. α must satisfy the bound constraint $d_{ij} + \alpha \leq d_{ij}^u$ for every $(i,j) \in P \cap S$.
2. To hold the budget constraint, α has to satisfy the inequality $\alpha \sum_{(i,j) \in P \cap S} r_{ij} \leq R'$ in which $R' = R - \sum_{(i,j) \in A} d_{ij}$ is the amount of remaining budget at previous iterations.
3. α has to be at most equal to the weight of second strict minimum st-cut. Notice that if not, at least one new edge is eligible to be added to S. So, it is essential that we first update S for some value less than α.

Satisfying the first and second cases is simply performed, but the third case is more challengeable. However, we present an approach which has not any need to find such a st-cut, directly. The approach is that we first compute α without regarding the third case as follows:

$$\alpha = \min \left\{ \frac{R'}{\sum_{(i,j) \in P \cap S} r_{ij}}, \min_{(i,j) \in P \cap S} \{ d_{ij}^u - d_{ij} \} \right\}. \quad (4)$$

Then, we increase the weight of edges of $P \cap S$ by α, and finally, we run the Dijkstra algorithm in the dual network to obtain a minimum st-cut with respect to new weights in the original network. If the weight of the current minimum st-cut is less than α plus the weight of the minimum st-cut in the previous iteration, then a second minimum st-cut of the previous iteration is now converted to a minimum st-cut due to these changes. In this case, we have to decrease α to the difference between them. After computing α, we update $d_{ij} = d_{ij} + \alpha$ for every $(i,j) \in P \cap$

S. Notice that this increases the length of each edge $(i,j) \in P \cap S$ by α. You can see a formal description of our proposed algorithm in Algorithm 2.

Let us discuss about the complexity of Algorithm 2.

Theorem 2: Algorithm 2 solves problem (2) in $O(m\ (S(n,m) + \bar{S}(n,m)))$ time in which $S(m,n)$ and $\bar{S}(n,m)$ are respectively the times required for solving shortest path problems in the original and dual network.

Proof. The proof is straightforward.

Algorithm 2

Input: *A planar graph $G(V,A)$, weights w_{ij} , costs r_{ij}, upper bounds d_{ij}^{u}, a budget R.*

Output: *an optimal weight increment vector d_{ij}.*

Set $d = 0, S = \emptyset, R' = R$.

while *$R' > 0$* ***do***

Apply Algorithm 1 to obtain the set S.

Set z to the weight of the current minimum st-cut.

Compute the length vector ***c*** *defined as (3).*

Find a shortest st-path P with respect to ***c***.

if *the length of P is infinity,* ***then***

Stop because there is a minimum st-cut whose length cannot be increased.

end if

Compute α *using (4).*

Find a minimum st-cut C with respect to $c_{ij} + d_{ij} + \alpha$ by finding the shortest path in the dual.

if *the weight of C is less than $z + \alpha$* ***then***

Set α *to the difference between the weight of C and the weight of the previous minimum st-cut.*

end if

Update $z = z + \alpha$ and $R' = R' - \alpha \sum_{(i,j)\in P\cap S} r_{ij}$.

for *(i,j) in $P \cap S$* ***do***

Update $d_{ij} = d_{ij} + \alpha$.

end for

end while

---

4 CONCLUSION

This paper investigated the minimum st-cut interdiction problem on planar networks. It developed an algorithm to solve the problem in strongly polynomial time. This algorithm is a significant contribution because the only available algorithm can solve the problem in (weakly) polynomial time [1].

It is notable that this paper considered the problem under linear costs (a linear budget constraint). In the case that the costs are fixed, it is proved that the problem is NP-hard [1]. So, the use of meta-heuristic or heuristic algorithms can be a convenient remedy in this case.

As future works, it will be meaningful to consider other combinatorial optimization interdiction problems, such as the assignment interdiction problem, and to design algorithms that are capable of solving them in polynomial time.

REFERENCES

[1] A. Abdolahzadeh, M. Aman, and J. Tayyebi, "Minimum st-cut interdiction problem," Computers & Industrial Engineering, vol. 148, 106708, 2020.

[2] R.K. Ahuja, L.T. Magnanti, and J.B. Orlin. Network Flows, Prentice-Hall, Englewood Cliffs, 1993.

[3] S.M. Bazaraa, J.J. Jarvis, and H.D. Sherali. Linear programming and network flows. John Wiley & Sons, 2008.

[4] J.C. Smith, and S. Yongjia, "A survey of network interdiction models and algorithms," European Journal of Operational Research, vol. 283.3, pp. 797-811, 2021.