On DDoS Attack Related Minimum Cut Problems

Qi Duan, Haadi Jafarian and Ehab Al-Shaer
Department of Software and Information Systems
University of North Carolina at Charlotte
Charlotte, NC, USA

Jinhui Xu
Department of Computer Science and Engineering
State University of New York at Buffalo
Buffalo, NY, USA

Abstract—In this paper, we study two important extensions of the classical minimum cut problem, called Connectivity Preserving Minimum Cut (CPMC) problem and Threshold Minimum Cut (TMC) problem, which have important applications in large-scale DDoS attacks. In CPMC problem, a minimum cut is sought to separate a source from a destination node and meanwhile preserve the connectivity between the source and its partner node(s). The CPMC problem also has important applications in many other areas such as emergency responding, image processing, pattern recognition, medical sciences. In TMC problem, a minimum cut is sought to isolate a target node from a threshold number of partner nodes. TMC problem is an important special case of network inhibition problem and has important applications in network security. We show that the general CPMC problem cannot be approximated within \( \log n \) unless \( \text{NP} = \text{P} \) has quasi-polynomial algorithms. We also show that a special case of two group CPMC problem in planar graphs can be solved in polynomial time. The corollary of this result is that the network diversion problem in planar graphs is in \( \text{P} \); a previously open problem. We show that the threshold minimum node cut problem can be approximated within ratio \( O(\sqrt{n}) \) and the threshold minimum edge cut problem can be approximated within ratio \( O(\log^2 n) \). We also answer another long standing open problem: the hardness of the network inhibition problem and network interdiction problem. We show that both of them cannot be approximated within any constant ratio, unless \( \text{NP} \not\subseteq \text{BPTIME}(2^{n^{o(1)}}) \).

I. INTRODUCTION

Distributed Denial of Service (DDoS) attacks have become one of the most preeminent threats of Internet today. The DDoS attacks against critical infrastructure (e.g., Internet backbone, power grid, financial services) are especially harmful. For example, the Crossfire attack \([1]\) can disable up to 53% of the total number of Internet connections of some US states, and up to about 33% of all the connections of the West Coast of the US. Link/node flooding is one important form of DDoS attacks. From the algorithmic point of view, link/node flooding is closely related to the minimum cut problem. The basic minimum cut problem is one of the most fundamental problems in computer science and has numerous applications in many different areas \([2,3,4,5]\). In this proposal we investigate two important generalizations of the minimum cut problem and their applications in link/node cut based DDoS attacks. The first generalization is denoted as the connectivity preserving minimum cut (CPMC) problem, which is to find the minimum cut that separates a pair (or pairs) of source and destination nodes and meanwhile preserve the connectivity between the source and its partner node(s). The second generalization is denoted as the Threshold Minimum Cut (TMC) problem in which a minimum cut is sought to isolate a target node from a threshold number of partner nodes. The basic minimum cut problem tries to find a minimum node/edge cut between a pair of nodes. If we want to find a minimum cut to separate two nodes (which are denoted as partner nodes) from a third node, the minimum cut may also separate the two partner nodes. As shown in Fig. 1 we want to find a node cut to separate two partner nodes \(s_1\) and \(s_2\) from another node \(t\). If we choose node \(s_3\) to be the cutting node, then \(s_1\) and \(s_2\) will still be connected, which means it is not a connectivity preserving cut. If we choose node \(s_4\) to be the cutting node, then \(s_1\) and \(s_2\) will be disconnected, which means it is not a connectivity preserving cut. In many applications it is important to find a connectivity preserving minimum cut since it is a natural requirement to maintain some connectivity when one wants to cut some links or nodes.

The CPMC problem has been recently studied in \([6]\) (a special case studied in \([7]\)). It has applications in many other areas, such as emergency responding, data mining, pattern recognition, and machine learning. It is very natural to have connectivity related constraints in many minimum cut based applications since any “cut” is destructive and one may need...
to limit the destructive aspects in the minimum cut. In applications related to emergency response, when a gun attack such as the Sandy Hook elementary school shooting \cite{1} happens in a building, the best response is to shut down certain passages, and at the same time to make sure that every one can have access to some exit and rescue personnel can reach the spot. This problem is closely related to CPMC. In medical science, the protein state transitions can be modeled as directed graphs called biological regulatory networks (BRN) \cite{2}. If one can identify the paths that lead to the states causing cancer or other possible diseases, then one should try to find the best way to prevent the system to reach the bad state. For example, one can try to find the optimal cut of possible paths leading to bad states, at the same time maintaining the paths that are needed for normal metabolism. This is exactly the CPMC problem. The network diversion problem \cite{3}, \cite{4}, \cite{5} is a problem to consider the minimum cut to divert traffic to a certain link or set of links/nodes. It has some similarity with the CPMC problem. However, the network diversion problem defines the problem only in the context of network diversion (none of the works recognizes the more important CPMC problem), while the CPMC problem defines the problem in a more natural way and has a lot more applications. In directed graphs, the network diversion problem is not even an NP optimization problem since the desired cut may not exist and it is NP-complete to judge if node disjoint paths between two pairs of nodes exist \cite{6}. The CPMC problem is an NP optimization problem in both undirected and directed graphs. Also, in planar graphs, the network diversion problem cannot be reduced to the CPMC problem. Even without considering all the applications, the CPMC is a very natural problem in pure graph theory and discrete mathematics.

The TMC problem arising naturally from DDoS attacks, threshold cryptography, and distributed data storage, which concerns with blocking a node from a threshold number of related nodes. Threshold cryptography and threshold related protocols have wide applications in network security such as secure and reliable cloud storage service \cite{7}, secure key generation and sharing in mobile ad hoc networks \cite{8}, \cite{9}, \cite{10}, etc. The natural optimization problem arises from threshold based protocols is to block a node from a threshold number of related nodes with minimum cost. This optimization problem is important for both attackers and defenders. From the attacker’s point of view, he/she needs to find an optimal way to thwart the execution of the threshold based protocol, or crack some information by compromising a threshold number of nodes or links. On the other hand, from the defender’s point of view, the defender may need to block the communication between a Bot master and a threshold number of Bots to thwart the Botnet attacks. In link cut DDoS attacks like Crossfire \cite{2} and Coremelt \cite{9}, the attacker may try to achieve the desired degradation ratio by flooding a set of critical links with minimum amount attacking flows. The problem can be easily converted to TMC. In distributed cloud storage, the attacker may try to disconnect the user from a threshold number of cloud servers to disrupt the services that require a certain number of available servers, the problem is exactly TMC. The TMC problem is an important special case of the network inhibition problem or network interdiction problem, and was first studied in \cite{11}.

The major results of this paper includes: We show that CPMC in directed graphs and multi-node CPMEC are hard to approximate, 3-node CPMNC is in \(P\), and network diversion problem in undirected planar graphs is in \(P\), which is a long standing open problem \cite{12}. We show that the TMC problem can be approximated within ratio \(\sqrt{n}\) and the TMEC problem can be approximated within ratio \(\log^2 n\). We reveal the relationship between the TMC problem and other closely related problems. We also give an answer to another long standing open problem: the hardness of the network inhibition problem \cite{13} and the network interdiction problem \cite{14}. We show that both of them cannot be approximated within any constant ratio unless \(NP \not\subseteq \cap_{\delta>0} BPTIME(2^{\delta n})\).

II. HARDNESS RESULTS OF CPMC

We adopt the notation from \cite{15}. The most simple case of CPMC is the 3-node CPMC. Informally speaking, in the 3-node CPMC problem, we are given a connected graph \(G = (V, E)\) with positive node (or edge) weights, a source node \(s_1\) and its partner node \(s_2\), and a destination node \(t\). The objective is to compute a cut with minimum weight to disconnect the source \(s_1\) and destination \(t\), and meanwhile preserve the connectivity of \(s_1\) and its partner node \(s_2\) (i.e., \(s_1\) and \(s_2\) are connected after the cut). The weights can be associated with either the nodes (i.e., vertices) or the edges, and accordingly the cut can be either a set of nodes, called a connectivity preserving node cut, or a set of edges, called a connectivity preserving edge cut. In the former case, a cut is a subset of vertices \(V\) whose removal (along with the edges incident to them) disconnects \(s_1\) and \(t\), but does not affect the connectivity of \(s_1\) and \(s_2\). Such a cut is called a connectivity preserving node cut (CPMNC). In the latter case, a cut is a subset of edges whose removal disconnects \(s_1\) and \(t\) and preserves the connectivity of \(s_1\) and \(s_2\). Such a cut is called a connectivity preserving minimum edge cut (CPMEC). The weight of a cut \(C\) is the total weight associated with the nodes or edges in \(C\). Note that we can easily extend the 3-node CPMC problem to the general case CPMC where one may have multiple pairs of source and destination nodes, and each source node may have multiple partner nodes.

First we note that the CPMNC problem is an NP optimization problem. To determine whether a valid cut exists, one just needs to check if \(t\) is connected to any bridge node between \(s_1\) and \(s_2\); if so, then no valid cut exists. Clearly, this can be done in polynomial time. Thus, we assume thereafter that a cut always exists.

The decision version of the 3-node CPMNC problem is as follows: given an undirected graph \(G = (V, E)\) with each node \(v_i \in V\) associated with a positive integer weight \(c_i\), a source node \(s_1\), a partner node \(s_2\), a destination node \(t\), and an integer \(B > 0\), determine whether there exists a subset of nodes in \(V\) with total weight less than or equal to \(B\) such that
the removal of this subset disconnects \( t \) from \( s_1 \) but preserves the connectivity between \( s_1 \) and \( s_2 \).

The decision version of the 3-node CPMEC problem can be defined similarly: given an undirected graph \( G = (V, E) \) with each edge \( e_i \in E \) associated with a positive integer weight \( c'_i \), a source node \( s_1 \), a partner node \( s_2 \), a destination node \( t \), and an integer \( B' > 0 \), determine whether there exists a subset of edges in \( E \) with total weight less than or equal to \( B' \) such that the removal of this subset disconnects \( t \) from \( s_1 \) but preserves the connectivity between \( s_1 \) and \( s_2 \).

The CPMEC has several key differences from CPMNC. First, the resulting graph after the node cut in CPMNC may be disintegrated into many connected components, where in CPMEC the resulting graph has exactly two connected components (otherwise there will be some redundant edges, and the cut cannot be the minimum one). Second, suppose the weight of the minimum edge cut between a single node \( s_1 \) and destination \( t \) is \( C_e(s_1, t) \), the weight of the minimum edge cut (not necessarily connectivity preserving) between two nodes \( s_1, s_2 \) and destination \( t \) is \( C_e(s_1, s_2, t) \), if \( C_e(s_1, t) + C_e(s_2, t) > C_e(s_1, s_2, t) \), then the minimum edge cut between two nodes \( s_1, s_2 \) and destination \( t \) must be connectivity preserving. Node cut does not have this property. These key differences mean that the node cut problem and the edge cut problem may have different hardness. In our NP-hardness proof of the CPMEC, we cannot modify it to get a proof for the CPMEC.

Given nodes \( s_1 \) and \( t \) in a graph, we can classify other nodes into several categories. If a node \( s_2 \) has the property \( C_e(s_1, t) > C_e(s_2, t) \), then it is easy to show that the minimum edge cut between two nodes \( s_1, s_2 \) and destination \( t \) must be connectivity preserving.

**Lemma 2.1:** For two points \( s_1 \) and \( s_2 \) in the graph, if \( C_e(s_1, t) + C_e(s_2, t) > C_e(s_1, s_2, t) \), then the minimum edge cut between two nodes \( s_1, s_2 \) and destination \( t \) must be connectivity preserving.

**Proof:** For a minimum edge cut between two nodes \( s_1, s_2 \) and destination \( t \), it must be the the union of two cuts: one is the cut between \( s_1 \) and \( t \), another is between \( s_2 \) and \( t \). The sum of this two cut is at least \( C_e(s_1, t) + C_e(s_2, t) \). If \( C_e(s_1, t) + C_e(s_2, t) > C_e(s_1, s_2, t) \), then the two cuts must have some common edges. But if a common edge exists, then we have two cases:

- case 1: The common edge is connected with the component of \( s_1 \) and \( s_2 \), then this edge can be removed from the cut, and the remaining cut is still valid, and \( s_1 \) and \( s_2 \) is now connected.
- case 2: The common edge is connected with the two components and the \( t \) component, in this case, \( s_1 \) and \( s_2 \) must be connected.

If \( C_e(s_1, t) + C_e(s_2, t) = C_e(s_1, s_2, t) = C_{ep}(s_1, s_2, t) \) (here \( C_{ep}(s_1, s_2, t) \) is the CPMEC between \( s_1, s_2, \) and \( t \)), we call \( s_2 \) a threshold node of \( s_1 \) time. If node \( s_2 \) satisfies \( C_{ep}(s_1, s_2, t) > C_e(s_1, t) + C_e(s_2, t) \), we call them outer points of \( s_1 \).

We also investigate the multiple-partner CPMEC problem.

In this case, we have \( u + 1 \) nodes \( s_1, \ldots, s_r, t \) in the graph, and the objective is to find a minimum edge cut that separates \( s_1, \ldots, s_r \) from \( t \), and at the same time keeps \( s_1, \ldots, s_r \) connected.

Note that this problem is still an NP optimization problem. If the removal of node \( t \) causes some of the \( s_i \) nodes to be disconnected from others, then no valid cut exists. Since this can be determined in polynomial time, we always assume that there exists a solution to the problem.

In some applications, we need to find a minimum cut to separate two connected components \( \Gamma_1 \) and \( \Gamma_2 \) from another connected component \( T \) in a graph, and keep \( \Gamma_1 \) and \( \Gamma_2 \) connected. This is a generalization of the original 3-node connectivity preserving minimum cut problem. We can show that the generalized problem has the same approximability as the original problem, in both the cases of node cut and edge cut.

**Theorem 2.2:** The generalized connectivity preserving minimum cut problem can be \( L \)-reduced to the original connectivity preserving minimum cut problem. This means the generalized problem has the same approximability as the original connectivity preserving minimum cut problem. In other words, if the connectivity preserving minimum cut problem can be approximated within \( f(n) \) (\( n \) is the input size), then the generalized problem can also be approximated within \( f(n) \).

**Proof:** To see this, we can transform the connected components \( \Gamma_1, \Gamma_2, \) and \( T \) to three nodes. For every connected component, we can shrink the whole component into one new node. All edges inside the connected component are deleted. For every edge connecting a node in the component and a node outside the component, we add an edge between the new node and the outside node. To make the graph still a simple graph, if there are multiple edges between two nodes, we add an intermediate node in the edge. For node cut, we set the weight of the intermediate node to be infinity. For edge cut, the two intermediate edges all have the same weight as the old edge. An example of component shrinking is shown in Fig. 2.

Now it is easy to see, any connectivity preserving minimum cut in the old graph is the connectivity preserving minimum cut for the new graph, and vice versa. Also note that after the shrinking procedure, the size of the graph decreased. So
the generalized problem has the same approximability as the original problem.

First, we give the definition of CPMC in directed graph.

**Definition 2.3 (3-node CPMC on directed graphs):** Let \( G = (V, E) \) be a directed graph with \( n \) nodes and \( m \) edges. Each edge \( e_i \in E(1 \leq i \leq m) \) is associated with a positive integer weight. Given three nodes \( s_1, s_2, t \), and a positive integer \( b \), the 3-node CPMC problem for \((s_1, s_2, t)\) is to seek a subset \( C \) of edges in \( E \) with total weight less than or equal to \( b \) such that after the removal of \( C \), there is no path from \( t \) to \( s_1 \) and \( s_2 \) and meanwhile there exists at least one path from \( s_1 \) and \( s_2 \).

Below we show that the 3-node CPMC problem on directed graphs cannot be approximated within a logarithmic ratio.

**Theorem 2.4:** In a directed graph \( G \), the 3-node connectivity preserving minimum edge cut problem cannot be approximated within a factor of \( \alpha \log n \) for some constant \( \alpha \) unless \( P = NP \).

**Proof:** To prove the theorem, we reduce the set cover problem to this problem. In the set cover problem, we have a ground set \( T = \{e_1, e_2, \ldots, e_n\} \) of \( n \) elements, and a set \( S = \{S_1, S_2, \ldots, S_k\} \) of \( k \) subsets of \( T \) with each \( S_i \in S \) associated with a weight \( w_i \). The objective is to select a set \( O \) of subsets in \( S \) so that the union of all subsets in \( O \) contains every element in \( T \) and the total weight of subsets in \( O \) is minimized.

Given an instance \( I \) of the set cover problem with \( n_1 \) elements and \( k \) sets, we construct a new graph. The new graph has an element gadget for every element, and every element gadget contains \( k_1 + 2 \) nodes, where \( k_1 \) is the number of sets that contains this element. In every gadget, there are two end points, and \( k_1 \) internal nodes are connected to the two end nodes in parallel. Every internal nodes of a gadget corresponds to a set that contains this element. All such \( n_1 \) gadgets are connected sequentially through their end points, with \( s_1 \) and \( s_2 \) at the two ends of the whole construction. We also construct an arc for every set. There is an arc from the ending point of every set arc to the corresponding set node in the element gadget, and there is another arc from \( t \) to the starting point of every set arc.

Figure 3 is the graph constructed for set cover instance with three elements \( x_1, x_2, \) and \( x_3 \), three sets \( A_1 = \{x_1, x_3\}, A_2 = \{x_2, x_3\}, \) and \( A_3 = \{x_1, x_2\} \).

Every set arc is assigned with weight \( w_i n_1 k \), where \( w_i \) is the weight of the set in the original set cover instance. All the arcs in a gadget are assigned with weight 1. All other arcs (the arcs connecting the \( s_1 \), the gadgets, and \( s_2 \), and the arcs connecting \( t \) with the set arcs) have weight infinity. We also let \( b = n_1 k D_1 + n_3 k - 1, \) where \( D_1 \) is the bound of weight in the set cover instance. First note that we only need to cut \( t \) from reaching \( s_2 \), since there is no way for \( t \) to reach \( s_1 \) in the constructed graph. Also note that one cannot put all arcs from the set node to the right end node of the gadget into the cut in an element gadget, otherwise \( s_1 \) and \( s_2 \) will be separated. Now we can see that if the set cover instance has a cover with weight no more \( D_1, \) then we can choose the following cut: The cut contains those set arcs contained in the cover and all the gadget arcs starting with the set node which are not in the set cover. The cut has a weight \( n_1 k D_1 + g_1, \) where \( g_1 < n_1 k. \) Similarly if we can find a cut with weight no more than \( n_1 k D_1 + n_1 k - 1, \) then we can find a set cover with weight no more than \( D_1. \) Furthermore, since set cover cannot be approximated within \( \alpha \log n \) for some constant \( \alpha \) unless \( NP = P \) \([7], [8]\), we can see that the connectivity preserving minimum cut problem in directed graph cannot be approximated within \( \alpha \log n \) for some constant \( \alpha \) unless \( NP = P. \)

For the set cover problem with \( n_1 \) elements and \( k = poly(n_1) \) sets, it cannot be approximated within \( \alpha \log n \) unless \( NP = P \) \([7], [8]\). Since \( k \) is bounded by some polynomial in \( n_1 \), we can see

\[
\frac{D_1}{D} < \frac{n_1 k D_1 + g_1}{n_1 k D + g_1} + o(1) \leq \alpha_1 \log(n_1 k),
\]

for some \( \alpha_1 \), then we have

\[
\frac{D_1}{D} < \frac{n_1 k D_1 + g_1}{n_1 k D + g_1} \leq \alpha_1 \log(n_1 k),
\]

Now we have a contradiction, which means that the problem cannot be approximated within \( \alpha \log n \) unless \( NP = P. \)

Next we consider the case of multiple partner nodes for the CPMC problem. In this case, we have \( u + 1 \) nodes \( s_1, \ldots, s_u, t \).
in the graph, and the objective is to find a minimum edge cut that separates \( s_1, \ldots, s_u \) from \( t \), and at the same time keeps \( s_1, \ldots, s_u \) connected.

Note that this problem is still an NP optimization problem. If the removal of node \( t \) causes some of the \( s_i \) nodes to be disconnected from others, then no valid cut exists. Since this can be determined in polynomial time, we always assume that there exists a solution to the problem.

**Theorem 2.5:** For an undirected graph \( G \), the CPMC problem with \( k \) partner nodes (where \( k \) is an integer, not necessarily constant, given in the input) cannot be approximated within a factor of \( \log n \) for some constant \( \alpha \) unless \( \text{NP} = \text{P} \).

**Proof:** We use similar reductions as in the proofs of Theorems 2.3. For the reduction of \( \log n \)-inapproximability, the only difference is that now the set node is replaced by a set edge, as shown in Figure 1.

As in the proof of Theorem 2.3, the set edge has weight \( w_{ij} n_j k \). Also in every gadget, the two edges connecting the end nodes in the gadget and the set node have weight 1, which are called gadget-set edges. All other edges have weight infinite. Now it is easy to see that to make \( s_j \) disconnected from \( t \), for every set node in a gadget, either the two gadget-set edges or the corresponding set edge must be included in the cut. The remaining proof is the same as that in the proof of Theorem 2.3.

Similarly we can apply the proof of polylog factor inapproximation of CPMNC in [2], using a similar construction, to show that directed CPMC and multi-node CPMEC cannot be approximated within a polylog factor. Due to space limitation, it is omitted here.

### III. CPMC in Planar Graphs

In [2] it is shown that 3-node CPMEC in planar graphs has polynomial time solutions and in [2] it is shown that multi-node CPMEC in planar graphs has polynomial time solutions.

With some modifications, the algorithm in [2] can also be applied to 3-node planar CPMNC, and we can show the algorithm also finds the optimal solution for 3-node planar CPMNC.

**Theorem 3.1:** The 3-node planar CPMNC can be solved in polynomial time.

**Proof:** The major difference between edge cut and node cut is that in any node cut the graph may be disintegrated into multiple connected components. For any cut between a node \( v \) and destination node \( t \), we define the connected component containing \( v \) after the cut as the principal cut component of node \( v \). We also introduce the perturbation technique of node weight that we used in the proof of 3-node planar CPMEC in [2]. In this way, any two cuts (or more generally, two subsets of edges) in \( G \) will have different weights unless they are completely identical. Based on this property, we have the following observations. (1) Any cut is unique. (2) Given any node \( v \) in \( V \), let \( C_v \) be the connected component containing \( v \) and resulting from the minimum edge cut between \( v \) and \( t \). Then all nodes in \( C_v \) can be uniquely determined due to the perturbation technique. Now note that any two principal cut components cannot enclose a hole. Suppose the CPMNC principal cut component between nodes \( s_1, v \) and \( t \) is \( C_{s_1, v} \).

If there exists a hole that is completely surrounded by the two CPMC principal cut components \( C_{s_1, A} \) and \( C_{s_1, A_1} \), then for the nodes adjacent to the boundary of the hole, we can divide the nodes into 3 types: The first type of is in the CPMC cutting of \( C_{s_1, A} \) but not cutting nodes of \( C_{s_1, A_1} \). We denote the total weight of this type of nodes as \( L_1 \). The second type is the cutting nodes of \( C_{s_1, A_1} \) but not cutting nodes of \( C_{s_1, A} \). We denote the total weight of this type of nodes as \( L_2 \). The third type of segments is the cutting nodes of both of \( C_{s_1, A_1} \) and \( C_{s_1, A} \). We have \( L_2 > L_1 \), \( L_1 > L \) or \( L_1 = L \). If \( L_1 = L \) then we remove the cutting nodes of type 1 and add cutting nodes of type 2 for the CPMNC between \( s_1, A \) and \( t \). Now we can get a smaller CPMNC, because the new cut will decrease by a value of \( L \) and increase by a value of \( L_1 \), and the overall effect is that the value of the cut will decrease by at least \( L - L_1 \). This is a contradiction. Note that the principal cut component \( C_{s_1, A} \) may not enlarge since the added nodes in the hole may not connect to component \( C_{s_1, A} \), but this does not affect the argument. The remaining two cases are similar.

So for planar CPMC, any two principal cut components cannot enclose a hole. The remaining proof of the 3-node planar CPMNC is similar to 3-node planar CPMEC when we replace the enclosed edge cut region \( C_{s_1, v} \) with principal cut component \( C_{s_1, v} \) for any node \( v \) which is not \( t \).

Next we show that 2-node versus 2-node planar CPMEC is in \( \text{P} \).

**Theorem 3.2:** Suppose in a planar graph \( G = (V, E) \), there
are two groups of partner nodes $s_1, s_2$ and $s'_1, s'_2$. Then we can find the CPMEC to separate the two groups of partner nodes in polynomial time.

Proof: In the algorithm for 3-node planar CPMEC, for every places when we need to compute a minimum cut between a shrunk node and the destination $t$ we can change it to “compute the CPMEC between the shrunk node and the partner nodes in another group”. We have the following observations: (1) In every step of the path growing procedure, the node $s_2$ may either be inside the component that contains $s'_1$ and $s'_2$ defined by the new found CPMEC or inside the component that contains $s_1$. (2) If at any step, the path cannot grow and node $s_2$ still not been added, this means it is infeasible to find a satisfactory cut. However, this infeasibility can be verified at the beginning of the algorithm. So if there exists a feasible cut, this situation will not happen. Since we can compute the 3-node planar CPMEC in polynomial time, the procedure will be able to find the desired cut in polynomial time. ■

Note that this result immediately implies that the network diversion problem is in $P$ in planar graphs, since the network diversion problem is a special case of 2-node versus 2-node CPMEC.

**Corollary 3.3:** The network diversion problem in planar graphs can be solved in polynomial time.

Proof: The network diversion problem in planar graphs is a special case of the 2-node versus 2-node CPMEC in planar graphs where node $s_1$ and $s'_1$ is connected. ■

Another result is that the two node location constrained shortest path (LCSP) problem [?] is in $P$. The original definition of LCSF defines the shortest path where one node should be above the path. The two node LCSP problem is to find the shortest path where one given node is located above the path, another given node is located below the path.

**Corollary 3.4:** The two node LCSP problem in planar graphs can be solved in polynomial time.

Proof: The two node LCSP problem is a special case of the 2-node versus 2-node CPMEC in planar graphs. If we add a dummy node $s_1$ that connects to all nodes that are in the boundary of the graph and above the two end points of the path, and a dummy node $s'_1$ that connects to all nodes that are in the boundary of the graph and below the two end points of the path, and consider the two position defining nodes as $s_2$ and $s'_2$ respectively, then the problem becomes a special case of the 2-node versus 2-node CPMEC in planar graphs where $s_1$ and $s'_1$ are in the same surface. ■

**IV. Threshold Minimum Cut Problem**

We first formalize the threshold minimum node cut problem as follows.

Suppose we have a graph $G = (V, E)$ and a set of $k$ service nodes $\Gamma = \{S_1, S_2, \ldots, S_k\}$, ($S_i \in V$ for $1 \leq i \leq k$), a client node $A \in V$, and a threshold integer $l$. Every node $v_i$ in $G$ has an associated cost $c_i$. How to find a minimum cost node cut such that at least $l$ out of the $k$ service nodes will be disconnected from $A$?

The decision version of the problem is:

Suppose we have a graph $G = (V, E)$ and a set of $k$ service nodes $\Gamma = \{S_1, S_2, \ldots, S_k\}$, ($S_i \in V$ for $1 \leq i \leq k$), a client node $A \in V$, a threshold integer $l$, and another integer $B$. Every node $v_i$ in $G$ has an associated cost $c_i$. Can one find a node cut such that at least $l$ out of the $k$ service nodes will be disconnected from $A$ and the total cost of the cut is no more than $B$?

We name this problem as the threshold minimum node cut problem (TMNC). The threshold minimum edge cut problem (TMEC) can also be similarly defined, the only difference is that every edge in $G$ has a cost and the cut is an edge cut.

**A. Hardness of the TMC problem**

The TMNC problem was shown to be NP-complete in [?]. We can also show that the TMEC problem is NP-complete.

**Theorem 4.1:** The TMEC problem is NP-complete.

Proof: We can use the reduction from minimum bisection problem (with unit edge cost). Given an instance of minimum bisection problem $G = (V, E)$, we can construct an instance of the threshold minimum edge cut problem. We construct a new graph $G'$ which contains all nodes in $G$ with an additional node $A$. Node $A$ is connected with every node in $G$ with an edge of cost $n^2$, where $n = |V|$. Without loss of generality, we assume $n$ is an even number. We also define the set $S$ of the TMEC instance to be $V$, and $l = n/2$. We have the following observations:

- The threshold minimum cut in $G'$ will make $A$ to be separated from at least $n/2$ nodes in $V$, that is, the threshold minimum cut will have a value that is at least $n^2 n/2 = n^3/2$.
- The threshold minimum cut in $G'$ will not make $A$ to be separated from more than $n/2$ nodes in $V$ since every additional edge in $G'$ has cost $n^2$, which is more than the number of total edges in $G$.

Now we can see that the minimum cut in the threshold minimum edge cut instance instance should include a minimum bisection of $G$ and exactly $n/2$ edges between $A$ and nodes in $V$. If we can find a threshold cut in $G'$ that is $u + n^3/2$, we can find the corresponding bisection of $G$ with value $u$. Conversely, if there exists a bisection of $G$ with value $u$, we can find threshold cut in $G'$ with value $u + n^3/2$. This finishes the reduction. ■

Based on the NP-completeness of the threshold minimum node cut problem, it is easy to see the following problem is NP-complete [?].

**Definition 4.2:** Given an undirected graph $G = (V, E)$ and a number $m$, can one find a subgraph with at least $m$ edges, and the number of nodes in the subgraph is minimized?

Note that this is the inverse problem of the maximum $k$-subgraph problem (unit weight case). And this inverse $k$-subgraph problem can be further generalized to the following set minimum cover problem [?]:

**Definition 4.3:** Given a set $S$ of $n$ elements, a collection $C$ of $m_1$ subsets of $S$, and positive integer $m \leq m_1$, can one
find $m$ subsets from $C$ such that the total number of distinct elements in the union of the $m$ subsets is minimized?

We can see the set minimum cover problem is the generalization of the inverse k-subgraph problem, so the set minimum cover problem is also NP-complete.

Similarly we can define the inverse problem of the set minimum cover problem as follows [2].

**Definition 4.4:** Given a set $S$ of $n$ elements, a collection $C$ of $m_1$ subsets of $S$, and a positive integer $n_1 \leq n$, can one find a subset $S'$ from $S$ such that the size of $S'$ is $n$ and the number of subsets (from $C$) fully covered by $S'$ is maximized?

Here “fully covered” means that every element of the subset is included in $S'$. We denote this problem as the set maximum cover problem. We can see this problem is the generalization of the maximum k-subgraph problem.

If every set in the set minimum cover problem has at most $\tau$ elements, we denote this special case as the $\tau$-minimum cover problem. Similarly we can define the $\tau$-maximum cover problem.

For the k-subgraph problem, the best known approximation algorithm is the $n^{1/4}$ ratio approximation algorithm in [2]. It is conjectured that the problem cannot be approximated within $n^\delta$ for some $0 < \delta < 1$, but currently the best known hardness result is that it has no polynomial time approximation scheme unless $NP \not\subset \cap_{\delta>0} BPTIME(2^{n^\delta})$ [2], [2].

The relationship of the approximability between the set minimum cover problem and the set maximum cover problem is shown in the next theorem, proved in [2].

**Theorem 4.5:** If the $\tau$-minimum cover problem can be approximated within ratio $1 + \epsilon$ ($0 < \epsilon < 1$), that is, there is an algorithm that can return a subset of $S$ with no more than $1 + \epsilon$ times number of elements compared with the optimal solution, then the $\tau$-maximum cover problem can be approximated within ratio $16^\epsilon$ (that is, suppose the optimal solution of the $\tau$-maximum cover problem is $OPT$, there exists an algorithm that can return a subset of $S$ that fully covers at least $4^{-\tau}OPT$ number of subsets of $C$.

Note that $4^{-\tau}$ is very close to 1 if $\epsilon$ is very close to 0.

**Corollary 4.6:** The set minimum cover problem does not have polynomial time approximation scheme unless $NP \not\subset \cap_{\delta>0} BPTIME(2^{n^\delta})$.

**Proof:** This follows from the above theorem and the result that max k-subgraph problem (special case of the $\tau$-maximum cover problem when $\tau = 2$) does not have polynomial time approximation scheme unless $NP \not\subset \cap_{\delta>0} BPTIME(2^{n^\delta})$.

**Corollary 4.7:** Both the threshold minimum node cut problem and the set minimum cover problem do not have polynomial time approximation scheme unless $NP \not\subset \cap_{\delta>0} BPTIME(2^{n^\delta})$.

We can further show that the maximum cover problem cannot be approximated within any constant ratio unless $NP \not\subset \cap_{\delta>0} BPTIME(2^{n^\delta})$.

**Theorem 4.8:** The maximum cover problem cannot be approximated within any constant ratio unless $NP \not\subset \cap_{\delta>0} BPTIME(2^{n^\delta})$.

**Proof:** First we show that if $2\tau$-maximum cover can be approximated within ratio $\zeta$ ($\zeta > 1$), then $\tau$-maximum cover can be approximated within ratio $\sqrt[3]{\zeta}$. For a $\tau$-maximum cover instance with set $S$, collection $C$, and integer $n_1$, we can construct a $2\tau$-maximum cover instance as follows: the set $S$ and integer $n_1$ is the same as that in the $\tau$-maximum cover instance. We define a new collection $C'$ which contains the union of all possible pairs of subsets in $C$, with all repetitions of subsets being kept. For example, if $C = \{\{a_1, a_2\}, \{a_2, a_3\}\}$, then

$$C' = \{\{a_1, a_2\}, \{a_2, a_3\}, \{a_1, a_2, a_3\}, \{a_1, a_2, a_3\}\}$$

We define $C'$ to be the subset collection of the $2\tau$-maximum cover instance. Suppose we have a ratio $\zeta$ approximation algorithm for the $2\tau$-maximum cover instance, we can apply the same solution (set of elements, denoted as $S'$) for the $\tau$-maximum cover instance. We have the following observations:

- If $S'$ fully covers $\eta$ subsets of $C$, then it fully covers $\eta^2$ subsets of $C'$.
- If $S'$ fully covers $\eta$ subsets of $C'$, then it fully covers $\sqrt{\eta^3}$ subsets of $C$.
- If the optimal solution of the $2\tau$-maximum cover instance can fully cover $OPT$ subsets of $C'$, then the optimal solution of the $\tau$-maximum cover instance can fully cover $\sqrt{OPT}$ subsets of $C$.

Actually we can further show that any subset of $S$ will fully cover a perfect square number of subset in $C'$. Suppose $S'$ fully covers $\eta$ subsets in $C$, then $S'$ will cover all those subsets that are union of two subsets in $C$, and $S$ will not fully cover any other subsets in $C'$ because at least one of the elements of these subsets will not be in $S'$. This means the number of fully covered subsets in $C'$ by $S'$ will always be a perfect square.

So if we have a ratio $\zeta$ approximation algorithm for the $2\tau$-maximum cover problem, then we will have a ratio $\sqrt[3]{\zeta}$ approximation algorithm for the $\tau$-maximum cover problem.

If there exists a constant ratio (denoted as $\zeta$) approximation algorithm for the set maximum cover problem, we have the following observation: given any instance of the set maximum cover problem with maximum number of elements in the collection of subsets to be $\tau$ and any number $\zeta_1 > 1$, we can construct a $2\mu\tau$-maximum cover instance where $\mu = \lceil \log(\zeta/\zeta_1) \rceil$, with the same $S$ and $n_1$, but the collection is defined to be $C'^\mu$. Here $C'^\mu$ is the resulting collection of subsets by applying the $C'^\mu$ operation $\mu$ times iteratively. For example, $C'^4$ is the resulting collection which contains the union of all possible pairs of subsets in $C'$. Since the $2\mu\tau$-maximum cover instance can be approximated within $\zeta$, then we can use its result to obtain the $2\sqrt[3]{\zeta} \leq \zeta_1$ approximation for the original $\tau$-maximum cover instance. This means if there exists a constant ratio approximation algorithm for the set maximum cover problem, then we can find a polynomial time approximation scheme for it. However we know that maximum $k$-subgraph problem (which is equivalent to the 2-maximum
cover problem) does not have polynomial time approximation scheme unless \(NP \not\subseteq \cap_{\delta > 0} \text{BPTIME}(2^n^\delta)\), so there exists no constant ratio approximation for the set maximum cover problem unless \(NP \not\subseteq \cap_{\delta > 0} \text{BPTIME}(2^n^\delta)\).

**Corollary 4.9:** The network inhibition problem (the goal is to find the most effective way to reduce the capacity of a network flow within fixed budget) and the network interdiction problem (a different version of network inhibition, and the goal is to choose a subset of arcs to delete, without exceeding the budget, that minimizes the maximum flow or other flow metrics that can be routed through the network induced on the remaining arcs) cannot be approximated within any constant ratio unless \(NP \not\subseteq \cap_{\delta > 0} \text{BPTIME}(2^n^\delta)\).

**Proof:** We can reduce the set maximum cover problem to the network inhibition problem and network interdiction problem. For an instance of the set maximum cover problem with set \(C\), collection \(C\) and integer \(n_1\), we can construct a directed graph as follows. First we create a source node \(U\) and destination node \(T\). For every subset in \(C\), we create a node. For every element in \(S\), we create an arc, which is again connecting to every node whose corresponding subset contains the element. The source is connected to the starting point of every arc that corresponds to every element. Every node that corresponds to a subset is connected to the destination. The blocking cost of the arcs that corresponds to the elements is 1, and the cost of all other arcs is infinity. We also set the capacity of the incoming arcs of destination to be 1, all other arcs have capacity infinity. As an example, suppose \(S = \{a_1, a_2, a_3\}, C = \{\{a_1, a_2\}, \{a_2, a_3\}\}\), then the constructed network inhibition/interdiction instance is shown in Fig. 5. In the figure, the first number corresponds to every arc is the capacity of the arc, and the second number corresponds to every arc is the blocking cost of the arc.

Now it is easy to see the maximum flow from \(U\) to \(T\) is \(|C|\). If a subset of \(S\) fully covers any subset in \(C\), the deletion of the corresponding arcs will reduce the maximum flow by 1. We can see that the set maximum cover in the original instance corresponds to a partial cut for the flow. In this way we reduce the set maximum cover problem to the network inhibition problem and network interdiction problem. So the network inhibition problem and the network interdiction problem cannot be approximated within any constant ratio unless \(NP \not\subseteq \cap_{\delta > 0} \text{BPTIME}(2^n^\delta)\).

**V. APPROXIMATION ALGORITHMS OF TMC**

First we present a ratio \(O(\sqrt{n})\) approximation algorithm for TMNC problem in Algorithm 1. In the algorithm, we assume that \(l \geq \sqrt{n}\), since it is trivial to have an \(\sqrt{n}\) approximation when \(l < \sqrt{n}\) (one just needs to sort \(S_1, \ldots, S_k\) according to their minimum cut values with \(A\) in ascending order, choose the first \(l\) nodes and find the minimum cut between these nodes and \(A\)).

**Algorithm 1:** The Approximation Algorithm for Threshold Minimum Node cut

1. Solve the following Linear Programming (LP):

   \[
   \text{Maximize} \quad \sum_{i \in \text{all nodes}} X_i c_i \\
   \text{Subject to} \\
   Y_i \leq X_i + Y_j, \text{ for all neighbors } v_j \text{ of } v_i, \forall v_i \\
   0 \leq X_i \leq 1, 0 \leq Y_i \leq 1, \forall v_i \\
   Y_A = 0 \\
   \sum_{i=1}^{k} Y_{S_i} \geq l
   \]

   After solving the LP, sorting the nodes \(S_1, S_2, \ldots, S_k\) according to the LP value \(Y_{S_i}\) in descending order.

2. Find the first \(S_i\) in the sorted list \(S_1, \ldots, S_k\) that is less than \(1/\sqrt{n}\).

3. If \(i > l\) then
   - Find the minimum cut between \(S_1, \ldots, S_i\) and \(A\), return this cut value.
   - Else
     - Sorting all \(S_j\) \((j > i)\) in descending order according to the cut value \(c(S_j)\), here \(c(S_j)\) is the minimum cut value between node \(S_j\) and node \(A\). Denote the first \(l - i + 1\) nodes in the sorted list as \(S'_1, \ldots, S'_{l-i+1}\).
     - Find the minimum cut between \(S_1, \ldots, S_i-1, S'_1, \ldots, S'_{l-i+1}\) and \(A\), return this cut value.

**Theorem 5.1:** Algorithm 1 achieves a ratio of \(O(\sqrt{n})\).

**Proof:** The LP defines a fractional cut between the nodes \(S_1, S_2, \ldots, S_k\) and \(A\), and the summation of the accumulated cut value of nodes \(S_1, S_2, \ldots, S_k\) is at least \(l\). If we sort the nodes \(S_1, S_2, \ldots, S_k\) according to their cut value \((Y_{S_i})\) in descending order and there are at least \(l\) nodes in the list than has cut value at least \(1/\sqrt{n}\), then the minimum cut between these \(l\) nodes and \(A\) will be at most \(\sqrt{n}\) times the fractional cut value returned by the LP in the algorithm (denoted as \(c(LP)\)). But \(c(LP) \leq \text{OPT}\), where \(\text{OPT}\) is the minimum threshold node cut value. So in this case the algorithm achieves the ratio \(\sqrt{n}\). If there less than \(l\) nodes in the sorted list that is less than \(1/\sqrt{n}\), in the algorithm...
the minimum cut between $S_1, \ldots, S_{i-1}$ and $A$ is less than $\sqrt{n}OPT$. The minimum cut between $S'_1, \ldots, S'_{i-1}$ and $A$ is also less than $\sqrt{n}OPT$, because there are at most $\sqrt{n}$ nodes (in $S_1, \ldots, S_k$) that have a cut value less than $1/\sqrt{n}$, otherwise the total cut value will be less than $l$ in the LP formulation. So in this case the solution returned by the algorithm will also be at most $2\sqrt{n}OPT$.

Next we present a ratio $O(\log^2 n)$ algorithm for the TMEC problem in Algorithm 2.

**Algorithm 2: The Approximation Algorithm for Threshold Minimum Edge cut**

1. Generate $k - 1$ cliques $\gamma_1, \ldots, \gamma_{k-1}$ with size $n^2$ and one clique $\gamma_k$ with size $(k - 1)n^2$. The cost of all edges in the cliques are set to be $n^2$.
2. for $i = 1..k$
   - Connect node $S_i$ to an arbitrary node in $\gamma_k$ with an edge of cost $n^2$, and connect each of the remaining $S_j$ ($j \neq i$) to one clique $\gamma_u$ ($1 \leq u \leq k - 1$), with an edge of cost $n^2$.
     - for $j = ((2l - 2)n^2 - n + l) \ldots (2(n - 1)n^2 + n - 2)$
       - Generate a clique $\gamma'$ with size $j$. The cost of all edges in $\gamma'$ are set to be $n^2$. Connect node $A$ to an arbitrary node in $\gamma'$ with an edge of cost $n^2$.
       - Denote the current graph (with $l$ + 1 cliques added) as $G'$. Apply the $O(\log^2 n)$ approximation algorithm on $G'$ to find the minimum bisection. Denote the value of the bisection as $B(i, j)$.
3. Find the minimum value of all $B(i, j)$, and return the bisection corresponds to this $B(i, j)$ as the threshold edge cut.

We can prove that this algorithm can achieve an approximation ratio $O(\log^2 n)$.

**Proof:** We can consider the minimum threshold edge cut. Suppose the the minimum threshold edge cut separate $S_{a_1}, \ldots, S_{a_w}$ from $A$ and $w \geq l$. In the minimum threshold edge cut, the maximum number of nodes (in $G$) that can be separated with $A$ can be as large as $n - 1$ ($A$ is separated from all other nodes), and as small as $l$. We denote this number as $\beta$. First we can see that no new added edges in $G'$ will be in the minimum bisection since all the new edges have a cost $n^2$, which is larger than the total number of edges in $G$. When $S_i \in \{S_{a_1}, \ldots, S_{a_w}\}$, one of the minimum bisection (in $G'$) with value $B(i, j)$ ($j$ ranges from $(2l - 2)n^2 - n + l$ to $2(n - 1)n^2 + n - 2$ ) is equivalent to the minimum threshold edge cut. When $\beta$ is $n - 1$, the minimum bisection $B(i, j)$ in $G'$ with $j = 2(k - 1)n^2 + n - 2$ corresponds to the minimum threshold edge cut in $G$, since in this case all the other $n - 1$ nodes in $G$ and all the cliques appended to $S_1, \ldots, S_k$ (total size $2(k - 1)n^2$) will be separated from $A$, but node $A$ and the appended clique of $A$ will have total size $2(k - 1)n^2 + n - 2 + 1 = 2(k - 1)n^2 + n - 1$. When $\beta$ is $l$, the minimum bisection $B(i, j)$ in $G'$ with $j = (2l - 1)n^2 - n + l$ corresponds to the minimum threshold edge cut in $G$, since in this case the $l$ nodes in $S_{a_1}, \ldots, S_{a_l}$ and all the cliques appended to then (total size $(l - 1 + k - 1)n^2$) will be separated from $A$, but there are $n - l + (k - l)n^2 + (2l - 2)n^2 - n + l = (l + k - 2)n^2$ remaining nodes in $G'$. So the minimum bisection in $G'$ corresponds to a minimum threshold edge cut in $G$. For all $\beta$ values between $l$ and $n - 1$, the algorithm will also find the corresponding minimum bisection with appropriate $j$. Since minimum bisection can be solved with approximation ratio $O(\log^2 n)$, the above algorithm also finds the minimum threshold edge cut with ratio $O(\log^2 n)$.

**VI. FUTURE WORK**

The hardness of general case 3-node CPMEC is still open, but we have some interesting observations for the problem:

- It is intriguing that the algorithm does not work for general graphs. It would be interesting to classify the types of graphs that the algorithm can find the optimal cut.
- We already have multiple hardness results for 3-node CPNC but the hardness of 3-node CPMEC is still open since the hardness proof of CPNC cannot be applied to CPMEC. For many other minimum cut based problems, such as the basic minimum cut and the minimum multi-terminal cut there is no big difference between the hardness of node cut and edge cut. So we conjecture that general 3-node CPMEC is also hard to solve, though it is not clear whether an NP-hard proof is available. Even if it is not NP-complete, it may not be in $P$, based on the assumption that $P \neq NP$.
- There are several minimum cut related problems which are NP-hard in general case but have polynomial algorithms in planar graphs. The max-cut problem and the minimum multi-terminal cut problem have polynomial time algorithm in planar graphs [7]. The hardness of Steiner tree problem in planar graphs is still open but it has polynomial time approximation scheme [7]. The minimum multi-way cut in planar graphs is NP-hard but has polynomial time approximation scheme [7]. The hardness of minimum bisection in planar graphs is still open [7]. It is important to investigate what kind of minimum cut related problems in planar graphs can be solved in polynomial time and what is the deep logic behind this. Further research on this can provide guidance on new problems related to planar minimum cut.
- There is another similarity between minimum multi-terminal cut problem and the CPMC in planar graphs. Actually if we adopt the perturbation method, we can have an algorithm for minimum 3-terminal cut in planar graphs. The CPMC problem can be considered as the “complementary” problem of the minimum multi-terminal cut problem. Further investigation of the relationship between the two problems will help the understanding of both problems.
It is rather surprising that the dynamic programming algorithm works for 3-node planar CPMEC. It is interesting to further investigate what kind of constrained minimum cut problem can be solved by similar algorithms. We conjecture that the 3-node CPMEC problem may belong to a class of problems that are neither in \( P \) nor \( NP \)-complete. Thus, the problem may be related to the central question of \( NP \) versus \( P \). For some special and practical graphs, we believe that there may exist efficient precise or approximation algorithms, which will be another future research direction. For TMC problem, there is much room to improve the approximation ratio and hardness result.