A Greedy Algorithm for Minimum Cut into Bounded Sets Problem

1nd Onur Ugurlu
Department of Fundamental Sciences
Izmir Bakircay University
Izmir, Turkey
onur.ugurlu@bakircay.edu.tr

2st Vahid Khalilpour Akram
International Computer Institute
Ege University
Izmir, Turkey
vahid.akram@ege.edu.tr

3rd Deniz Türsel Eliiyi
Department of Industrial Engineering
Izmir Bakircay University
Izmir, Turkey
deniz.eliiyi@bakircay.edu.tr

***Abstract*—Finding critical links and weak points is an important task in almost all types of networks. Minimum cuts provide useful information about the critical links. However, finding a minimum cut of a network may provide insufficient or misleading information on critical links since the number of disconnected nodes in the residual network is not taken into account in this problem. In this work, we study the minimum cut into bounded sets problem, which limits the number of nodes in portioned sets. Finding the minimum cut into bounded sets can provide useful information on important critical links in a different network, whose failure has a hard and unacceptable effect. The minimum cut into bounded sets problem is an open NP-Complete problem. We propose a greedy algorithm for this problem with $O(c \times n^2)$ time complexity and present computational results on random networks. To the best of our knowledge, the proposed algorithm is the first heuristic for the minimum cut into bounded sets problem.**

***Index Terms*—Network Connectivity, Minimum Cut, Critical Edges, Heuristics, Greedy Algorithms**

I. Introduction

Networks are important structures that have deep impacts on daily human life. Everyday, people use different types of networks for different purposes such as transportation, communication, business management, tasks automation, entertainment, and even social relationships. The internet, local area networks, road networks, railways, airlines, water networks, gas networks, power networks, social media networks, and satellite networks are some of the networks, failures, or inefficiency of which cause critical problems. Hence, different strategies, benchmarks, and methods have been developed almost for all types of networks to measure and increase reliability.

Generally, a node or link in a network is called critical if its removal partitions the network to some disconnected parts. In other words, failure of a critical node or a critical link completely separates some other nodes in the network. A network that has critical nodes or critical links is usually unreliable because failure in a single node or single link can destroy the connectivity of the network. Hence, finding the critical nodes or critical links can help to identify and resolve the weak points of the network.

One of the other properties that can help to measure network reliability is the identification of minimum cuts. A cut in any network is a subset of edges whose removal disconnects some nodes from others. A minimum cut is a cut with minimum cardinality. Therefore, the edges in a minimum cut can also be considered as critical because losing the edges in the minimum cut divides the network into the partitioned segments. Minimum cuts reveal useful information about networks, such as weak points, bottlenecks, and clusters. Hence, various studies have investigated the minimum cut problem on different network types. Based on the application, network type, and requirements, different variants of the minimum cut problem have been presented. A minimum s-t cut is the smallest subsets of edges whose removal separates nodes s and t from each other. As another variant, a minimum cut into bounded sets (MCBS) is the smallest set of edges whose removal disconnects special nodes s and t such that the number of nodes in each partition is smaller than a predefined value β.

In this study, we focus on the minimum cut into bounded sets problem and propose a greedy algorithm for finding a solution for the minimum s-t cut. The proposed algorithm finds the minimum s-t cut and the partitioned sets based on the initial cut. Then the algorithm repeatedly moves the nodes from the large set to the small set until all sets have less than the desired number of nodes. The algorithm runs in O($c \times n^2$) time, where n is the number of nodes and c is the initial s-t cut size. To the best of our knowledge, the proposed algorithm is the first algorithm for the minimum cut into bounded sets problem on general networks.

The remainder of this paper have been organized as follows. In Section II, we provide some applications and motivations for the study. Section III presents a brief survey of related works. Section IV includes the problem formulation and some background information. Section V illustrates the steps of the proposed algorithm and Section VI presents the results of performance evaluation on random graphs. Finally, we conclude the study in Section VII.

II. Motivation and Applications

The minimum cut into bounded sets problem has various applications in different network types. In this section, we provide motivations and some applications of the minimum cut into bounded sets problem on the different networks.

978-3-903176-36-2 ©2021 IFIP

A. Computer Networks

A computer network includes a large number of nodes such as client computers, servers, routers, and switches. The links between nodes can be wired cables, wireless channels or virtual connections. Finding the minimum cut between bounded sets of nodes can help to find the clusters or partitions of nodes that have minimal communication paths. Finding these clusters can be useful in distributing various resources such as cache servers and relay nodes in the networks. In addition, finding the minimum cut between bounded sets can be used to find bottlenecks in the network, which can affect the traffic flow between nodes s and t.

B. Wireless Ad-Hoc Networks

In a wireless Ad-Hoc network, the neighbor nodes may connect and communicate over radio messages if they are in the radio range of each other. To communicate with remote nodes, the nodes send messages over multi-hop links. Based on a routing protocol, each node forwards the incoming message to its neighbors until the message reaches the target node. Hence, the failure of some nodes may disconnect the communication paths between other nodes. Finding minimum cuts between bounded sets allows us to find the critical links between a set of nodes. Also, the minimum cuts of bounded sets may reveal clusters and partitions that can be used in routing and resource allocation.

C. Transportation Networks

Transportation networks, including road networks, railway networks, and airlines are important infrastructures that directly affect the daily life quality of people. A reliable and efficient road network can increase urban life quality by reducing transportation time. In a road network, the streets and roads are edges and the intersection of the streets are nodes. Finding the bounded minimum cut of a road network can reveal bottlenecks and critical junctions and roads in the city. Also, finding the bounded sets with minimum connectivity can provide useful information for different tasks such as resource allocation (transfer centers, hospitals, schools, etc.), city management, and disaster management.

D. Call Graphs

A call graph shows the calling relationship between the functions and classes of a computer program. In a call graph, the nodes are functions or classes, and the edges are the calling relationships. Call graphs have a wide range of applications in software engineering and software test tools. They also can be used to measure the complexity and modularity of programs. Detecting the bounded minimum cut of a call graph can reveal independent modules, which mainly call the functions from the same module. Detecting these independent modules is a key task in parallel and distributed systems because running independent modules in different computers can increase program efficiency with minimal communication overhead.

III. Related Works

Finding the critical edges in different type of networks has been the subject of many researches. The depth first search is one of the basic methods that can be used for finding the critical edges in the graphs [1]. After finding the depth first search tree of the graph, the (u, v) edge will be critical if there is no back edge from v or its descendants to vertex u or its ancestors. Also, there are some distributed algorithms that find critical edges in multi-hop networks by finding distributed depth first search tree or distributed breadth first tree in the network [2]–[4]. Besides the critical edges, the minimum cut problem is another well-known problem in graph theory that has many central and distributed approaches [5]–[7]. If a network has some critical edges, finding the minimum cuts can reveal the critical edges of that network because the minimum cut of such networks will be the set that has only one single edge. Hence, detecting the minimum cuts provides more detailed information about the reliability than finding the critical edges. A special case of the minimum cut problem is the minimum s-t cut problem, which aims at finding a minimum cut that disconnects node s from node t. The minimum s-t cut problem can be solved by using the maximum flow from node s to node t in $O(f \times m)$ time complexity [8, 9], where f is the maximum flow and m is the number of edges in the graph. To the best of our knowledge, the fastest maximum-flow-based algorithm for the minimum cut problem has $O(n \times m \times log(n^2/m))$ time complexity [10], where n is the number of nodes.

Some other algorithms use the merging method to find a minimum cut of a given graph. The proposed algorithm in [6] merges the most tightly-connected nodes in the graph until the graph has only two nodes. The cut size of each merging phase is the total weight of the eliminated edges. The final minimum cut is the cut with the smallest size. The time complexity of this algorithm is $O(n \times m + n^2 \times log(n))$. There are also some randomized algorithms that can approximate a close set to the minimum cut with lower time complexity [11, 12]. Despite of the existing various efficient algorithms for the minimum cut problem, to the best of our knowledge, the minimum cut into bounded sets problem on general undirected graphs is an open problem that has no specific algorithm. The only proposed algorithm for the minimum cut into bounded sets problem finds the bounded minimum cut of a cograph in O(n^2) time [13]. A cograph is a special graph that can be generated from single vertex graphs by complementation and disjoint union [14]. The proposed algorithm in [13] finds the minimum cut into bounded sets based on the maximum cuts in the cograph. The author proves that the minimum cut into bounded sets problem can be solved in polynomial time on cographs, but it is NP-Complete in general graphs. In this paper, we propose a greedy algorithm for the problem on general graphs that finds an approximated minimum cut of bounded sets.

IV. Problem Formulation

Any network can be modeled as a graph $G(V, E)$, where V is the set of nodes and E is the set of edges between the nodes.

For example in Fig. 1 we have $V = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ and $E = \{(0, 1), (0, 2), (1, 8), ..., (6, 8)\}$. Before presenting the formal definition of the minimum cut into bounded sets problem, we provide a definition for the minimum s-t cut problem:

***Definition** 1: Given an undirected unweighted graph $G(V, E)$, a minimum cut of G is the smallest subset of edges whose removal disconnects some nodes from the others.*

For example in Fig. 1, $C_1 = \{(0, 1), (0, 2)\}$ is a minimum cut, and we have 4 other minimum cuts (with size 2) in the network. Removing the edges in C_1 partitions the network to $V_1 = \{0\}$ and $V_2 = \{1, 2, 3, 4, 5, 6, 7, 8\}$ disconnected sets where $|V_1| = 1$ and $|V_2| = 8$. Sometimes we are interested in the minimum cuts that separate more than a specific number of nodes from others. For example, if Fig. 1 is an Ad-Hoc network, C_1 just separates a single node from the other nodes, where $C_2 = \{(2, 4), (3, 8)\}$ disconnects nodes $\{0, 1, 2, 8\}$ from the nodes $\{3, 4, 5, 6, 7\}$. Obviously, in most networks, C_2 leads to more serious and complicated situations than C_1 because the communication paths between a large number of nodes are terminated. If we limit the number of nodes in partitioned sets, we may identify the cuts that separate more nodes. For example, If we bound the number of nodes in the partitioned sets to 5, C_1 will be removed from the solution set because C_1 creates a partitioned set with size 8. A minimum cut into bounded sets is a cut that separates the nodes to the partitioned sets with a limited number of nodes. Formally, the minimum cut into bounded sets problem on undirected unweighted graphs can be defined as follows:

***Definition** 2: Given an undirected unweighted graph $G(V, E)$, the nodes $s \in V$ and $t \in V$, and an arbitrary integer $\beta < |V|$, the problem is finding the minimum cut that separates the nodes into $V_1 \subset V$ and $V_2 \subset V$, such that $s \in V$, $t \in V$, $|V_1| \leq \beta$, $|V_2| \leq \beta$.*

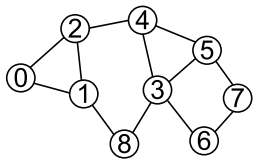


Fig. 1: Sample network with 5 minimum cuts.

For example, in Fig. 1, for $\beta = 3$, $s = 0$, and $t = 6$ we have two minimum cuts to bounded sets as $V_1 = \{(2, 4), (3, 8)\}$ and $\{(1, 8), (2, 4)\}$. A minimum cut into bounded sets finds the minimum cut that can lead to serious problems in the network. For example, in a network with 10000 nodes, losing the connection to a few nodes may be tolerable, while the disconnection of 4000 nodes from others is an undesirable situation definitely. Finding a minimum cut of a given graph is a polynomial problem that can be solved in $O(n \times m \times log(n^2/m))$ time complexity [10]. However, finding the minimum cut into bounded sets is an NP-Complete problem [13, 15]. In the next section, we propose a greedy algorithm for the minimum cut into bounded sets problem.

V. Proposed Algorithm

The minimum cut into bounded sets problem seeks for the minimum cut between a source node s and a target node t under the condition that the size of the disjoint sets of the residual graphs must be less than or equal to a predefined positive integer β (where $\beta < n$). This constraint defines an upper limit for the sizes of V_1 and V_2. When considering the following equation;

$$\beta + \alpha = n \tag{1}$$

α defines a lower limit for the sizes of V_1 and V_2. Thus, the MCBS problem can be considered as finding the minimum cut set S between $s - t$ such that $|V_1| \geq \alpha$ and $|V_2| \geq \alpha$. For $\alpha = 1$ ($\beta = n - 1$), the MCBS is converted into the classical minimum cut problem, which can be solved in polynomial time by a network flow algorithm.

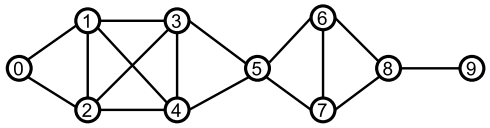
The proposed solution approach is based on the finding the minimum cut between $s - t$ for $\alpha = 1$ and iteratively constructing the V_1 and V_2 with regard to the given α value. After finding an initial solution, If $|V_2| < \alpha$, we add nodes from V_1 to V_2 until we get $|V_2| = \alpha$. In each iteration, among the nodes connected to V_2, the node with the least connections in V_1 is chosen. The pseudo-code of the proposed greedy algorithm is given in Algorithm 1.

Algorithm 1: The Greedy Algorithm for the MCBS

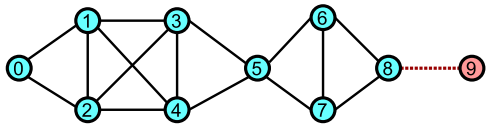
Input : G, s, t, α.
Output: Minimum cut into bounded sets.
1: Find the minimum s-t cut that splits G into two disjoint sets V_s and V_t.
2: $V_1 \leftarrow \max\{|V_s|,|V_t|\}$.
3: $V_2 \leftarrow \min\{|V_s|,|V_t|\}$.
4: **while** $|V_2| < \alpha$ **do**
5: Select the node $v^* \in V_1$ that has an edge to V_2 and has minimum number of neighbors in V_1.
6: $V_1 \leftarrow V_1 \setminus v^*$.
7: $V_2 \leftarrow V_2 \cup v^*$.
8: **end while**
9: **Return** the edges between V_1 and V_2.

Fig. 2 presents the steps of the proposed algorithm on a sample network with 10 nodes. The dashed edges in this figure show the candidate nodes that can be selected in the next iteration. To solve the MCBS on the sample graph, we set $s = 0$, $t = 9$ and $\alpha = 5$. The algorithm starts with finding the minimum cut between $s - t$ without any cardinality constraint for V_1 and V_2. The minimum s-t cut will be $S = \{(8, 9)\}$, thus, we get $|V_1| = 9$ and $|V_2| = 1$ (Fig. 2b). Since $|V_2| < \alpha$, a node must be added from V_1 to V_2. Considering V_1, node 8 is the only node that has a connection in V_2. Therefore, node 8 is deleted from V_1 and added to V_2, and we get $|V_1| = 8$ and $|V_2| = 2$ (Fig. 2c). In the next iteration, nodes 6 and 7 are the nodes that have connection to V_2. Since nodes 6 and 7 have one connection in V_1, we randomly select node 6, and

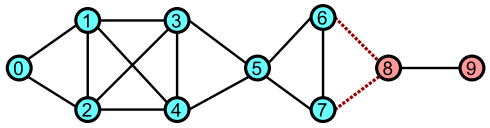
add it to V_2 (Fig. 2d). The procedure continues until the size of V_2 is equal to α and finally we get $V_1 = \{0, 1, 2, 3, 4\}$, $V_2 = \{5, 6, 7, 8, 9\}$ and $S = \{(3, 5), (3, 6)\}$ (Fig. 2f). Note that in Fig. 2, the red dotted edges represent the elements of S.



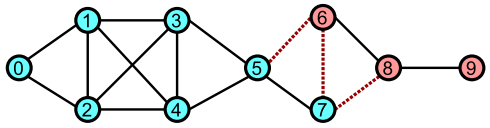
(a) Initial Graph



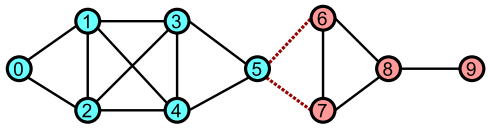
(b) $|V_1| = 9, |V_2| = 1, |S| = 1,$



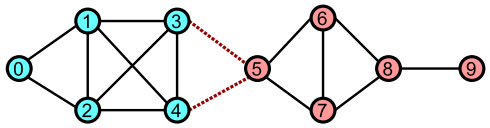
(c) $|V_1| = 8, |V_2| = 2, |S| = 2,$



(d) $|V_1| = 7, |V_2| = 3, |S| = 3$



(e) $|V_1| = 6, |V_2| = 4, |S| = 2$



(f) $|V_1| = 5, |V_2| = 5, |S| = 2$

Fig. 2: The steps of the proposed algorithm.

A. Complexity of the Proposed Algorithm

The first step of the proposed algorithm finds a minimum cut between nodes s and t. We may use the proposed algorithm in [9] to find such a cut in $O(f \times m)$ time complexity, where m is the number of edges and f is the maximum flow between nodes s and t. Since the maximum flow between nodes s and t is equal to the minimum cut size, we may conclude that the time complexity of line 1 is $O(c \times m)$, where c is the minimum cut size between nodes s and t. In the worst case, the 'while' loop (line 4) repeats α times and we may perform a search with $O(n)$ time complexity to find a node from V_1 that has a neighbor in V_2 and also has the minimum number of neighbors in V_1 (line 5). The remaining steps of the algorithm can be done in $O(1)$ time. Therefore, the time complexity of the proposed algorithm is $O(c \times m + \alpha \times n)$. Assuming $m \in O(n^2)$, we may conclude that the time complexity of the proposed algorithm is $O(c \times n^2)$.

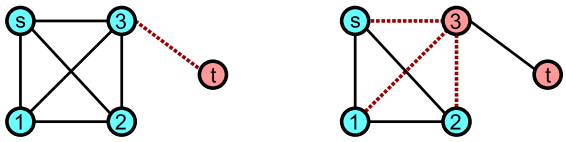
VI. Computational Results

This section presents the simulation results of the proposed greedy algorithm. To the best of our knowledge, there is no proposed algorithm for the simple minimum cut problem into bounded sets on general graphs. Thus, we cannot compare our greedy approach with the existing literature. Instead, we consider a special case, where $\alpha = 1$, as a tight lower bound for the MCBS. By definition, the optimum cut size of MCBS of any α value can not be smaller than the minimum cut size of the $s - t$. For $\alpha = 1$, the optimum solution of the MCBS equals to the minimum $s - t$ cut.

For some special graph types, the optimum solution of the MCBS can equal to minimum s-t cut for all α values. Considering the line graph in Fig. 3, the optimum solution of the MCBS is always equal to the minimum $s - t$ cut.

Fig. 3: A sample line graph.

However, in general, the cut set's size of the MCBS is expected to increase as the α value increases. The worst case of growth of the cut set is given in Fig. 4, where s belongs to a complete subgraph.



(a) $|S| = 1$ (b) $|S| = 3$

Fig. 4: Solutions of the MCBS for $\alpha = 1$ and $\alpha = 2$.

For the sample graph in Fig. 4, $s = v_3$ increases the size of the cut set by 2. Δ can be considered as an upper bound, and δ (the minimum node degree) can be considered as a lower bound for the increase in the size of the cut set. In the general case, each increase in α can be expected to increase the cut set's size by an average degree.

To investigate the performance of our greedy algorithm, we use 24 random graphs with different sizes and densities (the size of the graphs varies from 100 to 200, and the density of the graphs varies from 20 to 80). Note that the density is the ratio of the number of edges to the number of possible edges, and the average degree is the product of the density and size of the graph.

For each graph, we set $s = v_0$ and $t = v_{n-1}$, and we calculate the size of the cut set according to 4 different alpha values: 5,10,20 and 40. Besides, we report the results of $\alpha = 1$

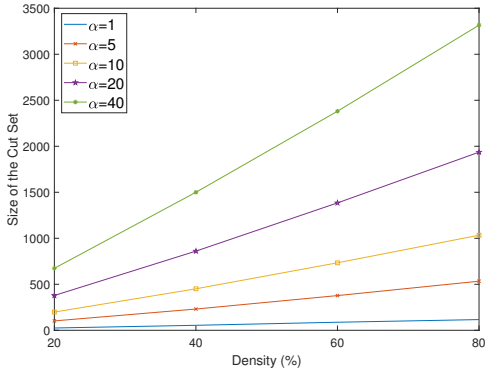


Fig. 5: The average results for each density.

for benchmarking. For a more precise analysis, we average the results according to the density and size of the graphs.

Fig. 5 depicts the average results for each density level. Note that the average graph's size is equal to 150 for each density level. Considering that each increase in α can be expected to increase the cut set's size by an average degree, it is predictable that the size of the cut set of $\alpha = 2c$ is double the size of the cut set of $\alpha = c$. When the results are examined, it can be seen that the solution found for each alpha value is less than double of the previous solution, and this indicates that the proposed greedy algorithm can find acceptable and near-optimal solutions. In addition, as the density increases, the difference between the solutions naturally increases as the average degree increases.

Fig. 6 shows the average results for each graph size. Note that, for each graph's size, the average density is equal to 0.5. It can be said that the size of the cut sets of each α value is a linear function of graph size.

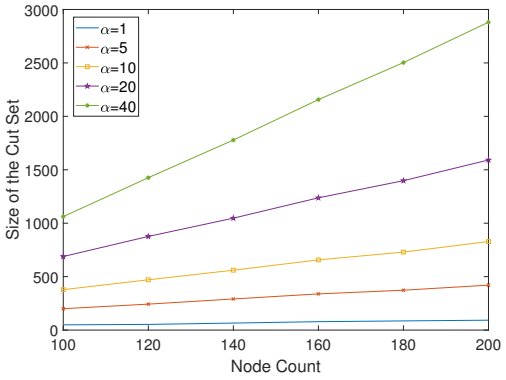


Fig. 6: The average results for each graph size.

VII. Conclusion

In this work, we study the simple minimum cut into bounded sets, which is an NP-Complete problem. Finding the minimum cut into bounded sets can provide useful information about the critical links in a network. Such a set also allows us to ignore unimportant weak links that have very limited negative impacts. Despite of existing various efficient algorithms for the minimum cut and minimum s-t cut problems, the minimum cut into bounded sets problem is an open problem. To the best of our knowledge, there is no specific solution for the minimum cut into bounded sets problem on general graphs. We proposed a greedy algorithm for the MCBS with $O(c \times n^2)$ complexity and presented the computational results on random graphs. To the best of our knowledge, the proposed algorithm is the first heuristic algorithm for the MCBS on general graphs.

As a future work, we plan to create a benchmark set with known optimum solutions and test the effectiveness of our greedy algorithm by more comprehensive computational analysis. Also, improving the algorithm to find closer solutions to optimal and finding the approximation ratio of the proposed algorithm can be considered as the future works of this study.

Acknowledgment

This research was supported by the TUBITAK (Scientific and Technical Research Council of Turkey) [Project number 121F092].

References

[1] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.

[2] O. Dagdeviren and V. K. Akram, "Energy-efficient bridge detection algorithms for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 9, no. 4, p. 867903, 2013.

[3] B. Milic and M. Malek, "Adaptation of the breadth first search algorithm for cut-edge detection in wireless multihop networks," in *Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, 2007, pp. 377–386.

[4] V. K. Akram and O. Dagdeviren, "Breadth-first search-based single-phase algorithms for bridge detection in wireless sensor networks," *Sensors*, vol. 13, no. 7, pp. 8786–8813, 2013.

[5] M. Brinkmeier, "A simple and fast min-cut algorithm," *Theory of Computing Systems*, vol. 41, no. 2, pp. 369–380, 2007.

[6] M. Stoer and F. Wagner, "A simple min-cut algorithm," *Journal of the ACM*, vol. 44, no. 4, pp. 585–591, 1997.

[7] V. K. Akram, "An asynchronous distributed algorithm for minimum s-t cut detection in wireless multi-hop networks," *Ad Hoc Networks*, vol. 101, p. 102092, 2020.

[8] G. Dantzig and D. R. Fulkerson, "On the max flow min cut theorem of networks," *Linear inequalities and related systems*, vol. 38, pp. 225–231, 2003.

[9] L. Ford and D. Fulkerson, "Maximal flow through a network," *Canadian journal of Mathematics*, vol. 8, no. 3, pp. 399–404, 1956.

[10] J. Hao and J. B. Orlin, "A faster algorithm for finding the minimum cut in a directed graph," *Journal of Algorithms*, vol. 17, no. 3, pp. 424–446, 1994.

[11] D. R. Karger and C. Stein, "A new approach to the minimum cut problem," *Journal of the ACM*, vol. 43, no. 4, pp. 601–640, 1996.

[12] A. A. Benczúr and D. R. Karger, "Approximating st minimum cuts in õ (n 2) time," in *Twenty-eighth annual ACM symposium on Theory of computing*. ACM, 1996, pp. 47–55.

[13] H. L. Bodlaender, *The maximum cut and minimum cut into bounded sets problems on cographs*. Unknown Publisher, 1987, vol. 87.

[14] C. A. Christen and S. M. Selkow, "Some perfect coloring properties of graphs," *Journal of Combinatorial Theory, Series B*, vol. 27, no. 1, pp. 49–59, 1979.

[15] M. R. Garey and D. S. Johnson, "Computers and intractability," *A Guide to the*, 1979.