The Minimum Vulnerability Problem

Sepehr Assadi[1], Ehsan Emamjomeh-Zadeh[1], Ashkan Norouzi-Fard[1],
Sadra Yazdanbod[1], and Hamid Zarrabi-Zadeh[1,2]*

[1]Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.
{s_asadi,emamjomeh,noroozifard,yazdanbod}@ce.sharif.edu,
zarrabi@sharif.edu
[2]Institute for Research in Fundamental Sciences (IPM), Tehran, Iran.

Abstract. We revisit the problem of finding k paths with a minimum number of shared edges between two vertices of a graph. An edge is called *shared* if it is used in more than one of the k paths. We provide a $\lfloor k/2 \rfloor$-approximation algorithm for this problem, improving the best previous approximation factor of $k-1$. We also provide the first approximation algorithm for the problem with a sublinear approximation factor of $O(n^{3/4})$, where n is the number of vertices in the input graph. For sparse graphs, such as bounded-degree and planar graphs, we show that the approximation factor of our algorithm can be improved to $O(\sqrt{n})$. While the problem is NP-hard, and even hard to approximate to within an $O(\log n)$ factor, we show that the problem is polynomially solvable when k is a constant. This settles an open problem posed by Omran *et al.* regarding the complexity of the problem for small values of k. We present most of our results in a more general form where each edge of the graph has a sharing cost and a sharing capacity, and there is vulnerability parameter r that determines the number of times an edge can be used among different paths before it is counted as a shared/vulnerable edge.

1 Introduction

In this paper, we investigate a family of NP-Hard network design problems. Our study is motivated by the *minimum shared edges* (MSE) problem, formally defined as follows:

Problem 1 (Minimum Shared Edges). Given a directed graph $G = (V, E)$, an integer $k > 0$, and two distinct vertices s and t in V, find k paths from s to t minimizing the number of shared edges. An edge is called *shared* if it is used in more than one of the k paths.

The minimum shared edges problem arises in a number of transportation and communication network design problems. As an example, consider a VIP who wishes to travel safely between two places of a network (see [10]). To achieve a minimum level of security assurance, the usual strategy is to pre-select k paths,

* This author's research was partially supported by IPM under grant No: CS1391-4-04.

and then, choose one of the k paths at random just before the actual trip. To bound the probability of being attacked by an adversary (who knows the strategy and the paths) to at most $1/k$, we need to put guards on high-risk edges, i.e., those edges shared among more than one of the pre-selected paths. To reduce the guarding cost, the obvious objective is to find paths with a minimum number of shared edges. A similar problem arises in the context of communication network design, e.g., in designing reliable client-server networks [13], reliable multicast communications [11], and distributed communication protocols [3].

In this work, we obtain results for a generalized version of the minimum shared edges problem. More precisely, we generalize MSE (Problem 1) in three directions. Firstly, we assign a cost c_e to each edge e, which represents the cost of guarding the edge. This weighted version is closer to the practical applications, in which guarding edges have different costs, depending on, say, the length of the edges. Secondly, we make the problem capacitated by assigning to each edge an upper bound specifying the maximum number of times an edge can be used among the k paths. Thirdly, we generalize the problem by adding a parameter r that specifies a threshold on the number of times an edge can be used before it becomes vulnerable, and needs to be guarded. The generalized problem, which we call *minimum vulnerability*, is formally defined as follows:

Problem 2 (Minimum Vulnerability). Given a directed graph $G = (V, E)$ with nonnegative edge costs c_e and maximum edge capacities U_e assigned to the edges $e \in E$, two distinct vertices $s, t \in V$, and two integers r and k with $0 \leqslant r < k$, find k paths from s to t so as to minimize the total cost of r-vulnerable edges. An edge is called *r-vulnerable* if it is used in more than r of the k paths.

Clearly, the minimum 1-vulnerability problem (i.e., when $r = 1$) is equivalent to the weighted capacitated MSE problem. Furthermore, the minimum 0-vulnerability problem is equivalent to the classic *minimum edge-cost flow* (MECF) problem, in which we are given a graph $G = (V, E)$ with nonnegative edge costs and capacities, and the goal is to find a min-cost subset $A \subseteq E$ so that the flow from s to t in (V, A) is at least a given value k. The MECF problem is one of the fundamental NP-hard problems in network design (see Garey and Johnson [4]). It includes several other interesting problems as special case, such as the Steiner tree problem [4] and some of its generalizations [5, 8].

Previous Work. The best previous approximation algorithm for the MSE problem has an approximation factor of $k - 1$ [10], which is based on a k-approximation algorithm for the MECF problem, proposed by Krumke *et al.* [9]. Both the MSE and MECF problems are known to be hard to approximate to within a factor of $2^{\log^{1-\varepsilon} n}$, for any constant $\varepsilon > 0$ [2, 10].

A restriction of the minimum vulnerability problem to the case where no r-vulnerable edge ($r > 0$) is allowed is equivalent to the well-known *disjoint paths* problem, which can be solved polynomially using a standard maximum flow algorithm (e.g., [6]). A closely related problem studied in the literature [13, 14] is the *minimum sharability* problem in which the cost of sharing each edge is equal to the number of times the edge is shared (i.e., the flow of the edge minus one) times the cost of the edge. This sharability problem can be solved efficiently using minimum-cost flow algorithms. Another related problem is the *fixed-charge flow* problem in which each edge has a fixed building cost as well as a per-unit flow cost, and the objective is to select a subset of edges to route a flow of size k between two nodes s and t such that the total cost of building the network and sending the flow is minimized. The best current approximation factor for this problem is $\beta(G) + 1 + \varepsilon$ where $\beta(G)$ is the size of a maximum s-t cut in the graph [1].

Our results. In this paper, we study the minimum vulnerability problem as a generalization of the MSE and MECF problems, and obtain several results, a summary of which is listed below.

- We present a primal-dual algorithm for the minimum r-vulnerability problem that achieves an approximation factor of $\lfloor \frac{k}{r+1} \rfloor$. This improves, in particular, the best previous approximation factor of $k - 1$ for the MSE problem to $\lfloor k/2 \rfloor$. It also yields an alternative k-approximation algorithm for the MECF problem.
- We show that for any $r \geqslant 0$ and $\varepsilon > 0$, the minimum r-vulnerability problem is hard to approximate to within a factor of $2^{\log^{1-\varepsilon} n}$ unless NP $\subseteq$ DTIME$(n^{\text{polylog}\, n})$. This eliminates the possibility of obtaining a poly-logarithmic approximation factor for the minimum vulnerability problem.
- Despite the fact that the minimum vulnerability problem is NP-hard (and even hard to approximate), we show that for any constant k and any $r > 0$, the minimum r-vulnerability problem can be solved exactly in polynomial time. This settles an open problem posed by Omran *et al.* [10] regarding the complexity of the MSE problem for small values of k. Our result indeed shows that the hardness of the minimum r-vulnerability problem, for any $r > 0$, crucially relies on the number of paths in the problem instance.
- For the MSE problem, we present an approximation algorithm that achieves an approximation guarantee of $O(n^{3/4})$, where n is the number of vertices in the graph. This improves upon the trivial factor-n approximation available for the problem, and is the first algorithm for the problem with a sublinear approximation factor. When the input graph is sparse—which is the case in most real-world applications, e.g., in road-map networks with bounded vertex-degrees—we show that the approximation factor of our algorithm can be further improved to $O(\sqrt{n})$.

Our results are mainly based on a clever use of max-flow min-cut duality. In Section 2, we use a primal-dual method to pick a bounded-cost set of edges, out of which the final vulnerable edges are selected. In Section 3, we find an ordered set of min-cuts that leads to an exact solution to the minimum r-vulnerability problem for any fixed k via a dynamic programming approach. In Section 4, we use a combination of the primal-dual method and a shortest path algorithm to obtain the first sublinear approximation factor for the MSE problem.

2 A Primal-Dual Algorithm

In this section, we present a primal-dual[1] algorithm for the minimum r-vulnerability problem with an approximation factor of $\lfloor \frac{k}{r+1} \rfloor$.

A *s-t cut* is defined as a minimal set of edges whose removal disconnects t from s. Let S be the set of all s-t cuts of size less than $\lceil k/r \rceil$ in G. For the special case of $r = 0$, we define S to be the set of all s-t cuts in G. An obvious constraint is that in any feasible solution, at least one edge from each cut $C \in S$ must be r-vulnerable. If not, at most $(\lceil k/r \rceil - 1) \times r < k$ paths can pass through C, making the solution infeasible. Let x_e be a 0/1 variable which is set to 1 if edge e is r-vulnerable in our solution, and is set to 0 otherwise. The minimum vulnerability problem with no capacity bounds (i.e., when $U_e = \infty$ for all edges) can be expressed as the following integer program:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e & & \qquad \text{(IP)} \\ \text{s.t.} \quad & \sum_{e \in C} x_e \geqslant 1 & \forall\, C \in S & \\ & x_e \in \{0,1\} & \forall\, e \in E & \end{aligned}$$

We relax the integer program to a linear program by replacing the constraint $x_e \in \{0,1\}$ with $x_e \geqslant 0$. The following is the dual of the resulting linear program:

$$\begin{aligned} \max \quad & \sum_{C \in S} y_C & \\ \text{s.t.} \quad & \sum_{C \ni e} y_C \leqslant c_e & \forall\, e \in E \\ & y_C \geqslant 0 & \forall\, C \in S \end{aligned}$$

Our primal-dual algorithm is presented in Algorithm 1. We start with a feasible dual solution $y = 0$, and an empty set of vulnerable edges R, that represents an infeasible primal solution. We initialize the capacity u_e of each edge to r, allowing each edge to pass at most r paths initially. We then iteratively improve the feasibility of the primal solution by choosing a s-t cut C whose capacity is less than k, and increase its corresponding variable y_C, until a dual constraint $\sum_{C \ni e} y_C \leqslant c_e$ becomes tight for some edge e. We then add e to the set of vulnerable edges, and set its capacity to U_e. The loop is terminated when all s-t cuts have capacity at least k, admitting a s-t flow f of value k, which is returned as the final solution.

Let OPT be the cost of an optimal solution for the minimum vulnerability problem, let Z_{IP} be the optimal value of the objective function of (IP), and APX be the cost of the solution returned by our algorithm. Obviously, $Z_{\text{IP}} \leqslant$ OPT, because every feasible solution to the capacitated problem is also a feasible solution for the uncapacitated one. We further prove the following.

[1] Readers not familiar with the primal-dual framework are referred to the textbooks on approximation algorithms, e.g., [12].

Algorithm 1 PRIMAL-DUAL

```
1: $y \leftarrow 0$, $R \leftarrow \emptyset$
2: set $u_e \leftarrow r$ for all $e \in E$
3: while there exists a $s$-$t$ cut $C$ of capacity less than $k$ in $G$ do
4:     increase $y_C$ until $\sum_{C \ni e} y_C = c_e$ for some edge $e$
5:     $R \leftarrow R \cup \{e\}$, $u_e \leftarrow U_e$
6: find an integral $s$-$t$ flow $f$ of value $k$ in graph $G$ with edge capacities $u_e$
7: return $f$
```

Lemma 3. $\mathrm{APX} \leqslant \lfloor \frac{k}{r+1} \rfloor \mathrm{OPT}$.

Proof. Let T be the set of edges carrying a flow more than r in f. Clearly, $T \subseteq R$. Now,

$$\begin{aligned}
\mathrm{APX} &= \sum_{e \in T} c_e \\
&= \sum_{e \in T} \sum_{C \ni e} y_C && \text{(by line 4 of algorithm)} \\
&= \sum_{C \in S} y_C \times |\{e \in T \cap C\}| \\
&\leqslant \left\lfloor \frac{k}{r+1} \right\rfloor \sum_{C \in S} y_C && (*) \\
&\leqslant \left\lfloor \frac{k}{r+1} \right\rfloor Z_{\mathrm{IP}} && \text{(by weak duality)}
\end{aligned}$$

where the inequality (*) holds, because at most $\lfloor \frac{k}{r+1} \rfloor$ edges of each cut C can have a flow more than r in f. The lemma follows by the fact $Z_{\mathrm{IP}} \leqslant \mathrm{OPT}$. □

Theorem 4. *There is a $\lfloor \frac{k}{r+1} \rfloor$-approximation algorithm for the minimum vulnerability problem that runs in $O(nm^2 \log(n^2/m))$ time on a graph with n vertices and m edges.*

Proof. The approximation factor of Algorithm 1 follows from Lemma 3. The main loop iterates at most m times. At each iteration, we need to compute a min-cut, which can be done in $O(nm \log(n^2/m))$ time [7]. Line 4 involves comparing at most k values, taking $O(k) = O(n)$ time. The total time is therefore $O(nm^2 \log(n^2/m))$. □

3 An Exact Algorithm for Fixed k

The minimum vulnerability problem is not only NP-hard, but is also hard to approximate to within a factor of $2^{\log^{1-\varepsilon} n}$, for any $\varepsilon > 0$ (the proof is omitted

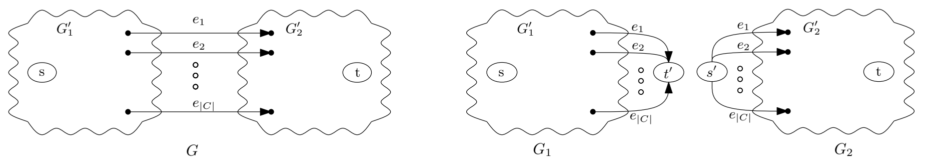

Fig. 1. A graph G with a s-t cut C is divided into two graphs G_1 and G_2.

in this version). Despite this fact, we show in this section that if k is a constant, then the minimum r-vulnerability problem, for any $r > 0$, can be solved exactly in polynomial time.

Given a directed graph G and a s-t cut C, we say that an edge $e = (u, v) \notin C$ is *before* C if there is a path from s to u not using any edge of C, and we say that e is *after* C if a path exists from v to t with no edge from C. Note that an edge cannot be both before and after C because C is a s-t cut. Given two s-t cuts C_1 and C_2, we write $C_1 \leqslant C_2$ if each edge of C_1 is either before or in C_2.

Consider an instance of the minimum r-vulnerability problem. We call a *capacity function* $u : E \to \mathbb{Z}^+$ *proper* if there exists a s-t flow f of value k such that $f(e) \leqslant u(e)$ for all edges $e \in E$. A proper capacity function is *minimal* if decreasing the capacity of any edge e with $u(e) > r$ makes u improper.

Lemma 5. *Given a minimal capacity function u, a sequence $C_1 \leqslant \cdots \leqslant C_\gamma$ of s-t cuts can be found such that $\sum_{e \in C_i} u(e) = k$ for all $1 \leqslant i \leqslant \gamma$, and that each edge $e \in E$ with $u(e) > r$ lies in at least one of the γ cuts.*

Proof. Pick an arbitrary edge $e \in E$ with $u(e) > r$ such that its head is not t and its tail is not s. If no such e exists, we are done. There must exist a s-t cut C containing e such that $\sum_{e \in C} u(e) = k$ by the minimality of u. We construct a graph G_1 from G by removing all edges after C, and then, merging all heads of the edges in C into a new vertex t' (see Figure 1). Similarly, we construct G_2 from G by removing all edges before C, and merging all tails of the edges in C into a new vertex s'. For any set P of k paths from s to t, all edges of P are either in G_1 or G_2. Therefore, the problem of finding k paths from s to t in G can be reduced into two subproblems: finding k paths from s to t' in G_1 and finding k paths from s' to t in G_2. By induction, there exists a sequence of cuts for each of G_1 and G_2 as stated in the lemma. Therefore, the sequence of cuts in G_1, followed by C and then the sequence of cuts in G_2 yields the desired cut sequence. □

Theorem 6. *If k is a constant, then the minimum r-vulnerability problem can be solved exactly in polynomial time for any $r > 0$.*

Proof. We define a *state* as a pair (C, θ), where C is a s-t cut, and θ is a $|C|$-tuple with $\sum_{e \in C} \theta_e = k$ and $\theta_e \leqslant U_e$. A set of k paths from s to t, represented by a s-t flow f of value k, is *suitable* for the state (C, θ) if $f(e) \leqslant \theta_e$ for all $e \in C$, and $f(e) \leqslant U_e$ for all $e \in E \setminus C$.

Algorithm 2 FIND-COST(C, θ)

```
1:  $\text{cost}_C \leftarrow \sum_{e \in C, \theta_e > r} c_e$
2:  if $(C, \theta)$ is a final state then
3:      return $\text{cost}_C$
4:  for each edge $e \in C$ do
5:      if $\theta_e > U_e$ then
6:          return $\infty$
7:  ans $\leftarrow \infty$
8:  for each cut $C'$ with $C \leqslant C'$ do
9:      for each $\theta' = (\theta'_1, \theta'_2, \ldots, \theta'_{|C'|})$ with $\sum_{i=1}^{|C'|} \theta'_i = k$ do
10:         if $(C', \theta')$ is immediately after $(C, \theta)$ then
11:             ans $\leftarrow$ min{ans, FIND-COST$(C', \theta') + \text{cost}_C$}
12: return ans
```

The *answer* to the state (C, θ) is defined as the minimum total cost of r-vulnerable edges either in C or after it, over all suitable sets of k paths. Obviously, if there exists a suitable set of k paths for a state (C, θ) such that no edge after C is r-vulnerable, then the answer to this state is equal to the total cost of r-vulnerable edges in C. We call such states the *final states*.

A state (C', θ') is *immediately after* (C, θ), if $C \leqslant C'$, and there exists a set of k paths suitable for both (C, θ) and (C', θ'), with no r-vulnerable edge between C and C' (i.e., after C and before C'). Given two states (C, θ) and (C', θ'), we define the following capacity function:

$$w(e) = \begin{cases} \theta_e & \text{if } e \in C \\ \theta'_e & \text{if } e \in C' \\ \min\{U_e, r\} & \text{if } e \text{ between } C \text{ and } C' \\ U_e & \text{otherwise} \end{cases}$$

Observe that the maximum flow in graph G with capacity function w is at least k if and only if (C', θ') is immediately after (C, θ). We use this observation to check whether a state is immediately after another one.

Algorithm 2 computes the answer to the state (C, θ) recursively, based on the answers to the states immediately after it. The algorithm works as follows. Given a state (C, θ), we find all states (C', θ') immediately after (C, θ), solve the problem recursively for (C', θ'), add the cost of the current cut (i.e., sum of costs of edges $e \in C$ with $\theta_e > r$), and then return the minimum of all these values. The final answer to the problem is the minimum answer to the states (C, θ) such that there exists a suitable set of k paths with no r-vulnerable edge before C.

The correctness of the algorithm follows from Lemma 5, since all possible sequences of s-t cuts that can be the sequence of cuts in an optimal solution are examined by the algorithm. We can use dynamic programming to store the answers to the states, and avoid recomputing. Note that the number of s-t cuts of size less than k is $O(m^{k-1})$ (where $m = |E|$), and the number of solutions to $\sum_{i=1}^{|C|} \theta_i = k$ is at most $\binom{2k-1}{k-1} = O(1)$, implying that the number of states is $O(m^{k-1})$. On the other hand, checking if a state is final, and checking if a state is immediately after another one can be both done in $O(n^3)$ time using a standard max flow algorithm. Therefore, the total running time of the algorithm is $O(m^{2(k-1)}n^3)$. □

4 A Sublinear Approximation Factor

In this section, we present an approximation algorithm for the MSE problem (Problem 1) with a sublinear approximation factor. Our algorithm is a combination of the primal-dual method presented in Section 2 and a simple shortest path algorithm. The pseudo-code is presented in Algorithm 3.

Algorithm 3 SUBLINEAR-APPROX

```
1: let $P_1$ be the output of Algorithm 1, having $w$ shared edges
2: let $P_2$ be a shortest $s$-$t$ path of length $\ell$
3: **return** $P_1$ if $w < \ell$ else $P_2$
```

The feasibility of the returned solution is clear, as we can route all the k paths through a shortest s-t path. Let P^* be an optimal solution to the MSE problem with a minimum number of used edges (i.e., edges carrying non-zero flow). Let $\mathcal{D}$ be the graph induced by P^*, and m^* be the number of its edges. Denote by OPT the number of shared edges in any optimal solution. We assume, w.l.o.g., that OPT $\neq 0$.

Lemma 7. *$\mathcal{D}$ is a DAG.*

Proof. Suppose that there is a cycle in $\mathcal{D}$. Reduce the capacity of each edge in the cycle by the minimum amount of flow along the edges of the cycle. This results in decreasing the number of edges in $\mathcal{D}$ by at least one without increasing the number of shared edges, contradicting the minimality of the number of edges in $\mathcal{D}$. □

Lemma 8. $\frac{k\ell - m^*}{k} \leqslant \text{OPT}$*, where ℓ is the length of a shortest s-t path.*

Proof. Let f be a s-t flow of value k in $\mathcal{D}$. We have:

$$\sum_{e \in E} \max\{0, f(e) - 1\} \leqslant k\text{OPT}, \tag{1}$$

where the the left-hand side counts the number of shared edges in f. On the other hand, any shared edge can be in at most k paths, so the inequality above holds. Furthermore:

$$\sum_{e \in E} \max\{0, f(e) - 1\} = \sum_{e \in E} f(e) - m^*. \tag{2}$$

The length of every s-t path is at least ℓ. Therefore, the total number of edges used in the k paths is at least $k\ell$. Combined with (1) and (2) we get:

$$k\ell - m^* \leqslant \sum_{e \in E} f(e) - m^* \leqslant k\mathrm{OPT}.$$

□

Lemma 9. *Let $G = (V, E)$ be a DAG with n vertices such that for all, but k vertices, in-degree equals out-degree. Then G has $O(kn\sqrt{n} + k^2)$ edges.*

(Proof can be found in the full version.)

Theorem 10. *Algorithm 3 is an $O(\min\{n^{\frac{3}{4}}, m^{\frac{1}{2}}\})$-approximation algorithm for the MSE problem.*

Proof. Let $\alpha = \ell/\mathrm{OPT}$ be the approximation factor achievable by just returning the shortest path. Algorithm 3 has therefore an approximation factor of $\min\{k, \alpha\}$. We consider two cases.

CASE 1. $k\ell \geqslant 2m^*$. By Lemma 8, $\alpha \leqslant \frac{k\ell}{k\ell - m^*}$. Therefore,

$$\alpha \leqslant \frac{k\ell}{k\ell - m^*} \leqslant \frac{2m^*}{2m^* - m^*} = 2.$$

Hence, $\min\{k, \alpha\} \leqslant 2$ in this case.

CASE 2. $k\ell < 2m^*$. In this case

$$k\alpha \leqslant k\alpha\mathrm{OPT} = k\ell < 2m^* \leqslant 2m.$$

Therefore, $\min\{k, \alpha\} < \sqrt{2m}$, implying an approximation factor of $O(\sqrt{m})$.

We can put a second upper bound on the approximation factor. Consider the DAG $\mathcal{D}$ defined earlier in this section. In this graph, except for the end-points of shared edges and s and t, all other vertices have equal in/out degrees, because each path enters a vertex with an edge, and exits it with another edge. By Lemma 9, the number of edges in $\mathcal{D}$ is $O(n\sqrt{n}\mathrm{OPT} + \mathrm{OPT}^2)$. Observe that if $\mathrm{OPT} \geqslant \sqrt{n}$, a shortest path is a $\sqrt{n}$-approximation to MSE, because $\ell \leqslant n$. If $\mathrm{OPT} < \sqrt{n}$, then m^* is upper-bounded by $O(n\sqrt{n}\mathrm{OPT})$, resulting in:

$$k\ell < 2m^* < cn\sqrt{n}\mathrm{OPT} \;\Rightarrow\; k\alpha\mathrm{OPT} < cn\sqrt{n}\mathrm{OPT}$$

$$\Rightarrow\; k\alpha < cn\sqrt{n} \;\Rightarrow\; \min\{k, \alpha\} < \sqrt{cn\sqrt{n}}$$

Combined with the previous result we get $\min\{k, \alpha\} = O(\min\{n^{\frac{3}{4}}, m^{\frac{1}{2}}\})$. □

Corollary 11. *For sparse graphs with $m = O(n)$, such as planar graphs and bounded-degree graphs, Algorithm 3 yields an $O(\sqrt{n})$ approximation factor.*

5 Conclusion

In this paper, we introduced the minimum vulnerability problem which is an extension of the two previously-known problems, MECF and MSE. We obtained a $\lfloor \frac{k}{r+1} \rfloor$-approximation algorithm for the problem in general form, and the first sublinear approximation factor for the MSE problem. While the problem is hard to approximate, we showed that the minimum $(r > 0)$-vulnerability problem can be solved exactly in polynomial time for any fixed k. We leave this question open whether a same poly-time algorithm can be obtained for the case of $r = 0$, i.e., for the MECF problem. Another open problem is whether better approximation factors can be obtained for MSE, and in general, for the minimum vulnerability problem.

References

1. R. D. Carr, L. K. Fleischer, V. J. Leung, and C. A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proc. 11th ACM-SIAM Sympos. Discrete Algorithms*, pages 106–115, 2000.
2. G. Even, G. Kortsarz, and W. Slany. On network design problems: fixed cost flows and the covering steiner problem. *ACM Trans. Algorithms*, 1(1):74–101, 2005.
3. M. K. Franklin. *Complexity and security of distributed protocols.* PhD thesis, Dept. of Computer Science, Columbia University, 1994.
4. M. Garey and D. S. Johnson. *Computers and intractability : A guide to the theory of NP-completeness.* W. H. Freeman, 1979.
5. N. Garg, R. Ravi, and G. Konjevod. A polylogarithmic approximation algorithm for the group Steiner tree problem. *J. Algorithms*, 37(1), 2000.
6. A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. *J. ACM*, 45(5):783–797, 1998.
7. A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35(4):921–940, 1988.
8. G. Konjevod, R. Ravi, and A. Srinivasan. Approximation algorithms for the covering steiner problem. *Random Structures & Algorithms*, 20(3):465–482, 2002.
9. S. O. Krumke, H. Noltemeier, S. Schwarz, H.-C. Wirth, and R. Ravi. Flow improvement and network flows with fixed costs. In *Proc. Internat. Conf. Oper. Res.: OR-98*, pages 158–167, 1998.
10. M. T. Omran, J.-R. Sack, and H. Zarrabi-Zadeh. Finding paths with minimum shared edges. In *Proc. 17th Annu. Internat. Conf. Computing and Combinatorics*, volume 6842 of *Lecture Notes Comput. Sci.*, pages 567–578, 2011.
11. J. Wang, M. Yang, B. Yang, and S. Q. Zheng. Dual-homing based scalable partial multicast protection. *IEEE Trans. Comput.*, 55(9):1130–1141, 2006.
12. D. P. Williamson and D. B. Shmoys. *The design of approximation algorithms.* Cambridge University Press, 2011.
13. B. Yang, M. Yang, J. Wang, and S. Q. Zheng. Minimum cost paths subject to minimum vulnerability for reliable communications. In *Proc. 8th Internat. Symp. Parallel Architectures, Algorithms and Networks*, ISPAN'05, pages 334–339. IEEE Computer Society, 2005.
14. S. Q. Zheng, J. Wang, B. Yang, and M. Yang. Minimum-cost multiple paths subject to minimum link and node sharing in a network. *IEEE/ACM Trans. Networking*, 18(5):1436–1449, 2010.