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 $k-1$ available for the problem. 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 a $O(\log n)$ 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, which is defined as follows:

Problem 1 (Minimum Shared Edges). Given a graph $G = (V, E)$, an integer $k > 0$, and two distinct vertices s and t of 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 [11]). To achieve a minimum level of security assurance, the usual strategy is to pre-select k paths, and then, choose one of the k paths at random just before the actual travel. 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 guard 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, for example, in designing reliable client-server networks [13], reliable multicast communications [12], and distributed communication protocols [4].

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 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 [5]). It includes several other interesting problems as special case, such as the Steiner tree problem [5] and some of its generalizations [6, 9].

Previous Work. The best previous approximation algorithm for the MSE problem has an approximation factor of $k - 1$ [11], which is based on a k-approximation algorithm for the MECF problem, proposed by Krumke *et al.* [10]. Both the MSE and MECF problems are shown to be hard to approximate to within a factor of $2^{\log^{1-\varepsilon} n}$, for any constant $\varepsilon > 0$ [3, 11].

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., [7]). 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 (e.g., [1]). 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 [2].

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.* [11] 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 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 practical instances of the problem where the input is a real road-map network 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 us 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 algorithm for the minimum r-vulnerability problem with an approximation factor of $\lfloor \frac{k}{r+1} \rfloor$.

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. Consider a *s-t cut* to be 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 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 a cut, making the solution infeasible. The minimum vulnerability problem with no capacity bounds (i.e., when $U_e = \infty$ for all edges) can be therefore expressed as the following integer program:

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

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 dual solution $y = 0$, which is feasible because $c_e \geqslant 0$ for all edges $e \in E$, and an empty set of vulnerable edges V, 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, 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, $Z_{\text{IP}} \leqslant \text{OPT}$, because every feasible solution to the capacitated problem is also a feasible solution for the uncapacitated one. We further prove the following.

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

Algorithm 1 PRIMAL-DUAL

1: $y \leftarrow 0$, $V \leftarrow \emptyset$
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: $V \leftarrow V \cup \{e\}$, $u_e \leftarrow U_e$
6: find an integral s-t flow f of value k in G
7: **return** f

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

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

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 $O(nm(n^2/m))$ time [8]. 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 $\varepsilon > 0$ (see appendix). 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.

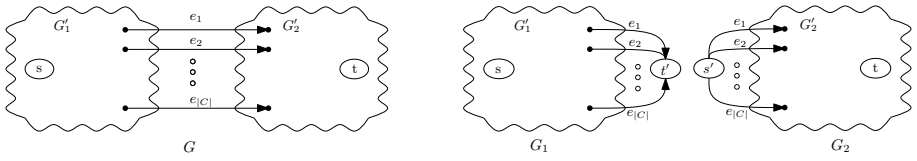

Fig. 1. A graph G with a s-t cut C is divided into two graphs G_1 and G_2.

Given a directed graph G and a s-t cut C, we say that an edge $e = (u, v)$ 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. For two s-t cuts C_1 and C_2, we write $C_1 \leqslant C_2$ if all edges of C_1 are either before or on 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 neither before C nor on it, 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 neither after C nor on it, and merging all tails of the edges in C into a new vertex s'. We claim that for any set P of k paths from s to t, all edges in P are either in G_1 or G_2 or in both. Assume, by way of contradiction, that an edge $e = (u, v) \in E$ is neither before C, nor after it, nor on it. According to our definition of before and after, any path from s to u passes through C. The same holds for any path from v to t. If e is on any path of P, then that path must go through C more than once, contradicting the fact that the total capacity of C is k.

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 graphs 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 $C = \{e_{i_1}, \ldots, e_{i_{|C|}}\}$ is a s-t cut and θ is a $|C|$-tuple with $\sum_{j=1}^{|C|} \theta_j = k$ and $\theta_j \leqslant U(e_{i_j})$. A set of k paths from

Algorithm 2 EXACT-COST(C, θ)

```
1: cur ← ∑_{e_{i_j} ∈ C, θ_j > r} c(e_{i_j})
2: if (C, θ) is a final state then
3:     return cur
4: for each 1 ⩽ j ⩽ |C| do
5:     if θ_j > U(e_{i_j}) then
6:         return ∞
7: ans ← ∞
8: for each cuts C' in G with C ⩽ C' do
9:     for each θ' = (θ'_1, θ'_2, ..., θ'_{|C'|}) where ∑_{i=1}^{|C|} θ'_i = k do
10:        if (C', θ') is next to (C, θ) then
11:            ans ← min{ans, EXACT-COST(C', θ')+ cur}
12: return ans
```

s to t, represented by a s-t flow f of value k, is *suitable* for the state (C, θ) if $f(e_{i_j}) \leqslant \theta_j$, for all $1 \leqslant j \leqslant |C|$, and $f(e) \leqslant U(e)$ for all edges $e \in E$.

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

A state (C', θ') is *next to* (C, θ), if $C \leqslant C'$ and there exists a set of k paths which is suitable for both, with no r-vulnerable edge between C and C' (i.e., after C and before C'). For two given states (C, θ) and (C', θ'), in order to check whether (C', θ') is next to (C, θ), we define a function $w : E \to \mathbb{N}_0$ as belows:

- $w(e) = U(e)$ if e is before C or after C'
- $w(e) = \min\{U(e), r\}$ if e is between C and C'
- $w(e_{i_j}) = \theta_j$ if e_{i_j} is in C
- $w(e_{i'_j}) = \theta'_j$ if $e_{i'_j}$ is in C'
- $w(e) = 0$ otherwise

One can easily check that the maximum flow in graph G with capacity of each edge e equal to $w(e)$ is at least k if and only if (C', θ') is next to (C, θ).

Algorithm 2 computes the answer to the state (C, θ) recursively, based on the answers to the states next to it. The algorithm works as follows. Given a state (C, θ), we find all states (C', θ') next to (C, θ), solve the problem recursively for (C', θ'), and add the sum of costs of edges $e_{i_j} \in C$ with $\theta_j > r$, 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 with size less than k is $O(m^{k-1})$ and the number of solutions of $\sum_{i=1}^{|C|} \theta_i = k$ is $O(1)$, implying that the number of states is $O(m^{k-1})$ (where $m = |E|$). On the other hand, checking if a state is final and also checking whether a state is next to another or not are both solvable in $O(n^3)$ time by using a standard max-flow. Therefore, the total running time of the algorithm is $O(m^{2k-2}n^3)$. □

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 ℓ
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 $\text{OPT} \neq 0$.

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, a contradiction to 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:

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

where the the right hand side counts the number of paths a shared edges is in minus one for every shared edge. On the other hand, any shared edge can be in at most k paths, so the inequality above holds. Furthermore:

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

because we have take an optimal solution with a minimum number of used edges. 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 $k\ell$, implying the following inequalities:

$$k\ell - m^* \leqslant \sum_{e \in E} f(e) - m^* = \sum_{e \in E} \max\{0, f(e) - 1, \} \leqslant k\mathrm{OPT}. \qquad \square$$

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 appendix.)

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. $k\ell < 2m^*$. We know that $m^* \leqslant m$. Therefore:

$$k\ell < 2m \;\Rightarrow\; k\alpha\mathrm{OPT} < 2m \;\Rightarrow\; k\alpha < 2m.$$

Therefore, $\min\{k, \alpha\} < \sqrt{2m}$ in this case.

We can put a second upper bound on m^*. 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:

$$\exists\, c: \; k\ell < 2m^* < cn\sqrt{n}\mathrm{OPT} \;\Rightarrow\; k\alpha\mathrm{OPT} < c_1 n\sqrt{n}\mathrm{OPT}$$

$$\Rightarrow\; k\alpha < c_1 n\sqrt{n} \;\Rightarrow\; \min\{k, \alpha\} < \sqrt{c_1 n\sqrt{n}}$$

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

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

Remark. We can extend Algorithm 3 to obtain a factor $O(\sqrt{rm})$ approximation for the uncapacitated unweighted version of minimum r-vulnerability problem, for any $r > 0$. Details omitted in this version.

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. K. Ahuja, A. V. Goldberg, J. B. Orlin, and R. E. Tarjan. Finding minimum-cost flows by double scaling. *Mathematical Programming*, 53(1):243–266, 1992.
2. R. Carr, L. Fleischer, V. Leung, and C. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proc. 11th ACM-SIAM Sympos. Discrete Algorithms*, pages 106–115, 2000.
3. 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.
4. M. K. Franklin. *Complexity and security of distributed protocols.* PhD thesis, Dept. of Computer Science, Columbia University, 1994.
5. M. Garey and D. S. Johnson. *Computers and intractability : A guide to the theory of NP-completeness.* W. H. Freeman, 1979.
6. N. Garg, R. Ravi, and G. Konjevod. A polylogarithmic approximation algorithm for the group Steiner tree problem. *J. Algorithms*, 37(1), 2000.
7. A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. *J. ACM*, 45(5):783–797, 1998.
8. A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35(4):921–940, 1988.
9. G. Konjevod, R. Ravi, and A. Srinivasan. Approximation algorithms for the covering steiner problem. *Random Structures & Algorithms*, 20(3):465–482, 2002.
10. 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.
11. 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.
12. J. Wang, M. Yang, B. Yang, and S. Zheng. Dual-homing based scalable partial multicast protection. *IEEE Trans. Comput.*, 55(9):1130–1141, 2006.
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. 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.

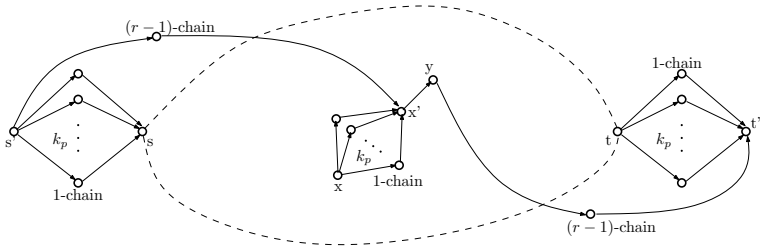

Fig. 2. Reduction from MSE to the r-minimum vulnerability problem with uniform edge costs. Each edge (x, y) in G is replaced by k 1-chains between (x, x') and one edge (x', y).

A Inapproximability

Even *et al.* [3] proved that the uniform-cost MECF problem does not admit any $2^{\log^{1-\varepsilon} n}$-ratio approximation, for any $\varepsilon > 0$, in a graph with n vertices, unless NP $\subseteq$ DTIME($n^{\text{polylog}\, n}$). Omran *et al.* [11] extended this result to the MSE problem which is an instance of the uniform-cost minimum 0-vulnerability with no upper bound on edges. Using a similar reduction, we prove in the following the inapproximability of the minimum r-vulnerability problem, for any $r > 1$.

Theorem 12. *The minimum vulnerability problem does not admits any* $2^{\log^{1-\varepsilon} n}$*-ratio approximation, for any* $\varepsilon > 0$*, unless* NP $\subseteq$ DTIME($n^{\text{polylog}\, n}$).

Proof. Let p be an instance of the MSE(G_p, k_p) problem, where $k_p \leqslant |E|$, for E being the set of edges in G_p. We want to solve p by a instance of the minimum r-vulnerability problem called q. To do so, we take $k_q = k_p + (r-1) \times |E|$, and then build a graph G_q based on G_p as follows.

We define a *u-chain* to be a path of size two having an upper bound of u on each of its edges. In order to build G_q, we first add two new vertices s' and t'. We then add a new vertex x' for each edge (x, y) in G_p. We call such a vertex a *mid vertex* corresponding to the edge (x, y). Now, for each edge from x to y in G_p, we insert an edge from x' to y with an upper bound of k_q (which is equivalent to ∞ because k_q is the total number of required paths), and k_p 1-chains from x to x'. Also, for each edge in G_p, one $(r-1)$-chain is added from s' to the corresponding mid vertex, and one from the head of the edge to t' (see Figure 2). Finally, k_p 1-chains from s' to s and k_p 1-chains from t to t' are added to the graph.

The idea behind adding k_p 1-chains in G_q is that these kind of edges can pass at most k_p paths without becoming r-vulnerable, and each of them cannot route more than one path, so none of them can be r-vulnerable. We prove that any optimal solution to p on (G_p, k_p) is an optimal solution to q on (G_q, k_q), and vice versa.

Firstly, assume that p has an optimal solution with k_p paths from s to t. To find an equivalent solution to q, for any edge (x, y) in G_p, we first pass $r-1$ paths from s' to the mid vertex corresponding to that edge, and route it through

y to t'. Then, we take k_p paths from s' to s and route them from s to t exactly in the same way as in p, and then route the paths from t to t'. So, this way there will be $(r-1) \times |E| + k_p = k_q$ paths from s' to t' in G_q. Regarding the r-vulnerable edges, observe that only the edges of G_q that are from a mid vertex can be r-vulnerable, since the upper bound of other edges is not exceeding r. Note that, if in a solution to p more than one path go through an edge, in G_q more than r paths can pass through the corresponding edge, because $r-1$ paths comes from s' to it and continue toward t', thus making it a r-vulnerable edge and vice versa. Therefore, the number of shared edges in the solution to p is equal to the number of r-vulnerable edges in the corresponding solution to q.

Secondly, assume that q has an optimal solution with k_q paths from s' to t', using λ r-vulnerable edges. These paths can be seen as a flow from s' to t', with λ edges carrying a flow more than r. Note that the sum of upper bounds of out-edges of s' is exactly k_q. This also holds for the incoming edges of t', and hence, those edges are completely full. Now, for any edge from a mid vertex x' to y, decrease the flow of the edge by $r-1$ units. Then, remove $r-1$ units of flow from s' to x', and from y to t', making them completely empty. It can be easily seen that this does not violate the feasibility of the flow. After performing this procedure for all such edges, $(r-1) \times |E|$ units of flow from s' to t' is removed, and there is k_p units of flow from s to t. It means that a k_p flow exists in G_p, in the same way as in G_q. Also, if an edge currently carries more than one unit of flow, it means that in the original G_q, it has more than r units of flow, because of the $r-1$ flow from s' to x' and from y to t', and vice versa. Therefore, the number of r-vulnerable edges in the solution to q is equal to the number of shared edges in the later solution to p.

The number of edges in G_q is at most $O(k_p|E|)$ times the number of edges in G_p. Moreover, according to [11], MSE is polynomially solvable for $k_p > |E|$. Thus, in our reduction we can assume that $k_p = O(|E|)$. So, if q admits a $2^{\log^{1-\varepsilon} n}$-ratio approximation algorithm, for any $\varepsilon > 0$, that approximation algorithm exactly works for p, contradicting the inapproximability of MSE by such factor. □

B Proof of Lemma 9

Proof. We replace each vertex v_i with unequal in- and out-degrees by two vertices s_i and t_i such that $d^+(s_i) = d^+(v_i)$, $d^-(t_i) = d^-(v_i)$, and $d^-(s_i) = d^+(t_i) = 0$. Let S and T be the set of s_i's and t_i's, respectively. We topologically sort the vertices in $V \setminus (S \cup T)$, and then, pack vertices in this order into groups of $\sqrt{n}$ vertices (see Figure 3). The maximum number of edges within the vertices of each group is n, and thus, there is at most $n\sqrt{n}$ edges in all groups.

Now, consider each group i as a new vertex v'_i in a graph $G' = (V', E')$ which can have multiple edges between vertices. We claim that in a graph G' with a maximum number of edges, each vertex v'_i only has edges to the vertex v'_{i+1}. Assume otherwise that there is an edge (v'_i, v'_j) with $j > i+1$. Then this edge can be split into two edges (v'_i, v'_{i+1}) and (v'_{i+1}, v'_j), preserving the in/out-degree equality constraints, contradicting the maximality of edges in G'.

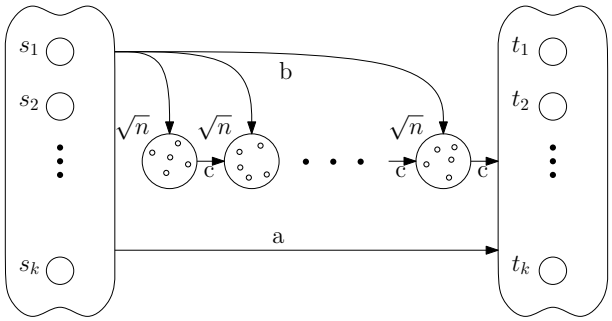

Fig. 3. A topologically sorted DAG with a maximum number of edges. Each edge is in one of four categories: (a) from s_i to t_j, (b) from s_i to any group in the middle, (c) from any group in the middle to the group right after it, (d) within each group.

By the previous claim, for any vertex v'_i, its out-degree is equal to the out-degree of v'_{i-1} plus the number of edges from S to v'_i which is at most $|S|\sqrt{n}$. By induction, out-degree of $v'_{\sqrt{n}}$ is at most $n|S|$, and therefore, the total number of edges between groups is $O(n|S| \times \sqrt{n})$. Considering the edges between S and T, the total number of edges in G' is $O(|S||T| + |S|n\sqrt{n})$, which proves the lemma statement for G. □