The Checkpoint Problem

MohammadTaghi Hajiaghayi[1], Rohit Khandekar[2], Guy Kortsarz[3⋆], and Julián Mestre[4]

[1] AT&T Labs– Research & University of Maryland, hajiagha@research.att.com.
[2] IBM T.J. Watson Research Center, rohitk@us.ibm.com.
[3] Rutgers University-Camden, guyk@camden.rutgers.edu.
[4] Max-Planck-Institut für Informatik, jmestre@mpi-inf.mpg.de.

Abstract. In this paper we consider the *checkpoint problem*. The input consists of an undirected graph G, a set of source-destination pairs $\{(s_1, t_1), \ldots, (s_k, t_k)\}$, and a collection $\mathcal{P}$ of paths connecting the (s_i, t_i) pairs. A feasible solution is a multicut E'; namely, a set of edges whose removal disconnects every source-destination pair. For each $p \in \mathcal{P}$ we define $\mathsf{cp}_{E'}(p) = |p \cap E'|$. In the *sum checkpoint* (*SCP*) problem the goal is to minimize $\sum_{p\in\mathcal{P}} \mathsf{cp}_{E'}(p)$, while in the *maximum checkpoint* (*MCP*) problem the goal is to minimize $\max_{p\in\mathcal{P}} \mathsf{cp}_{E'}(p)$. These problem have several natural applications, e.g., in urban transportation and network security. In a sense, they combine the *multicut problem* and the *minimum membership set cover problem*.

For the sum objective we show that weighted *SCP* is equivalent, with respect to approximability, to undirected multicut. Thus there exists an $O(\log n)$ approximation for *SCP* in general graphs.

Our current approximability results for the max objective have a wide gap: we provide an approximation factor of $O\big(\sqrt{n \log n/\mathsf{opt}}\big)$ for *MCP* and a hardness of 2 under the assumption $\mathrm{P} \neq \mathrm{NP}$. The hardness holds for trees, in which case we can obtain an asymptotic approximation factor of 2.

Finally we show strong hardness for the well-known problem of *finding a path with minimum forbidden pairs*, which in a sense can be considered the dual to the checkpoint problem. Despite various works on this problem, hardness of approximation was not known prior to this work. We show that the problem cannot be approximated within $c\,n$ for some constant $c > 0$, unless $\mathrm{P} = \mathrm{NP}$. This is the strongest type of hardness possible. It carries over to directed acyclic graphs and is a huge improvement over the plain NP-hardness of Gabow (*SIAM J. Comp 2007, pages 1648–1671*).

1 Introduction

In many countries, trains and urban transport operate largely on the honor system with enforcement by roving inspectors or conductors. The typical transaction consists of a user buying a ticket from a vending machine or a salesperson in advance and then time stamping it with a validating machine at the station just

⋆ Partially supported by NSF grant number 0829959.

before use. Inspectors check the tickets at certain stations (or indeed on the train in between stations) called *checkpoints* that might vary from day to day and fine people without validated tickets. In these scenarios, the transportation companies generally want to make sure a ticket is checked at least once, (in all routes, even those that carry small number of people) but avoid many checkpoints at popular source-destination travel paths. Due to the inconvenience of checking tickets for passengers many times, potential delays, and lack of resources, we consider the problem of placing checkpoints to minimize the average or maximum checks of tickets for some popular source-destination paths.

This problem can be modeled as follows. We are given an undirected graph $G(V, E)$ corresponding to stations and their connections via the transit system. We are also given a set of source-destinations $\{(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)\}$ and a set of fixed paths $\mathcal{P}$ between them. The goal is to find a set of *checkpoint* edges E' that forms a *multicut*, i.e., for every i, s_i and t_i are in different connected components in $G(V, E \setminus E')$ and minimizes the average (equivalently sum) or minimizes the maximum intersection with each path $p \in \mathcal{P}$. In this paper, we consider this problem, which we call the *checkpoint problem*. We note that the problem has other potential applications beyond our motivating example in transportation networks; for instance, in network security, we may want to check certain malicious source-destinations pair without incurring too much delay along certain critical paths.

1.1 Related work

Closely related to the checkpoint problem are the more common multicut problems in which given an edge-weighted (undirected or directed) graph and a collection of pairs $\{(s_i, t_i)\}_{i=1}^{k}$, the goal is to find a subset $E' \subseteq E$ of minimum cost so that in $E \setminus E'$ there is no s_i-t_i path for any i. The only difference between our problems and these is the objective function.

The literature on undirected multicut (*UM*) problems is extensive. Garg *et al.* [7] showed that *UM* is at least as hard to approximate as the well-known vertex-cover problem even if the underlying graph is a star. This implies that unless $\mathrm{P} = \mathrm{NP}$, it is hard to approximate the undirected multicut problem on stars within a factor better than $10\sqrt{5} - 21$ [4]. Garg *et al.* [7] also gave a 2 approximation for *UM* in trees via the primal dual approach. The best known approximation for *UM* in general undirected graphs is $O(\log n)$ [8]. Conditional on the Unique Game Conjecture [11], Chawla *et al.* [1] proved that the *UM* problem admits no constant approximation ratio for any constant. A stronger version of the conjecture implies that the *UM* problem can not be approximated within a factor $\Omega(\sqrt{\log \log n})$.

The multicut problem can also be viewed as a set cover problem in which we want to cover all paths between specific source-destination pairs (as elements) by a minimum number of edges (as sets). Set cover problems in which we are restricted to cover elements are also considered. In the *minimum membership set-cover problem* of Kuhn *et al.* [16], we want to cover all elements while minimizing the maximum number of sets covering an element. In the *unique coverage problem* of Demaine *et al.* [3], we want to maximize elements which are covered exactly once. Both these problems have applications concerning interference reduction in cellular networks. Roughly speaking, our checkpoint problem combines multicut and minimum membership set cover.

Multicut problems are associated with a dual multicommodity flow problem, where instead of disconnecting pairs, the objective is to connect them. In our setting, these flow problems are quite hard even if the input contains a single pair. We consider the well-known problem of *finding a path with minimum forbidden pairs (PAFP)*, a problem that has been studied since the seventies [15]. The input consists of a (directed or undirected) graph $G(V, E)$, a pair (s, t) of vertices, and a collection of forbidden pairs $\mathcal{F} = \{b_i b'_i\}_{i=1}^{\ell}$ where the forbidden pairs are the pairs of vertices *that may not appear simultaneously on the solution path*. A vertex may appear in many forbidden pairs. The goal is to find an s-t path with the minimum number of pairs $b_i b'_i \in \mathcal{F}$ such that *both* $b_i \in p$ and $b'_i \in p$.

The *PAFP* problem is particularly important for its relation to automatic software testing and validation [15, 19], and its applications in bioinformatics [2]. In [5] it is proved that *PAFP* is NP-complete on directed acyclic graphs. Yinnone [22] studied the problem in directed graphs under the so called skew-symmetry condition constraining the set of edges and the set of forbidden pairs. Yinnone gives a polynomial algorithm for the problem under that restriction. Chen *et al.* [2] study a special case of the problem coming from protein identification via tandem mass spectronomy. Kolman *et al.* [12] study *PAFP* under the so-called halving structure for which they prove the problem remains NP-complete, and also under the hierarchical structures condition for which they give a polynomial-time algorithm.

Notation and problem definitions. In this section, we define useful notations and formally define the problems considered in this paper. Let OPT be the optimum solution and opt be its value for the problem and instance at hand. For the rest of the paper we fix a collection $\mathcal{H} = \{(s_i, t_i)\}_{i=1}^{k}$ of k source-destination pairs. The given set of s_i-t_i paths will throughout be denoted by $\mathcal{P}$. We require that every (s_i, t_i) pair has at least one path in $\mathcal{P}$. Generally we assume that $|\mathcal{P}|$ is polynomial in n, unless stated otherwise. When working with trees, $\mathcal{P}$ is

uniquely defined by $\mathcal{H}$ because there is a single path connecting every source-sink pair.

Let p be a path in $\mathcal{P}$ and e be an edge in p. We will say that e *stabs* or *covers* p. For a set $E' \subseteq E$ we denote by $\mathsf{cp}_{E'}(p) = |p \cap E'|$ the number of edges in E' that stab p.

Definition 1. *The sum checkpoint (SCP) problem is to find a multicut $E' \subseteq E$ minimizing $\sum_{p\in\mathcal{P}} \mathsf{cp}_{E'}(p)$. The max checkpoint (MCP) problems is to find a multicut $E' \subseteq E$ minimizing* $\max_{p\in\mathcal{P}} \mathsf{cp}_{E'}(p)$.

The checkpoint value cp treats all edges uniformly since it simply counts the *number* of checkpoints in the multicut. In some cases, though, edges may be endowed with weights. In these cases, cp can be defined as the weight of edges chosen in the multicut that are in the path . We explore this variant for *SCP*.

Definition 2. *Given a (directed or undirected) graph G, a pair st, and a collection $\mathcal{F} = \{b_i b_i'\}_{i=1}^{\ell}$ of forbidden pairs of vertices, the path with minimum forbidden pairs (PAFP) problem is to find a path p from s to t minimizing the number of pairs $b_i b_i' \in \mathcal{F}$ such that* both $b_i \in p$ *and* $b_i' \in p$.

Note that a forbidden pair $b_i b_i' \in \mathcal{F}$ such that *at most* one of b_i or b_i' lies on p does not contribute towards the *PAFP* objective function.

Our results. First, we study *MCP* in trees. A tree input for *MCP* is said to have *ascending paths* if for all $(s_i, t_i) \in \mathcal{H}$ either s_i is an ancestor of t_i or vise-versa. *MCP* on ascending path tree inputs can be solved in polynomial time by linear programming. This follows from the well-known fact [21] that the edge-path incident matrix is totally unimodular. However, such a solution would have a very large running time. Even if T is a path, it is non-trivial to come up with purely combinatorial algorithms. We develop a linear-time algorithm for *MCP* in trees with ascending paths, which gives a solution with cost $\mathsf{opt} + 1$. Then we build upon this to obtain a combinatorial polynomial-time exact algorithm, which runs orders of magnitude faster than the obvious linear-programming based algorithm.

Beyond this special case, the problem becomes hard. We prove that unless $\mathrm{P} = \mathrm{NP}$, *MCP* in trees does not admit an approximation ratio better than 2. This solves an open problem of [17]. On the positive side, using standard techniques one can show a nearly matching approximation ratio.

For general graphs, we design an $O\left(\sqrt{\frac{n \log n}{\mathsf{opt}}}\right)$-approximation algorithm for *MCP* using a more sophisticated approach. Our algorithm is based on a somewhat unusual application of sphere growing. First the sphere growing is

combinatorial, that is, we grow spheres on the graph itself rather than on the LP solution à la Garg *et al.* [8]. Second, we use an LP solution to remove some edges in order to ensure that the every source-sink pair is "far apart". Combining these two ingredients, we guarantee that when the neighborhood of a set S is removed to disconnect a source-sink pair, the set S contains no "uncut" pairs.

Then we focus our attention on the weighted version of *SCP*. We show that weighted *SCP* is equivalent to *UM* from the point of view of approximability. In particular, *SCP* admits an $O(\log n)$ approximation ratio in general graphs and a 2 approximation ratio in trees.

Finally, we give a strong hardness of approximation for *PAFP* for undirected graphs. We show that unless $\mathrm{P} \neq \mathrm{NP}$, *PAFP* admits no $c \cdot n$ approximation ratio for some $c > 0$. Moreover, our construction can be easily modified to give the same hardness of approximation on directed acyclic graphs. This represents a huge improvement over the plain NP-hardness result of Gabow [6]. In fact, such a linear lower bound is one of the largest that can be found in the literature.

We close the section by mentioning that, independently, Nelson [17] also studied *MCP*. He designed an exact algorithm for paths, an asymptotic 2 approximation for trees, and showed 1.5-hardness for general graphs. The algorithm in Section 2 is a generalization of Nelson's algorithm for paths. We thank him for letting us include his result here. Our 2-approximation for general trees is slightly different. Our hardness result improves the one of Nelson in two aspects. First, our hardness ratio is slightly better; second, our proof is for trees and Nelson's is for general graphs. In fact establishing whether the problem is hard on trees is stated as an open problem in [17].

2 The *MCP* problem in trees with ascending paths

In this subsection we consider *MCP* in trees with ascending paths. That is, we look at instances where G is a rooted tree and for each pair (s_i, t_i) we have a unique path connecting them where s_i is an ancestor of t_i. For a given path $p \in \mathcal{P}$ we denote with $s(p)$ the *starting point* (closest vertex to the root) of p and with $f(p)$ the *finishing point* (furthest vertex from the root) of p. We call the edge $e \in p$ that is adjacent to $f(p)$ the *furthest* edge in p. For a given set X of paths, we define

$$F(X) = \cup_{p \in X} \{\text{the furthest edge in } p\}. \tag{1}$$

For a path $p \in \mathcal{P}$ and a set of paths A, we define $I_A(p)$ to be the number of paths in A that are contained in p.

Additive one approximation. Our main algorithm builds upon the following greedy procedure for computing a set paths. First, we show that the set of paths found by GREEDY can be used to produce a solution for *MCP* that is close to optimum. Later, we show how this algorithm can be used to find an optimal solution.

Algorithm GREEDY($\mathcal{P}$)

1. $A \leftarrow \emptyset$
2. **for** $p \in \mathcal{P}$ in increasing depth of $f(p)$ **do**
3. **if** $p \cap F(A) = \emptyset$
4. **then** $A \leftarrow A \cup \{p\}$
5. **return** A

Let A be the set returned by GREEDY. Notice that taking the furthest edge of each path in A yields a feasible solution: For any path $p \in \mathcal{P}$, if $p \in A$ then it is clear that $F(A)$ stabs p; otherwise, by Line 3, we know that $F(A)$ stabs p. The next lemma shows that the set $F(A)$ is a good approximation of the optimum.

Lemma 1. *Let A be the set returned by* GREEDY *and p be an arbitrary path in $\mathcal{P}$. Then every feasible solution stabs p at least $I_A(p)$ times and $F(A)$ stabs p at most $I_A(p) + 1$ times.*

Proof. We claim that the intervals in A contained in p are pairwise disjoint. Suppose, for the sake of contradiction, that there are paths $a, a' \in A$ contained in p that share an edge. Assume without loss of generality that a was added to A before a'. Thus we have that either $f(a) = f(a')$ or $f(a)$ is a proper ancestor of $f(a')$. Now because both paths lie in p and they intersect, it must be the case that the furthest edge of a stabs a'—here we use the property that all paths are ascending. Thus, we reach the contradiction that a' was not added to A. We conclude that the paths in A are pairwise disjoint.

Let a be a path in A whose furthest edge stabs p. Because the paths are ascending, either $s(a)$ is a proper ancestor of $s(p)$, or $s(a)$ belongs to p (that is, a lies inside p); let us call these paths of type 1 and 2, respectively. The key observation is that there is at most one path of type 1 (otherwise the furthest edge of one would stab the other and hence they could not be disjoint). Also, all paths of type 2 are disjoint and lie inside p; that is, $I_A(p)$ equals the number of type 2 paths. Therefore, furthest edges of paths of type 2 stab p exactly $I_A(p)$ times and the type 1 path, if any, can stab p one more time.

Notice that any solution must stab the type 2 paths using different edges. Therefore, any solution must stab p at least $I_A(p)$ times. □

It follows that $F(A)$ is a feasible solution that uses at most one extra checkpoint than the optimal solution. In addition, GREEDY can be implemented to run in linear time. The proof of this fact is omitted due to lack of space.

Lemma 2. *There is an $O(n + k)$ time additive-1 approximation for MCP in trees with ascending paths.*

Algorithm ITERATIVE-REFINEMENT($\mathcal{P}$)

1. $A \leftarrow B \leftarrow$ GREEDY($\mathcal{P}$)
2. $M \leftarrow \max_{p \in \mathcal{P}} I_A(p)$
3. **while** $\max_{p \in P} I_A(p) = M$ **do**
4. **if** $\exists$ bad pair $a \in A, p \in \mathcal{P}$
5. **then** $f(a) \leftarrow s(p)$ and $A \leftarrow$ GREEDY($\mathcal{P}$)
6. **else** return $F(A)$ // opt $= M$
7. **return** $F(B)$ // opt $= M + 1$

From approximate to optimal. Our exact algorithm is based on the idea of trying to weed out the structure that forces the previous algorithm to use an extra checkpoint. We call $a \in A, p \in \mathcal{P}$ a *bad pair* if $I_A(p) = M$ and $s(a)$ is proper ancestor of $s(p)$, and $s(p)$ is a proper ancestor of $f(a)$, and $f(a)$ is a proper ancestor of $f(p)$; notice that in this case the furthest edge of a stabs p. From the proof of Lemma 1, it immediately follows that the solution $F(A)$ has cost $M + 1$ if and only if there is a bad pair because if p is involved in a bad pair, then it will be stabbed by the furthest edge of a.

Lemma 3. *If $a \in A$, $p \in \mathcal{P}$ is a bad pair and there is a feasible solution with cost M then the solution is also feasible for the modified instance where $f(a) \leftarrow s(p)$. Also, any feasible solution to the modified instance is feasible for the original instance.*

Proof. Recall that there are $I_A(p) = M$ disjoint paths in A inside of p. These paths together with a form a set of disjoint paths. Suppose X is solution with cost M. Since p is stabbed only M times in X then it must be that a is stabbed in $a \setminus p$. Therefore, X remains feasible after we set $f(a) \leftarrow s(p)$.

The second part is trivial since after the modification, a is a subset of its original self. □

With this observation in hand, an algorithm follows suit. Compute A and iteratively try to find a bad pair. If we cannot find a bad pair then $F(A)$ has cost M and this is optimal. Otherwise, we modify the instance as described in Lemma 3 and recompute A. If $\max_{p \in \mathcal{P}} I_A(p)$ becomes $M + 1$ then the new instance cannot have a solution with cost M and hence our implicit assumption that the original instance admitted a solution with cost M must have been wrong.

Theorem 1. *There is a polynomial-time algorithm for MCP in trees with ascending paths.*

Proof. As mentioned above the correctness follows directly from repeatedly applying Lemma 3. To bound the running time we note that each iteration runs in $O(n + k)$ time and that there could be at most k^2 iterations since once a bad pair (a, p) is fixed, it never again becomes a bad pair. We note that the number of iterations can be brought down to $\min\{n, k^2\}$ if we are more aggressive when handling a bad pair (a, p). □

3 Hardness

In this section we show hardness of approximation for *MCP* in trees via a gap-inducing reduction from *1-in-3-SAT*. Recall that a 3-CNF formula belongs to *1-in-3-SAT* if there exists a satisfying assignment where each clause has exactly one true literal. Schaefer [18] proved that *1-in-3-SAT* is NP-complete. Our reduction maps yes (no) instances of *1-in-3-SAT* to instances of *MCP* with cost two (one). Due to lack of space we state our result without proof.

Theorem 2. *Unless* $\mathrm{P} = \mathrm{NP}$*, MCP in trees admits no better than ratio* 2 *approximation.*

4 Approximations for *MCP*

LP formulation In this section we present our approximation results for *MCP*. Our algorithms are based on the following linear programming relaxation. Let $\mathcal{Q}$ be the full set of paths connecting the source-sink pairs. (Recall that $\mathcal{P}$ is just a subset of $\mathcal{Q}$.)

$$\text{minimize } z \qquad \text{(LP1)}$$

subject to

$$\sum_{e \in q} x_e \geq 1 \qquad \text{for all } q \in \mathcal{Q} \qquad (2)$$

$$\sum_{e \in p} x_e \leq z \qquad \text{for all } p \in \mathcal{P} \qquad (3)$$

$$x_e \geq 0 \qquad e \in E$$

Variable x_e indicates whether edge e is chosen in the multicut. Constraint (2) enforces that the set of edges chosen indeed forms a multicut. The objective is to minimize z, the maximum number of edges any one path sees (3). For general graph, the set $\mathcal{Q}$ can be exponentially large. The program (LP1) can be solved in polynomial time by running the Ellipsoid algorithm on its dual.

Due to lack of space the presentation of our approximation algorithm for *MCP* in trees is deferred for the journal version.

Theorem 3. *There is a polynomial-time algorithm for MCP in trees that returns a solution with cost no more than* $2 \cdot \mathsf{opt} + 2$.

4.1 *MCP* in general graphs

Algorithm APPROXIMATING-MCP$(G, \mathcal{H})$

1. $x \leftarrow$ fraction optimal solution for (LP1)
2. $Sol \leftarrow \left\{ e \in E : x_e \geq \frac{1}{2}\sqrt{\frac{\mathsf{opt}}{n \cdot (\ln n + 1)}} \right\}$
3. remove the edges Sol from G
4. **while** Sol is not a multicut **do**
5. $\quad S \leftarrow \{s\}$, for some arbitrary uncut source s
6. $\quad$ **while** $|N(S)| \geq \left(1 + \sqrt{\frac{\mathsf{opt} \cdot (\ln n + 1)}{n}}\right)|S|$ **do**
7. $\quad\quad S \leftarrow N(S)$
8. $\quad Sol \leftarrow Sol \cup E(N(S) \setminus S)$
9. $\quad$ remove $E(N(S) \setminus S)$ from G
10. **return** Sol

Throughout this section, Sol will denote the partial solution accumulated by our algorithm. We say that a source s_i is *uncut*, if $G(V, E \setminus Sol)$ contains an s_i-t_i path. For simplicity, we assume that opt, the value of the optimal solution, is known. This value can easily be guessed, or, alternatively, we can use the value of the optimal fractional solution instead.

Along the way, we prove the following result: If the minimum distance between every (s_i, t_i) pair is ℓ, then there exists a vertex cut of size at most $\widetilde{O}(n/\ell)$ whose deletion disconnects all pairs. We believe this fact is known, but are not aware of any specific reference. Some results along these lines are known for the directed case; for example, it was shown independently in [20] and [13] that if every pair in a *directed graph* has distance at least ℓ, then there is an *edge cut* separating all pairs whose size is at most $\widetilde{O}(n^2/\ell^2)$.

Given a fractional solution x to (LP1), we denote the *fractional checkpoint value* of a path p by $\mathsf{cp}_x(p) = \sum_{e \in p} x(e)$. Let $dist(u, v)$ denote the length of the shortest path in G between u and v measured by the number of edges. The following operators are used by our algorithm:

$$N(X) = X \cup \{v \in V : \exists\, u \in X \text{ s.t. } (u, v) \in E)\},$$

and

$$E(X) = \{(u, v) \in E : u \in X \vee v \in X\}.$$

In other words, $N(X)$ equals X and its all neighbors, while $E(X)$ equals the set of edges with at least one endpoint in X. We note that both operators are defined with respect to the graph $G(V, E)$. As the algorithm progresses and removes edges from G, these operators change accordingly.

The algorithm can be thought off as having two main parts: a *filtering step* and a *region-growing step*. The next two lemmas, which we state without proof, establish some important properties of the first step.

Lemma 4. *Consider the value of Sol and G right after Line 3. Then $dist(s_i, t_i) > 2\sqrt{\frac{n\cdot(\ln n+1)}{\mathsf{opt}}}$ for all $(s_i, t_i) \in \mathcal{H}$.*

Lemma 5. *Consider Sol right after Line 2. Then $\mathsf{cp}_{Sol}(p) = \mathsf{cp}_x(p)\cdot O\left(\sqrt{\frac{n\log n}{\mathsf{opt}}}\right)$ for all $p \in \mathcal{P}$.*

After the initial filtering step (after the initial Sol is computed in Line 2), the algorithm iteratively finds sets $S_1, S_2, \ldots$, using a region-growing procedure out of uncut sources $s_1, s_2, \ldots$, respectively. We note that our approach is related to that of Garg *et al.*[8]. There are, however, two major differences. First, instead of "growing our regions on the LP solution", we do so in the input graph itself. Second, instead of using edge cuts, we use vertex cuts—indeed, the edges removed in Line 9 correspond to removing the vertices $N(X) \setminus S$.

Lemma 6. *The sets $S_1, S_2, \ldots$ are pair-wise disjoint.*

Proof. Consider an arbitrary set S_i. Upon existing the while loop in Line 6, the algorithm adds $E(N(S_i) \setminus S_i)$ to Sol. This effectively disconnects S_i from the rest of the graph defined by $E \setminus Sol$.

We claim that the number of iterations of the while loop in Line 6 needed to compute S_i is at most $\sqrt{\frac{n\cdot(\ln n+1)}{\mathsf{opt}}}$. Indeed, since the size of $|S|$ increases by $1+\sqrt{\frac{\mathsf{opt}\cdot(\ln n+1)}{n}}$ factor in each iteration, if the while loop were to run for $\sqrt{\frac{n\cdot(\ln n+1)}{\mathsf{opt}}}$ iterations then we would reach the contradiction that

$$|S| > \left(1+\sqrt{\frac{\mathsf{opt}\cdot(\ln n+1)}{n}}\right)^{\sqrt{\frac{n\cdot(\ln n+1)}{\mathsf{opt}}}} \geq n \geq |S|.$$

A corollary of this, is that the diameter of the graph induced by S_i is most $2\sqrt{\frac{n\cdot(\ln n+1)}{\mathsf{opt}}}$.

Now consider a set S_j constructed in some subsequent iteration. If $s_j \notin S_i$ then clearly S_j and S_i must be disjoint. We claim that this is the only option. Indeed, if $s_j \in S_i$ then, since the diameter of S_i is at most $2\sqrt{\frac{n\cdot(\ln n+1)}{\mathsf{opt}}}$, it follows that right after S_i is created $dist(s_j, t_j) \leq 2\sqrt{\frac{n\cdot(\ln n+1)}{\mathsf{opt}}}$, which contradicts Lemma 4. □

Everything is in place to prove the main result of this section.

Theorem 4. *The MCP problem admits a polynomial-time $O\left(\sqrt{\frac{n \log n}{\mathsf{opt}}}\right)$ approximation algorithm.*

Proof. Let p be an arbitrary path in $\mathcal{P}$. Notice that when $E(N(S_i) \setminus S_i)$ is added to Sol, since p is simple, the value of $\mathsf{cp}_{Sol}(p)$ increases by at most $|N(S_i) \setminus S_i|$. Therefore, in order to bound total increase in $\mathsf{cp}_{Sol}(p)$ due to edges to Sol after Line 2, we need to bound $\sum_i |N(S_i) \setminus S_i|$:

$$\sum_i |N(S_i) \setminus S| < \sum_i \sqrt{\frac{\mathsf{opt} \cdot (\ln n + 1)}{n}} \cdot |S_i| \leq \sqrt{n \cdot \mathsf{opt} \cdot (\ln n + 1)}, \quad (4)$$

where the first inequality follows from the exit condition of the while loop in Line 6, and the second, from Lemma 6. Putting (4) and Lemma 4 together, we conclude that

$$\mathsf{cp}_p(Sol) = \mathsf{cp}_x(Sol) \cdot O\left(\sqrt{\frac{n \ln n}{\mathsf{opt}}}\right) + \sqrt{n \cdot \mathsf{opt} \cdot (\ln n + 1)} = \mathsf{opt} \cdot O\left(\sqrt{\frac{n \ln n}{\mathsf{opt}}}\right). \qquad \square$$

5 Approximation for *SCP*

For this problem we allow the graph to be weighted, in which case $\mathsf{cp}_{E'}(p)$ is the combined weight of edges in $E' \cap p$. Recall that $w(e)$ denotes the edge e.

Theorem 5. *Any ρ approximation for UM gives a ρ approximation for weighted SCP, and vise-versa*

Proof. We first show the forward direction. We construct edge capacities c as follows: For every edge e let $p(e)$ be the number of paths $p \in \mathcal{P}$ that use e; notice that if for a fixed pair (s_i, t_i) the set $\mathcal{P}_i$ contains many paths going trough e, each one will contribute towards $p(e)$. We give edge e capacity $c(e) = p(e)w(e)$. We show that capacity of a multicut E' equals the min-sum checkpoint value, that is, $\sum_{e \in E'} c(e) = \sum_{p \in \mathcal{P}} \mathsf{cp}_{E'}(p)$. Given an edge $e \in E'$, we charge $w(e)$ to each of the $p(e)$ paths containing that edge; this exhausts the $c(e)$ term in the cost of the *UM* objective. Therefore, $\sum_{e \in E'} c(e) = \sum_{p \in \mathcal{P}} \mathsf{cp}_{E'}(p)$ and every ratio ρ that applies to *UM* also applies to *SCP*.

In the other direction, assume we have a ρ approximation for *SCP*. We approximate *UM* by a reduction to *SCP* as follows. Create a *SCP* instance with every e having capacity $w(e)/p(e)$. For any multicut E', $w(e)$ will be counted $p(e)$ times thus the checkpoint cost of E' is $c(E')$. Thus the best solution is the minimum capacity multicut. Thus, *SCP* and *UM* are equivalent with respect to approximation. $\square$

Corollary 1. *SCP admits an $O(\log n)$ approximation in general graphs and a 2 approximation in trees.*

6 A lower bound for *PAFP*

Recall that in *PAFP* we are given a pair (s,t) to connect and a collection of forbidden pairs $\{(b_i,b'_i)\}_{i=1}^{\ell} \subseteq V \times V$. The goal is to find an s-t path minimizing the number of pairs (b_i,b'_i) such that both belong to the path. To disallow a zero cost solution we may arbitrarily define (s,t) as a forbidden pair or we can define the cost of a solution as the maximum between the number of bad pairs in the path and 1.

Background. The *LABELCOVER-MAX* problem is introduced in [10, Chapter 10] for presenting one-round two-provers systems. Here we use an alternative formulation, called *MAX-REP*, defined in [14]. In *MAX-REP*, we are given a bipartite graph $G(V_1,V_2,E)$. The sets V_1 and V_2 are partitioned into a disjoint union of q sets: $V_1 = \bigcup_{i=1}^{q} A_i$ and $V_2 = \bigcup_{j=1}^{q} B_j$. The bipartite graph and the partition of V_1 and V_2 induce a super-graph $\mathcal{H}$ in the following way: The vertices in $\mathcal{H}$ are the sets A_i and B_j. Two sets A_i and B_j are connected by a super-edge in $\mathcal{H}$ if and only if there exist $a_i \in A_i$ and $b_j \in B_j$ which are adjacent in G. In *MAX-REP* we are to select a unique *representative* vertex $a_i \in A_i$ from each subset A_i, and a unique *representative* vertex $b_j \in B_j$ from each B_j. We say that a super-edge (A_i,B_j) is *covered* if the two corresponding representatives are neighbors in G; that is, $(a_i,b_j) \in E$. The goal is to select unique representatives so as to maximize the number of super-edges covered. Håstad's breakthrough hardness for *3-SAT-5* [9] translates into the following hardness for *MAX-REP*.

Theorem 6 ([9]). *There is a polynomial time reduction that maps each instance ϕ of SAT into an instance G of MAX-REP with n' vertices and $h = \Theta(n')$ super-edges. If ϕ is satisfiable then there exists a set of unique representatives of G that covers all h super-edges. If ϕ is not satisfiable then every set of unique representatives of G covers at most $\frac{23}{24}h$ super-edges.*

Reduction. The reduction from *MAX-REP* to *PAFP* is relatively simple. Arbitrarily order the super-vertices from left to right: $X_1, X_2, \ldots, X_{2q}$. Join X_i to X_{i+1} with a complete bipartite graph for every $1 \leq i \leq 2q-1$. Let (A,B) be a super-edge in our *MAX-REP* instance. For each $a \in A$ and $b \in B$ such that $(a,b) \notin E$, we create a forbidden pair (a,b). Thus, forbidden pairs correspond

to vertices that *are not* connected in the *MAX-REP* graph and whose corresponding super-nodes *are* connected in the super-graph. Finally, join a vertex s to all the vertices of X_1 and join a vertex t to all the vertices of X_h. This defines the *PAFP* instance.

Theorem 7. *Unless* $\mathrm{P} = \mathrm{NP}$*, PAFP on undirected graphs admits no* $c \cdot n$ *approximation ratio, where* n *is the number of vertices and* $c > 0$ *is some constant. The same holds for directed acyclic graphs.*

Proof. Consider the reduction above. We show that a solution for the *PAFP* instance with t forbidden pairs translates into solution for the *MAX-REP* instance covering $h-t$ super-edges, and vice-versa. Without loss of generality we restrict our attention to *PAFP* solutions that use a single vertex from each super-vertex X_i. Under this restriction, there is a clear one-to-one correspondence between solutions for the *PAFP* instance (s-t paths) and solutions for the *MAX-REP* instance (unique representative choices). Let X be a unique representative choice and p its corresponding s-t path. Let (A, B) be an arbitrary super-edge, and let a and b be the representatives of A and B respectively. If (A, B) is covered by X then none of the forbidden pairs induced by (A, B) appear in p. Otherwise, if (A, B) is not covered by X, we know that (a, b) is a forbidden pair. It follows that the number of super-edges covered by X is h minus the number of forbidden pairs in p.

In Theorem 6, satisfiable formulas map to instances of *MAX-REP* that have a perfect cover, which in turn our reduction maps to instances of *PAFP* that have a path with no forbidden pairs, which have value 1 (recall that the cost of a path is the maximum of 1 and the number of forbidden pairs.) On the other hand, unsatisfiable formulas map to instances of *MAX-REP* with value at most $\frac{23}{24}h$, which in turn map to instances of *PAFP* having value at least $\frac{h}{24}$. In addition, Theorem 6 tells us that the *MAX-REP* instance has $h = \Theta(n)$, where n is the number of vertices in the *PAFP* instance. This finishes the proof for undirected order.

Finally, in order to get the result on directed acyclic graphs, just direct all the edges from s to t. □

7 Discussion and open problems

Can the approximation for *SCP* be used to approximate *MCP*? If the optimum for *SCP* is opt_s then the optimum for *MCP* is at least $\mathsf{opt}_s/|\mathcal{P}|$. Therefore, by approximating the *SCP* objective, we obtain a lower bound for the *MCP* objective. The multicut for the *SCP* problem, however, cannot be used directly as a solution for the *MCP* problem since the path with the largest checkpoint value

may be well above the average checkpoint value. One could try to deal with these "expensive" paths in a later stage, but this may increase the checkpoint value of paths previously having low checkpoint value in the *SCP* solution. Indeed, the *MCP* problem seems highly non-separable.

Finally, it would be interesting to know whether *MCP* admits a polylog(n) approximation, or whether it has a hardness is similar to *MAX-REP*.

References

1. S. Chawla, R. Krauthgamer, R. Kumar, Y. Rabani, and D. Sivakumar. On the hardness of approximating multicut and sparsest-cut. *Computational Complexity*, 15(2):94–114, 2006.
2. T. Chen, M. Y. Kao, M. Tepel, J. Rush, and G. Church. A dynamic programming approach to de novo peptide sequencing via tandem mass spectrometry. *Journal of Computational Biology*, 8(3):325–337, 2001.
3. E. D. Demaine, U. Feige, M. T. Hajiaghayi, and M. Salavatipour. Combination can be hard: approximability of the unique coverage problem. *SIAM J. Comp.*, 38(4):1464–1483, 2008.
4. I. Dinur and S. Safra. The importance of being biased. In *STOC*, pages 33–42, 2002.
5. H. Gabow, S. Maheswari, and L. Osterweil. On two problems in the generation of program test paths. *IEEE Trans. Software Eng.*, 2(3):227–231, 1976.
6. H. N. Gabow. Finding paths and cycles of superpolylogarithmic length. *SIAM J. Comp.*, 36(6):1648–1671, 2007.
7. N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.
8. N. Garg, V. V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25(2):235–251, 1996.
9. J. Håstad. Some optimal inapproximability results. *J. of the ACM*, 48(4):798–859, 2001.
10. D. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co, 1997.
11. S. Khot. On the unique games conjecture. In *FOCS*, page 3, 2005.
12. P. Kolman and O. Pangrac. On the complexity of paths avoiding forbidden pairs. *Discrete applied math*, 157:2871–2867, 2009.
13. Y. Kortsarts, G. Kortsarz, and Z. Nutov. Greedy approximation algorithms for directed multicuts. *Networks*, 45(4):214–217, 2005.
14. G. Kortsarz. On the hardness of approximating spanners. *Algorithmica*, 30(3):432–450, 2001.
15. K. Krause, R. Smith, and M. Goodwin. Optimal software test planning through authomated search analysis. In *IEEE Symp. Computer Software Reliability*, pages 18–22, 1973.
16. F. Kuhn, P. von Rickenbach, R. Wattenhofer, E. Welzl, and A. Zollinger. Interference in cellular networks: The minimum membership set cover problem. In *COCOON*, pages 188–198, 2005.
17. J. Nelson. Notes on min-max multicommodity cut on paths and trees. Manuscript, 2009.
18. T. J. Schaefer. The complexity of satisfiability problems. In *Proc. of the 10th of the Tenth Annual ACM Symposium on Theory of Computing*, pages 216–226, 1978.
19. P. Strimani and B. Sinha. Impossible pair-constrained test path generation in a program. *Information Sciences*, 28:87–103, 1982.
20. K. Varadarajan and G. Venkataraman. Graph decomposition and a greedy algorithm for edge-disjoint paths. In *SODA*, pages 379–380, 2004.
21. M. Yannakakis. On a class of totally unimodular matrices. In *FOCS*, pages 10–16, 1980.
22. H. Yinnone. On paths avoiding forbidden pairs of vertices in a graph. *Discrete Appl. Math.*, 74(1):85–92, 1997.