The Checkpoint Problem*

MohammadTaghi Hajiaghayi† Rohit Khandekar‡ Guy Kortsarz§ Julián Mestre¶

Abstract

In this paper we consider the *checkpoint problem*. The input consists of an undirected graph G, a set of source-destination pairs $\{(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)\}$, 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. This solves an open problem of [25]. We complement the lower bound by an almost matching upper bound with an asymptotic approximation factor of 2. On trees with all s_i, t_i having an ancestor descendant relation, we give a combinatorial exact algorithm. Besides the algorithm being combinatorial, its running time improves by many orders of magnitude the LP algorithm that follows from total unimodularity.

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, 36(6), pages 1648–1671,2007*).

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 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 a 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.

*A preliminary version of this paper appeared in the Proceedings of the 13th International. Workshop on Approximation Algorithms for Combinatorial Optimization Problems – APPROX 2010

†University of Maryland, College Park & AT&T Labs– Research, hajiagha@cs.umd.edu. Supported in part by NSF CAREER award 1053605, ONR YIP award N000141110662, DARPA/AFRL award FA8650-11-1-7162, and a University of Maryland Research and Scholarship Award (RASA).

‡IBM T.J. Watson Research Center, rohitk@us.ibm.com

§Rutgers University-Camden, guyk@camden.rutgers.edu. Partially supported by NSF grant number 0829959.

¶The University of Sydney, Australia, mestre@it.usyd.edu.au.

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.* [13] 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 $10\sqrt{5} - 21$ [10]. Garg *et al.* [13] 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 $O(\log n)$ [14]. In [15, 23, 24], the p-*UM* problem was considered. In this problem, in addition to the *UM* input, we are given an additional integer $p \leq k$. The problem is to multicut at least p of the pairs; thus, *UM* is the special case $p = k$. Using the scheme of Golovin *et al.* [15] and a recent result of Räcke [27], one can get an $O(\log n)$-approximation algorithm for p-*UM*. Conditional on the Unique Game Conjecture [17], Chawla *et al.* [5] 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 $\Omega(\sqrt{\log \log n})$.

Next we discuss works related to directed multicut (*DM*). In [8] it is shown that unless $\mathrm{NP} \subseteq \mathrm{ZPP}$ the *DM* problem admits no $2^{\log^{1-\varepsilon} n}$ ratio for any constant ε. The first non-trivial approximation for DM was an $O(\sqrt{n \log n})$ ratio by Cheriyan *et al.* [7]. This ratio was improved to an $O(\sqrt{n})$ by Gupta [16]. Recently, the $O(\sqrt{n})$ barrier was broken by Agrawal *et al.* [1] to an $\tilde{O}(n^{11/23})$ ratio approximation[1]. In [19], an $O(n^{2/3}/\mathsf{opt}^{1/3})$ approximation algorithm for DM with unit capacities is presented. In case $\mathsf{opt} \geq n^{0.566}$, the ratio of [19] is better than the one of [1]. The [19] algorithm is also the only non-trivial *combinatorial* approximation algorithm for DM.

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 there are restrictions in the way we can cover elements are also considered. In the *minimum membership set-cover problem* of Kuhn *et al.* [22], we want to cover all elements while minimizing the maximum number of sets covering an element. Kuhn *et al.* designed an $O(\log |\mathcal{U}|)$ approximation for this problem, where $\mathcal{U}$ is the ground set of elements that we are to cover. In the *unique coverage problem* of Demaine *et al.* [9], 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. It is worth noting that the result of Kuhn *et al.* [22] does not apply to our setting since the ground set we are to cover (the set of paths connecting all source-sink pairs) can be exponential in the size of the instance.

Multicut problems are associated with a dual multicommodity flow problem, where instead of disconnecting pairs, the objective is to connect them.

It might be that the most natural dual problem is finding a collection of s_i, t_i paths which minimizes the sum of length of the paths or minimizes the maximum length of a path. But then we get a problem that is trivial for a single pair as it is the shortest path problem.

[1] The notation $\widetilde{O}(f(n))$ ignores the factors polylogarithmic in n.

Since this problem is hardly known, instead we consider the well-known problem of *finding a path with minimum forbidden pairs (PAFP)*, a problem that has been studied since the seventies [21]. 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 $b'_i \in p$. Unlike the previously discussed problem, we are able to give a strong hardness result for the path with forbidden pairs problem, with only one pair s_1, t_1.

The *PAFP* problem is particularly important for its relation to automatic software testing and validation [21, 29], and its applications in bioinformatics [6]. In [11] it is proved that *PAFP* is NP-complete on directed acyclic graphs. Yinnone [32] 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.* [6] study a special case of the problem coming from protein identification via tandem mass spectronomy. Kolman *et al.* [18] 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 (s, t), 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* on trees. A tree input for *MCP* is said to have *ascending paths* if for all $(s_i, t_i) \in \mathcal{H}$ the node s_i is an ancestor of t_i. *MCP* on ascending path tree inputs can be solved in polynomial time by linear programming. This follows from the well-known fact [31] that the edge-path incident matrix is totally unimodular. However, such a solution would have a very large running time.

Therefore we are interested in purely combinatorial 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 at most $\mathsf{opt}+1$. Then we build upon this to obtain a combinatorial polynomial-time *optimum* algorithm, which runs many orders of magnitude faster than the obvious linear-programming based algorithm.

Beyond this special case, but still on trees, 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 [25]. On the positive side, using standard techniques one can show a asymptotically 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 of Garg *et al.* [14]. Second, we use an LP solution to remove some edges in order to ensure that every source-sink pair

is "far apart". Combining these two ingredients, we guarantee that when the neighborhood of a set S of vertices is removed to disconnect a source-sink pair, the set S contains no "uncut" pairs. An interesting aspect of our algorithm is that it performs best when opt is large and worst when opt is small. This leaves the door open for possible improvements; for example, an $O(\mathsf{opt})$ approximation would immediately imply a $\widetilde{O}(\sqrt[3]{n})$ approximation. At the moment, however, we are not able to provide such a guarantee.

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. Even though this equivalence holds true only in the weighted case, we can approximate unweighted *SCP* using *UM* with non-uniform capacities.

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 [12]. 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 [25] 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 [25].

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 incident to $f(p)$ the *furthest* edge in p. For a given set X of paths, we define

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

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 fully contained in p, that is,

$$I_A(p) = |\{q \in A : q \subseteq p\}|\,.$$

Additive one approximation. Our main algorithm builds upon the following greedy procedure for computing a set of 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** $F(A)$

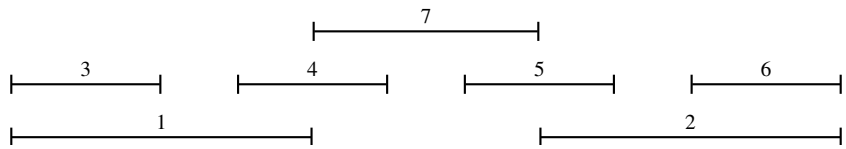


Figure 1: An instance of *MCP*. The underlying graph is a path and there are 7 source-sink pairs.

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 2.1. *Let A be the set computed 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 paths 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'. 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 contained in p 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 would not have been added). 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 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 whose value is at most one more than the value of an optimal solution. In addition, GREEDY can be implemented to run in linear time.

Lemma 2.2. *There is an $O(n + k)$ time additive-1 approximation for MCP in trees with ascending paths.*

Proof. It follows from Lemma 2.1 that a lower bound on the cost of the optimum solution is $M = \max_{p \in \mathcal{P}} I_A(p)$ and that each path p is stabbed at most $I_A(p) + 1 \leq M + 1$.

The implementation details to get the claimed running time are as follow. First we find the depth of every vertex in T. Then we use bucket sort to order the paths in increasing depth of their finishing endpoint. For each vertex u we keep track of how many edges in $F(A)$ there are in the path from u to the root. Initially, this information is known only for the root, for which this quantity equals 0. When processing a path p in the while loop, if this information is not available at $t(p)$ then we walk up the tree until we find the first vertex, call it v, that has this information and we replicate this value to all vertices from v to $t(p)$. Using this information we can easily find out how many edges in $F(A)$ stab a given path p. (When adding a path p to A, we need to update this information for $t(p)$.) The overall time spent is $O(n + k)$. □

So far we know that opt is either M or $M + 1$, where $M = \max_{p \in \mathcal{P}} I_A(p)$. If the above algorithm returns a solution with cost M then we know it is optimal; otherwise, we need to find out whether or not $M + 1$ checkpoints are really necessary. It is instructive to look at an example in order to understand what sort of challenges an optimal algorithm has to overcome. Consider the example from Figure 1, where the tree is in fact a path. If we assume the root is the leftmost node then $A = \{3, 4, 5, 6\}$ and $M = 1$, which is attained by $\{1, 2\}$. Notice that $F(A)$ has a cost of 2 since path 2 is stabbed twice. In this case, however, this is the best possible solution: If we insist that paths 1 and 2 are stabbed once then path 7 must be stabbed twice. The example shows that it is not enough to guarantee that paths with $I_A(p) = M$ do not get an extra checkpoint, we also need to enforce, for example, that paths with $I_A(p) = M - 1$ do not get two extra checkpoints.

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 $t(a)$, and $t(a)$ is a proper ancestor of $t(p)$; notice that in this case the furthest edge of a stabs p. From the proof of Lemma 2.1, 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.

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 $F(A)$ // $\mathsf{opt} = M$
7. **return** $F(B)$ // $\mathsf{opt} = M + 1$

Lemma 2.3. *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 2.3 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.

Let us see how the algorithm proceeds in the instance from Figure 1. At the beginning the only bad pair is $(5, 2)$. After path 5 is shortened, $(4, 7)$ become a bad pair since now $I_A(5) = 1$. After path 4 is shortened, $I_A(1)$ becomes 2, signaling that the instance does not admit a solution with cost 1.

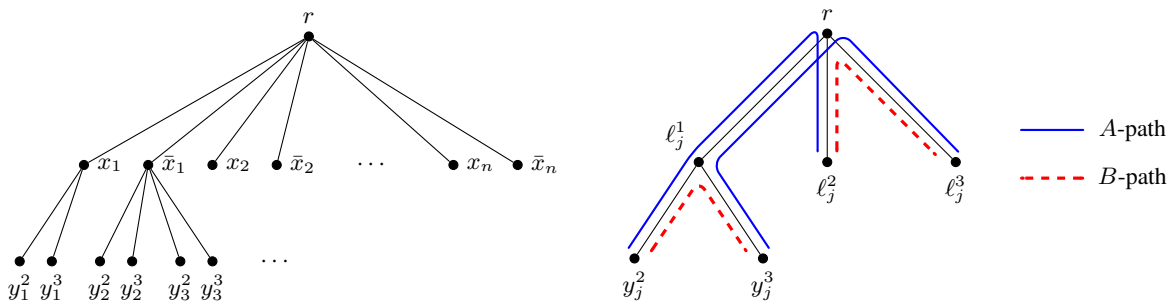
Theorem 2.4. *There is a polynomial-time algorithm for MCP in trees with ascending paths.*

Proof. As mentioned above the correctness follows directly from repeatedly applying Lemma 2.3. 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.

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). Recall that if $\mathsf{opt} = M$ then none of the edges in $a \cap p$ can be chosen to be checkpoints. Thus when we find a bad pair (a, p) we can contract the edges $a \cap p$ to a single vertex. Since we reduce the number of edges by at least 1 in each iteration, there are at most n iterations. □

3 Hardness

In this section we show hardness of approximation for *MCP* in trees via a gap-introducing 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 [28] proved that *1-in-3-SAT* is NP-complete.



(a) The tree used in the reduction.

(b) Paths for clause $C_j = \ell_j^1 \vee \ell_j^2 \vee \ell_j^3$.

Figure 2: The reduction from *1-in-3-SAT* to *MCP*.

Reduction. Given a 3-CNF formula $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ on variables $x_1, x_2, \ldots, x_n$, we will construct an *MCP* instance consisting of a tree T and pairs $\{(s_1, t_1), \ldots, (s_k, t_k)\}$ such that the optimal solution of the *MCP* instance has value 1 if ϕ belongs to *1-in-3-SAT* and has value 2 otherwise.

For ease of presentation we divide the paths into two types: A-paths and B-paths. Paths of type A are required to be stabbed exactly once, paths of type B are required to be stabbed at most once. We say the *MCP* instance is *feasible* if there is set of edges that satisfy all path requirements. It is straightforward to reduce this problem to regular *MCP* where the goal is to stab every path *exactly* once[2].

Our tree T has three levels: On level 0 we have r, the root of the tree; on level 1 we have for each variable two vertices x_i and $\bar{x}_i$ connected to the root r; on level 2 we have for each clause $C_i = \ell_1 \vee \ell_2 \vee \ell_3$ two vertices y_i^2 and y_i^3 connected to ℓ_1. The set of paths is as follows: For each variable x_i we have an A-path $(x_i, \bar{x}_i)$, and for each clause $C_j = \ell_j^1 \vee \ell_j^2 \vee \ell_j^3$ we have B-paths (y_j^2, y_j^3) and (ℓ_j^2, ℓ_j^3), and A-paths (ℓ_j^2, y_j^2) and (ℓ_j^3, y_j^3). The reader is referred to Figure 2 for a pictorial view of the tree and the paths.

Theorem 3.1. *Unless* $\mathrm{P} = \mathrm{NP}$*, MCP in trees admits no better than ratio* 2 *approximation.*

Proof. Using the reduction above, we show a one-to-one correspondence between solutions to the *MCP* instance satisfying the path requirements and truth assignments for ϕ where every clause is satisfied by exactly one literal. Given a feasible set $S \subseteq T$ for *MCP*, we construct a truth assignment $x_i = \top$ if and only if $(x_i, r) \in S$; notice that this is well defined because we have an A-path $(x_i, r, \bar{x}_i)$ for each variable x_i. Now consider a clause $C_j = \ell_j^1 \vee \ell_j^2 \vee \ell_j^3$. Notice that since there is a path going trough every pair of literals in C_j, if S satisfies the path requirements then C_j is satisfied at most once in the assignment. To see that each clause is indeed satisfied, suppose, for the sake of contradiction that C_j is not satisfied. Because of the A-paths (y_j^2, ℓ_j^2) and (y_j^3, ℓ_j^3), it must be the case that (y_j^2, ℓ_j^1) and (y_j^3, ℓ_j^1) belong to S. Thus the B-path (y_j^2, y_j^3) would be stabbed twice contradicting the feasibility of S.

For the other direction, given a feasible assignment for ϕ, it is straightforward to check that the solution $\{(x_i, r) : x_i = \top\} \cup \{(\bar{x}_i, r) : x_i = \bot\}$ indeed satisfies all the path requirements. $\square$

Critically, the paths used in the reduction are not ascending. Also, the length of the longest path is three. This is necessary because if all paths have length 2, then $\mathsf{opt} \in \{1, 2\}$ and checking whether opt equals 1 can be reduced to the question of whether a certain 2-CNF formula is satisfiable, which in turn can be solved in polynomial time [28].

Recall that later in Subsection 4.1, we give an almost matching upper bound. We give a polynomial time approximation algorithm that always returns a solution of value at most $2\mathsf{opt} + 1$ with opt the value of the best solution. Thus the problem is nearly resolved on trees.

[2]For each s-t path of type B attach a dummy node d_{st} to s and use the path d_{st}-t instead.

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 $\mathcal{Q}$.)

$$\text{minimize } z \tag{LP1}$$

subject to

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

$$\sum_{e \in p} x_e \leq z \qquad \text{for all } p \in \mathcal{P} \tag{2}$$

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

Variable x_e indicates whether edge e is chosen in the multicut. Constraint (1) 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 (2). 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: We can design a separation oracle for constraints (1) by giving each edge e in the graph a weight of x_e and running an all-pairs shortest path algorithm to check if there is a path $q \in \mathcal{Q}$ such that $\sum_{e \in q} x_e < 1$; the remaining constraints (2) are polynomially many, so they can be checked individually.

4.1 *MCP* in trees

In this section we consider *MCP* in trees with unrestricted paths; that is, a path is allowed to go up and down the tree. The main idea is to "round" the instance to another related problem whose LP formulation is integral; this is similar to the approach that Golovin *et al.*[15] used for the k-multicut problem in trees. Let (x^*, z^*) be an optimal fraction solution for (LP1). We denote by $x^*(a, b) = \sum_{e \in (a,b)} x^*(e)$, the total fractional value of the edges in the unique path connecting a and b in the tree. The new instance is constructed as follows: For each path $p \in \mathcal{P}$ going from s_i to t_i let a_i be the lowest common ancestor in the tree. Without loss of generality we assume $x^*(a_i, s_i) \geq x^*(a_i, t_i)$. We ask that each (a_i, s_i) path is cut at least once, that the path (a_i, s_i) is cut at most $\lceil 2x^*(a_i, s_i) \rceil$ times, and that the path (a_i, t_i) is cut at most $\lceil 2x^*(a_i, t_i) \rceil$ times. The LP formulation for the new problem is given below.

$$\text{no objective} \tag{LP2}$$

subject to

$$\sum_{e \in (a_i, s_i)} y_e \geq 1 \qquad \forall\, (s_i, t_i) \in \mathcal{P} \tag{3}$$

$$\sum_{e \in (a_i, s_i)} y_e \leq \lceil 2x^*(a_i, s_i) \rceil \qquad \forall\, (s_i, t_i) \in \mathcal{P} \tag{4}$$

$$\sum_{e \in (a_i, t_i)} y_e \leq \lceil 2x^*(a_i, t_i) \rceil \qquad \forall\, (s_i, t_i) \in \mathcal{P} \tag{5}$$

$$y_e \geq 0 \qquad \forall\, e \in T$$

In this new program, some of these paths must be cut (3) and each path has an upper bound on the number of edges that can be chosen in the multicut (4, 5). Notice that all paths in the new instance are ascending.

We claim that this new program is feasible and integral. Furthermore, any solution for (LP2) is feasible for the original problem and offers a good approximation for the maximum number of checkpoints.

Theorem 4.1. *There is a polynomial-time algorithm for MCP in trees that returns a solution with cost no more than* $2 \cdot \mathsf{opt} + 1$.

Proof. The paths used in (LP2) are ascending. It is well known that the path-edge incident matrix is totally unimodular (see for example [31]), which in turn implies that the linear program (LP2) is integral. Notice that $2x^*$ is a feasible fractional solution for (LP2). Therefore, an integral solution must exists and we can compute such a solution in polynomial time.

First we show the slightly weaker result that the solution returned has cost no more than $2 \cdot \mathsf{opt} + 2$. Let y^* be the integral solution for (LP2). Clearly, since y^* cuts (a_i, s_i) it also must cut (s_i, t_i). Furthermore, for any path (s_i, t_i) we have

$$y^*(s_i, t_i) \leq \lceil 2x^*(a_i, s_i) \rceil + \lceil 2x^*(a_i, t_i) \rceil \leq 2x^*(s_i, t_i) + 2 \leq 2z^* + 2.$$

Recalling that $z^* \leq \mathsf{opt}$ finishes the proof of the weaker bound.

Let us finish the proof by arguing that the additive term in the approximation guarantee is in fact 1. Since opt is an integer, we have $\lceil z^* \rceil \leq \mathsf{opt}$ and we assume without loss of generality that z^* is also an integer. Consider a source-sink pair such that $x^*(s_i, t_i) = z^*$. Suppose that $x^*(a_i, s_i) = \frac{k_1}{2} + \delta_1$ and $x^*(a_i, t_i) = \frac{k_2}{2} + \delta_2$ where $0 < \delta_1 \leq 1/2$ and $0 \leq \delta_2 < 1/2$. It follows by our assumptions that $\delta_1 + \delta_2 = 1/2$. Our solution stabs (s_i, t_i) at most

$$k_1 + k_2 + 2 = 2x^*(s_i, t_i) + 1 = 2z^* + 1 \leq 2 \cdot \mathsf{opt} + 1$$

times. □

4.2 *MCP* in general graphs

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 (we run the algorithm on all $n-1$ values of opt and return the best solution found), 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 [30] and [19] 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 all its 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.

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

1. $x \leftarrow$ fractional 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

The algorithm can be thought of as having two main parts: a *filtering step* and a *region-growing step*. The next two lemmas establish some important properties of the first step.

Lemma 4.2. *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* $(s_i, t_i) \in \mathcal{H}$.

Proof. Let q be an arbitrary path in $\mathcal{Q}$. By Constraint (1) in (LP1), we know there must be an edge e in q such that $x_e \geq \frac{1}{|q|}$. If $|q| \leq 2\sqrt{\frac{n \cdot (\ln n + 1)}{\mathsf{opt}}}$, at least one edge of q must belong to Sol. Therefore, after Line 3 the remaining paths connecting source-sink pairs must be longer than $2\sqrt{\frac{n \cdot (\ln n + 1)}{\mathsf{opt}}}$. □

Lemma 4.3. *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* $p \in \mathcal{P}$.

Proof. Let p be a path in $\mathcal{P}$. Note that each edge of p that belongs to Sol contributes at least $\frac{1}{2}\sqrt{\frac{\mathsf{opt}}{n \cdot (\ln n + 1)}}$ towards $\mathsf{cp}_x(p)$. Therefore, $\mathsf{cp}_{Sol}(p) \leq \mathsf{cp}_x(p) \cdot 2\sqrt{\frac{(\log n + 1) \cdot n}{\mathsf{opt}}}$. □

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.*[14]. 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 4.4. *The sets* $S_1, S_2, \ldots$ *are pair-wise disjoint.*

Proof. Consider an arbitrary set S_i. Upon exiting 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 at least $1 + \sqrt{\frac{\mathsf{opt} \cdot (\ln n + 1)}{n}}$ 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 4.2. □

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

Theorem 4.5. *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 added to Sol after Line 2, we need to bound $\sum_i |N(S_i) \setminus S_i|$:

$$\sum_i |N(S_i) \setminus S| < \sum_i \sqrt{\frac{\mathsf{opt}\cdot(\ln n+1)}{n}} \cdot |S_i| = \sqrt{\frac{\mathsf{opt}\cdot(\ln n+1)}{n}} \sum_i |S_i| \leq \sqrt{n \cdot \mathsf{opt} \cdot (\ln n + 1)}, \tag{6}$$

where the first inequality follows from the exit condition of the while loop in Line 6, and the second, from Lemma 4.4. Putting (6) and Lemma 4.3 together, we conclude that

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

□

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 weight of edge e.

Theorem 5.1. *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 through e, each one will contribute towards $p(e)$. We give edge e capacity $c(e) = p(e)w(e)$. We show that the 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 weight $c(e)/p(e)$. For any multicut E', $w(e)$ will be counted $p(e)$ times therefore the checkpoint cost of E' is $c(E')$. Thus the best solution is the minimum capacity multicut. We conclude that *SCP* and *UM* are equivalent with respect to approximation. □

Corollary 5.2. *SCP admits an $O(\log n)$ approximation in general graphs and a 2 approximation in trees.*

Proof. This follows from Theorem 5.1 and the fact that *UM* admits a ratio 2 approximation in trees [13] and an $O(\log n)$ approximation in general graphs [14]. □

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. We define the cost of a solution as the maximum between the number of forbidden pairs in the path and

1. This assures that there are no instances of value 0. Indeed if a value 0 is possible outcome for a yes instance then even if the the no instance has a single forbiidden pair the ratio is infinite. This is despite the fact that the two solutions are very close. Thus, the above definition helps to better reflects the gap between a yes and a no instance.

We show that the problem can not be approximated within $\Theta(n)$. This gap is perhaps the largest we can hope for and is rather rare.

Background. We provide hardness by giving a reduction from one-round two-prover interactive proof systems. The satisfiability problem (*SAT*) is defined as follows: A CNF Boolean formula ϕ is given, and the question is whether there is an assignment satisfying all clauses in ϕ. In the *3-SAT* problem, every clause has three literals. It is an immediate corollary from the PCP theorem [4] that there is exists some universal constant $\delta > 0$ so that *3-SAT* admits no $1 - \delta$ approximation, unless $P = NP$.

We need a variant of *3-SAT* called *3-SAT-5* in which every variable appears in at most 5 clauses (perhaps in negated form). In [26], an expander argument is used to show that there exists a universal $\varepsilon > 0$ so that *3-SAT-5* admits no $1 - \varepsilon$ approximation, unless $P = NP$.

The following a one-round two-prover system was introduced by [2] in order to translate the above theorem into one round two provers language. We start with the *3-SAT-5* instance above. The verifier chooses at random a clause C and then chooses a random variable x (or $\bar{x}$) in that clause. The verifier sends C to the first prover and x (or $\bar{x}$) to the second prover. The clause prover assigns values to its three literals in C and in particular to x. The literal prover assigns value to x. The verifier accepts if the two given values for x by the two provers match. A *strategy* is an a truth assignment to all variables. Let k be the number of answers and h be the number of queries. In [3] it is shown that in the case of a yes instance there is always a strategy (the same strategy for the two provers) that makes the verifier accept with probability 1. However, if the instance corresponds to a no instance, for any strategy, the probability of acceptance is at most $1 - \varepsilon/3$ with ε the constant derived from the reduction to *3-SAT-5*. In [3] a graph theoretic version of this problem is represented in which answers to the provers are called *labels*. The name given to the above problem in [3] is the *Labelcover* problem.

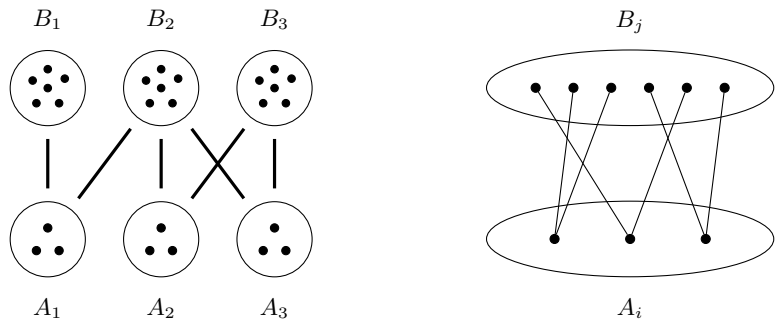
An even more explicit graph theoretic representation of *Labelcover* is given in [20]. The problem resulting is called the *MAX-REP* problem. We are given a bipartite graph $G(V_1, V_2, E)$. The sets V_1 and V_2 are partitioned into a disjoint union of q sets: $V_1 = \bigcup_{i=1}^{q} A_i$ and $V_2 = \bigcup_{j=1}^{q} B_j$. Two sets (A_i, B_j) form a *superedge* if there exists $a \in A_i$ and $b \in B_j$ so that $(a_i, b_j) \in E$. See Figure 3 for an example. 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 $(a_i, b_j) \in E$. The goal is to select unique representatives so as to maximize the number of super-edges covered.

Remark: The relation to one-round two-provers can be shown as follows. The A_i, B_j are question for the first and second prover, respectively. The sets A_i, B_j contain all possible answers of the respective questions A_i and B_j for the two provers. A superedge (A_i, B_j) is a query, namely two question that may be simultaneously sent to the two provers. Say that (A_i, B_j) is query. Two answers $a_i \in A_i$ and $b_j \in B_j$ are joined by an edge in E if and only if they are consistent, namely will cause the verifier to accept. This gives the *MAX-REP* graph.

Theorem 6.1 ([4, 26])**.** *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 $(1 - \varepsilon/3) \cdot h$ super-edges with ε the universal constant of the PCP theorem.*

Reduction. The reduction from *MAX-REP* to *PAFP* is as follows: 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 a super-edge. 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 6.2. *Unless* $\mathrm{P} = \mathrm{NP}$*, PAFP on undirected graphs admits no $(1 - \rho) \cdot n$ approximation ratio, where n is the number of vertices and $\rho > 0$ is a universal constant. The same holds for directed acyclic graphs.*



(a) The super graph. (b) A super edge (A_i, B_j).

Figure 3: An example of a *MAX-REP* instance.

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; indeed, using more vertices can only increase the number of forbidden pairs. 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 6.1, 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, Theorem 6.1 says that unsatisfiable formulas map to instances of *MAX-REP* with value at most $c' \cdot h$, for some $c' > 0$, which in turn map to instances of *PAFP* having value at least $c' \cdot h$. In addition, we are guaranteed 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 graphs.

Finally, 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.

Our hardness result for *MCP* rules out the possibility of a $2 - \varepsilon$ approximation for trees. Is it possible to get a better-than-2 asymptotic approximation ratio?

It would be interesting to study other natural variations of *MCP*. Consider, for example, the following *shortest distance* variant. After selecting a multicut Sol, each (s_i, t_t) computes its shortest checkpoint distance, where edges in Sol have length 1 and others have length 0. The objective is to minimize maximum distance across all s-t pairs. While this variant seems quite natural, we are unable at the moment to give it any non-trivial ratio. The main difficulty here is that a meaningful LP for this problem does not seem to exist. Even if we stick to the collection $\mathcal{P}$ of paths, every (s_i, t_i) is charged with $\min_{p \in \mathcal{P}_i}\{\mathsf{cp}(p)\}$, and then we take the maximum

over these values. There does not seem to be a meaningful LP for that variant either. Hence, at the moment, we do not know how to give a non-trivial approximation for these two natural variants of the checkpoint problem.

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*.

Acknowledgement The first author thanks Erik Demaine and Jelani Nelson for several fruitful discussions especially on initiating the problem.

References

[1] A. Agarwal, N. Alon, and M. Charikar. Improved approximation for directed cut problems. In *STOC*, pages 671–680, 2007.

[2] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. Syst. Sci.*, 54(2):317–331, 1997.

[3] S. Arora and C. Lund. *In Approximation Algorithms for NP-hard problems edited by D. Hochbaum*. PWS Publishing, 1996. Chapter10: 'Hardness on Approximation'.

[4] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *Journal of the ACM*, pages 501–555, 1998.

[5] 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.

[6] 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.

[7] J. Cheriyan, H. Karloff, and Y. Rabani. Approximating directed multicuts. *Combinatorica*, 25(3):251–269, 2005.

[8] J. Chuzhoy and S. Khanna. Polynomial flow-cut gaps and hardness of directed cut problems. *J. ACM*, 56(2), 2009.

[9] 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.

[10] I. Dinur and S. Safra. The importance of being biased. In *ACM Symposium on Theory of Computing (STOC)*, pages 33–42, 2002.

[11] 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.

[12] H. N. Gabow. Finding paths and cycles of superpolylogarithmic length. *SIAM J. Comp.*, 36(6):1648–1671, 2007.

[13] N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.

[14] 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.

[15] D. Golovin, V. Nagarajan, and M. Singh. Approximating the k-multicut problem. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 621–630, 2006.

[16] A. Gupta. Improved results for directed multicut. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 454–455, 2003.

[17] S. Khot. On the unique games conjecture. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, page 3, 2005.

[18] P. Kolman and O. Pangrac. On the complexity of paths avoiding forbidden pairs. *Discrete applied math*, 157:2871–2867, 2009.

[19] Y. Kortsarts, G. Kortsarz, and Z. Nutov. Greedy approximation algorithms for directed multicuts. *Networks*, 45(4):214–217, 2005.

[20] G. Kortsarz. On the hardness of approximating spanners. *Algorithmica*, 30(3):432–450, 2001.

[21] 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.

[22] F. Kuhn, P. von Rickenbach, R. Wattenhofer, E. Welzl, and A. Zollinger. Interference in cellular networks: The minimum membership set cover problem. In *International Computin and Combinatoics Conference (COCOON)*, pages 188–198, 2005.

[23] A. Levin and D. Segev. Partial multicuts in trees. *Theor. Comput. Sci.*, 369(1-3):384–395, 2006.

[24] J. Mestre. Lagrangian relaxation and partial cover (extended abstract). In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 539–550, 2008.

[25] J. Nelson. Notes on min-max multicommodity cut on paths and trees. Manuscript, 2009.

[26] C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Math. Oper. Res.*, 18(1):1–11, 1993.

[27] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *ACM Symposium on Theory of Computing (STOC)*, pages 255–264, 2008.

[28] T. J. Schaefer. The complexity of satisfiability problems. In *ACM Symposium on Theory of Computing (STOC)*, pages 216–226, 1978.

[29] P. Strimani and B. Sinha. Impossible pair-constrained test path generation in a program. *Information Sciences*, 28:87–103, 1982.

[30] K. Varadarajan and G. Venkataraman. Graph decomposition and a greedy algorithm for edge-disjoint paths. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 379–380, 2004.

[31] M. Yannakakis. On a class of totally unimodular matrices. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 10–16, 1980.

[32] H. Yinnone. On paths avoiding forbidden pairs of vertices in a graph. *Discrete Appl. Math.*, 74(1):85–92, 1997.