Generalized Metric Repair on Graphs

Chenglin Fan
Department of Computer Science; University of Texas at Dallas; Richardson, TX 75080, USA
cxf160130@utdallas.edu

Anna C. Gilbert
Department of Mathematics; University of Michigan; Ann Arbor, MI 48109, USA
annacg@umich.edu

Benjamin Raichel
Department of Computer Science; University of Texas at Dallas; Richardson, TX 75080, USA
benjamin.raichel@utdallas.edu

Rishi Sonthalia
Department of Mathematics; University of Michigan; Ann Arbor, MI 48109, USA
rsonthal@umich.edu

Gregory Van Buskirk
Department of Computer Science; University of Texas at Dallas; Richardson, TX 75080, USA
greg.vanbuskirk@utdallas.edu

Abstract

Many modern data analysis algorithms either assume or are considerably more efficient if the distances between the data points satisfy a metric. However, as real data sets are noisy, they often do not possess this fundamental property. For this reason, Gilbert and Jain [10] and Fan et al. [9] introduced the closely related *sparse metric repair* and *metric violation distance* problems. Given a matrix, representing *all* distances, the goal is to repair as few entries as possible to ensure they satisfy a metric. This problem was shown to be APX-hard, and an $O(OPT^{1/3})$-approximation was given, where OPT is the optimal solution size.

In this paper, we generalize the problem, by describing distances by a possibly *incomplete* positively weighted graph, where again our goal is to find the smallest number of weight modifications so that they satisfy a metric. This natural generalization is more flexible as it takes into account different relationships among the data points. We demonstrate the inherent combinatorial structure of the problem, and give an approximation-preserving reduction from MULTICUT, which is hard to approximate within any constant factor assuming UGC. Conversely, we show that for any fixed constant ς, for the large class of ς-chordal graphs, the problem is fixed parameter tractable, answering an open question from previous work. Call a cycle *broken* if it contains an edge whose weight is larger than the sum of all its other edges, and call the amount of this difference its *deficit*. We present approximation algorithms, one depending on the maximum number of edges in a broken cycle, and one depending on the number of distinct deficit values, both quantities which may naturally be small. Finally, we give improved analysis of previous algorithms for complete graphs.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures → Computational geometry

Keywords and phrases Approximation, FPT, Hardness, Metric Spaces

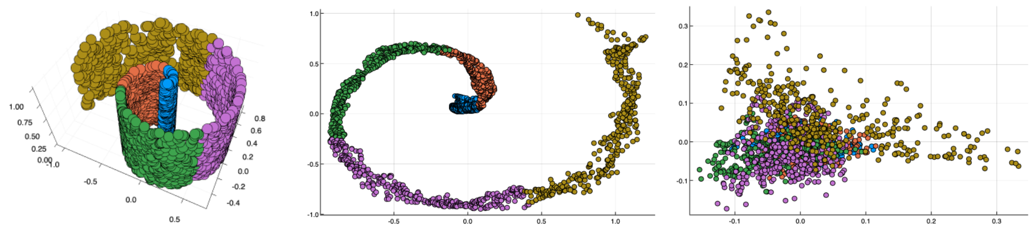
Digital Object Identifier 10.4230/LIPIcs.SWAT.2020.24

Funding C. Fan, B. Raichel, and G. Van Buskirk supported by NSF awards 1566137 and 1750780.

1 Introduction

Given a set of distances determined by a collection of data points, one of the most basic questions we can ask is whether the distances satisfy a metric. This basic property is fundamental to a large number of computational geometry and machine learning tasks such as metric learning, dimensionality reduction, and clustering (see for example [17, 2]). It is fortuitous when the underlying distances arise from a metric space or are at least well modeled by one, as certain tasks become provably easier over metric data (e.g., approximating the optimal TSP tour), and moreover it allows us to use a number of computational tools such as metric embeddings. However, due to noise, missing data, and other corruptions, in practice these distances often do not adhere to a metric.

As a motivating example, consider the following standard manifold learning task ([3, 13, 15]). Given a high dimensional data set, we wish to uncover its intrinsic lower dimensional structure, allowing us to visualize and to understand the geometry of the data. Isomap [15] is one of the standard embedding tools used to find this lower dimensional structure, and Figure 1.1 shows how Isomap nicely recovers the 2d spiral when embedding a 3d Swiss roll data set. However, as shown on the right in Figure 1.1, if we perturb even a small fraction of the distances this structure is lost in the embedding produced by Isomap.



(a) Original Swissroll data. **(b)** Embedded true distances. **(c)** Embedded corrupted distances.

Figure 1.1 (a) 2000 data points in the Swissroll. For (b) and (c) we took the pairwise distance matrix and added $2\mathcal{N}(0,1)$ noise to 5% of the distances. We then constructed the 30-nearest-neighbor graph G from these distances, where roughly 8.5% of the edge weights of G were perturbed. For (b) we used the true distances on G as the input to Isomap. For (c) we used the perturbed distances.

Motivated by the above applications, the problem of minimally fixing the distances to uncover the data metric was previously considered. Specifically, Fan et al. [9] and Gilbert and Jain [10] respectively formulated the *Metric Violation Distance (MVD)* and the *Sparse Metric Repair (SMR)* problems, where in both cases one is given a *full* distance matrix, and the goal is to modify as few entries as possible so that the repaired distances satisfy a metric.

More generally, however, the underlying distance graph will be incomplete as data may be missing or the constructed distance graph is inherently sparse, as the above manifold example demonstrates. Working directly with this incomplete graph is not only computationally more desirable when the graph is sparse, but also may be necessary to uncover the ground truth. For example, observe that for any graph we can attempt to fill in its missing edges by assigning them weights according to their shortest path distance. Thus naively one could attempt to fix the input graph, by solving MVD/SMR on this complete graph, and afterwards dropping any selected edges that were not in the original graph. Figure 1.2 shows that doing so, however, can produce radically different and sub-optimal solutions.

Thus to appropriately capture this more general problem, we define the *Graph Metric Repair* problem as the natural graph theoretic generalization of the MVD and SMR problems:

> Given a positively weighted undirected graph $G = (V, E, w)$ and a set $\Omega \subseteq \mathbb{R}$, find the smallest set of edges $S \subseteq E$ such that by modifying the weight of each edge in S, by adding a value from Ω, the new distances satisfy a metric.

The additional graph structure introduced in the generalized problem lets us incorporate different types of relationships amongst data points and gives us more flexibility in its

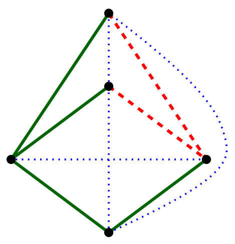


Figure 1.2 Original graph is the four solid green edges of weight 1, and two dashed red edges of weight 4. Added blue dotted edges all have weight 2. The original graph is repaired by increasing the lower left green edge, but the optimal solution in the complete graph decreases the two red edges.

structure, and hence avails itself to be applicable to a richer class of problems. This general graph structure also elucidates the deep connections to cutting problems, which underlie several results in this paper, and which were not previously observed in [9, 10]. In particular, as discussed below, our problem is closely related to MULTICUT (Problem 4.1), a generalization of the standard s-t cut problem to multiple s-t pairs, as well as LB-CUT (Problem 4.2), where only s-t paths with lengths up to a given threshold L must be cut.

It should also be noted that Graph Metric Repair, as well as MVD and SMR, are related to a large number of other previously studied problems. A short list includes: metric nearness, seeking the metric minimizing the sum of distance value changes [5]; metric embedding with outliers, seeking the fewest points whose removal creates a metric [14]; matrix completion, seeking to fill missing matrix entries to produce a low rank [6]; and many more. See [9] for a more detailed discussion of these and other problems.

Contributions and Results: The main contributions of this paper are as follows:

- We transition all previously known structural results about SMR and MVD to the new graph theoretic version. In particular, we provide a *characterization* for the support of solutions to the increase ($\Omega = \mathbb{R}_{\geq 0}$) and general ($\Omega = \mathbb{R}$) versions of the problem. Furthermore, we provide a new structural result showing that the increase only problem reduces to the general one, where it is unknown if such a result holds for SMR and MVD.
- For any fixed constant ς, by parameterizing on the size of the optimal solution, we present a *fixed parameter tractable* algorithm for the case when G is ς-chordal. This not only answers an open question posed by [9] for complete graphs, but significantly extends it to the larger ς-chordal case (see [7] for characterizations of such graphs, many of which are the complements of a variety of families of graphs). Moreover, we get an upper bound on the number of optimal supports, as each one is seen by some branch of the algorithm.
- We give polynomial-time approx-preserving reductions from MULTICUT and LB-CUT to graph metric repair. This connection to the well studied MULTICUT problem is interesting in its own right, but by [8] it also implies graph metric repair is NP-hard, and cannot be approximated within any constant factor assuming the Unique Games Conjecture (UGC).
- We give approximation algorithms, parameterized by different measures of how far the input is from a metric. Significantly, our approximations mirror our hardness results. Call a cycle *broken* if it contains an edge whose weight is larger than the sum of all its other edges, and call the amount of this difference its *deficit.* We give an L-approximation, where L is the maximum number of edges in a broken cycle, while LB-CUT gives $\Omega(\sqrt{L})$-hardness. We give an $O(\kappa \log n)$-approximation, where κ is the number of distinct cycle deficit values, while in general the best known approximation for MULTICUT is $O(\log n)$.

- Finally, we give improved analysis of previous algorithms for the complete graph case. To keep the focus on our main results, this entire section has been moved to Appendix C.

2 Preliminaries

2.1 Notation and problem definition

Let us start by defining some terminology. Throughout the paper, the input is an undirected and weighted graph $G = (V, E, w)$. A subgraph $C = (V', E')$ is called a k-cycle if $|V'| = |E'| = k$, and the subgraph is connected with every vertex having degree exactly 2. We often overload this notation and use C to denote either the cyclically ordered list of vertices or edges from this subgraph. Let $C \setminus e$ denote the set of edges of C after removing the edge e, and $\pi(C \setminus e)$ denote the corresponding induced path between the endpoints of e.

A cycle C is **broken** if there exists an edge $h \in C$ such that $w(h) > \sum_{e \in C \setminus h} w(e)$. In this case, we call the edge h the **heavy** edge of C, and all other edges of C are called **light** edges. We call a set of edges a **light cover** if it contains at least one light edge from each broken cycle. Similarly, we call it a **regular cover** if it contains at least one edge from each broken cycle. We say that a weighted graph $G = (V, E, w)$ **satisfies a metric** if there are no broken cycles. Finally, let $\mathrm{Sym}_n(\Omega)$ be the set of $n \times n$ symmetric matrices with entries drawn from $\Omega \subseteq \mathbb{R}$. Note that the weight function w can be viewed as an $n \times n$ symmetric matrix (missing edges get weight ∞), and thus for any $W \in \mathrm{Sym}_n(\Omega)$, the matrix sum $w + W$ defines a new weight function. Now we can define the generalized graph metric repair problem as follows. In the following, $\|W\|_0$ is the number of non-zero entries in the matrix W, i.e., the ℓ_0 pseudonorm when viewing the matrix W as a vector.

▶ **Problem 2.1.** *Given $\Omega \subseteq \mathbb{R}$ and a positively weighted graph $G = (V, E, w)$ we want to find*

$$\operatorname*{arg\,min}_{W \in \mathrm{Sym}(\Omega)} \|W\|_0 \text{ such that } G = (V, E, w + W) \text{ satisfies a metric, or return NONE,}$$

if no such W exists. Denote this problem as graph metric repair or MR(G, Ω).

A matrix W is an *optimal solution* if it realizes the arg min in the above, and is a *solution* (without the *optimal* prefix) if $G = (V, E, w + W)$ satisfies a metric, but $\|W\|_0$ is not required to be minimum. The **support** of a matrix $W \in \mathrm{Sym}(\Omega)$, denoted S_W, is the set of edges corresponding to non-zero entries in W. As we will see in Proposition 7, given a support for a solution W, we can easily find satisfying entries. Thus, the main difficulty lies in finding the support. Throughout we use OPT to denote the size of the support of an optimal solution.

We also need the following basic graph theory definitions: K_n is the complete graph on n vertices. C_n is the cycle n vertices. A **chord** of a cycle is an edge connecting two non-adjacent vertices. For a given value ς, a graph G is called a ς-**chordal** if the size of the largest chordless cycle in G is $\leq \varsigma$.

Let the **deficit** of a broken cycle C, denoted $\delta(C)$, be the weight of its heavy edge minus the sum of the weights of all other edges in C. Similarly, $\delta(G)$ denotes the maximum of $\delta(C)$ over all broken cycles. Finally, let $L + 1$ be the maximum number of edges in a broken cycle (i.e., L counts the light edges). Note δ and L are both parameters measuring the extent to which cycles are broken, δ with respect to weights and L with respect to the number of edges.

In several places we compute all pairs shortest paths (APSP). Let T_{APSP} denote the time to do so, where $T_{\text{APSP}} = O(mn + n^2 \log n)$ using Dijkstra's algorithm and Fibonacci heaps.

2.2 Previous results

Fan et al. [9] and Gilbert and Jain [10] studied the special case of $\mathrm{MR}(G,\Omega)$ where $G = K_n$. Three sub-cases based on Ω were considered, namely $\Omega = \mathbb{R}_{\leq 0}$ (decrease only), $\mathbb{R}_{\geq 0}$ (increase only), and $\mathbb{R}$ (general). Various structural, hardness, and algorithmic results were presented for these cases. In particular, the major results from these previous works are as follows. (Note the notation and terminology here differs slightly from [9, 10].)

▶ **Theorem 1.** *[9, 10] The problem $MR(K_n, \mathbb{R}_{\leq 0})$ can be solved in $O(T_{\mathrm{APSP}})$ time.*

▶ **Theorem 2.** *[9] For a complete positively weighted graph $K_n = (V, E, w)$ and $S \subseteq E$:*
1. *S is a regular cover if and only if S is the support to a solution to $MR(K_n, \mathbb{R})$.*
2. *S is a light cover if and only if S is the support to a solution to $MR(K_n, \mathbb{R}_{\geq 0})$.*

▶ **Theorem 3.** *[9, 10] Given the support S of a solution to $MR(K_n, \mathbb{R}_{\geq 0})$ or $MR(K_n, \mathbb{R})$, in polynomial time one can find a weight assignment to the edges in S which is a solution.*

[10] Moreover, for $MR(K_n, \mathbb{R}_{\geq 0})$, if $K_n - S$ is connected, then for any edge $uv \in S$, setting the weight of uv to be the shortest distance between u and v in $K_n - S$ is a solution.

▶ **Theorem 4.** *[9] The problems $MR(K_n, \mathbb{R}_{\geq 0})$ and $MR(K_n, \mathbb{R})$ are APX-Complete, and moreover permit $O(OPT^{1/3})$ approximation algorithms.*

3 Transitioning to Graph Metric Repair

In this section we generalize Theorems 1, 2, and 3 to the case when G is any graph, and additionally show that for general graphs $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ reduces to $\mathrm{MR}(G, \mathbb{R})$. Subsequently, in the later sections of paper, we provide a number of new stronger hardness and approximation results for $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ and $\mathrm{MR}(G, \mathbb{R})$ for general graphs, as well as an FPT algorithm for ς-chordal graphs, in effect generalizing and strengthening Theorem 4, and answering previously unresolved questions.

For $\mathrm{MR}(G, \mathbb{R}_{\leq 0})$ we have the following generalization of Theorem 1. Moreover, we observe the hardness proof of [9] implies if weights are allowed to increase even by a single value, the problem is APX-Complete. The proof of the theorem below follows fairly directly from previous work, and so has been moved to Appendix A.1, which contains additional corollaries.

▶ **Theorem 5.** *The problem $MR(G, \mathbb{R}_{\leq 0})$ can be solved in $O(T_{\mathrm{APSP}})$ time.*

Moreover, the problem becomes hard if even a single positive value is allowed. That is, if $0 \in \Omega$ and $\Omega \cap \mathbb{R}_{>0} \neq \emptyset$ then $MR(G, \Omega)$ is APX-Complete.

3.1 Structural results

Theorem 2 suggests that the problem is mostly combinatorial in nature. We shall see that, in general, the difficult part of the problem is finding the support of an optimal solution. Next, we present a characterization of the support of all solutions to the graph metric repair problem, generalizing Theorems 2, 3. It should be noted that the proof of the following is significantly simpler than the proof of Theorem 2 in [9]. The key insight is:

(i) If the shortest path between two adjacent vertices is not the edge connecting them, then this edge is the heavy edge of a broken cycle.

▶ **Theorem 6.** *For any positively weighted graph $G = (V, E, w)$ and $S \subseteq E$:*
1. *S is a regular cover if and only if S is the support to a solution to $MR(G, \mathbb{R})$.*

2. *S is a light cover if and only if S is the support to a solution to $MR(G, \mathbb{R}_{\geq 0})$.*

Proof. First, assume that S is the support of a solution to $\mathrm{MR}(G, \mathbb{R})$ ($\mathrm{MR}(G, \mathbb{R}_{\geq 0})$). Suppose C is a broken cycle in G. If S does not contain any (light) edges from C, then changing (increasing) the weights on S could not have fixed C. Hence, S must be a regular (light) cover thus proving the "if" direction of both parts of the theorem.

For the "only if" direction, we are given a regular (light) cover $S \subseteq E$ which we use to define a graph $\hat{G} = (V, E \setminus S, w)$. Note that since S is either a regular or light cover, S contains at least one edge from all broken cycles of G. Thus, since $\hat{G}$ is G with the edges of S removed, $\hat{G}$ has no broken cycles. Therefore, the shortest path between all adjacent vertices in $\hat{G}$ is the edge connecting them.

Now we define another graph $G' = (V, E, w')$ where $w'(e) = w(e)$ for all $e \in E \setminus S$ and for all $e \in S$, $w'(e)$ is the length of the shortest path between its end points in $\hat{G}$ or $\|w\|_\infty$ (the maximum edge weight in $\hat{G}$) if no path exists.

To prove **1.**, it suffices to show G' satisfies a metric, since G' is G with only weights from edges in S modified. For any edge $e \in E$, if $w'(e)$ is the shortest path between its nodes in G' then e is not a heavy edge in G'. Therefore, edges that are in both G' and $\hat{G}$ and edges that are in G' whose weight was set to length of the shortest path between its end points in $\hat{G}$ are not heavy edges. Thus, we only need to look at edges in G' whose weight is $\|w\|_\infty$. These are edges that connect two disconnected components in $\hat{G}$. Thus, any cycle in G' with such an edge must involve another edge between components which also has weight $\|w\|_\infty$. However, a cycle with two edges of maximum weight cannot be broken, and thus such edges cannot be heavy in G'. Therefore, there are no heavy edges in G', and so G' satisfies a metric.

To prove **2.**, it now suffices to show that for all $e \in E$, we have that $w'(e) \geq w(e)$. For all $e \in E \setminus S$, we know that $w'(e) = w(e)$. Now, suppose for contradiction that for some $e \in S$, we have $w'(e) < w(e)$. Note if we set $w'(e) = \|w\|_\infty$, then we cannot have $w'(e) < w(e)$. Thus, $w'(e)$ must be the weight of the shortest path between the end points of e in $\hat{G}$. Let P be this shortest path in $\hat{G}$. This implies G has a broken cycle $C = P \cup \{e\}$ for which e is the heavy edge. Since S is a light cover, it has a light edge from each broken cycle. So, S must have a light edge from C, but then P could not have existed in $\hat{G}$, a contradiction. Hence, $w'(e) \geq w(e)$ and we have an increase only solution with such a set S. ◀

Furthermore, given a weighted graph G and a potential support S_W for a solution W, in $O(T_{\mathrm{APSP}})$ time we can determine if there exists a valid (increase only or general) solution on that support, and if so, find one. This is a generalization of Theorem 3, interestingly improving upon the linear programming approach of [9]. Its proof is related to the above theorem, and again uses insight (i), though due to space has been moved to Appendix A.2.

Algorithm 1 Verifier

```
1: function VERIFIER($G = (V, E, w), S$)
2:     $M = \|w\|_\infty$, $\hat{G} = (V, E, \hat{w})$
3:     For each $e \in S$ set $\hat{w}(e) = M$ and for each $e \in E \setminus S$, set $\hat{w}(e) = w(e)$
4:     For each $(u, v) \in E$, update $w(u, v)$ to be length of the shortest path from $u$ to $v$ in $\hat{G}$
5:     if Only edges in $S$ had weights changed (or increased for increase only case) then
6:         return $w$
7:     else
8:         return NULL
```

▶ **Proposition 7.** *The* Verifier *algorithm, given a weighted graph G and a potential support for a solution S, determines in $O(T_{\text{APSP}})$ time whether there exists a valid (increase only or general) solution on that support and if so finds one.*

3.2 Reducing MR$(G, \mathbb{R}_{\geq 0})$ to MR$(G, \mathbb{R})$

We now show that $\text{MR}(G, \mathbb{R}_{\geq 0})$ reduces to $\text{MR}(G, \mathbb{R})$. In later sections, this lets us focus on $\text{MR}(G, \mathbb{R})$ for our algorithms and $\text{MR}(G, \mathbb{R}_{\geq 0})$ for our hardness results. Note that whether an analogous statement holds for the previously studied $G = K_n$ case, is not known, and the following does not immediately imply this as it does not construct a complete graph.

▶ **Theorem 8.** *There is an approximation-preserving, polynomial-time reduction from $MR(G, \mathbb{R}_{\geq 0})$ to $MR(G, \mathbb{R})$.*

Proof. Let $G = (V, E, w)$ be an instance of $\text{MR}(G, \mathbb{R}_{\geq 0})$. Find the set $H = \{(s_1, t_1), \ldots, (s_{|H|}, t_{|H|})\}$ of heavy edges of all broken cycles by comparing the weight of each edge to the shortest path distance between its endpoints. We now construct an instance, $G' = (V', E', w)$, of $\text{MR}(G, \mathbb{R})$. For all $1 \leq i \leq |H|$ and $1 \leq j \leq |E|+1$, let $Q = \{v_{ij}\}_{i,j}$ be a vertex set, and let $F_l = \{(s_i, v_{ij})\}_{i,j}$ and $F_r = \{(t_i, v_{ij})\}_{i,j}$ be edge sets. Let $V' = V \cup Q$ and $E' = E \cup F_l \cup F_r$, where all (s_i, v_{ij}) edges in F_l have weight $Z = 1 + \max_{e \in E} w(e)$, and for any i all (t_i, v_{ij}) edges in F_r have weight $Z - w((s_i, t_i))$.

Let C be any broken cycle in G with heavy edge (s_i, t_i) for some i. First, observe that $C' = (C \setminus (s_i, t_i)) \cup \{(s_i, v_{ij}), (t_i, v_{ij})\}$ is a broken cycle with heavy edge (s_i, v_{ij}), for any j. To see this, note that $w((s_i, v_{ij})) = Z = w((t_i, v_{ij})) + w((s_i, t_i))$. Thus since C is broken,

$$w((s_i, v_{ij})) = w((t_i, v_{ij})) + w((s_i, t_i)) > w((t_i, v_{ij})) + w(C \setminus (s_i, t_i)),$$

and thus by definition C' is broken with heavy edge (s_i, v_{ij}). Hence each broken cycle C in G, with heavy edge (s_i, t_i), corresponds to $|E| + 2$ broken cycles in G', namely, C itself and the cycles obtained by replacing (s_i, t_i) with a pair $(s_i, v_{ij}), (t_i, v_{ij})$, for any j.

We now show the converse, that any broken cycle C' in G' is either also a broken cycle C in G, or obtained from a broken cycle C in G by replacing (s_i, t_i) with $(s_i, v_{ij}), (t_i, v_{ij})$ for some j. First, observe that for any i, any cycle containing the edge (s_i, v_{ij}) must also contain the edge (t_i, v_{ij}), and moreover, if a cycle containing such a pair is broken, then its heavy edge must be (s_i, v_{ij}) as $w((s_i, v_{ij})) = Z$. Similarly, any cycle containing more than one of these pairs of edges (over all i and j) is not broken, since such cycles then would contain at least two edges with the maximum edge weight Z. So let C' be any broken cycle containing exactly one such (s_i, v_{ij}), (t_i, v_{ij}) pair. Note that C' cannot be the cycle $((s_i, v_{ij}), (t_i, v_{ij}), (s_i, t_i))$, as this cycle is not broken because $w((s_i, v_{ij})) = w((t_i, v_{ij})) + w((s_i, t_i))$. Thus, $C = C' \setminus \{(s_i, v_{ij}), (t_i, v_{ij})\} \cup \{(s_i, t_i)\}$ is a cycle, and C' being broken implies C is broken with heavy edge (s_i, t_i), implying the claim. This holds since

$$w(s_i, t_i) = w((s_i, v_{ij})) - w((t_i, v_{ij})) > w(C' \setminus (s_i, v_{ij})) - w((t_i, v_{ij})) = w(C \setminus (s_i, t_i)).$$

Now consider any optimal solution M to the $\text{MR}(G, \mathbb{R}_{\geq 0})$ instance G, which by Theorem 6 we know is a minimum cardinality light cover of G. By the above, we know that M is also a light cover of G', and hence is also a regular cover of G'. Thus by Theorem 6, M is a valid solution to the $\text{MR}(G, \mathbb{R})$ instance. Conversely, consider any optimal solution M' to the $\text{MR}(G, \mathbb{R})$ instance G', which by Theorem 6 is a minimum cardinality regular cover of G'. The claim is that M' is also a light cover of G, and hence is a valid solution to the $\text{MR}(G, \mathbb{R}_{\geq 0})$ instance. To see this, observe that since all broken cycles in G are broken cycles in G', M' must be a regular cover of all broken cycles in G, and we now argue that it is in fact a light cover. Specifically, consider all the broken cycles in G which have a common heavy edge (s_i, t_i). Suppose there is some cycle in this set, call it C, which is not light covered by M'. As M' is a regular cover for G', this implies that for any j, the broken cycle described above determined by removing the edge (s_i, t_i) from C and adding edges (s_i, v_{ij}) and (t_i, v_{ij}), must be covered either with (s_i, v_{ij}) or (t_i, v_{ij}). However, as j ranges over $|E| + 1$ values, and these edge pairs have distinct edges for different values of j, M' has at least $|E| + 1$ edges. This is a clear contradiction with M' being a minimum sized cover, as any light cover of G is a regular cover of G', and G only has $|E|$ edges in total. ◀

4 Hardness

Previously, [9] gave an approximation-preserving reduction from Vertex Cover to both $\mathrm{MR}(K_n, \mathbb{R})$ and $\mathrm{MR}(K_n, \mathbb{R}_{\geq 0})$. Thus, both are APX-complete, and in particular are hard to approximate within a factor of $2 - \varepsilon$ for any $\varepsilon > 0$, assuming UGC [11]. Since these hardness results were proven for complete graphs, they also immediately apply to the general problems $\mathrm{MR}(G, \mathbb{R})$ and $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$. Here we give stronger hardness results for $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ and $\mathrm{MR}(G, \mathbb{R})$ by giving approximation-preserving reductions from MULTICUT and LB-CUT.

▶ **Problem 4.1** (MULTICUT). *Given an undirected unweighted graph $G = (V, E)$ on $n = |V|$ vertices together with k pairs of vertices $\{s_i, t_i\}_{i=1}^k$, compute a minimum size subset of edges $M \subseteq E$ whose removal disconnects all the demand pairs, i.e., in the subgraph $(V, E \setminus M)$ every s_i is disconnected from its corresponding vertex t_i.*

Chawla *et al.* [8] proved that if UGC is true, then it is NP-hard to approximate MULTICUT within any constant factor $L > 0$, and assuming a stronger version of UGC, within $\Omega(\sqrt{\log \log n})$. (The MULTICUT version in [8] allowed weights, but they remark their hardness proofs extend to the unweighted case.)

▶ **Theorem 9.** *There is an approximation-preserving, polynomial-time reduction from MULTICUT to $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$.*

Proof. Let $G = (V, E)$ be an instance of MULTICUT with k pairs of vertices $\{s_i, t_i\}_{i=1}^k$. First, if $(s_i, t_i) \in E$ for any i, then that edge must be included in the solution M. Thus, we can assume no such edges exists in the MULTICUT instance, as assuming this can only make it harder to approximate the optimum value of the MULTICUT instance. We now construct an instance of $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$, $G' = (V', E', w)$. Let $V' = V$ and $E' = E \cup \{s_i, t_i\}_{i=1}^k$ where the edges in E have weight one and the edges (s_i, t_i), for all $i \in [k]$, have weight $n = |V|$.

If a cycle in G' has exactly one edge of weight n, then it must be broken since there can be at most $n - 1$ other edges in the cycle. Conversely, if a cycle C has no edge or more than one edge with weight n, then C does not have a heavy edge, and so is not broken.

Note that the edges from G are exactly the weight one edges in G', and thus, the paths in G are in one-to-one correspondence with the paths in G' which consist of only weight one edges. Moreover, the weight n edges in G' are in one-to-correspondence with the (s_i, t_i) pairs from G. Thus, the cycles in G' with exactly one weight n edge followed by paths of all weight one edges connecting their endpoints, which by the above are exactly the set of broken cycles, are in one-to-one correspondence with paths between (s_i, t_i) pairs from G. Therefore, a minimum cardinality subset of edges which light cover all broken cycles, i.e., an optimal $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ support, corresponds to a minimum cardinality subset of edges from E which cover all paths from s_i to t_i for all i, i.e., an optimal solution to MULTICUT. ◀

▶ **Problem 4.2** (LB-CUT). *Given a value L and an undirected unweighted graph $G = (V, E)$ with source s and sink t, find a minimum size subset of edges $M \subseteq E$ such that no s-t-path of length less than or equal to L remains in the graph after removing the edges in M.*

An instance of LB-CUT with length L, is referred to as an instance of L-LB-CUT. For any fixed L, Lee [12] showed that it is hard to approximate L-LB-CUT within a factor of $\Omega(\sqrt{L})$. Using a similar reduction as above, we argue the following.

▶ **Theorem 10.** *For any fixed value L, there is an approximation-preserving, polynomial-time reduction from L-LB-CUT to MR$(G, \mathbb{R}_{\geq 0})$.*

Proof. Let $G = (V, E)$ be an instance of L-LB-CUT with source s and sink t. First, if $(s, t) \in E$, then that edge must be included in the solution M. Thus we can assume that edge is not in the LB-CUT instance, as assuming this can only make it harder to approximate the optimum value of the LB-CUT instance. We now construct an instance of MR$(G, \mathbb{R}_{\geq 0})$, $G' = (V', E', w)$. Let $V' = V$ and $E' = E \cup \{(s, t)\}$ where the edges in E have weight 1 and the edge (s, t) has weight $L + 1$.

First, observe that any cycle containing the edge (s, t) followed by $\leq L$ unit weight edges is broken, as the sum of the unit weight edges will be $< L + 1 = w((s, t))$. Conversely, any broken cycle must contain the edge (s, t) followed by $\leq L$ unit weight edges. Specifically, if a cycle does not contain (s, t) then it is unbroken since all edges would then have weight 1. Moreover, if a cycle contains (s, t) and $> L$ other edges, then the total sum of those unit edges will be $\geq L + 1 = w((s, t))$.

Note that the edges from G are exactly the weight one edges in G', and thus the paths in G are in one-to-one correspondence with the paths in G' which consist of only weight one edges. Moreover, the edge (s, t) in G' corresponds with the source and sink from G. Thus by the above, the broken cycles in G' are in one-to-one correspondence with s-t-paths with length $\leq L$ in G. Therefore, a minimum cardinality subset of edges which light cover all broken cycles, i.e., an optimal support to MR$(G, \mathbb{R}_{\geq 0})$, corresponds to a minimum cardinality subset of edges from E which cover all paths from s to t of length $\leq L$, i.e., an optimal solution to LB-CUT. ◀

In the L-LB-CUT to MR$(G, \mathbb{R}_{\geq 0})$ reduction of Theorem 10, one edge (the s, t pair) has weight $L + 1$ and all other edges have unit weight. Moreover, in the reduction from MR$(G, \mathbb{R}_{\geq 0})$ to MR$(G, \mathbb{R})$ of Theorem 8, the max edge weight increases by 1. Thus, by these reductions, and previous hardness results, we have the following summarizing theorem.

▶ **Theorem 11.** *MR$(G, \mathbb{R}_{\geq 0})$ and MR$(G, \mathbb{R})$ are APX-complete, and moreover assuming UGC neither can be approximated within any constant factor.*

For any positive integer L, consider the problem defined by the restriction of MR$(G, \mathbb{R})$ to integer weight instances with maximum edge weight L and minimum edge weight 1, or the further restriction of MR$(G, \mathbb{R}_{\geq 0})$ to instances where all weights are 1 except for a single weight L edge. Then assuming UGC these problems are hard to approximate within $\Omega(\sqrt{L})$.

5 Fixed Parameter Analysis for ς-Chordal Graphs

Let ς be a fixed constant, and let F_ς be the family of all ς-chordal graphs. Here we provide an FPT for MR$(G, \mathbb{R})$ for any $G \in F_\varsigma$, parameterized on the optimal solution size OPT.

By Theorem 6, we seek a minimum sized cover of all broken cycles. First, we argue below that if G has a broken cycle, then it has a broken chordless cycle. This seems to imply a natural FPT algorithm for constant ς. Namely, find an uncovered broken chordless cycle and recursively try adding each one of its edges to our current solution.[*] However, it is possible to cover all broken chordless cycles while not covering the broken chorded cycles. These cycles are difficult to cover as they may be much larger than ς, though again by Theorem 6 they must be covered.

Consider an optimal solution W, with support S_W. Suppose that we have found a subset $S \subsetneq S_W$, covering all broken chordless cycles in G. Intuitively, if we add to each edge in S its weight from W, then any remaining broken chordless cycle must be covered further, in effect revealing which edges to consider from the chorded cycles from the original graph G. The challenge, however, is of course that we don't know W a priori. We argue that despite this one can still identify a bounded sized subset of edges containing an edge from a cycle needing to be covered further.

▶ **Lemma 12.** *If G has a broken cycle, then G has a broken chordless cycle.*

Proof. Let $C = v_1, \ldots, v_k$ be the broken cycle in G with the fewest edges, with $v_1 v_k$ being the heavy edge. If C is chordless, then the claim holds. Otherwise, this cycle has at least one chord $v_i v_j$. Now there are two paths P_1 and P_2 from v_i to v_j on the cycle. Let P_1 be the path containing the heavy edge of C. If $w(v_i, v_j) > \sum_{e \in P_2} w(e)$, then P_2 together with the edge $v_i v_j$ defines a broken cycle with fewer edges than C. On the other hand, if $w(v_i, v_j) \leq \sum_{e \in P_2} w(e)$ then P_1 together with the edge $v_i v_j$ defines a broken cycle with fewer edges than C. In either case we get a contradiction as C was the broken cycle with the fewest edges. ◀

Our FPT is shown in Algorithm 2, where we recursively build a potential support S up to our current guess at the optimal size k. The following lemma is key to arguing correctness.

▶ **Lemma 13.** *Consider any optimal solution W and its support S_W to an instance of metric repair for $G = (V, E, w) \in F_\varsigma$. If $S \subsetneq S_W$, then $F(G, S, OPT)$ adds at least one edge in $S_W \setminus S$ to P.*

Proof. Consider the auxiliary graph $G_S = (V, E, \tilde{w})$, which has the same vertex and edge sets as G, but with the modified weight function:

$$\tilde{w} = \begin{cases} w(e) & e \notin S \\ W(e) + w(e) & e \in S \end{cases}$$

Since $S \subsetneq S_W$, we have that G_S has a broken cycle. Thus, by Lemma 12, G_S has a chordless broken cycle. Suppose there is a chordless broken cycle in G_S that is edge disjoint from S (which occurs if and only if it is also broken in G), in which case, line 4 finds such a cycle. As this is a broken cycle, it must be covered by some edge in $S_W \setminus S$, and thus, we have added some edge in $S_W \setminus S$ to P.

Let us assume otherwise, that any chordless broken cycle in G_S has non-empty intersection with S. Let C be any such chordless broken cycle with $C \cap S \neq \emptyset$. Observe that as C is broken in G_S, it must be that $|C \cap S| < |C|$, as otherwise it would imply W was not a solution. Thus, as $G \in F_\varsigma$, we know that $|C| \leq \varsigma$, and so $|C \cap S| < \varsigma$. This implies in some for loop iteration, $C \in \mathcal{C}$ on line 7.

Let h be the heavy edge, in G_S, of the broken cycle C. We now have two cases:

[*]One might construe this as FPT kernelization. The edges of the broken chordless cycles do form a kernel but its size is not bounded in our parameter. As an example, take $G = K_n$, set one edge weight to $n+1$, and all other weights to 1. There are $2n-3$ edges in the kernel while the optimal solution has size 1.

Algorithm 2 FPT

```
1: function F($(G, S, k)$)
2:     if $|S| = k$ then return VERIFIER($(G,S)$)
3:     $P = \emptyset$
4:     if there exists a broken chordless cycle $C$ such that $C \cap S = \emptyset$ then $P = C$
5:     else
6:         for $s \subseteq S$ such that $|s| \leq \varsigma - 1$ do
7:             Let $\mathcal{C} = \{$Chordless cycles $C$ such that $C \cap S = s\}$
8:             $C_1 \leftarrow \arg\min_{C \in \mathcal{C}} \sum_{e \in C \setminus s} w(e)$
9:             $C_2 \leftarrow \arg\max_{C \in \mathcal{C}} w(h) - \sum_{e \in C \setminus (s \cup \{h\})} w(e)$, where $h = \arg\max_{f \in C \setminus s} w(f)$
10:            Add $(C_1 \cup C_2) \setminus S$ to $P$
11:    for $e \in P$ do
12:        $X = \mathrm{F}(G, S \cup \{e\}, k)$
13:        if $X \neq$ NULL then return $X$
14:    return NULL

15: function FPTWRAPPER($(G)$)
16:    for $k = 1, 2, \ldots$ do
17:        $X = \mathrm{F}(G, \emptyset, k)$
18:        if $X \neq$ NULL then return $X$
```

Case 1: $h \in S$. In this case we have that

$$W(h) + w(h) > \underbrace{\sum_{e \in C \setminus S} w(e)}_{(1)} + \sum_{e \in S} W(e) + w(e).$$

On line 8 we found a cycle C_1 that minimized (1). Thus, since C is broken in G_S, C_1 is also broken in G_S, and so must be covered by some edge in $S_W \setminus S$. Hence, we added some edge in $S_W \setminus S$ to P.

Case 2 $h \notin S$. In this case h has the maximum weight of all edges in $C \setminus s$. We have that

$$\underbrace{w(h) - \sum_{e \in C \setminus (S \cup \{h\})} w(e)}_{(2)} > \sum_{e \in S} W(e) + w(e).$$

On line 9 we found a cycle C_2 maximizing (2). Thus, if C is broken in G_S, then C_2 is broken in G_S, and so must be covered by some edge in $S_W \setminus S$. Hence, we added some edge in $S_W \setminus S$ to P. ◀

▶ **Lemma 14.** *Any time we call F, we have that $|P| \leq 2\varsigma|S|^\varsigma$*

Proof. Note $|P|$ is upper bounded by ς multiplied by the number of chordless cycles we add. If the conditional on line 4 is true then we add only a single chordless cycle to P. Otherwise, for each $s \subseteq S$ such that $|s| \leq \varsigma - 1$ we find two cycles. There are at most

$$\sum_{i=1}^{\varsigma-1} \binom{|S|}{i} \leq \sum_{i=1}^{\varsigma-1} |S|^i \leq |S|^\varsigma$$

many such subsets, and thus we add at most $2|S|^\varsigma$ many cycles, implying the claim. ◀

▶ **Theorem 15.** *For any fixed constant ς, Algorithm 2 is an FPT algorithm for MR$(G, \mathbb{R})$ for any $G \in F_\varsigma$, when parameterized by OPT. The running time is $\Theta((2\varsigma OPT^{\varsigma})^{OPT+1} n^{\varsigma})$.*

Proof. FPTWrapper iteratively calls $F(G, \emptyset, k)$ for increasing values of k until it returns a non-Null value. First, we argue that while $k < OPT$, $F(G, \emptyset, k)$ will return Null. In the initial call to F, we have $S = \emptyset$. F then adds exactly one edge in each recursive call until $|S| = k$, at which point it returns Verifier(G, S). Thus, as $k < OPT$, by proposition 7, NULL is returned.

Now we argue that when $k = OPT$ an optimal solution is returned. Fix any optimal solution W and its support S_W to the given instance G. By Lemma 13, if $S \subsetneq S_W$ (which is true initially as $S = \emptyset$) then at least one edge in $S_W \setminus S$ is added to P. Thus, as F makes a recursive call to $F(G, S \cup \{e\}, k)$ for every edge $e \in P$, in at least one recursive call an edge of S_W is added to S. Thus there is some path in the tree of recursive calls to F in which all $k = OPT$ edges from S_W are added, at which point F returns Verifier(G, S), which returns an optimal solution by proposition 7. (Note this recursive call may not be reached, if a different optimal solution is found first.)

Now we consider bounding the running time. Observe that in each call to F, a set P is constructed, and then recursive calls to $F(G, S \cup \{e\}, k)$ are made for each $e \in P$. By Lemma 14, $|P| \leq 2\varsigma|S|^{\varsigma} \leq 2\varsigma k^{\varsigma}$ at all times. So in the tree of all recursive calls made by any initial call to $F(G, \emptyset, k)$, the branching factor is always bounded by $2\varsigma k^{\varsigma}$, and the depth is k. Thus there are $O((2\varsigma k^{\varsigma})^k)$ nodes in our recursion tree.

Now we bound the time needed for each node in the recursion tree. If Verifier is called then it takes $O(T_{\text{APSP}})$ time by proposition 7. Otherwise, note that there are $O(n^{\varsigma})$ chordless cycles. Thus it takes $O(\varsigma n^{\varsigma})$ time to enumerate and check them on line 4. Similarly $|\mathcal{C}| = O(n^{\varsigma})$ on line 7, and so the run time of each iteration of the for loop is $O(\varsigma n^{\varsigma})$. There are $O(|S|^{\varsigma}) = O(k^{\varsigma})$ iterations of the for loop, thus the total time per node is $O(\varsigma k^{\varsigma} n^{\varsigma})$.

Thus the total time for each call to $F(G, \emptyset, k)$ is $O((2\varsigma k^{\varsigma})^k \varsigma k^{\varsigma} n^{\varsigma}) = O((2\varsigma k^{\varsigma})^{k+1} n^{\varsigma})$. Since FPTWrapper calls $F(G, \emptyset, k)$ for $k = 1, \dots, OPT$, the overall running time is

$$O\left(\left(\sum_{k=1}^{OPT} (2\varsigma k^{\varsigma})^{k+1}\right) \cdot n^{\varsigma}\right) = O((2\varsigma OPT^{\varsigma})^{OPT+1} n^{\varsigma}). \qquad \blacktriangleleft$$

As lemma 13 holds for any optimal solution, the bound on the recursion tree size in the above proof actually bounds the number of optimal solutions.

▶ **Corollary 16.** *If $G \in F_\varsigma$ then there are at most $(2\varsigma OPT^{\varsigma})^{OPT}$ subsets $S \subset E$ such that S is the support of an optimal solution to MR$(G, \mathbb{R})$.*

▶ Remark 17. Using the approximation-preserving reduction from MR$(G, \mathbb{R}_{\geq 0})$ to MR$(G, \mathbb{R})$ in Theorem 8, the above also yields an FPT for MR$(G, \mathbb{R}_{\geq 0})$. This holds since the reduction does not change the optimal solution size, nor ς as it only adds triangles. Alternatively, the above algorithm can be carefully modified to consider light covering broken cycles.

6 Approximation Algorithms

In this section we present approximation algorithms for MR$(G, \mathbb{R}_{\geq 0})$ and MR$(G, \mathbb{R})$.

By Theorem 6, the support of an optimal solution to MR$(G, \mathbb{R})$ is a minimum cardinality regular cover of all broken cycles. This naturally defines a hitting set instance $(E, \mathcal{C})$, where the ground set E is the edges from G, and $\mathcal{C}$ is the collection of the subsets of edges determined by broken cycles. Unfortunately, constructing $(E, \mathcal{C})$ explicitly is infeasible as there may be an exponential number of broken cycles. In general just counting the number of paths in a graph is #P-Hard [16], though it is known how to count paths of length up to roughly $O(\log n)$ using color-coding. (See [1, 4] and references therein.) Moreover, observe our situation is more convoluted as we wish to count only paths corresponding to broken cycles.

Despite these challenges, we argue there is sufficient structure to at least roughly apply the standard greedy algorithms for hitting set. Our first key insight, related to insight (i), is:

(ii) One can always find *some* broken cycle, if one exists, by finding any edge whose weight is more than the shortest path length between its endpoints (using APSP).

Thus we have a polynomial time oracle, returning an arbitrary set in $\mathcal{C}$. Recall the greedy algorithm for hitting set, which repeatedly picks an arbitrary uncovered set, and adds all its elements to the solution. If $L = \max_{c\in\mathcal{C}} |c|$ is the largest set size, this gives an L-approximation, as each time we take the elements of a set, we get at least one element of the optimal solution. Below we apply this approach to approximate $\mathrm{MR}(G, \mathbb{R})$ and $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$.

We would prefer, however, to have an oracle for the number of broken cycles that an edge $e \in E$ participates in as using such an oracle would yield an $O(\log n)$-approximation algorithm for $\mathrm{MR}(G, \mathbb{R})$ (regardless of the size of L) by running the standard greedy algorithm for hitting set which repeatedly selects the element that hits the largest number of uncovered sets. Towards this end, we have the following key insight:

(iii) We can find the *most* broken cycle (i.e., with maximum deficit) and, more importantly, count how many such maximum deficit cycles each edge is in.

To argue that insight (iii) is true, first we observe that the cycle with the largest deficit value corresponds to a shortest path. This in turn, argued over several lemmas, allows us to quickly get a count when restricting to such cycles. Thus, if κ denotes the number of distinct cycle deficit values, the above insight implies an $O(\kappa \log n)$-approximation, by breaking the problem into κ hitting set instances, where for each instance we can run the greedy algorithm.

6.1 L-approximation

In this section, we consider the problems defined by restricting $\mathrm{MR}(G, \mathbb{R})$ and $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ to the subset of instances where the largest number of light edges in a broken cycle is L. We present an $(L+1)$-approximation algorithm for $\mathrm{MR}(G, \mathbb{R})$ which runs in $O(T_{\mathrm{APSP}} \cdot OPT)$ time, which also will imply an L-approximation for $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ with the same running time.

As mentioned above, the main idea comes from insight (ii). In particular, the following algorithm, Short Path Cover (SPC), can be easily understood by viewing it as running the standard L-approximation for the corresponding instance $(E, \mathcal{C})$ of hitting set, where we have an oracle for finding a set $c \in \mathcal{C}$. In the following, APSP is a subroutine returning a shortest path distance function $d(u, v)$, and a function $P(u, v)$ giving the set of edges along *any* shortest path from u to v.

Algorithm 3 Short Path Cover (SPC) for $\mathrm{MR}(G, \mathbb{R})$

```
1: function SPC($G = (V, E, w)$)
2:     $H = (V_H = V, E_H = E, w_H = w)$
3:     while True do
4:         $d, P = \mathsf{APSP}(H)$
5:         if $\exists\ e = (u, v) \in E_H$ such that $w(e) > d(u, v)$ then $E_H = E_H \setminus (P(u, v) \cup \{e\})$
6:         else return Verifier($G, E \setminus E_H$)
```

▶ **Theorem 18.** SPC *gives an* $(L+1)$*-approximation for* $MR(G, \mathbb{R})$ *in* $O(T_{\text{APSP}} \cdot OPT)$ *time.*

Proof. First, note that if there is a broken cycle in H, then for some edge $e = (u, v)$, $w(e) > d(u, v)$, and moreover, in this case $P(u, v) \cup \{e\}$ is a broken cycle. Thus, when the algorithm terminates there are no broken cycles in H. Also, for any broken cycle in G, if all of its edges are still in H, then it will be a broken cycle in H. Thus, when the algorithm terminates at least one edge from each broken cycle in G is in $E \setminus E_H$, which by Theorem 6 implies $E \setminus E_H$ is a valid support.

Note that removing edges does not create any new broken cycles, thus, any broken cycle in H is also a broken cycle in G. Thus, the support of any optimum solution must contain at least one edge from each broken cycle in H (again by Theorem 6), and so every time we remove the edges of a broken cycle $P(u, v) \cup \{e\}$, we remove at least one optimum edge. As the largest broken cycle length is $L + 1$, this implies overall we get an $(L + 1)$-approximation. The same argument implies the while loop can get executed at most OPT times, and as APSP takes $O(T_{\text{APSP}})$ time, and line 5 takes $O(m)$ time, we obtain the running time in the theorem statement. ◀

▶ Remark 19. If we modify SPC so that in line 5 we only remove $P(u, v)$ from E_H (rather than $P(u, v) \cup \{e\}$), then by the second part of Theorem 6, the same argument implies that SPC is an L-approximation for $\text{MR}(G, \mathbb{R}_{\geq 0})$ that runs in $O(T_{\text{APSP}} \cdot OPT)$ time.

▶ Remark 20. Theorem 11 restricts $\text{MR}(G, \mathbb{R}_{\geq 0})$ and $\text{MR}(G, \mathbb{R})$ to integer instances with max weight L, implying any broken cycle has $\leq L$ edges. As this is a subset of the instances here, SPC is an L or $L + 1$ approx for instances that are hard to approximate within $\Omega(\sqrt{L})$.

6.2 $O(\kappa \log n)$-approximation

Using insight (iii), our approach is to iteratively cover cycles by decreasing deficit value, ultimately breaking the problem into multiple hitting set instances. We present the algorithm for $\text{MR}(G, \mathbb{R})$ first and then remark on the minor change needed to apply it to $\text{MR}(G, \mathbb{R}_{\geq 0})$.

For any pair of vertices $s, t \in V$, let $d(s, t)$ denote their shortest path distance in G, and $\#\mathsf{sp}(s, t)$ denote the number of shortest paths from s to t. It is straightforward to show that $\#\mathsf{sp}(s, t)$ can be computed in $O(m + n)$ time given all $d(u, v)$ values have been precomputed.

▶ **Lemma 21** (Proof in Appendix B). *Let G be a positively weighted graph, where for all pairs of vertices u, v one has constant time access to the value $d(u, v)$. Then for any pair of vertices s, t, the value $\#\mathsf{sp}(s, t)$ can be computed in $O(m + n)$ time.*

Recall that for a broken cycle C with heavy edge h, the deficit of C is $\delta(C) = w(h) - \sum_{e \in (C \setminus h)} w(e)$. Moreover, $\delta(G)$ denotes the maximum deficit over all cycles in G. For any edge e, define $N_h(e, \alpha)$ to be the number of distinct broken cycles of deficit α whose heavy edge is e. Similarly, let $N_l(e, \alpha)$ denote the number of distinct broken cycles with deficit α which contain the edge e, but where e is not the heavy edge. While for general α it is not clear how to even approximate $N_l(e, \alpha)$ and $N_h(e, \alpha)$, we argue that when $\alpha = \delta(G)$ these values can be computed exactly.

▶ **Lemma 22.** *For any edge $e = (s, t)$, if $w(e) = d(s, t) + \delta(G)$ then $N_h(e, \delta(G)) = \#\mathsf{sp}(s, t)$, and otherwise $N_h(e, \delta(G)) = 0$.*

Proof. If $w(e) \neq d(s, t) + \delta(G)$, then as $\delta(G)$ is the maximum deficit over all cycles, it must be that $w(e) < d(s, t) + \delta(G)$, which in turn implies any broken cycle with heavy edge e has deficit strictly less than $\delta(G)$. Now suppose $w(e) = d(s, t) + \delta(G)$, and consider any path $p_{s,t}$

from s to t such that e together with $p_{s,t}$ creates a broken cycle with heavy edge e. If $p_{s,t}$ is a shortest path then $w(e) - w(p_{s,t}) = w(e) - d(s,t) = \delta(G)$, and otherwise $w(p_{s,t}) > d(s,t)$ and so $w(e) - w(p_{s,t}) < w(e) - d(s,t) = \delta(G)$. Thus $N_h(e, \delta(G)) = \#\mathsf{sp}(s,t)$ as claimed. ◀

As G is undirected, every edge $e \in E$ correspond to some unordered pair $\{a, b\}$. However, often we write $e = (a, b)$ as an ordered pair, according to some fixed arbitrary total ordering of all the vertices. We point this out to clarify the following statement.

▶ **Lemma 23.** *Fix any edge $e = (s,t)$, and let $X = \{f = (a,b) \mid w(f) = d(a,s) + w(e) + d(t,b) + \delta(G)\}$, and $Y = \{f = (a,b) \mid w(f) = d(b,s) + w(e) + d(t,a) + \delta(G)\}$. Then*

$$N_l(e, \delta(G)) = \left(\sum_{(a,b)\in X} \#\mathsf{sp}(a,s) \cdot \#\mathsf{sp}(t,b) \right) + \left(\sum_{(a,b)\in Y} \#\mathsf{sp}(b,s) \cdot \#\mathsf{sp}(t,a) \right).$$

Proof. Consider any broken cycle C containing $e = (s,t)$, with heavy edge $f = (a,b)$ and where $\delta(C) = \delta(G)$. Such a cycle must contain a shortest path between a and b, as otherwise it would imply $\delta(G) > \delta(C)$. Now if we order the vertices cyclically, then the subset of C's vertices $\{a, b, s, t\}$, must appear either in the order a, s, t, b or b, s, t, a. In the former case, as the cycle must use shortest paths, $w(f) = d(a,s) + w(e) + d(t,b) + \delta(G)$, and the number of cycles satisfying this is $\#\mathsf{sp}(a,s) \cdot \#\mathsf{sp}(t,b)$. In the latter case, $w(f) = d(b,s) + w(e) + d(t,a) + \delta(G)$, and the number of cycles satisfying this is $\#\mathsf{sp}(b,s) \cdot \#\mathsf{sp}(t,a)$. Note also that the set X from the lemma statement is the set of all $f = (a,b)$ satisfying the equation in the former direction, and Y is the set of all $f = (a,b)$ satisfying the equation in the later direction. Thus summing over each relevant heavy edge in X and Y, of the number of broken cycles of deficit $\delta(G)$ which involve that heavy edge and e, yields the total number of broken cycles with deficit $\delta(G)$ containing e as a light edge. ◀

▶ **Corollary 24** (Appendix B). *Given constant time access to $d(u,v)$ and $\#\mathsf{sp}(u,v)$ for any vertices u and v, $N_h(e, \delta(G))$ can be computed in $O(1)$ time and $N_l(e, \delta(G))$ in $O(m)$ time.*

■ **Algorithm 4** Finds a valid solution for MR$(G, \mathbb{R})$.

```
1:  function APPROX($G = (V, E, w)$)
2:      Let $S = \emptyset$
3:      while True do
4:          For every pair $s, t \in V$ compute $d(s,t)$
5:          Compute $\delta(G) = \max_{e=(s,t)\in E}\ w(e) - d(s,t)$
6:          if $\delta(G) = 0$ then return VERIFIER($G, S$)
7:          For every edge $(s,t) \in E$ compute $\#\mathsf{sp}(s,t)$
8:          For every $e \in E$ compute $count(e) = N_h(e, \delta(G)) + N_l(e, \delta(G))$
9:          Set $f = \arg\max_{e\in E} count(e)$
10:         Update $S = S \cup \{f\}$ and $G = G \setminus f$
```

▶ **Theorem 25.** *For any positive integer κ, consider the set of MR$(G, \mathbb{R})$ instances where the number of distict deficit values is at most κ, i.e., $|\{\delta(C) \mid C$ is a cycle in $G\}| \le \kappa$. Then Algorithm 4 gives an $O((T_{\mathrm{APSP}} + m^2) \cdot OPT \cdot \kappa \log n)$ time $O(\kappa \log n)$-approximation.*

Proof. Observe that the algorithm terminates only when $\delta(G) = 0$, i.e., only once there are no broken cycles left. As no new edges are added, and weights are never modified, this implies that when the algorithm terminates it outputs a valid regular cover S. (The algorithm must terminate as every round removes an edge.) Therefore, by Theorem 6, S is a valid $\mathrm{MR}(G, \mathbb{R})$ support, and so we only need to bound its size.

Let the edges in $S = \{s_1, \ldots, s_k\}$ be indexed in increasing order of the loop iteration in which they were selected. Let $G_1, \ldots, G_{k+1}$ be the corresponding sequence of graphs produced by the algorithm, where $G_i = G \setminus \{s_1, \ldots, s_{i-1}\}$. Note that for all i, $G_i = (V, E_i)$ induces a corresponding instance of hitting set, $(E_i, \mathcal{C}_i)$, where the ground set is the set of edges from the $\mathrm{MR}(G, \mathbb{R})$ instance G_i, and $\mathcal{C}_i = \{E_i(C) \mid C \text{ is a broken cycle in } G_i\}$ (where $E_i(C)$ is the set of edges in C).

Let $D = \{\delta(C) \mid C \text{ is a cycle in } G\}$, where by assumption $|D| \leq \kappa$. Note that any cycle C in any graph G_i, is also a cycle in G. Thus as we never modify edge weights, $\delta(G_1), \ldots, \delta(G_{k+1})$ is a non-increasing sequence. Moreover $X = \{\delta(G_i)\}_i \subseteq D$, and in particular $|X| \leq \kappa$. For a given value $\delta \in X$, let $G_\alpha, G_{\alpha+1}, \ldots, G_\beta$ be the subsequence of graphs with deficit δ (which is consecutive as the deficit values are non-increasing). Observe that for all $\alpha \leq i \leq \beta$, the edge s_i is an edge from a cycle with deficit $\delta = \delta(G_i)$. So for each $\alpha \leq i \leq \beta$, define a sub-instance of hitting set $(E'_i, \mathcal{C}'_i)$, where E'_i is the set of edges in cycles of deficit δ from G_i, and $\mathcal{C}'_i$ is the family of sets of edges from each cycle of deficit δ in G_i.

The claim is that for the hitting set instance $(E'_\alpha, \mathcal{C}'_\alpha)$, that $\{s_\alpha, \ldots, s_\beta\}$ is an $O(\log n)$ approximation to the optimal solution. To see this, observe that for any $\alpha \leq i \leq \beta$ in line 8, *count*(e) is the number of times e is contained in a broken cycle with deficit $\delta = \delta(G_i)$, as by definition $N_h(e, \delta(G_i))$ and $N_l(e, \delta(G_i))$ count the occurrences of e in such cycles as a heavy edge or light edge, respectively. Thus s_i is the edge in E'_i which hits the largest number of sets in $\mathcal{C}'_i$, and moreover, $(E'_{i+1}, \mathcal{C}'_{i+1})$ is the corresponding hitting set instance induced by removing s_i and the sets it hit from $(E'_i, \mathcal{C}'_i)$. Thus $\{s_\alpha, \ldots, s_\beta\}$ is the resulting output of running the standard greedy hitting set algorithm on $(E'_\alpha, \mathcal{C}'_\alpha)$ (that repeatedly removes the element hitting the largest number of sets), and it is well known this greedy algorithm produces an $O(\log n)$ approximation.

The bound on the size of S now easily follows. Specifically, let $I = \{i_1, i_2, \ldots, i_{|X|}\}$ be the collection of indices, where i_j was the first graph considered with deficit $\delta(G_{i_j})$. By the above, S is the union of the $O(\log n)$-approximations to the sequence of hitting set instance $(E'_{i_1}, \mathcal{C}'_{i_1}), \ldots, (E'_{i_{|X|}}, \mathcal{C}'_{i_{|X|}})$. In particular, note that for all i_j, $(E'_{i_j}, \mathcal{C}'_{i_j})$ is a hitting set instance induced from the removal of a subset of edges from the initial hitting set instance $(E_1, \mathcal{C}_1)$, and then further restricted to sets from cycles with a given deficit value. Thus the size of the optimal solution on each of these instances can only be smaller than on $(E_1, \mathcal{C}_1)$. This implies that the total size of the returned set S is $O(OPT \cdot |X| \log n) = O(OPT \cdot \kappa \log n)$.

As for the running time, first observe that by the above, there are $O(OPT \cdot \kappa \log n)$ while loop iterations. Next, the single call to Verifier in line 6 takes $O(T_{\mathrm{APSP}})$. For a given loop iteration, computing all pairwise distances in line 4 also takes $O(T_{\mathrm{APSP}})$ time. Computing the graph deficit in line 5 can then be done in $O(m)$ time. For any given vertex pair s, t, computing $\#\mathsf{sp}(s, t)$ takes $O(m + n)$ time by Lemma 21. Thus computing the number of shortest paths over all edges in line 7 takes $O(m^2 + mn)$ time. For each edge e, by Corollary 24, *count*$(e) = N_h(e, \delta(G)) + N_l(e, \delta(G))$ can be computed in $O(m)$ time, and thus computing all counts in line 8 takes $O(m^2)$ time. As the remaining steps can be computed in linear time, each while loop iteration in total takes $O(T_{\mathrm{APSP}} + mn + m^2) = O(T_{\mathrm{APSP}} + m^2)$ time, thus implying the running time bound over all iterations in the theorem statement. ◀

▶ Remark 26. If we modify line 8 to instead set *count*$(e) = N_l(e, \delta(G))$, by Theorem 6, we get the same result for $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$. If instead we used the reduction from $\mathrm{MR}(G, \mathbb{R}_{\geq 0})$ to $\mathrm{MR}(G, \mathbb{R})$ of Theorem 8, the graph size increases by a linear factor, giving a slower run time.

References

1 N. Alon and S. Gutner. Balanced families of perfect hash functions and their applications. *ACM Trans. Algorithms*, 6(3):54:1–54:12, 2010.

2 S. Baraty, D. Simovici, and C. Zara. The impact of triangular inequality violations on medoid-based clustering. In *Foundations of Intelligent Systems*, pages 280–289. Springer, 2011.

3 M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, June 2003.

4 C. Brand, H. Dell, and T. Husfeldt. Extensor-coding. In *Symposium on Theory of Computing (STOC)*, pages 151–164, 2018.

5 J. Brickell, I Dhillon, S. Sra, and J. Tropp. The metric nearness problem. *SIAM Journal on Matrix Analysis and Applications*, 30(1):375–396, 2008.

6 E. Candès and B. Recht. Exact matrix completion via convex optimization. *Commun. ACM*, 55(6):111–119, June 2012.

7 L. Sunil Chandran, Vadim V. Lozin, and C. R. Subramanian. Graphs of low chordality. *Discrete Mathematics and Theoretical Computer Science*, 7:25–36, 2005.

8 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.

9 C. Fan, B. Raichel, and G. Van Buskirk. Metric violation distance: Hardness and approximation. In *Symposium on Discrete Algorithms (SODA)*, pages 196–209, 2018.

10 A. Gilbert and L. Jain. If it ain't broke, don't fix it: Sparse metric repair. *ArXiv*, 2017.

11 S. Khot. On the power of unique 2-prover 1-round games. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 767–775, 2002.

12 E. Lee. Improved hardness for cut, interdiction, and firefighter problems. In *44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 92:1–92:14, 2017.

13 S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

14 A. Sidiropoulos, D. Wang, and Y. Wang. Metric embeddings with outliers. In *Proc. Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 670–689, 2017.

15 J. Tenenbaum, V. Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

16 L. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

17 F. Wang and J. Sun. Survey on distance metric learning and dimensionality reduction in data mining. *Data Mining and Knowledge Discovery*, 29(2):534–564, Mar 2015.

A Transitioning to Graph Metric Repair

A.1 The decrease only case

For the problem $\mathrm{MR}(G, \mathbb{R}_{\leq 0})$, consider the following simple algorithm, used in previous works for the special case when $G = K_n$.

Algorithm 5 Decrease Metric Repair (DMR)

```
1: function DMR($G = (V, E, w)$)
2:     Let $W = w$
3:     For any edge $e = uv \in E$, set $W(e)$ = weight of a shortest path between $u$ and $v$
4:     return $W - w$
```

Theorem 5. *The problem $MR(G, \mathbb{R}_{\leq 0})$ can be solved in $O(T_{\mathrm{APSP}})$ time by the* DMR *algorithm.*
Moreover, the problem becomes hard if even a single positive value is allowed. That is, if $0 \in \Omega$ and $\Omega \cap \mathbb{R}_{>0} \neq \emptyset$ then $MR(G, \Omega)$ is APX-Complete.

Proof. For the first part, let $e \in G$ be an edge whose edge weight is bigger than the shortest path between the two end points of e. Then in this case e is the heavy edge in a broken cycle. Hence, any decrease only solution must decrease this edge. Thus all edges decreased by DMR are edges that must be decreased.

By the same reasoning we see that this new weighted graph has no broken cycles. Thus, we see that our algorithm gives a sparsest solution to $\mathrm{MR}(G, \mathbb{R}_{\leq 0})$ in $\Theta(T_{\mathrm{APSP}})$ time.

For the second part, the reduction is the same as that of Fan et al. [9]. However, we make the observation that for any value $\alpha > 0$, by appropriately scaling the weights of the reduction in Fan et al. [9], $\mathrm{MR}(G, \mathbb{R}_{\leq 0})$ is still APX-Hard in the extreme case when $\Omega = \{0, \alpha\}$. ◀

▶ **Corollary 27.** *For any $G = (V, E, w)$* DMR *returns the smallest solution for any ℓ_p norm for $p \in [1, \infty)$.*

Proof. The proof of Theorem 5 actually shows that there is a unique support for the sparsest solution, i.e., the set of all heavy edges. In fact any decrease only solution must contain all of these edges in its support. We can also see that DMR decreases these by the minimum amount so that the cycles are unbroken. Thus, this solution is in fact the smallest for any ℓ_p norm. ◀

A.2 Structural results

Proposition 7. *The* VERIFIER *algorithm (Algorithm 1), given a weighted graph G and a potential support for a solution S, determines in $O(T_{\mathrm{APSP}})$ time whether there exists a valid (increase only or general) solution on that support and if one exists finds one.*

Proof. Let $G = (V, E, w)$ be the original graph and let M be the maximum edge weight from the graph G. The algorithm defines a new graph $\hat{G} = (V, E, \hat{w})$, with the following weights

$$\hat{w}(e) = \begin{cases} w(e) & e \notin S \\ M & e \in S \end{cases}$$

For each $e = (v_1, v_2) \in E$, line 4 sets $w(e)$ to be the weight of the shortest path in $\hat{G}$ from v_1 to v_2. Thus, at the end of the algorithm $w(e)$ satisfies the shortest path metric of $\hat{G}$. As the algorithm outputs w if and only if only edge weights in S are modified (increased), it suffices to argue S is a regular cover (light cover) if and only if only edge weights in S are modified (increased).

Assume that S is a regular or light cover. We argue line 4 only updates the weights of the edges in S. Note that $G \setminus S$ has no broken cycles. Thus, for any $e = (v_1, v_2) \in G \setminus S$ we have that the shortest path from v_1 to v_2 must be e. Now consider any path P from v_1 to v_2 in $\hat{G}$. If $P \cap S = \emptyset$, then $w(P) \geq w(e)$. On the other hand if $P \cap S \neq \emptyset$, then let $\tilde{e} \in P \cap S$. Then, we have that

$$w(P) \geq w(\tilde{e}) = M \geq w(e)$$

Thus, in either case, $w(P) \geq w(e)$. Hence for all $e \in G \setminus S$ we do not change its weight.

If S is a light cover, we also need to argue that the weights only increased. Let $e = (v_1, v_2) \in S$. Let P be a path of smallest weight in $\hat{G}$. Suppose $P \cap S \neq \emptyset$, then, we have that $w(P) \geq M \geq w(e)$. Thus, in this case we could not have decreased the weight. Thus, assume that $P \cap S = \emptyset$. If we still have that $w(P) \geq w(e)$, then we could not have decreased the weight. Thus, let us further assume that $w(P) < w(e)$. In this case, P along with e form a broken cycle in G, with e as the heavy edge. But then since S is a light cover, we have that $P \cap S \neq \emptyset$. Thus, we have a contradiction and this case cannot occur. Thus, if S is a light cover, then we only increase the edge weights.

Now assume S is not a regular cover (light cover). Then there exists a broken cycle C such that none of its (light) edges are in S. Thus, there is a broken cycle C in $\hat{G}$. Let e be the heavy edge of C, then on line 4 the weight of e will be decreased, and thus our algorithm will return NULL. ◀

The next theorem shows that once we know the support, the set of all possible solutions on that support is a nice space.

▶ **Theorem 28.** *For any weighted graph G and support S we have that the set of solutions with support S is a closed convex subset of $\mathbb{R}^{n\times n}$. Additionally, if $G - S$ is a connected graph or we require an upper bound on the weight of each edge, then the set of solutions is compact.*

Proof. Let x_{ij} for $1 \leq i, j \leq n$ be our coordinates. Then the equations $x_{ij} = c_{ij}$ for (i, j) not in the support and $x_{ij} \leq x_{ik} + x_{kj}$ define a closed convex set. Thus, we see the first part. For the second part we just need to see that set is bounded to get compactness. If we have that $G - S$ is connected then for all $e \in S$ there is a path between end points of e in $G - S$. Thus, the weight of this path is an upper bound. On the other hand 0 is always a lower bound. Thus, we get compactness if $G - S$ is connected. ◀

B Approximation Algorithms

Here we give the missing proofs from our $O(\kappa \log n)$-approximation algorithm.

Lemma 21. *Let G be a positively weighted graph, where for all pairs of vertices u, v one has constant time access to the value $d(u, v)$. Then for any pair of vertices s, t, the value $\#\mathsf{sp}(s, t)$ can be computed in $O(m + n)$ time.*

Proof. Let $V = \{v_1, v_2, v_3, ..., v_n\}$, and let $N(v_i)$ denote the set of neighbors of v_i. Define $X_i = \{v_j \in N(v_i) \mid w(v_i, v_j) + d(v_j, t) = d(v_i, t)\}$, that is, X_i is the set of neighbors of v_i

where there is a shortest path from t to v_i passing through that neighbor. Thus we have,

$$\#\mathsf{sp}(v_i, t) = \sum_{v_j \in X_i} \#\mathsf{sp}(v_j, t).$$

Note that any shortest path from v_i to t can only use vertices v_j which are closer to t than v_i. So consider a topological ordering of the vertices, where edges are conceptually oriented from smaller to larger $d(v_i, t)$ values. Thus if we compute the $\#\mathsf{sp}(v_i, t)$ values in increasing order of the index i, then each $\#\mathsf{sp}(v_i, t)$ value can be computed in time proportional to the degree of v_i, and so the overall running time is $O(m + n)$. ◀

Corollary 24. *Given constant time access to $d(u, v)$ and $\#\mathsf{sp}(u, v)$ for any pair of vertices u and v, $N_h(e, \delta(G))$ can be computed in $O(1)$ time and $N_l(e, \delta(G))$ in $O(m)$ time.*

Proof. By Lemma 22, in constant time we can check whether $w(e) = d(s, t) + \delta(G)$, in which case set $N_h(e, \delta(G)) = \#\mathsf{sp}(s, t)$, and otherwise set $N_h(e, \delta(G)) = 0$. By Lemma 23, we can compute $N_l(e, \delta(G))$ with a linear scan of the edges, where for each edge f in constant time we can compute whether $w(f) = d(a, s) + w(e) + d(t, b) + \delta(G)$ and if so add $\#\mathsf{sp}(a, s) \cdot \#\mathsf{sp}(t, b)$ to the sum over X, and if $w(f) = d(b, s) + w(e) + d(t, a) + \delta(G)$ add $\#\mathsf{sp}(b, s) \cdot \#\mathsf{sp}(t, a)$ to the sum over Y. ◀

C Improved Analysis for Complete Graphs

Here we consider the special case when $G = K_n$, improving parts of the analysis from [9, 10]. First, we consider the $O(OPT^{1/3})$-approximation algorithm of [9], which works for both $\mathrm{MR}(K_n, \mathbb{R})$ and $\mathrm{MR}(K_n, \mathbb{R}_{\geq 0})$. The running time of this algorithm is $\Theta(n^6)$, since at some point it enumerates all cycles of length ≤ 6. With a more careful analysis, we observe it suffices to consider cycles of length ≤ 5, improving the running time to $\Theta(n^5)$. For $\mathrm{MR}(K_n, \mathbb{R}_{\geq 0})$ we consider a simple, appealing algorithm with good empirical performance from [10], referred to as IOMR-FIXED. We prove that unfortunately it is an $\Omega(n)$ approximation.

C.1 5 Cycle Cover

Here we argue the running time of the $O(OPT^{1/3})$-approximation algorithm of [9], which works for both $\mathrm{MR}(K_n, \mathbb{R})$ and $\mathrm{MR}(K_n, \mathbb{R}_{\geq 0})$, can be improved from $\Theta(n^6)$ to $\Theta(n^5)$. The algorithm presented in [9] has 3 major steps. The first two steps are used to approximate the support of the optimal solution and then the last step is actually used to find a solution given this support. We shall focus on the first 2 steps as these are where we make modifications.

First Step: In the first step, [9] find a cover for all broken cycles of length $\leq m$. In particular, the authors use the case when $m = 6$. As described in [9], we can obtain an $m - 1$ approximation of the optimal cover for all broken cycles of length $\leq m$ in $O(n^m)$ time. Denote this cover by $S_{\leq m}$.

Second Step: For this step, we need to first define unit cycles. Given a broken cycle C with heavy edge h, let e be a chord of C. Then e divides C into 2 cycles, one that contains h, denoted heavy(C, e) and one that does not contain h denoted light(C, e). We say this cycle is a unit cycle if for all chords e, e is not the heavy edge of light(C, e).

From the definition of a unit cycle, a light cover of all unit cycles light covers all broken cycles. Hence, step 2 of the algorithm from [9] light covers all unit cycles not covered by $S_{\leq 6}$ as follows. Let C be such a unit cycle. Now we know that C has at least 7 edges. Consider the red C_4 shown in Figure C.1. We know that for each $e \in C_4$, we have that heavy(C, e) is

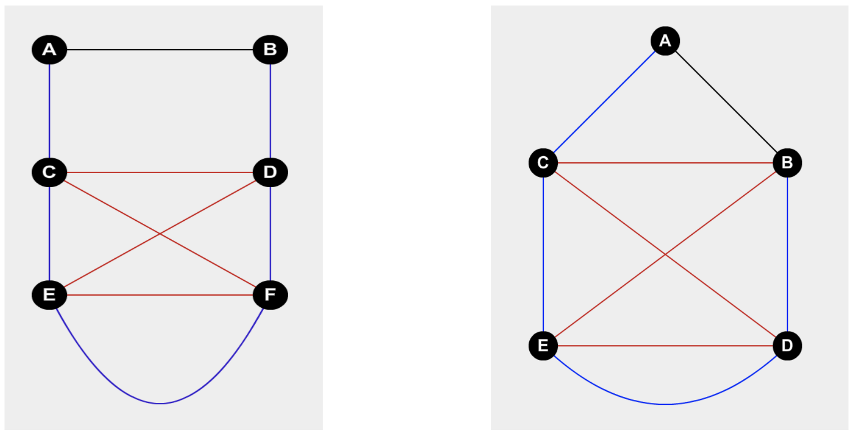


Figure C.1 Left: Embedding from [9]. Right: Our modified embedding for a smaller cycle. Here the black edge is the heavy edge. The blue edges are the light edges and the red edges are the embedded 4 cycle. The curved blue edge indicates that there are more vertices along that path

a broken cycle with at most 6 edges. Hence, we must have at least 1 edge in $S_{\leq 6}$. But since C has no light edges in $S_{\leq 6}$, we must have $e \in S_{\leq 6}$. Thus, we know all edges in C_4 are edges in $S_{\leq 6}$. Moreover, observe that either chord of C_4 is a light edge of C. Thus it suffices to compute a cover with least one chord of every four cycle from the edges in $S_{\leq 6}$, a step which the authors in [9] denote $chord4(S_{\leq 6})$.

In Figure C.1, we observe that the same 4 cycle can be embedded in a 6 cycle instead of a 7 cycle. Thus, our modified algorithm is shown in Algorithm 6.

Algorithm 6 5-Cycle Cover

```
1: function 5 CYCLE COVER($G = (V, E, w)$)
2:     Compute a regular cover of $S_{\leq 5}$ of all broken cycles with $\leq 5$ edges
3:     Compute a cover $S_c = chord4(S_{\leq 5})$
4:     return VERIFIER($G, S_c \cup S_{\leq 5}$)
```

C.2 IOMR-fixed

We will now show that IOMR-FIXED is an $\Omega(n)$ approximation algorithm. The algorithm presented in Gilbert and Jain [10] is as follows:

Algorithm 7 IOMR Fixed

```
Require: $D \in \mathrm{Sym}_n(\mathbb{R}_{\geq 0})$
1: function IOMR-FIXED(D)
2:     $\hat{D} = D$
3:     for $k \leftarrow 1$ to $n$ do
4:         for $i \leftarrow 1$ to $n$ do
5:             $\hat{D}_{ik} = \max(\hat{D}_{ik}, \max_{j<i}(\hat{D}_{ij} - \hat{D}_{jk}))$
6:     return $\hat{D} - D$
```

▶ **Lemma 29.** *For every n, there exists a weighted graph G such that* IOMR-FIXED *repairs $\binom{n-1}{2}$ edge weights while an optimal solutions repairs at most $(n-2)$ edge weights.*

Proof. Consider a matrix D where

$$D_{ij} = \begin{cases} 0 & \text{if } i \neq 1, j \neq 1 \\ 2^i & \text{if } j = 1, i > 1 \\ 2^j & \text{if } i = 1, j > 1 \end{cases}$$

This matrix D will be the weight matrix for the input graph K_n.

First, we claim that all entries of the form D_{s1} will never be updated as entries will only be updated the first time they are seen. Thus

$$D_{s1} = \max(D_{s1}, \max_{t<s}(D_{s1} - D_{1t})) = \max(2^s, \max_{t<s}(2^s - 2^t)) = 2^s$$

Now we just have to verify that the rest of the non-diagonal entries are updated. Let us look at the first time an entry D_{rs} is updated. (Here $r < s$.) Then we have that

$$\begin{aligned} \hat{D}_{rs} &= \max(D_{rs}, \max_{t<s}(D_{st} - D_{tr})) = \max_{t<s}(D_{st} - D_{tr}) && \text{[Since } D_{rs} = 0\text{]} \\ &\geq D_{s1} - D_{1r} = 2^s - s^r > D_{rs}. \end{aligned}$$

Thus all other non-diagonal entries will be updated the first time seen. Thus, for the solution $W = \hat{D} - D$ that IOMR-FIXED returns, we see that $W_{ij} > 0$ for exactly all $1 < i, j \leq n$ and $i \neq j$. Thus, we repaired $\binom{n-1}{2}$ edge weights.

Finally, a sparser increase only solution W can be obtained as follows. For all $s > 1$ we set

$$W_{1s} = W_{s1} = 2^n - D_{s1}$$

and all other entries of W are 0. This then gives us the desired result. ◀

▶ **Corollary 30.** IOMR-FIXED *is an* $\Omega(n)$ *approximation algorithm.*