


Approximation Algorithms for the Generalized Point-To-Point Problem

Zachary Friggstad 

Department of Computing Science, University of Alberta, Edmonton, Canada

Mohammad R. Salavatipour 

Department of Computing Science, University of Alberta, Edmonton, Canada

Hao Sun 

Department of Computer Science, University of Houston, TX, USA

Abstract

We consider the GENERALIZED POINT-TO-POINT (GP2P) problem in which we have an edge-weighted graph G with (possibly negative) node charges $\phi(v) \in \mathbb{Z}$. The goal is to find a minimum-cost set of edges such that each component has nonnegative total charge. Viewing the positive charges as specifying supply and negative charges as demand quantities at various nodes, the problem is equivalent to build the cheapest network so that it is possible to satisfy all demands by routing supplies across the network.

This problem is a significant generalization of other network design problems such as the well-studied STEINER FOREST problem. Even the special case of only having one single demand point (having charge $-k$ and all the other nodes having charge $+1$) is capturing the k -MINIMUM SPANNING TREE problem. Earlier work by Hajiaghayi et al. (2016) [10] gave an $O(\log n)$ approximation in pseudo-polynomial time with further improved guarantees if the total supply is not much larger than the total demand, and also a 2-approximation if the total supply equals the total demand.

Our contributions are four-fold: (a) we show how known k -MINIMUM SPANNING TREE approximations can be extended to GP2P approximations while losing only a ϵ -factor if the number of demand points in the instance is bounded by a constant, (b) we improve the running time to be Fixed-Parameter Tractable (FPT) in the number of demand points in constant-dimensional Euclidean metrics, (c) we give a 2-approximation in instances where edge costs are all 1 and $\phi(v) = \pm 1$ for each node v and show such instances are **APX**-hard, and (d) we show how the logarithmic approximations in earlier work can be modified to run in truly polynomial time.

2012 ACM Subject Classification Theory of computation \rightarrow Graph algorithms analysis; Theory of computation \rightarrow Approximation algorithms analysis

Keywords and phrases Point-to-Point Network design, Approximation, Steiner Forest, k -MST

Digital Object Identifier 10.4230/LIPIcs.WADS.2025.28

Funding Zachary Friggstad: Supported by NSERC.

Mohammad R. Salavatipour: Supported by NSERC.

1 Introduction

Consider the following setting in a network: some locations have a finite supply of goods and some locations have a given demand for goods. The goal is, naturally, to route supplies to the demand locations in order to satisfy all demands. For example, in the MINIMUM-COST TRANSPORTATION the goal is to do this in a way that minimizes the total travel cost of all goods. However, it is natural to consider variations of this problem. A network-design version would be to install a minimum-cost set of connections (edges between locations) so that it is possible to find a routing satisfying demands such that each unit of supply only travels along installed connections. That is, we only pay one for each link. Such a setting is captured by the following problem.



© Zachary Friggstad, Mohammad R. Salavatipour, and Hao Sun;
licensed under Creative Commons License CC-BY 4.0

19th International Symposium on Algorithms and Data Structures (WADS 2025).

Editors: Pat Morin and Eunjin Oh; Article No. 28; pp. 28:1–28:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

► **Definition 1.** In the *GENERALIZED POINT TO POINT CONNECTION* problem (GP2P), we are given a tuple $(G = (V, E), c, \phi)$ where G is an undirected graph with edge costs $c(u, v) \geq 0$, for all $e \in E$ and (possibly negative) node charges $\phi(v) \in \mathbb{Z}$, for all $v \in V$. The goal is to find a minimum-cost $F \subseteq E$ such that $\sum_{v \in C} \phi(v) \geq 0$ for each connected component $C \subseteq V$ of (V, F) .

A convenient view is that $\phi(v) < 0$ corresponds to a **demand point**, $\phi(v) > 0$ corresponds to a **supply point**, and $\phi(v) = 0$ is a location that is a **Steiner point**. For brevity, throughout this paper when the instance is clear from the context we let $\Phi = \sum_{v \in V} \max\{1, |\phi(v)|\}$. This way, Φ represents the total charge in absolute value but also ensures to count the zero-charge nodes.

It seems the first study of GP2P was actually for a directed variant of the problem and was conducted by Di Gaspero et al. [5] in the course of studying a practical problem related to scheduling shifts. The undirected version itself was formally proposed by Even, Kortsarz, and Slany [6] under the name INFINITE CAPACITY MINIMUM EDGE COST FLOW problem and an $O(\log \Phi)$ -approximation was given. The most recent work on GP2P is by Hajiaghayi et al. [10] where they give a 2-approximation when $\phi(V) = 0$ and an $O(\log(\min\{n, \phi(V) + 2\}))$ -approximation in general time that is polynomial in Φ . This is achieved by an adaptation of the primal-dual algorithm of [9] for the classic Steiner Forest problem and its generalization (e.g. when we have a proper function). As pointed out in [6], many special cases of GP2P were already well-studied. For example:

- **STEINER FOREST:** We are given terminal pairs $(s_1, t_1), \dots, (s_k, t_k)$ in an edge-weighted graph and the goal is to buy the cheapest set of edges so each pair has both endpoints in the same component. This is modelled by GP2P by letting $\phi(s_i) = 2^i$ and $\phi(t_i) = -2^i$ (and $\phi(v) = 0$ for all non-terminals). Goemans and Williamson give a 2-approximation for STEINER FOREST [9].
- **k-MINIMUM SPANNING TREE (k-MST):** We are given a particular *root* node $r \in V$ and an integer $k \geq 0$. The goal is to find the cheapest tree including r and at least k other nodes. This is modelled by GP2P by letting $\phi(r) = -k$ and $\phi(v) = 1$ for all other $v \in V - \{r\}$. Note, this instance of GP2P has only a single demand point. The history of approximating k-MST is storied, currently the best approximation is a 2-approximation [8, 4].
- **KNAPSACK COVERING:** We are given n items with sizes s_1, \dots, s_n and costs c_1, \dots, c_n plus a demand value D . The goal is to find a minimum-cost set of items with total size at least D . This modelled by GP2P by using a star with center r and leaf nodes v_1, \dots, v_n . The cost of rv_i is c_i and the charges are $\phi(r) = -D$ and $\phi(v_i) = s_i$. An FPTAS for KNAPSACK COVERING is known, e.g. by slightly modifying the standard dynamic programming based FPTAS for standard KNAPSACK.

Notably, all these cases have constant-factor approximations or better. An open problem is to determine if GP2P has a constant-factor approximation even if we allow pseudo-polynomial running time, i.e. running time that is polynomial in Φ ; note for the purposes of getting an $O(1)$ -approximation we can assume without loss of generality that the edge costs are bounded by a polynomial in n (see Lemma 19 in Appendix A). Even special cases of GP2P that generalize some of the classic problems stated above are open. For instance, for generalization of k -MST to the situation where we have more than one root node, say r_1, r_2, \dots, r_d , and the goal is to find a minimum cost subgraph where the connected component of each r_i has at least k nodes, it is not known if one can get an $O(1)$ -approximation or not. This is one of the cases we study in this paper.

1.1 Our Results

For brevity, we say an instance (G, c, ϕ) of GP2P is **metric** (METRIC-GP2P) if G is a complete graph and edge costs satisfy the triangle inequality $c(u, v) \leq c(u, w) + c(w, v)$ for any distinct $u, v, w \in V$. As is usual with single-connectivity network design problems, e.g. STEINER FOREST, assuming the graph is metric can be done without loss of generality. The standard proof is found in Lemma 18 in Appendix A. We also say an instance (G, c, ϕ) of GP2P is **graphical** (GRAPHICAL-GP2P) if all edge costs are 1 but G is not necessarily complete.

Our results need approximation algorithms for slight generalizations of k-MST.

► **Definition 2.** In **k-MST with required nodes** (k -MST-R), instead of a single root node $r \in V$ we are given a subset $R \subseteq V$. The goal is to find the cheapest tree spanning R and at least k other nodes in $V - R$. In **weighted k-MST-R** (W - k -MST-R), each vertex $v \in V - R$ is given an integer weight $w_v \geq 0$ and the goal is to find the cheapest tree spanning R plus a subset of nodes in $V - R$ with total weight at least k .

We note all k-MST approximations we consider in this paper can be readily adapted to the generalization k-MST-R. One can also extend them to W - k -MST-R that would run in pseudo-polynomial time (the simple reduction can be found in Appendix A).

► **Observation 3.** If there is a polynomial-time α -approximation for k -MST-R then there is an α -approximation for W - k -MST-R whose running time is polynomial in the number of nodes and the total weight of all nodes.

Let n_d be the number of demand points in a given instance, i.e. $|\{v \in V : \phi(v) < 0\}|$. Recall from above that k-MST is a special case of GP2P when there is only one demand point, i.e. $n_d = 1$. We consider the case of GP2P with $n_d \geq 1$. This is akin to generalizing k-MST to multiple roots, each with its own size requirement, but there is one key difference. One might imagine a multiple-root k-MST generalization would want to keep the trees disjoint but in GP2P we can have multiple demand points in a single component as long as the total supply in the component is at least the total demand in the component.

Our main result shows how an α -approximation for W - k -MST-R can be used to give a $(1 + \epsilon) \cdot \alpha$ -approximation for GP2P.

► **Theorem 4.** For any $\epsilon > 0$, given an α -approximation algorithm \mathcal{A} for W - k -MST-R, there is a $(1 + \epsilon)\alpha$ -approximation for GP2P. This algorithm makes $n^{O(n_d/\epsilon)}$ calls to \mathcal{A} . For D -dimensional Euclidean metrics the running time of the GP2P approximation can be improved to $O((16 \cdot n_d/\epsilon)^{D \cdot n_d})$ calls to \mathcal{A} .

We also show how existing algorithms for k-MST (the $(2 + \epsilon)$ -approximation for k-MST in [3] for general metrics and the PTAS for k-MST in constant-dimensional Euclidean metrics [2, 11]) can be suitably adapted to given the same approximation guarantees for W - k -MST-R, which combined with the above theorem imply the following:

► **Corollary 5.** For any $\epsilon > 0$, there is a $(2 + \epsilon)$ -approximation for GP2P that runs in time is $n^{O(n_d/\epsilon)}$. For D -dimensional Euclidean metrics there is a $(1 + \epsilon)$ -approximation for GP2P with running time $O((16 \cdot n_d/\epsilon)^{D \cdot n_d} \cdot \Phi^{O(D/\epsilon)})$

Note that the $(2 + \epsilon)$ -approximation for W - k -MST-R and for GP2P runs in truly polynomial time for any fixed n_d but the $(1 + \epsilon)$ -approximation for the Euclidean metrics is polynomial in Φ , hence only pseudo-polynomial, as it is not clear how to get a PTAS for W - k -MST-R in the Euclidean metrics.

We also mention that this would also extend to doubling metrics using the quasi-polynomial time $(1 + \epsilon)$ -approximation for k -MST in metrics with constant doubling dimension by Talwar [12]. The running time would be quasipolynomial in Φ but still FPT in n_d .

Our next collection of results is for GRAPHICAL-GP2P. First, we observe that particular restrictions of GP2P remain essentially as hard to approximate as the general problem.

► **Observation 6.** *If there is a polynomial-time α -approximation for instances of GP2P with $\phi(v) \in \{-1, +1\}$ for each $v \in V$, then there is a 2α -approximation for general instances of GP2P with running time being polynomial in Φ .*

► **Observation 7.** *If there is a polynomial-time α -approximation for instances of GRAPHICAL-GP2P with $\phi(v) \in \{-1, 0, +1\}$ for each $v \in V$, then for any $\epsilon > 0$ there is a $(1 + \epsilon) \cdot \alpha$ -approximation for general instances of GP2P with running time being polynomial in $1/\epsilon$ and Φ .*

These simple observations are proven in Appendix B. Our main contribution here is that the intersection of these two restrictions of GP2P is still **APX**-hard but does at least admit a simple approximation.

► **Theorem 8.** *The restriction of GRAPHICAL-GP2P with $\phi(v) \in \{-1, +1\}$ for each $v \in V$ is **APX**-hard and admits a polynomial-time 2-approximation.*

Finally, we improve the running time of the logarithmic approximations in [10] by making them run in truly polynomial time (i.e. removing the dependence on Φ).

► **Theorem 9.** *There is an $O(\log(\min\{n, \phi(V) + 2\}))$ -approximation for GP2P running in polynomial time.*

1.2 Notation

For a graph $G = (V, E)$ and a subset of edges $F \subseteq E$, we let $V(F)$ denote the set of all nodes in V that appear as the endpoint of at least one edge in F and say that F **spans** $V(F)$. We refer to a component of G as a subset of vertices corresponding to a connected component of G . For a given instance $(G = (V, E), c, \phi)$ of GP2P, let $V_s = \{v \in V : \phi(v) > 0\}$ be the supply points, $V_d = \{v \in V : \phi(v) < 0\}$ be the demand points, and $V_0 = \{v \in V : \phi(v) = 0\}$ be the Steiner points. Correspondingly, let n_s, n_d , and n_0 denote their sizes with $n := |V| = n_s + n_d + n_0$. We also use the convention $c(F) := \sum_{e \in F} c(e)$ for $F \subseteq E$ and, similarly, $\phi(C) := \sum_{v \in C} \phi(v)$ for $C \subseteq V$. Note $\phi(V)$ differs from $\Phi := \sum_{v \in V} \max\{1, |\phi(v)|\}$, the former measures the excess of positive charge in the input and the latter measures the absolute value of all charges while ensuring Steiner points are still counted.

1.3 Organization

The algorithms proving Theorem 4 appear in Section 2. Theorem 8 is proven in Section 3 and Section 4 then concludes the paper with the proof of Theorem 9. The proofs of some supporting results as well as Observations 6 and 7 appear in the appendix.

2 Approximations for Constant n_d

The algorithms proving Theorem 4 are obtained by reducing to W- k -MST-R. Recall that in W- k -MST-R, we are given a set of required nodes R along with (integer) node weights $w_v \geq 0$ and the goal is to find a minimum cost tree spanning R such that the total weight of the nodes in the tree is at least a given integer k . The special case of $|R| = 1$ and $w_v = 1$

for all $v \neq r$ is the traditional k -MST problem. Despite its generality, W - k -MST- R can be approximated as well (or nearly as well) as the simpler k -MST problem by adapting the known algorithms.

► **Theorem 10.** *The k -MST algorithm described [3] can be adapted to give a polynomial-time $(2 + \epsilon)$ -approximation for W - k -MST- R for any constant $\epsilon > 0$. Similarly, the PTAS for k -MST in D -dimensional Euclidean metrics described in [11] with running time $O(n^{O(D/\epsilon)})$ can be adapted to give a PTAS for k -MST- R with the same asymptotic running time and a $(1 + \epsilon)$ -approximation for W - k -MST- R with running time $O((n\Phi)^{O(D/\epsilon)})$.*

These adaptations are discussed in Appendix C. We emphasize both algorithms run in polynomial time (for constant ϵ) even if the number of required nodes R is not bounded by a constant. It is possible that the 2-approximations for k -MST in [8] or [4] could be extended to W - k -MST- R and that the PTAS for Euclidean k -MST in [2] could be similarly extended. We focus on these particular algorithms due to the ease in describing how to extend them.

We present the proof of Theorem 4 in two parts. First for the general metrics and then the improved FPT time for D -dimensional Euclidean metrics.

2.1 Theorem 4: General metrics

Throughout, let OPT denote the cost of an optimum solution. Fix $\epsilon > 0$ to be a sufficiently small constant, $\epsilon < 1/5$ suffices. From Lemma 18 (Appendix A), we may assume G is a complete graph with edge costs satisfying the triangle inequality. For any $r \geq 0$ and any $v \in V$ we let $B(v, r) = \{u \in V : c(u, v) \leq r\}$ be the **ball** of radius r around v .

The intuition for our algorithm is the following. If the optimum solution consisted of only a single tree containing all nodes with negative charge, we could treat it as a W - k -MST- R instance where $R = V_d$ and $k = -\phi(R)$ while setting $w_v = 0$ for $v \in R$ and $w_v = \phi(v)$ for $v \notin R$. However, this may not be the case. To cope, we show how to decompose the instance into disjoint subproblems that have single-tree optimal solutions whose total costs are close to OPT .

To perform the decomposition, we first show that a near-optimum solution to the original instance (G, c, ϕ) exists such that any two trees in this solution notably far apart. Then we guess a small “net” of points in each tree, the union of balls with a particular common radius around each net point covers all trees, but two balls from different trees are disjoint. This allows us to partition the instance into disjoint instances, one per tree in the near optimum solution.

The first step is to show some near-optimum solution has its components being bounded away from each other.

► **Lemma 11.** *There is a solution $F' \subseteq E$ with $c(F') \leq (1 + \epsilon) \cdot OPT$ such that for any two components C_1, C_2 of (V, F') with $C_1 \cap V_d \neq \emptyset$ and $C_2 \cap V_d \neq \emptyset$ we have $c(u, v) \geq \frac{\epsilon}{n_d} \cdot OPT$ for any $u \in C_1, v \in C_2$.*

Proof. Let F' initially be an optimum solution. While there are two components C_1, C_2 of (V, F') with $C_1 \cap V_d \neq \emptyset$ and $C_2 \cap V_d \neq \emptyset$ such that $c(u, v) < \frac{\epsilon}{n_d} \cdot OPT$ for some $u \in C_1$ and $v \in C_2$, add uv to F' . Each such addition increases the cost of F' by at most $\frac{\epsilon}{n_d} \cdot OPT$ and this procedure will be executed fewer than n_d times since there are initially at most n_d components containing a node in V_d . ◀

Of course, we may also assume, by discarding edges, that F' is a minimal solution meaning $F' - \{e\}$ is not feasible for any $e \in F'$. So F' consists of $m \leq n_d$ vertex-disjoint trees, say $T_1, \dots, T_m \subseteq F'$ that collectively span all of V_d plus some other nodes in $V_0 \cup V_s$. Any other node v not in a tree has $\phi(v) \geq 0$ and forms an isolated component of (V, F') .

Next, we identify a *net* of nodes in the trees T_1, \dots, T_m that will be small enough for our algorithm to guess.

► **Lemma 12.** *For any tree T and any $r > 0$ there is a set $N \subseteq V(T)$ with $|N| \leq 1 + c(T)/r$ such that $c(v, N) \leq r$ for each $v \in V(T)$.*

Similar constructions have been considered many times before, one example is in the $(2 + \epsilon)$ -approximation for k-MST [3]. We include a proof for completeness.

Proof. We prove this by induction on $|V(T)|$. Root T at an arbitrary node w . The base case is when $c(w, v) \leq r$ for each $v \in V(T)$. If so, we simply let $N = \{w\}$.

Otherwise, for two $u, v \in V(T)$ let $c_T(u, v)$ be the cost of the unique $u - v$ path in T . Pick any $v \in V(T)$ that has maximum value $c_T(w, v)$. Note $c_T(w, v) \geq c(w, v) > r$. Now let u be the furthest node along the $v - w$ path such that $c_T(v, u) \leq r$. Since $c(v, u) \leq c_T(v, u) \leq r$, then $u \neq w$. Let $p(u)$ be the parent of u in T . By our choice of u we also have $c_T(v, p(u)) > r$.

Let T_u denote the subtree of T rooted at u . For any $v' \in V(T_u)$ we have $c(v', u) \leq c_T(v', u) \leq c_T(v, u) \leq r$. Let T' be the tree obtained by removing the subtree rooted at u from T . This removes all edges of the $v - p(u)$ path, so $c(T') \leq c(T) - r$. By induction, we can find a set $N' \subseteq V(T')$ with $|N'| \leq 1 + c(T')/r \leq c(T)/r$ such that $c(w', N) \leq r$ for each $w' \in V(N')$.

Finally, set $N = \{u\} \cup N'$ and note $|N| = 1 + |N'| \leq 1 + c(T)/r$. Every $u' \in V(T)$ is now within distance r from some node in N , as required. ◀

The entire algorithm is summarized in Algorithm 1. The high-level idea is that it guesses a value ν very close to OPT and then guesses the “nets” N_1, \dots, N_m from Lemma 12 applied to each tree of the near-optimum solution F' using $r = \frac{\epsilon}{4 \cdot n_d} \cdot \nu$. Below, we show for the proper guess of N_1, \dots, N_m that the sets V_i obtained by the union of balls $B(v, r)$ for $v \in V_i$ are disjoint. This instance of GP2P then naturally decomposes into disjoint instances of W-k-MST-R. Supporting results demonstrating the performance of our algorithm are found below.

■ **Algorithm 1** GP2P approximation.

```

1: if  $F^0 = \{e \in E : c(e) = 0\}$  is feasible then
2:   return  $F^0$ 
3: for each integer  $b$  such that  $\nu := (1 + \epsilon)^b \in [\min_e c(e), \sum_e c(e)]$  do
4:   for each  $1 \leq m \leq n_d$  and each  $m$ -tuple  $N_1, \dots, N_m \subseteq V$  with  $\sum_{i=1}^m |N_i| \leq 5 \cdot n_d / \epsilon$ 
     do
5:     let  $r := \frac{\epsilon}{4 \cdot n_d} \cdot \nu$  and  $V_i := \cup_{v \in N_i} B(v, r)$  for each  $1 \leq i \leq m$ 
6:     if  $V_i \cap V_j \neq \emptyset$  for distinct  $i, j$  or if  $V_d \not\subseteq \cup_{i=1}^m V_i$  then
7:       continue to the next iteration
8:     for each  $1 \leq i \leq m$  do
9:       construct the W-k-MST-R instance on the subgraph  $G[V_i]$  with  $R = V_d \cap V_i$ 
        and  $k = -\phi(R)$ 
10:      let  $T'_i$  be the tree obtained by running an  $\alpha$ -approximation on this instance
11:      Record  $\cup_{i=1}^m T'_i$  as a candidate solution.
12: return the cheapest candidate solution found over all iterations.
```

We first discuss the running time. The number of iterations of the outer loop is logarithmic in the ratio $c(E) / \min_e c(e)$, which is polynomial in the number of bits used to represent the costs in the instance. There are clearly only n_d possible values for m and the number of

m -tuples satisfying the stated bounds is at most $n^{O(5 \cdot n_d/\epsilon)}$. So when n_d is regarded as a constant, the total number of iterations is polynomial in the input size and, thus, the entire algorithm makes a polynomial number of calls to a k-MST-R approximation on instances with at most n nodes and, otherwise, runs in polynomial time.

Towards the performance guarantee, we show for the “correct” guess of values in the loops the algorithm will perform well.

► **Lemma 13.** *Let $\nu \in [OPT, (1 + \epsilon) \cdot OPT]$ and T_1, \dots, T_m be the trees in the near-optimum solution F' from Lemma 11. For each $1 \leq i \leq m$, let N_i be the set identified by Lemma 12 when applied to T_i using $r = \frac{\epsilon}{4 \cdot n_d} \cdot \nu$. Also let $V_i = \cup_{v \in N_i} B(v, r)$ for each $1 \leq i \leq m$.*

Then (a) $\sum_{i=1}^m |N_i| \leq 5 \cdot n_d/\epsilon$, (b) $V_i \cap V_j = \emptyset$ for distinct i, j , and (c) $V_d \subseteq \cup_{i=1}^m V_i$.

Proof. For (a), we have

$$\sum_i |N_i| \leq m + \frac{c(F')}{r} \leq n_d + \frac{(1 + \epsilon) \cdot OPT}{OPT} \cdot \frac{4n_d}{\epsilon} \leq \frac{5}{\epsilon} \cdot n_d.$$

For (b), if, say, $w \in V_i \cap V_j$ for some distinct i, j then there would be some $u \in N_i$ and $v \in N_j$ such that

$$c(u, v) \leq c(u, w) + c(w, v) \leq r + r = \frac{\epsilon}{2 \cdot n_d} \cdot \nu \leq \frac{\epsilon(1 + \epsilon)}{n_d} \cdot OPT \leq \frac{\epsilon}{n_d} \cdot OPT.$$

But this contradicts Lemma 11, which showed $c(u, v) > \frac{\epsilon}{n_d} \cdot OPT$. Finally, (c) follows because the balls $B(v, r)$ for $v \in N_i$ collectively cover all of $V(T_i)$ and each node of V_d lies on some T_i . ◀

To finish the analysis, consider the iteration of the algorithm for the particular setting of ν and T_1, \dots, T_m described in Lemma 13. With these values, the algorithm proceeds to run the k-MST-R approximations. In the instance corresponding to V_i , we know T_i itself is a feasible solution so the returned tree T'_i satisfies $c(T'_i) \leq \alpha \cdot c(T_i)$. Thus, the candidate GP2P solution found in this iteration has cost $\sum_{i=1}^m c(T'_i) \leq \alpha \cdot c(F') \leq \alpha \cdot (1 + \epsilon) \cdot OPT$.

2.2 Theorem 4: Euclidean metrics

We simply describe how to modify Algorithm 1. Clearly, we can use a $(1 + \epsilon)$ -approximation for W-k-MST-R in Euclidean metrics to make Algorithm 1 a $(1 + \epsilon)$ -approximation in Euclidean metrics. The pseudo-polynomial running time in the statement of Theorem 4 comes from the fact that we only know how to adapt k-MST PTASes to k-MST-R and then rely on Observation 3 to get a pseudo-polynomial time W-k-MST-R $(1 + \epsilon)$ -approximation. One small comment is that even though the distances are not necessarily rational numbers, the number of iterations of the outer loop is still polynomial in the number of bits used to describe the locations of the points in V .

To improve the running time to be FPT in n_d , we change how the nets are guessed. Let D be the dimension of the metric, recall that D is assumed to be a constant. For brevity, let $\delta := \epsilon/(16n_d)$. Note $4\delta \cdot \nu$ is the value r from Algorithm 1. The idea of the improvement is the following. For simplicity let's consider the case of $D = 2$. If one considers a square of side length L , the number of disjoint balls of radius $\epsilon \cdot L/n_d$ that can be placed inside that square is $O((n_d/\epsilon)^2)$. This simple packing argument can be used to bound the number of guessed points for the nets N to be bounded by $O((n_d/\epsilon)^D)$.

For each guess ν in the outer loop, we first let N be any $\delta \cdot \nu$ -net of V . That is, every $v \in V$ has $c(v, N) \leq \delta \cdot \nu$ yet $c(u, v) > \delta \cdot \nu$ for any $u, v \in N$. Such a set N can be constructed by greedily adding points while maintaining the property that $c(u, v) > \delta \cdot \nu$ until no more points can be added. For each $v \in V$, let $\tau(v)$ be its closest point in N . So $\tau(v) = v$ for $v \in N$ and, otherwise, we at least know $c(v, \tau(v)) \leq \delta \cdot \nu$.

Now let N_1, \dots, N_m be the sets identified by applying Lemma 12 using $r = \delta \cdot \nu$ and let $N'_i = \{\tau(v) : v \in N_i\}$. For $u \in N'_i$ and $v \in N'_j$ for $i \neq j$ we still have that $B(u, \delta) \cap B(v, \delta) = \emptyset$. That is, suppose otherwise and let $u' \in N_i$ be such that $\tau(u') = u$ and $v' \in N_j$ be such that $\tau(v') = v$. Then when $\nu \in [OPT, (1 + \epsilon) \cdot OPT]$ we have

$$c(u', v') \leq c(u, u') + c(u', v') + c(v', v) \leq \delta \cdot \nu + 2\delta \cdot \nu + \delta \cdot \nu \leq \frac{\epsilon}{4n_d} \cdot \tau \leq \frac{\epsilon}{n_d} \cdot OPT$$

which contradicts Lemma 11 and the fact that u' and v' lie in different trees in F' .

So it suffices to guess N'_1, \dots, N'_m in the algorithm. But now we leverage packing property of Euclidean metrics to help reduce the number of guesses to a constant depending on D, n_d and ϵ .

► **Lemma 14.** *For each $v \in V$, $|B(v, \nu) \cap N|$ is bounded by $O((4/\delta)^D)$.*

Proof. The Euclidean balls of radius $\delta/2 \cdot \nu$ about points in $B(v, \nu) \cap N$ are disjoint by construction of N and are completely contained in the Euclidean ball of radius $(1 + \delta/2) \cdot \nu \leq 2 \cdot \nu$ about v . The volume a D -dimensional ball with radius r is within an absolute constant factor of $f(r) := \frac{1}{D} \cdot \left(\frac{2\pi e}{D}\right)^{D/2} \cdot r^D$. Therefore, $|B(v, \nu) \cap N|$ is at most a constant factor times $f(2\nu)/f(\delta/2 \cdot \nu) = (4/\delta)^D$. ◀

The steps for guessing N'_1, \dots, N'_m are to first try all ways to partition V_d into m nonempty groups, a coarse upper bound on the number of such choices is $n_d^{n_d}$. For each such partition, let $v_1, \dots, v_m \in V_d$ be any particular representatives from the m parts. We try all tuples N'_1, \dots, N'_m where each $N'_i \subseteq B(v_i, \nu) \cap N$ such that $\sum_i |N'_i| \leq 17/\epsilon \cdot n_d$ (as opposed to $5/\epsilon \cdot n_d$ as in Lemma 13 since the radius δ is smaller). For each such tuple that passes the other requirements of Lemma 13, we partition the instance into disjoint Euclidean k-MST-R instances and approximate these with a PTAS. A coarse upper bound on the number of such tuples N'_1, \dots, N'_m is $O((4/\delta)^{D \cdot n_d})$.

3 Approximation Algorithms and Hardness for Graphical-GP2P with ± 1 Charges

Observations 6 and 7 show GP2P is not much easier to approximate if we assume either that $\phi(v) \in \{-1, +1\}$ for each $v \in V$ or that the instance is graphical and $\phi(v) \in \{-1, 0, +1\}$ for each $v \in V$. We begin this section by showing the common intersection of these two special cases does admit a 2-approximation. After this, we complete the proof of Theorem 8 by showing such GP2P instances remain **APX**-hard.

3.1 Graphical Instances with Unit Charges

Let $(G = (V, E), c, \phi)$ be an instance of GRAPHICAL-GP2P where $\phi(v) \in \{-1, +1\}$ for each $v \in V$. As usual, let OPT denote the optimum solution cost which, in this case, is just measuring the minimum size feasible solution $F \subseteq E$.

► **Theorem 15.** *Let $F \subseteq E$ be any minimal feasible solution (i.e. $F - \{e\}$ is not feasible for any $e \in F$). Then $|F| \leq 2 \cdot OPT$.*

A 2-approximation is then straightforward as one could simply start with F being any spanning tree and then iteratively try to drop an edge from F while retaining feasibility until no such drop is possible.

Proof. Recall $V_d = \{v \in V : \phi(v) = -1\}$ and $n_d = |V_d|$. So $V_s = V - V_d$ as no charge is zero in this case. We claim $n_d \leq OPT$ and that any minimal solution has size at most $2 \cdot n_d$. Thus, any minimal solution F satisfies $|F| \leq 2 \cdot n_d \leq 2 \cdot OPT$, as required.

For the first bound, let F^* be an optimum solution and C any connected component in (V, F^*) that contains at least one node in V_d . If C has, say, $n_d^C \geq 1$ nodes in V_d then C must contain at least n_d^C nodes in V_s as well. That is, $|C| \geq 2n_d^C$ so F^* contains at least $2n_d^C - 1 \geq n_d^C$ edges in component C . Summing over all components that contain at least one node in V_d , we see $|F^*| \geq n_d$.

Now let F be any minimal feasible solution. Let C be any connected component of (G, F) and let F_C be the edges of F in component C . If $C \cap V_d = \emptyset$ then minimality of F means C is a single node v with $\phi(v) = +1$ (i.e. it has no edges). If $C \cap V_d \neq \emptyset$, we claim $\phi(C) = 0$. If so, then $|C \cap V_d| = |C \cap V_s|$ so $|F_C| < 2 \cdot |C \cap V_d|$ as F_C is a tree (by minimality). Summing over all components would complete the claim that $|F| < 2 \cdot n_d$.

To see $\phi(C) = 0$, again recall F_C is a tree. Further, for each $e \in F_C$ we have that deleting e would produce a component with negative charge but it could not be that both components have negative charge since $\phi(C) \geq 0$. Consider the orientation of edges that directs each edge e toward the side that would have negative charge if e was deleted. Since (C, F_C) is a tree, the orientation of edges produces a directed acyclic graph. Let r be any source node in this DAG.

View the tree (C, F_C) as being rooted at r and say r has m children. Let $C_1, \dots, C_m \subseteq C$ be such that C_i are the nodes in the subtree under the i 'th child of r . Since $C \cap V_d \neq \emptyset$ and $\phi(C) \geq 0$, we know C has at least two nodes so $m \geq 1$.

By the orientation of edges, we have $\phi(C_i) \leq -1$ for each $1 \leq i \leq m$. Therefore

$$\phi(C) = \phi(r) + \sum_{i=1}^m \phi(C_i) \leq \phi(r) - m \leq \phi(r) - 1 \leq 0$$

which, by feasibility of F , means $\phi(C) = 0$. ◀

It may be possible to get a better-than-2 approximation through a more involved approach. For example, notice in our proof $n_d \leq OPT$ is only tight if all components C of the optimum solution F^* with $C \cap V_d \neq \emptyset$ had $|C| = 2$ (i.e. F^* is a matching). So if the optimal solution F^* has at least, say, $0.01 \cdot n_d$ nodes of V_d in components of size greater than 2 then any minimal solution would be a 1.99-approximation. Otherwise, nearly all nodes of V_d are in components that consist of a single edge. In this case, a maximum-size matching between V_d and V_s would then create components of charge 0 that collectively include most nodes of V_d . One can even show there is a way to “alternate” the matching so that a greedy algorithm yields a good approximation (i.e. by alternating so our matching edges largely agree with the optimum matching edges), but efficiently finding such an alternation seems challenging.

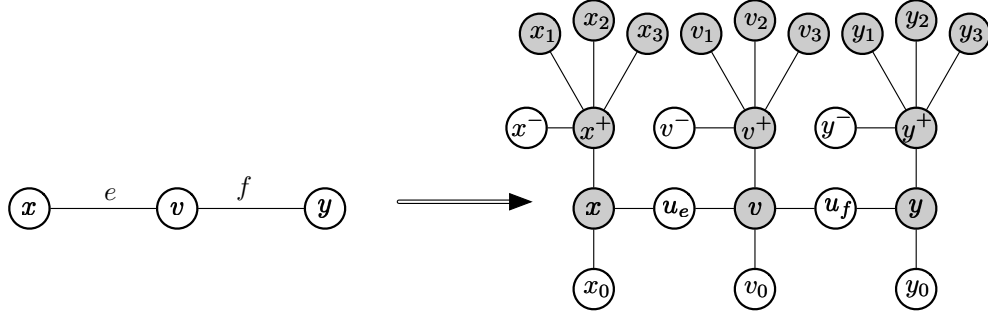
One would reasonably wonder if instances of GP2P with unit-cost edges and ± 1 charges are actually hard. Indeed, we conclude this section by showing **APX**-hardness.

► **Theorem 16.** *The restriction of GRAPHICAL-GP2P to instances with unit edge costs and ± 1 charges is **APX**-hard.*

Proof. We reduce from the MINIMUM VERTEX COVER PROBLEM in simple, cubic graphs which is known to be **APX**-hard [1]. That is, for some constants $0 \leq \beta < \alpha \leq 1$ it is **NP**-hard to distinguish if a cubic graph on n vertices has a vertex cover of size at most $\beta \cdot n$ or if all vertex covers have size exceeding $\alpha \cdot n$.

So let $G = (V, E)$ be an n -vertex cubic graph with $m = 3n/2$ edges. We obtain a graph $H = (V', E')$ and set the charges of $v \in V'$ as follows.

Initially let $H = G$ and set $\phi(v) = +1$ for every $v \in V$. Then subdivide each edge e with a single vertex u_e with $\phi(u_e) = -1$. Then for each v we add new vertices $v^+, v^-, v_0, v_1, v_2, v_3$ and edges $\{v_0v, vv^+, v^+v^-, v^+v_1, v^+v_2, v^+v_3\}$. Here, v, v^+, v_1, v_2, v_3 all have positive charge and v_0, v^- have negative charge. See Figure 1 for an illustration of this process.



■ **Figure 1** An illustration of the reduction to a vertex v and two of its neighbours. In the right picture, positively-charged nodes are shown in grey and negatively-charged nodes are shown in white. Intuitively, selecting vv^+ corresponds to using v in the vertex cover because it now permits using the positive charge nodes v_1, v_2, v_3 to satisfy the demands of some negative charge nodes u_e .

We claim G has a vertex cover of size k if and only if the GRAPHICAL-GP2P instance given by H and ϕ has a solution of size $2n + 2m + k = 5n + k$. So if G has a vertex cover of size at most βn then there is a GP2P solution of size at most $(5 + \beta) \cdot n$ and if all vertex covers in G have more than $\alpha \cdot n$ then all solutions to this GP2P instance have size more than $(5 + \alpha) \cdot n$. This shows there is no $\frac{5+\alpha}{5+\beta}$ -approximation for GP2P with unit edge costs and charges ± 1 unless **P** = **NP**.

To see the claim, first let $C \subseteq V$ be a vertex cover of G with size k . To get a corresponding GRAPHICAL-GP2P solution, buy the following edges:

- For each $v \in C$, purchase vv^+ .
- For each $v \in V$, purchase v_0v and v^+v^- .
- For each $e \in E$, let v be any endpoint of e in C . Purchase $u_e v$ and any edge of the form $v_i v^+$ that was not purchased by another edge this way. Since G is cubic, this is always possible.

In total, this purchases $k + 2n + 2m = k + 5n$ edges. This can be seen to be a feasible solution by matching each negative-charged vertex with a positively-charged vertex in its component in a one-to-one fashion. Such a mapping is immediate from the construction: each v_0 can be matched with v , each v^- can be matched with v^+ , and each u_e can be matched with the corresponding v_i purchased in the description above.

For the converse, we first claim there is a well-structured optimal solution.

▷ **Claim 17.** There is an optimal solution F such that: (a) for each $v \in V$ both v_0v and v^+v^- are in F , (b) F has exactly m edges of the form $v_i v^+$ where $v \in V$ and $i = 1, 2, 3$, (c) each $e \in E$ has $vv^+ \in F$ for at least one endpoint v of e , and (d) for each edge $e = vw$ exactly one of $u_e v \in F$ or $u_e w \in F$.

If so, then $C = \{v : vv^+ \in F\}$ is a vertex cover of G and $|F| = |C| + 2n + 2m = |C| + 5n$, as required to complete the proof.

Proof of Claim 17. Let F be an optimal solution. First observe we must have $v_0v \in F$ and $v^+v^- \in F$ for every $v \in V$ otherwise some negatively-charged node would be isolated. So in each component we must have the number of nodes of the form u_e for various $e \in E$ is at most the number of nodes of the form v_i for various $v \in V, i = 1, 2, 3$. By optimality of F and the fact each v_i is a pendant node, we have that these counts are in fact equal in each component.

Let τ be any minimum-cost pairing of nodes $u_e, e \in E$ with nodes of the form v_i lying in the same component as u_e . Here, the cost of pairing two nodes is the length of their shortest path using only edges in F . For each $e \in E$, let P_e denote the corresponding shortest path from u_e to $\tau(u_e)$. Finally, let $v(e)$ be the first vertex after u_e along P_e , notice $v(e)$ corresponds to an endpoint of e in the original cubic graph G .

Now consider an alternative pairing of the u_e vertices with the v_i vertices (which do not necessarily need to lie in the same component as u_e , for now). For each u_e , pair it with any vertex of the form $v(e)_i$ for some $i = 1, 2, 3$ that has not been paired before. Such a pairing is possible since G is cubic. Let F' be obtained by modifying F in the following way. From F , first delete all edges of the form $v^+v_i \in F$ then add all edges of the form v^+v_i where v_i is paired under the new pairing.

This does not change the size of F . To ensure it is feasible, do the following. For each $e \in E$ such that P was not initially paired with one of $v(e)_1, v(e)_2, v(e)_3$, it must have been that $v(e)u_{e'}$ was the second edge along P for some $e \neq e'$ and that $v(e') \neq v(e)$ (otherwise P_e and $P_{e'}$ both use $v(e)u_{e'}$ but in opposite directions, meaning we can uncross the paths to get a cheaper pairing than τ). So we remove $v(e)u_{e'}$ and add $v(e)v(e)^+$ (if it was not already there). Doing so for all $e \in E$ will ensure it is now feasible since each u_e can now reach the vertex it is paired with while not increasing the size of F because an edge of the form $v(e)v(e)^+$ is added only after an edge of the form $v(e)u_{e'}$ is removed. This also ensures all properties required by the claim now hold. \triangleleft

◀

4 A Polynomial-Time Logarithmic Approximation

We show that a slight variation of algorithm of [10] yields an $O(\log n)$ -approximation in truly polynomial time. The algorithm in [10] begins by observing, using standard metric embeddings [7], that it suffices to given an $O(1)$ -approximation if the graph is a tree which, after rooting at some vertex, has exactly two children per node. Then they present an exact algorithm using dynamic programming where one index of the DP table considers values up to Φ . Roughly speaking, for every vertex v and every $-\Phi \leq p \leq \Phi$ they consider the subtree T_v rooted at v and compute the cheapest subset of edges in the tree such that the component with v has charge p and every other component in the subtree has nonnegative charge.

We simply point out that we can flip the roles of charges and costs in the DP table. First, we use the reduction in Lemma 19 (using, say, $\epsilon = 1/2$) to instances where each edge has its cost being bounded by a polynomial in n . It is easy to verify this reduction produces a tree if the input graph was a tree. Let (T, c, ϕ) be the resulting instance of GP2P, i.e. T is a tree and each edge cost is a positive integer bounded by a polynomial in n .

Root T at an arbitrary node r and for each node v let T_v be the subtree under v . Our dynamic programming table is the following: for each node v and each $0 \leq c \leq \sum_{e \in E} c_e$ let $f[v, c]$ be the maximum p such that in the subtree T_v , it is possible to purchase edges with total cost at most c such that the component with v has charge at least p and every other component has nonnegative charge. Given these values, the optimum solution cost is then the minimum c such that $f[r, c] \geq 0$.

To compute the $f[v, c]$ values:

- If v is a leaf node then $f[v, c] = \phi(v)$.
- Otherwise, say u, w are the two children of v . Intuitively, we try all subsets of $\{uv, vw\}$ to delete and try all ways to split the remaining budget between the subproblems and keep the best solution found overall. That is, we try all ways to purchase a subset $S \subseteq \{uv, vw\}$ such that $c(S) \leq c$ and all $0 \leq c_u, c_w$ such that $c_u + c_w + c(S) = c$ and such that $f[u, c_u] \geq 0$ if $uv \notin S$ and $f[w, c_w] \geq 0$ if $vw \notin S$ (i.e. if we do not purchase the corresponding parent edge, then the budget in the subproblem better be able to buy a feasible solution in the subtree).

Then $f[v, c]$ is the maximum of the following expression over such S, c_u, c_w :

$$\phi(v) + \mathbb{I}[uv \in S] \cdot f[u, c_u] + \mathbb{I}[vw \in S] \cdot f[w, c_w]$$

where $\mathbb{I}[\cdot]$ is the $\{0, 1\}$ -indicator for the logical expression enclosed by the brackets. The number of distinct subproblems is polynomial in n since the edge costs are integers at most n and there are at most $4 \cdot (c + 1)$ different ways to select (S, c_u, c_w) in a subproblem, the algorithm runs in polynomial time. In particular, if C denotes the total edge cost in the tree then the number of distinct subproblems is $O(C \cdot n)$ and processing each entry $f[v, c]$ takes $O(c)$ time (including $O(c)$ recursive calls) so the total running time is $O(C^2 \cdot n)$. Finally, if one only permits recursive calls to subproblems $f[v, c]$ where c is at most the total edge cost in the subtree T_v and the loops over the split $c_u + c_w = c$ only iterate over values where c_u and c_w are at most the total edge cost of their respective subtrees, the running time is improved to $O(C^2)$.

This polynomial-time approximation for GP2P in trees can be used in a black-box fashion to improve the running time of the $O(\log(\min\{n, \phi(V) + 2\}))$ in [10] to run in true polynomial time.

References

- 1 Paola Alimonti and Viggo Kann. Some apx-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1):123–134, 2000. doi:10.1016/S0304-3975(98)00158-3.
- 2 Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, September 1998. doi:10.1145/290179.290180.
- 3 Sanjeev Arora and George Karakostas. A $2 + \epsilon$ approximation algorithm for the k-mst problem. *Math. Program.*, 107(3):491–504, July 2006.
- 4 Emmett Breen, Renee Mirka, Zichen Wang, and David P. Williamson. Revisiting garg’s 2-approximation algorithm for the k -mst problem in graphs. In *2023 Symposium on Simplicity in Algorithms, SOSA 2023*, pages 56–68. SIAM, 2023. doi:10.1137/1.9781611977585.CH6.
- 5 Luca Di Gaspero, Johannes Gärtner, Guy Kortsarz, Nysret Musliu, Andrea Schaerf, and Wolfgang Slany. The minimum shift design problem. *Annals of operations research*, 155:79–105, 2007. doi:10.1007/S10479-007-0221-1.
- 6 Guy Even, Guy Kortsarz, and Wolfgang Slany. On network design problems: fixed cost flows and the covering steiner problem. *ACM Trans. Algorithms*, 1(1):74–101, July 2005. doi:10.1145/1077464.1077470.

- 7 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004. Special Issue on STOC 2003. doi:10.1016/j.jcss.2004.04.011.
- 8 Naveen Garg. Saving an epsilon: a 2-approximation for the k-mst problem in graphs. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 396–402, 2005. doi:10.1145/1060590.1060650.
- 9 Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995. doi:10.1137/S0097539793242618.
- 10 Mohammadtaghi Hajiaghayi, Rohit Khandekar, Guy Kortsarz, and Zeev Nutov. On fixed cost k-flow problems. *Theor. Comp. Sys.*, 58(1):4–18, January 2016. doi:10.1007/s00224-014-9572-6.
- 11 Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999. doi:10.1137/S0097539796309764.
- 12 Kunal Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 281–290. ACM, 2004. doi:10.1145/1007352.1007399.

A Standard Reductions

Proof of Observation 3. Each $v \in V - R$ with $w_v \geq 1$ is replaced with $(n + 1) \cdot w_v$ colocated copies and each $v \in V - R$ with $w_v = 0$ is left as well. In the new k-MST-R instance, use $k' := (n + 1) \cdot k$. Consider any tree T (say, one obtained using a k-MST-R approximation) in the new instance that spans R and at least k' other nodes. By adding 0-cost edges if necessary, we may assume that T contains all colocated copies of any node it spans.

Since at most n nodes have $w_v = 0$, then T spans at least $(n + 1) \cdot k - n > (n + 1) \cdot (k - 1)$ nodes from groups of colocated copies of original nodes. Since each group has a size that is a multiple of $n + 1$, then T spans at least $(n + 1) \cdot k$ such nodes, i.e. viewing T as a tree in the original graph after contracting colocated copies of nodes it spans yields a feasible solution. \blacktriangleleft

► **Lemma 18.** *Let $(G = (V, E), c, \phi)$ be an instance of GP2P and let $G' = (V, E')$ be the complete graph over V with metric edge costs $c'(uv)$ given by the minimum-cost $u - v$ path. Any feasible solution to the GP2P instance (G, c, ϕ) can be mapped to a feasible solution to the METRIC-GP2P instance (G', c', ϕ) with no greater cost and vice-versa.*

Proof. For each $e \in E$ we have $c'(e) \leq c(e)$ since e is one possible path between its endpoints. So for any feasible solution F to (G, c, ϕ) we have $c'(F) \leq c(F)$, as required. Conversely, for any feasible solution $F' \subseteq E'$ to (G', c', ϕ) if we let F be the union of all shortest $u - v$ paths in G for each $uv \in F'$ then F then $c(F) \leq c'(F)$. Also, two nodes that were in the same connected component in (V, F') still lie in the same connected component of (V, F) . That is, each connected component C of (V, F) is the union of one or more connected components $\{C'_1, \dots, C'_a\}$ in (V, F') so $\phi(C) = \sum_{i=1}^a \phi(C'_i) \geq 0$. \blacktriangleleft

► **Lemma 19.** *For any constant $\epsilon > 0$, if there is an α -approximation for instances of GP2P where every edge cost $c(e)$ is an integer at most $n^2/\epsilon + 1$, then there is a $(1 + \epsilon) \cdot \alpha$ -approximation for general instances of GP2P.*

Proof. Let $(G = (V, E), c, \phi)$ be a general instance of GP2P with optimum solution $F^* \subseteq E$ with cost OPT . By contracting 0-cost edges (which does not change the optimal solution value), we assume $c(e) > 0$ for each $e \in E$.

First, compute smallest value ν such that every connected component C in the graph G_ν with edges $\{e \in E : c(e) \leq \nu\}$ has $\phi(C) \geq 0$. We claim $\nu \leq OPT \leq n \cdot \nu$: the first bound is because any feasible solution must use at least one edge of cost $\geq \nu$ by our choice of ν and the other is because a spanning forest of G_ν is a feasible solution using fewer than n edges each of cost at most ν .

Let $E' = \{e \in E : c(e) \leq n \cdot \nu\}$; we have $F^* \subseteq E'$ since $OPT \leq n \cdot \nu$. Define new edge costs $c'(e) := \lceil n \cdot c(e) / (\epsilon \cdot \nu) \rceil$ for each $e \in E'$. Notice $c'(e)$ is a positive integer (since $c(e) > 0$) and $c'(e) \leq n \cdot c(e) / (\epsilon \cdot \nu) + 1 \leq n^2 / \epsilon + 1$ by construction of E' .

Let OPT' denote the optimum solution cost for the GP2P instance $((V, E'), c', \phi)$. We have $OPT' \leq c'(F^*) \leq \sum_{e \in F^*} (n \cdot c(e) / (\epsilon \cdot \nu) + 1) = n \cdot OPT / (\epsilon \cdot \nu) + n$. Therefore, running an α -approximation on this new instance finds a set of edges $F' \subseteq E'$ with $c'(F') \leq \alpha \cdot (n \cdot OPT / (\epsilon \cdot \nu) + n)$. Since $c(e) \leq \epsilon \cdot \nu \cdot c'(e) / n$ for every $e \in E'$,

$$c(F') \leq \epsilon \cdot \nu \cdot c'(F') / n \leq \alpha \cdot (OPT + \epsilon \cdot \nu) \leq (1 + \epsilon) \cdot \alpha \cdot OPT. \quad \blacktriangleleft$$

B **Restricted Instances of GP2P**

Proof of Observation 6. Let $(G = (V, E), c, \phi)$ be an instance of GP2P. As noted in the introduction, we may assume this is an instance of METRIC-GP2P. Finally, let $V' = V - V_0$ (i.e. the nodes with non-zero $\phi(v)$) and G' be the subgraph of G induced by V' . Notice the restriction of (G', c, ϕ) to H is an instance of METRIC-GP2P.

Let $F^* \subseteq E$ be an optimal solution for the original METRIC-GP2P instance. For each tree T in the forest F^* , let C_T be the tour obtained by doubling the edges of T and shortcutting the resulting Eulerian tour past nodes in V_0 . In this way, C_T spans all nodes in $V - V_0$ that are spanned by T and $c(C_T) \leq 2 \cdot c(T)$. Thus, the optimal solution cost in the restriction to H is at most twice the optimal solution cost for (G, c, ϕ) .

To complete the reduction, for each vertex v of H if $\phi(v) \geq 1$ then replace v with $\phi(v)$ collocated copies each having charge 1 and if $\phi(v) \leq -1$ then replace v with $-\phi(v)$ collocated copies each having charge -1 . Note this steps takes pseudopolynomial time. \blacktriangleleft

Proof of Observation 7. Consider any constant $0 < \epsilon' \leq 1$. Let $(G = (V, E), c, \phi)$ be an instance of GP2P. First we consider some preprocessing. If the 0-cost edges form a feasible solution, then it must be optimal so there is nothing more to do. Otherwise, apply Lemma 19 with $\epsilon := \epsilon' / 3$ and let $(G' = (V, E'), c', \phi)$ be the resulting graph with positive integer edges costs being bounded by a polynomial in n .

Let K be a positive integer to be specified later. Form $(G'' = (V'', E''), c'', \phi'')$ by performing the following operations to G' .

- Subdivide each $e \in E'$ into a path of length $K \cdot c(e)$ of unit cost edges. Each new vertex v' in the subdivision has $\phi''(v') = 0$.
- For each $v \in V'$ with $\phi(v) \geq 1$, append a path P_v to v using $\phi(v) - 1$ new vertices and edges: each edge has cost 1 and each vertex on P_v , including v itself, has $\phi''(v) = 1$. Note, the other endpoint of P_v is a pendant.
- Similarly each $v \in V'$ with $\phi(v) \leq -1$, append a path P_v to v using $-\phi(v) - 1$ new vertices and edges: each edge has cost -1 and each vertex on P_v , including v itself, has $\phi''(v) = -1$.
- Each $v \in V'$ with $\phi(v) = 0$ also has $\phi''(v) = 0$.
- Finally, each edge e of this new graph G'' has $c''(e) = 1$.

Observe (G'', c'', ϕ'') is an instance of GRAPHICAL-GP2P with $\phi(v) \in \{-1, 0, +1\}$ for each $v \in V''$.

Note a solution $F' \subseteq E'$ naturally maps to a solution in the final instance with cost at most $\Phi + K \cdot c'(F')$ by including all pendant paths P_v and all subdivided paths corresponding to edges in F' . Conversely, consider any solution $F'' \subseteq E''$. Let $F' \subseteq E$ be the set of edges of G' such that their entire subdivision is included in F'' . It is easy to verify that F' is a feasible solution with cost at most $c''(F'')/K$.

Let OPT' be the optimal solution value for instance (G', c', ϕ) and let F'' be the result of using α -approximation on (G'', c'', ϕ'') . By the preceding discussion, this yields a feasible solution F' to (G', c', ϕ) with

$$c'(F') \leq c''(F'')/K \leq \alpha \cdot (\Phi + K \cdot OPT')/K = \alpha \cdot OPT' + \alpha \cdot \Phi/K.$$

By setting $K = \lceil 3 \cdot \Phi/\epsilon' \rceil$ and noting that $1 \leq OPT$ as edge costs are positive integers in (G', c', ϕ) , this is at most $(1 + \epsilon'/3) \cdot \alpha \cdot OPT'$.

Finally, by accounting for the application of Lemma 19 at the start of this proof we see that we would have an approximation for the original instance (G, c, ϕ) with guarantee $(1 + \epsilon'/3)^2 \cdot \alpha \leq (1 + \epsilon') \cdot \alpha$. ◀

C Adapting k-MST Approximations

We only sketch how the algorithms can be adapted. We refer the reader to the respective papers for their details.

► **Lemma 20** (Slight adaptation of Arora and Karakostas [3]). *There is a polynomial-time $(2 + \epsilon)$ -Approximation for W-k-MST-R.*

Proof. The algorithm in [3] explicitly guesses a subset of vertices that appear in the optimum solution and builds that into the LP relaxation they write. So it can already handle the situation where we have a larger set of required nodes R . The only thing to mention is how it can be extended to handle node weights $w_v \geq 0$ for $v \notin R$. We can assume each node with weight w_v is implicitly a collection of w_v many nodes connected using 0-cost edges in a star fashion. The algorithm of [3] first guesses an additional set of size $O(1/\epsilon)$ of vertices of OPT to be required. The next step of the algorithm is to run a “primal dual” like algorithm to find a tree T . This step works without modification in our setting. ◀

The final step in [3] is to modify T appropriately. That is, the manner in which T was constructed actually provides us with two options: include some subset of nodes or not. One choice would result in fewer than k non-required nodes being spanned and the other would result in at least k non-required nodes being spanned. In [3], it is mentioned how to pick the correct number of nodes contiguously from this “optional” portion so that grafting them in to the remaining portion of T yields a feasible solution with the required number of nodes. The grafting only costs $O(\epsilon \cdot OPT)$ due to the guess of the net at the start of the algorithm. This can also be done in polynomial time if one implicitly maintains a 0-cost tree spanning each group of colocated points.

► **Lemma 21** (Slight adaptation of Arora [2]). *There is a PTAS for k-MST-R*

We first comment that a similar adaptation could be made to Mitchell’s PTAS [11]. We chose this one because it was slightly easier to describe. We also emphasize that this adaptation is only for *unweighted* k-MST-R. Combining this with Observation 3 yields a pseudo-polynomial time approximation scheme for W-k-MST-R.

28:16 **Approximation Algorithms for the Generalized Point-To-Point Problem**

Proof. The PTAS for k -MST in constant-dimensional Euclidean plane uses a dynamic programming routine through a quadtree dissection of the plane (an in higher dimensions a 2^D -tree dissection in D -dimensional spaces). The DP table entries for each square, roughly speaking, describe the interface of the optimal solution across the boundary of that square through “portals” and also include the guess for how many nodes should be covered within the square. If there are required nodes R , we can use the same DP table and simply insist that the subproblem’s solution also span any nodes of R in the square. The base cases are trivially extended to this setting and the combination of subproblems (i.e. the recurrence) for a non-base case is identical to before. ◀