Approximate Earth Mover's Distance in Truly-Subquadratic Time

Lorenzo Beretta, Aviad Rubinstein

Abstract

We design an additive approximation scheme for estimating the cost of the min-weight bipartite matching problem: given a bipartite graph with non-negative edge costs and $\varepsilon > 0$, our algorithm estimates the cost of matching all but $O(\varepsilon)$-fraction of the vertices in truly subquadratic time $O(n^{2-\delta(\varepsilon)})$.

- Our algorithm has a natural interpretation for computing the Earth Mover's Distance (EMD), up to a ε-additive approximation. Notably, we make no assumptions about the underlying metric (more generally, the costs do not have to satisfy triangle inequality). Note that compared to the size of the instance (an arbitrary $n \times n$ cost matrix), our algorithm runs in *sublinear* time.
- Our algorithm can approximate a slightly more general problem: max-cardinality bipartite matching with a knapsack constraint, where the goal is to maximize the number of vertices that can be matched up to a total cost B.

1 Introduction

Earth Mover's Distance (EMD - sometimes also Optimal Transport, Wasserstein-1 Distance or Kantorovich–Rubinstein Distance) is perhaps the most important and natural measure of similarity between probability distributions over elements of a metric space [PC+19; San15; Vil+09]. Formally, given two probability distributions μ and ν over a metric space $(\mathcal{M}, d)$ their EMD is defined as

$$\mathrm{EMD}(\mu,\nu) = \min\left\{ \mathrm{E}_{(x,y)\sim\zeta}[d(x,y)] \;\middle|\; \zeta \text{ is a coupling}^1\text{of } \mu \text{ and } \nu \right\}. \tag{1}$$

When μ and ν are discrete distributions with support size n (perhaps after a discretization preprocessing), a straightforward algorithm for estimating their EMD is to sample $\Theta(n)$ elements from each, compute all $\Theta(n^2)$ pairwise distances, and then compute a bipartite min-weight perfect matching. This algorithm clearly takes at least $\Theta(n^2)$ time (even ignoring the computation of the matching), and incurs a small additive error due to the sampling.

Our main result is an asymptotically faster algorithm for estimating the EMD:

[1] A distribution ζ over $\mathcal{M}^2$ is a coupling of μ and ν if $\mu(x) = \int_{\mathcal{M}} \zeta(x,y)\,dy$ and $\nu(y) = \int_{\mathcal{M}} \zeta(x,y)\,dx$.

Theorem 1 (Main Theorem)**.** *Suppose we have sample access to two distributions μ, ν over metric space $(\mathcal{M}, d)$ satisfying $d(\cdot,\cdot) \in [0,1]$ and query access to d. Suppose further that μ, ν have support size at most n.*

For each constant $\gamma > 0$ there exists a constant $\varepsilon > 0$ and an algorithm running in time $O(n^{2-\varepsilon})$ that outputs $\widehat{\mathrm{EMD}}$ such that

$$\widehat{\mathrm{EMD}} \in [\mathrm{EMD}(\mu,\nu) \pm \gamma].$$

Moreover, such algorithm takes $\tilde{O}(n)$ samples from μ and ν.

Notably, our algorithm makes no assumption about the structure of the underlying metric. In fact, it can be an arbitrary non-negative cost function, i.e. we do not even assume triangle inequality.

Beyond bounded support size. Support size is a brittle matter, indeed two distributions that are arbitrarily close in total variation (TV) distance (or EMD) can have completely different support size. Moreover, for continuous distributions the notion of support size is not even defined yet we would like to compute their EMD through sampling. To obviate this issue, Theorem 2 generalize Theorem 1 to distributions that are *close* in EMD to some distributions with support size n.

Theorem 2. *Suppose we have sample access to two distributions μ, ν over metric space $(\mathcal{M}, d_{\mathcal{M}})$ satisfying $d(\cdot,\cdot) \in [0,1]$ and query access to d. Suppose further that there exist μ', ν' with support size n such that $\mathrm{EMD}(\mu,\mu'), \mathrm{EMD}(\nu,\nu') \leqslant \xi$, for some $\xi > 0$.*

For each constant $\gamma > 0$ there exists a constant $\varepsilon > 0$ and an algorithm running in time $O(n^{2-\varepsilon})$ that outputs $\widehat{\mathrm{EMD}}$ such that

$$\widehat{\mathrm{EMD}} \in [\mathrm{EMD}(\mu,\nu) \pm (4\xi + \gamma)].$$

Moreover, such algorithm takes $\tilde{O}(n)$ samples from μ and ν.

For continuous μ, requiring that μ is close to a distribution with bounded support size is equivalent to saying that μ can be discretized effectively for EMD computation. Thus, such assumption is natural while computing EMD between continuous distribution through discretization.

For discrete μ on an large number of points, we can think of such bounded-support-size approximation of μ as a *coreset* for EMD computations. Such coresets naturally exist, for instance, for discrete distributions with low doubling dimension.

We stress that the algorithm in Theorem 2 does not assume knowledge of μ' (nor ν') beyond its support size n. Indeed, the empirical distribution over $\tilde{O}(n)$ samples from μ (resp. ν) makes a good approximation in EMD. Finally, the sample complexity in Theorem 1 and Theorem 2 is optimal, up to polylog(n) factors. Indeed, Theorem 1 in [VV10] implies a lower bound of $\tilde{\Omega}(n)$ on the sample complexity of testing EMD closeness[2].

Matching with knapsack constraint. Applying our main algorithm to a graph-theory setting, we give an approximation scheme for a knapsack bipartite matching problem, where our goal is to estimate the number of vertices that can be matched subject to a total budget constraint.

Theorem 3 (Main theorem, graph interpretation)**.** *For each constant $\gamma > 0$, there exists a constant $\varepsilon > 0$, and an algorithm running in time $O(n^{2-\varepsilon})$ with the following guarantees. The algorithm takes as input a budget B, and query access to the edge-cost matrix of an undirected, bipartite graph G over n vertices. The algorithm returns an estimate $\widehat{M}$ that is within $\pm\gamma n$ of the size of the maximum matching in G with total cost at most B.*

[2]While deriving the lower bound from [VV10] takes some work, Remark 5.13 in [Can20] explicitly states a $\Omega(n/\log n)$ lower bound for TV closeness testing.

1.1 Relaed Work

Computing EMD is an important problem in machine learning [PC+19] with some exemplary applications in computer vision [ACB17; RTG00; Sol+15] and natural language processing [Kus+15; Yur+19]. See [PC+19] for a comprehensive overview.

Exact solution. Computing EMD between two sets of n point boils down to computing the minimum cost of a perfect matching on a bipartite graph, a problem with a 70-years history [Kuh55]. Min-weight bipartite perfect matching can be cast as a min-cost flow instance and to date we can solve it in $n^{2+o(1)}$ time (namely, near-linear in the size of the distance matrix) [Che+22a]. Apparently, any exact algorithm requires inspecting the entire distance matrix, thus $\Theta(n^2)$ time is the best we can hope for. In addition, even in d-dimensional Euclidean space, where the input has size $d \cdot n \ll n^2$, no $O(n^{2-\varepsilon})$ algorithm exists[3], unless SETH is false [Roh19].

Multiplicative approximation. A significant body of work has investigated multiplicative approximation of EMD [Aga+22; AIK08; And+09; And+14; AS14; AZ23; Cha02; Che+22b; Ind03], where the most commonly studied setting is the Euclidean space (or, more generally, ℓ_p). If the dimension is constant we have near-linear time approximation schemes [Aga+22; And+14; FL22; SA12], whereas the high-dimensional case is more challenging. Only recently [AZ23] broke the $O(n^2)$ barrier for $(1+\varepsilon)$-approximation of EMD, building on [HIS13].

The landscape is much less interesting for general metrics. Indeed, a straightforward counterexample from [Bǎd+05] shows that any $O(1)$-approximation requires $\Omega(n^2)$ queries to the distance matrix. This suggests that for general metrics we should content ourselves with a additive approximation.

Additive approximation. Additive approximation for EMD has been extensively studied by optimization and machine learning communities [ANR17; Bla+18; Cut13; DGK18; Le+21; LXH23; Pha+20].

An extremely popular algorithm to solve optimal transport in practice is Sinkhorn algorithm [Cut13] (see [Le+21; Pha+20] for recent work). Sinkhorn distance SNK is defined by adding an entropy regularization term $-\eta \cdot H(\zeta)$ to the EMD objective in Equation (1). Approximating SNK via Sinkhorn algorithm provably yields a εr-additive approximation to EMD and takes $O_\varepsilon(n^2)$ time, where r is the dataset diameter [ANR17].

Graph-theoretic approaches also led to εr-additive approximations [LMR19] in $O_\varepsilon(n^2)$ time. Notice that even though all previous approximation algorithms have roughly the same complexity as the MCF-based exact solution they are backed by experiments showing their practicality, whereas exact algorithms for EMD are still largely impractical .

Breaking the $O(n^2)$ barrier. In a recent paper Charikar *et al.*[Cha+23] write:

> *"There are no algorithmic approaches that achieve $(1\pm\varepsilon)$-approximations or εr-additive approximations for either EMD nor SNK[4] in time $n^{1.99}$. In addition, there is some reason to believe that this may be impossible for EMD [Roh19]".*

[3] The lower bound in [Roh19] holds in dimension $d = 2^{\Omega(\log^* n)}$.

[4] SNK is the Sinkhorn distance or entropy-regularized optimal transport defined in [Cut13].

Both [AZ23] and our Theorem 1 disprove this conjecture. Indeed [AZ23] shows a $(1+\varepsilon)$-multiplicative-approximation for Euclidean metrics, while Theorem 1 provides an ε-additive approximation for general metrics and both run in $n^{2-\Omega_\varepsilon(1)}$ time.

We stress that despite [AZ23] and this work proving similar results, they use a completely different set of techniques. Indeed, in [AZ23] they approximate the complete bipartite weighted graph induced by Euclidean distances with a $(1+\varepsilon)$-multiplicative spanner of size $n^{2-\Omega_\varepsilon(1)}$. Their spanner construction is based on LSH and so it hinges on the Euclidean structure. Then, they run a near-linear time MCF solver [Che+22a] to solve the matching problem on the metric induced by the spanner. In this work, instead, we build on sublinear algorithms for max-cardinality matching [Beh+23; Beh22; BKS23a; BKS23b; BRR23] and do not leverage any metric property, not even triangle inequality. Section 2 contains a detailed explanation of our techniques.

It is worth to notice that since [AZ23] operates over d-dimensional Euclidean space the input representation takes $d \cdot n$ space, and so it *does not* run in sublinear time. On the contrary, our algorithm assumes query access to the distance matrix and runs in sublinear time.

Sublinear algorithms. Most previous work in sublinear models of computation focuses on streaming Euclidean EMD [AIK08; And+09; Bac+20; Cha02; Che+22b; Ind04], where the latest work [Che+22b] achieves $\tilde{O}(\log n)$-approximation in polylogarithmic space. Some other work [Ba+11] addresses the sample complexity of testing EMD on low-dimensional ℓ_1.

In this work we focus on a different access model: we do not make any assumption on the ground metric and we assume query access to the distance matrix. This model is natural whenever the underlying metric is expensive to evaluate. For example, in [ALT21] they consider EMD over a shortest-path ground metric and experiment with heuristics to avoid computing all-pair distances, which would be prohibitively expensive.

Comparison with MST. MST and EMD are two of the most studied optimization problems in metric spaces. It is interesting to observe a separation between the sublinear-time complexity of MST and EMD for general metrics. Indeed, [CS09] shows a $\tilde{O}_\varepsilon(n)$ time algorithm approximating the *cost* of MST up to a factor $1+\varepsilon$, whereas no $O(1)$-approximation for EMD can be computed in $o(n^2)$ time [Bǎd+05]. Essentially, this is due to the fact that MST cost is a more *robust* problem than EMD, in that deleting input points can cause EMD to increase while this does not happen to MST cost.

A valuable take-home message from this work is that allowing additive approximation makes EMD more robust. A natural question is whether we can find a ε-additive approximation to EMD in linear time, thus matching the above result on MST cost. The $\Omega(n^{1.2})$ lower bound on max-cardinality matching from [BRR23] suggests that this should not be possible[5]. However, it is not clear how to embed the lower bound from [Beh22] into a metric space.

2 Technical Overview

Computing Earth Mover's Distance between two sets of n points in a metric space can be achieved by solving Min-Weight Perfect Matching (MWPM) on the complete bipartite graph where edge-

[5]The lower bound of [BRR23] is proven in a slightly different model of adjacency list, but it seems plausible that it can be extended to the adjacency matrix model.

costs are given by the metric $d(\cdot,\cdot)$. Here we seek a suitable notion of approximation for MWPM that recovers Theorem 1.

Min-weight perfect matching with outliers. Consider the following problem: given a constant $\gamma > 0$, find a matching M of size $(1-\gamma)n$ in a bipartite graph such that the cost of M is at most the minimum cost of a perfect matching. A natural interpretation of this problem is to label a γ fraction of vertices as outliers and leave them unmatched; so we dub this problem MWPM *with outliers*.

Assuming $d(\cdot,\cdot) \in [0,1]$, solving MWPM with a γ fraction of outliers immediately yields a γ additive approximation to EMD, proving Theorem 1.

The main technical contribution of this work is the following theorem, which introduces an algorithm that solves MWPM with outliers in sublinear time. For the sake of this overview, the reader should instantiate Theorem 4 with $\beta = 1$ and think of $\gamma = (1-\alpha)$ as the fraction of allowed outliers.

Theorem 4. *For each constants $0 \leqslant \alpha < \beta \leqslant 1$ there exists a constant $\varepsilon > 0$ and an algorithm running in time $O(n^{2-\varepsilon})$ with the following guarantees.*

The algorithm has adjacency-matrix access to an undirected, bipartite graph $G = (V_0 \cup V_1, E)$ and random access to the edge-cost function $c : E \to \mathbb{R}^+$. The algorithm returns $\hat{c}$ such that, whp,

$$c(M^\alpha) \leqslant \hat{c} \leqslant c(M^\beta)$$

where M^α is a minimum-weight matching of size αn and M^β is a minimum-weight matching of size βn.

Moreover, the algorithm returns a matching oracle data structure that, given a vertex u returns, in $n^{1+f(\varepsilon)}$ time, an edge $(u,v) \in \hat{M}$ or $\perp$ if $u \notin V(\hat{M})$, where $f(\varepsilon) \to 0$ when $\varepsilon \to 0$. The matching $\hat{M}$ satisfies $\alpha n \leqslant |\hat{M}| \leqslant \beta n$ and $c(M^\alpha) \leqslant c(\hat{M}) \leqslant c(M^\beta)$.

Notice that the algorithm in Theorem 4 does *not* output the matching $\hat{M}$ explicitly. However, it returns a matching oracle data structure which implicitly stores $\hat{M}$. The rest of this overview sketches the proof of Theorem 4.

Our algorithm, in a nutshell. A new set of powerful techniques was recently developed to approximate the size of a max-cardinality matching in sublinear time [Beh+23; Beh22; BKS23a; BKS23b; BRR23]. Our main contribution is a sublinear-time algorithm which leverages the techniques above to implement (a certain step of) the classic Gabow-Tarjan [GT89] algorithm for MWPM. Since the techniques above return *approximate* solutions, the obtained matching will be approximate as well, in the sense that we have to disregard a fraction of outliers when computing its cost to recover a meaningful guarantee. Careful thought is required for relaxing the definitions of certain objects in the Gabow-Tarjan algorithm so as to accommodate their computation in sublinear time. The bulk of our analysis is devoted to proving that these relaxations combine well and lead to the guarantee in Theorem 4.

Roadmap. First, we will review (a certain step of) the Gabow-Tarjan algorithm that we will use as our template algorithm to be implemented in sublinear time. Then, we will review some recent sublinear algorithms for max-cardinality matching. Finally, we will sketch how to combine these tools to approximate the value of minimum-weight matching.

2.1 A Template Algorithm

The original Gabow-Tarjan algorithm operates on several scales and this makes it (slightly) more involved. We focus on a simpler case where all our edge weights are integers in $[1, C]$, for $C = O(1)$. We will see in Section 6 that we can reduce to this case (incurring a small additive error). Here we describe our template algorithm, at a high level.

A high-level description. Essentially, our template algorithm is a primal-dual algorithm which (implicitly) maintains a pair (M, φ), where M is a partial matching (so primal infeasible), and $\{\varphi(v)\}_{v \in V}$ is a vertex potential function, or an (approximately) feasible dual solution. Moreover, for each $e \in M$ the dual constraint corresponding to e is tight. In other words, the pair (M, φ) satisfies complementary slackness. The algorithm progressively grows the dual variables $\{\varphi(v)\}_{v \in V}$ and the size of M. When M has size $\geqslant (1-\gamma)n$ then we are done. Indeed, throwing out γn vertices (as well as their associated primal constraints) we have that (M, φ) is a (approximately) feasible primal-dual pair that satisfies complementary slackness, thus it is (approximately) optimal.

The algorithm. First, recall the linear program for MWPM together with its dual. Here we consider a complete bipartite bipartite graph with vertex set $V = V_0 \cup V_1$ and cost function $c(\cdot, \cdot) \in [1, C]$.

Primal

$$\begin{aligned} \text{Minimize} \quad & \sum_{u \in V_0, v \in V_0} x_{u,v} \cdot c(u,v) \\ \text{subject to} \quad & \sum_{v \in V_1} x_{u,v} \geqslant 1 \quad \forall u \in V_0 \\ & \sum_{u \in V_0} x_{u,v} \geqslant 1 \quad \forall v \in V_1 \\ & x_{u,v} \geqslant 0 \quad \forall u \in V_0,\ v \in V_1. \end{aligned}$$

Dual

$$\begin{aligned} \text{Maximize} \quad & \sum_{u \in V} \varphi_u \\ \text{subject to} \quad & \varphi_u + \varphi_v \leqslant c(u,v) \quad \forall u \in V,\ v \in V_1 \\ & \varphi_u \geqslant 0. \quad \forall u \in V,\ v \in V_1. \end{aligned}$$

We maintain an initially empty matching M. Inspired by the dual, we define a potential function $\varphi : V \to \mathbb{Z}$ and we enforce a relaxed version of the dual constraints: $\varphi(u) + \varphi(v) \leqslant c(u,v) + 1$ for each $(u, v) \in E$. Moreover, we maintain that $\varphi(u) + \varphi(v) = c(u, v)$ for each $(u, v) \in M$ (complementary slackness). Let T be the set of edges s.t. the constraints above are tight. Orient the edges in T so that all edges in $M \subseteq T$ are oriented from V_0 to V_1 and all edges in $T \backslash M$ are oriented from V_1 to V_0. We denote the set of free (unmatched) vertices F and let $F_0 = F \cap V_0$ $F_1 = F \cap V_1$. We say that a path $P = (v_0 \to \cdots \to v_1)$ is an augmenting path if $v_0 \in F_0$, $v_1 \in F_1$ and P alternates between edges in $T \backslash M$ and M. When we say that we augment M wrt P we mean that we set $M \leftarrow M \oplus P$. We alternate between the following two steps:

1. Find a maximal set of node-disjoint augmenting paths $\{P_1 \dots P_\ell\}$. Augment M wrt these paths. Decrement $\varphi(v)$ `-=` 1 for each $v \in \bigcup_i P_i \cap V_1$, to ensure the relaxed dual constraints are satisfied.

2. Define R as the set of vertices that are T-reachable[6] from F_0. Increment $\varphi(r_0)$ `+=` 1 for each

[6] Recall that T is oriented.

$r_0 \in R \cap V_0$, and decrement $\varphi(r_1)$ `-=` 1 for each $r_1 \in R \cap V_1$. This preserves the relaxed dual constraints and (eventually) adds some more edges to T.

After $O_{\gamma,C}(1)$ iterations, we have $|F| \leqslant \gamma n$.

Analysis sketch. It is routine to verify that steps 1 and 2 preserve the relaxed dual constraints. At any point the pair (M, φ) satisfies, $c(M) \leqslant \sum_{v \in V_0 \cup V_1} \varphi(v) \leqslant c(M') + n$ for any perfect matching M'. We can content ourselves with this additive approximation, indeed in Section 6 we will see how to charge it on the outliers. To argue that we have few free vertices left after $O_\gamma(1)$ iterations, notice that at iteration t we have $\varphi|_{F_0} \equiv t$ and $\varphi|_{F_1} \equiv 0$. Computing a certain function of potentials along $(M \oplus M')$-augmenting paths shows that $|F| \cdot t \leqslant O(n)$. Thus, $O_\gamma(1)$ iterations are sufficient to obtain $|F| \leqslant \gamma n$. The arguments above are sufficient to show that our template algorithm finds an (almost) perfect matching with (almost) minimum weight. We will shove both *almost* under the outlier carpet in Section 6.

2.2 Implementing the Template in Sublinear Time

Our sublinear-time implementation of the template algorithm hinges on matching oracles.

Matching oracles. Given a matching M' we define a *matching oracle* for M' as a data structure that given $u \in V$ returns $v \in V$ if $(u, v) \in M'$ and $\perp$ otherwise. Note that given a matching oracle for M', if we are promised that $|M'| = \Omega(n)$ then $O_\gamma(\log n)$ calls to such oracle are enough to estimate $|M'| \pm \gamma n$. We stress that all matching oracles that we use have sublinear query time.

Finding large matchings in sublinear time. An important ingredient in our algorithm is the `LargeMatching`$(G, A, \varepsilon, \delta)$ subroutine (Theorem 6), which is due to [BKS23a]. Given $A \subseteq V$, `LargeMatching`$(G, A, \varepsilon, \delta)$ returns either $\perp$ or a matching oracle for some matching M' in $G[A]$. If there exists a matching in $G[A]$ of size δn, then `LargeMatching` returns a matching oracle for some M' in $G[A]$ with $|M'| = \Omega_\delta(n)$. Else, if there are no matchings of size δn in $G[A]$ `LargeMatching` returns $\perp$. The parameter ε controls the running time and essentially guarantees that `LargeMatching` runs in $O(n^{2-\varepsilon})$ time while the matching oracle it outputs runs in $O(n^{1+\varepsilon})$.

We will use `LargeMatching` to implement both step 1 and step 2 in the template algorithm. However, this requires us to relax our notions of maximal set of node-disjoint augmenting paths, as well as that of reachability. A major technical contribution of this work is to find the right relaxation of these notions so that:

1) We can analyze a variant of the template algorithm working with these relaxed objects and still recover a a solution which is optimal if we neglect a γ fraction of outliers.

2) We can compute these relaxed objects in sublinear time using `LargeMatching` as well as previously constructed matching oracles.

These relaxed notions are introduced in Section 3, point (1) is proven in Section 4 and point (2) is proven in Section 5.

Implementing step 1 in sublinear time. In [BKS23a] the authors implement McGregor's algorithm [McG05] for streaming Max-Cardinality Matching (MCM) in a sublinear fashion using `LargeMatching` (see Theorem 7 in this work). McGregor's algorithm finds a size-$\Omega(n)$ set of node-disjoint augmenting paths of fixed constant length, whenever there are at least $\Omega(n)$ of them. This notion is weaker than that of a maximal node-disjoint set of augmenting paths required in step 1 of our template algorithm in two regards: first, it only finds augmenting paths of fixed constant length; second, it finds only a constant fraction of such paths (as long as we have a linear number of them).

In our template algorithm, the invariant $\varphi|_{F_1} \equiv 0$ is maintained (in step 2) because $R \cap F_1 = \varnothing$. In turn, $R \cap F_1 = \varnothing$ holds exactly because in step 1 we augment M with a maximal node-disjoint set of augmenting paths. Since our sublinear implementation of step 1 misses some augmenting paths, the updates performed in step 2 will violate the invariant $\varphi(v) = 0$ for some $v \in F_1$.

A careful implementation of step 2 (see next paragraph) guarantees that only missed augmenting paths that are short lead to a violation of $\varphi|_{F_1} \equiv 0$. Moreover, repeatedly running the sublinear implementation of McGregor's algorithm from [BKS23a], we ensure that we miss at most γn short paths, for γ arbitrary small. Thus, we can flag all vertices that belong to missed short augmenting paths as outliers since we have only a small fraction of them.

Implementing step 2 in sublinear time. We implement an approximate version of the reachability query in step 2 as follows. We initialize the set of reachable vertices R as $R \leftarrow F_0$. Then, for a constant number of iterations: we compute a large matching $M' \subseteq T \backslash M$ between the vertices of $R \cap V_0$ and $V_1 \backslash R$; then we add to R all matched vertices in $\bigcup M'$ as well as their M-mates, namely $\texttt{mate}_M(u)$ for each $u \in \bigcup M'$. Notice that if a $\Omega(n)$-size matching $\subseteq T \backslash M$ between $R \cap V_0$ and $V_1 \backslash R$ exists, then we find a matching $\subseteq T \backslash M$ between $R \cap V_0$ and $V_1 \backslash R$ of size at least $\Omega(n)$. This ensures that: (i) after a constant number of iterations `LargeMatching` returns $\perp$; (ii) when `LargeMatching` returns $\perp$ there exists a vertex cover $\mathcal{C}$ of $((R \cap V_0) \times (V_1 \backslash R)) \cap T \backslash M$ of size γn. Only constraints corresponding to edges incident to $\mathcal{C}$ might be violated during step 2. Furthermore, $|\mathcal{C}| = \gamma n$ is small and so we can just label vertices in $\mathcal{C}$ as outliers.

As we pointed out in the previous paragraph, the invariant $\varphi|_{F_1} \equiv 0$ might be violated in step 2 if $R \cap F_1 \neq \varnothing$. We already showed that whenever we the missed augmenting path causing the violation of $\varphi|_{F_1} \equiv 0$ is short we can charge this violation on a small set of outliers. To make sure that no long augmenting path leads to a violation of $\varphi|_{F_1} \equiv 0$ we set our parameters so that the depth of the reachability tree built in step 2 is smaller than the length of "long" paths. Thus, any long path escapes R and cannot cause a violation.

Everything is an oracle. The implementation of both step 1 and step 2 operates on the graph T of tight constraints. To evaluate $(u, v) \in T$, we need to compute $\varphi(u)$ and $\varphi(v)$. In turn, the potential values depend on previous iterations of the algorithm. None of these iterations outputs an explicit description of the objects described in the template (potentials, matchings, augmenting paths or sets of reachable vertices). Indeed, these objects are output as oracle data structures, which internals call (eventually multiple) matching oracles output by `LargeMatching`. We prove that essentially all these oracles have query time $O(n^{1+\varepsilon})$ for some small $\varepsilon > 0$. A careful analysis is required to show that we can build the oracles at iteration $i + 1$ using the oracles at iteration i without blowing up their complexity.

Paper organization. In Section 3 we define some fundamental objects that we will use throughout the paper. In Section 4 we present a template algorithm to be implemented in sublinear time, and prove its correctness. In Section 5 we implement the template algorithm in sublinear time. In Section 6 we put everything together and prove the main theorems stated in the introduction.

3 Preliminaries

We use the notation $[a, b] := \{a \dots b-1\}$, $[b] = [0, b]$, and $(a \pm b) := [a-b, a+b]$ meaning that $c \cdot (a \pm b) = (ac \pm bc)$. We denote our undirected bipartite graph with $G = (V, E)$, and the bipartition is given by $V = V_0 \cup V_1$. Our original graph is complete and for each $(u, v) \in V_0 \times V_1$ we denote with $c(u, v)$ the cost of the edge (u, v). We stress that none of our algorithms require $c(\cdot, \cdot)$ to be a metric. Given a matching M we denote its combined cost with $c(M)$. For each $u \in V$ we say that $u = \mathtt{mate}_M(v)$ iff $(u, v) \in M$. When the matching M is clear from the context we denote with F the set of unmatched (or *free*) vertices, and set $F_i := F \cap V_i$ for $i = 0, 1$.

When we say that an algorithm runs in time t we mean that both its computational complexity and the number of queries to the cost matrix $c(\cdot, \cdot)$ are bounded by t. The computational complexity of our algorithms is always (asymptotically) equivalent to their query complexity, so we only analyse the latter. All our guarantees in this work hold with high probability.

Definition 3.1 (Augmenting paths)**.** *Given a matching M over $G = (V, E)$ we say that $P = (v_0, v_1 \dots v_{2\ell+1})$ is an augmenting path w.r.t. M if $(v_{2i}, v_{2i+1}) \in E \backslash M$ for each $i = 0 \dots \ell$ and $(v_{2j+1}, v_{2j+2}) \in M$ for each $j = 0 \dots \ell - 1$. When we say that we augment M w.r.t. P we mean that we set $M \leftarrow M \oplus P$, where $\oplus$ is the exclusive or.*

We use the same notion of 1-feasible potential as in [GT89].

Definition 3.2 (1-feasibility conditions)**.** *Given a potential $\varphi : V \longrightarrow \mathbb{Z}$ we say that it satisfies 1-feasibility conditions with respect to a matching M if the following hold.*

(i) For each $u \in V_0, v \in V_1$ $\varphi(u) + \varphi(v) \leqslant c(u, v) + 1$.

(ii) For each $(u, v) \in M$, $\varphi(u) + \varphi(v) = c(u, v)$.

Definition 3.3 (Eligibility Graph)**.** *We say that an edge (u, v) is eligible w.r.t. M if: $(u, v) \notin M$ and $\varphi(u) + \varphi(v) = c(u, v) + 1$ or; $(u, v) \in M$ and $\varphi(u) + \varphi(v) = c(u, v)$. We define the eligibility graph as the directed graph $G_{\mathcal{E}} = (V, E_{\mathcal{E}})$ that orients the eligible edges so that, for each eligible $(u, v) \in V_0 \times V_1$, we have $(u, v) \in E_{\mathcal{E}}$ if $(u, v) \notin M$ and $(v, u) \in E_{\mathcal{E}}$ if $(u, v) \in M$.*

Notice that, whenever a potential is 1-feasible w.r.t. M, then all edges in M are eligible.

Definition 3.4 (Forward Graph)**.** *We define the forward graph $G_F = (V, E_F)$ as the subgraph of the eligibility graph containing only edges from V_0 to V_1. That is, we remove all edges (v, u) such that $(u, v) \in M$.*

Now, we introduce two quite technical definitions, which provide us with approximate versions of the notion of "maximal set of node-disjoint augmenting paths" and "maximal forest".

Definition 3.5 ((k, ξ)-Quasi-Maximal Set of Node-Disjoint Augmenting Paths)**.** *Given a graph $G = (V, E)$ and a matching $M \subseteq E$ we say that a set $\mathcal{P}$ of augmenting paths of length at most k is a (k, ξ)-QMSNDAP if for any $\mathcal{Q}$ such that $\mathcal{Q} \cup \mathcal{P}$ is a set of node-disjoint augmenting paths of length $\leqslant k$ we have $|\mathcal{Q}| \leqslant \xi n$.*

Intuitively, $\mathcal{P}$ is a (k, ξ)-QMSNDAP if we can add only a few more node-disjoint augmenting paths of length $\leqslant k$ to $\mathcal{P}$ before it becomes a maximal. Next we introduce an approximate notion of "maximal forest" $\mathcal{F}$ in the eligibility graph $G_{\mathcal{E}}$ rooted in F_0. $\mathcal{F}$ is obtained starting from the vertices in F_0 and adding edges (in a way that we will specify later) so as to preserve the $\mathcal{F}$ has $|F_0|$ connected components and has no cycles. This construction will ensure that the connected component of our forest have small diameter and small size. We maintain that whenever $v \in V_1$ is added to $\mathcal{F}$, then $\texttt{mate}_M(v)$ is also added to $\mathcal{F}$. $\mathcal{F}$ is approximately maximal in the sense that the cut $(\mathcal{F}, V \backslash \mathcal{F})$ in $G_{\mathcal{E}}$ admits a small vertex cover.

Definition 3.6 ((k, δ)-Quasi-Maximal Forest)**.** *Given the eligibility graph $G_{\mathcal{E}} = (V, E_{\mathcal{E}})$ w.r.t. the matching M, and the set of vertices $F_0 \subseteq V_0$ we say that $\mathcal{F}$ is a (k, δ)-QMF rooted in F_0 if:*

1. *$F_0 \subseteq \mathcal{F}$*
2. *For each $v \in V_1 \cap \mathcal{F}$ we have $\texttt{mate}_M(v) \in \mathcal{F}$*
3. *For each $u \in \mathcal{F}$ there exists $v \in F_0$ at hop distance from u at most k.*
4. *Every connected component of $\mathcal{F}$ has size at most 2^k.*
5. *The edge set $E_{\mathcal{E}} \cap (\mathcal{F} \times V \backslash \mathcal{F})$ has a vertex cover of size at most δn.*

Now, we introduce a few results from past work on sublinear-time maximum caridnality matching. The following theorem, which is the main technical contribution of [BKS23a], states that we can compute a εn-additive approximation of the size of a maximum-cardinality matching in strongly sublinear time.

Theorem 5 (Theorem 1.3, [BKS23a])**.** *There is a randomized algorithm that, given the adjacency matrix of a graph G, in time $n^{2-\Omega_\varepsilon(1)}$ computes with high probability a $(1, \varepsilon n)$-approximation $\tilde{\mu}$ of $\mu(G)$. After that, given a vertex v, the algorithm returns in $n^{1+f(\varepsilon)}$ time an edge $(v, v') \in M$ or $\perp$ if $v \notin V(M)$ where M is a fixed $(1, \varepsilon n)$-approximate matching, where f is an increasing function such that $f(\varepsilon) \to 0$ when $\varepsilon \to 0$.*

The algorithm in Theorem 5 does not exactly output a matching, but rather a *matching oracle*. Namely, it outputs a data structure that stores a matching M implicitly. We formalize the notion of matching oracle below.

Definition 3.7 (Matching Oracle)**.** *Given a matching M, we define the matching oracle $\texttt{match}_M(\cdot)$ as a data structure such that $\texttt{match}_M(u) = v$ if $(u, v) \in M$ and $\texttt{match}_M(u) = \perp$ otherwise. Throughout the paper we denote with t_M the time complexity of $\texttt{match}_M(\cdot)$.*

Similarly to matching oracles, we make use of membership oracles $\texttt{mem}_A(\cdot)$ and potential oracles $\texttt{eval}_\varphi(\cdot)$ where $A \subseteq V$ and φ is a potential function defined on V. As expected, $\texttt{mem}_A(u)$ returns $\mathbb{1}_{u \in A}$ and $\texttt{eval}_\varphi(u)$ returns $\varphi(u)$. We denote their running time with t_A and t_φ respectively. Now, we recall two theorems from [BKS23a] that constitutes fundamental ingredients of our sublinear-time algorithm for minimum-weight matching.

Theorem 6 roughly says that, in sublinear time, we can find a matching oracle for a size-$\Omega(n)$ matching, whenever a size-$\Omega(n)$ matching exists.

Theorem 6 (Essentially Theorem 4.1, [BKS23a]). *Let $G = (V, E)$ be a graph, $A \subseteq V$ be a vertex set. Suppose that we have access to adjacency matrix of G and an A-membership oracle $\texttt{mem}_A$ with t_A query time. We are given as input a sufficiently small $\varepsilon > 0$ and $\delta_{\text{in}} > 0$.*

There exists an algorithm `LargeMatching`$(G, A, \varepsilon, \delta_{\text{in}})$ *that preprocesses G in $\tilde{O}_{\delta_{\text{in}}}((t_A+n)\cdot n^{1-\varepsilon})$ time and either return $\perp$ or construct a matching oracle $\texttt{match}_M(\cdot)$ for a matching $M \subset G[A]$ of size at least $\delta_{\text{out}} n$ where $\delta_{\text{out}} = \frac{1}{2000}\delta_{\text{in}}^5$ that has $\tilde{O}_{\delta_{\text{in}}}((t_A + n)n^{4\varepsilon})$ worst-case query time. If $\mu(G[A]) \geqslant \delta_{\text{in}} n$, then $\perp$ is not returned. The guarantee holds with high probability.*

Theorem 7 roughly says that, in sublinear time, we can increase the size of our current matching (oracle) by $\Omega(n)$, whenever there are $\Omega(n)$ short augmenting paths.

Theorem 7 (Essentially Theorem 5.2, [BKS23a]). *Fix two constants $k, \gamma > 0$. For any sufficiently small $\varepsilon_{in} > 0$, there exists $\varepsilon_{out} = \Theta_{k,\gamma}(\varepsilon_{in})$ such that the following holds. There exists an algorithm* `Augment`$(G, M^{in}, k, \gamma, \varepsilon_{in})$ *that makes $O_{k,\gamma}(1)$ calls to* `LargeMatching` *which take $\tilde{O}_{k,\gamma}\left(n^{2-\varepsilon_{in}}\right)$ time in total. Further, either it returns an oracle $\texttt{match}_{M^{out}}(\cdot)$ with query time $\tilde{O}_{k,\gamma}(n^{1+\varepsilon_{out}})$, for some matching M^{out} in G of size $|M^{out}| \geqslant |M^{in}| + \Theta_{k,\gamma}(1) \cdot n$ (we say that it "succeeds" in this case), or it returns* FAILURE. *Finally, if the matching M^{in} admits a collection of $\gamma \cdot n$ many node-disjoint length $(2k+1)$-augmenting paths in G, then the algorithm succeeds whp.*

Theorem 7 differs from Theorem 5.2 in [BKS23a] in that it specifies that the only way `Augment` accesses the graph is through `LargeMatching`. We will use this property crucially to prove Lemma 5.2.

4 A Template Algorithm

In this section we study min-weight matching with integral small costs $c(\cdot, \cdot) \in [1, C]$, where C is constant. We will see how to lift this restriction in Section 6. Algorithm 1 gives a template algorithm realising Theorem 4 that assumes we can implement certain subroutines; in Section 5 we will see how to implement these subroutines in sublinear time.

Comparison with Gabow-Tarjan. Intuitively, our template algorithm implements the Gabow-Tarjan algorithm [GT89] for a fixed scale in an approximate fashion. Indeed, instead of finding a maximal-set of node-disjoint augmenting paths we find a (k, ξ)-QMSNDAP and instead of growing a forest in the eligibility graph we grow a (k, δ)-QMF. See Figure 1 for a representation of step 1 and step 2.

Analysis. Here we analyse Algorithm 1 and show that it satisfies the following theorem.

Theorem 8. *Fix a constant $\gamma > 0$. Suppose that we have adjacency-matrix access to the bipartite graph $G = (V_0 \cup V_1, E)$ and random access to the cost function $c : E \to [1, C]$, with $C = O(1)$. Then, with high probability, Algorithm 1 returns $\hat{c}$ such that*

$$c(M^{1-\gamma}) \leqslant \hat{c} \leqslant c(M^{OPT})$$

where $M^{1-\gamma}$ is a min-weight matching of size $(1-\gamma)n$ and M^{OPT} is a min-weight matching of size n.

To prove Theorem 8, we need a series of technical lemmas.

Algorithm 1 Template Algorithm.

Input: A bipartite graph $G = (V_0 \cup V_1, E)$ and a cost function $c : E \to [1, C]$.
Set $T = C/\gamma^3$, $\xi = \frac{\gamma}{Tk2^k}$, $\delta = \frac{\gamma}{T}$ and $k = 6000(2T+1)^{10}/\delta^5$.
Initialize $M \leftarrow \varnothing$ and $\varphi(v) \leftarrow 0$ for each $v \in V$.
Let F_0 denote the set of M-unmatched vertices in V_0.
For each $e \in E$ update $c(e) \leftarrow c(e)/\gamma$ (this is implemented lazily).
Execute the following two steps for T iterations:

- *Step 1.* Find a (k, ξ)-QMSNDAP $\mathcal{P}$ in the eligibility graph $G_{\mathcal{E}}$. Augment M w.r.t. paths in $\mathcal{P}$. Set $\varphi(v) \leftarrow \varphi(v) - 1$ for each $v \in V_1 \cap \bigcup_{P \in \mathcal{P}} P$.
- *Step 2.* Find a (k, δ)-QMF$\mathcal{F}$ rooted in F_0 in the eligibility graph $G_{\mathcal{E}}$. Set $\varphi(u) \leftarrow \varphi(u) + 1$ for each $u \in V_0 \cap \mathcal{F}$ and $\varphi(v) \leftarrow \varphi(v) - 1$ for each $v \in V_1 \cap \mathcal{F}$.

Sample a set S of $O_{\gamma,C}(\log n)$ edges in M with replacement.
Discard the $3\gamma|S|$ edges with highest costs and let Σ be the sum of costs of remaining edges.
Output: $\hat{c} = \frac{n}{|S|}\Sigma$.

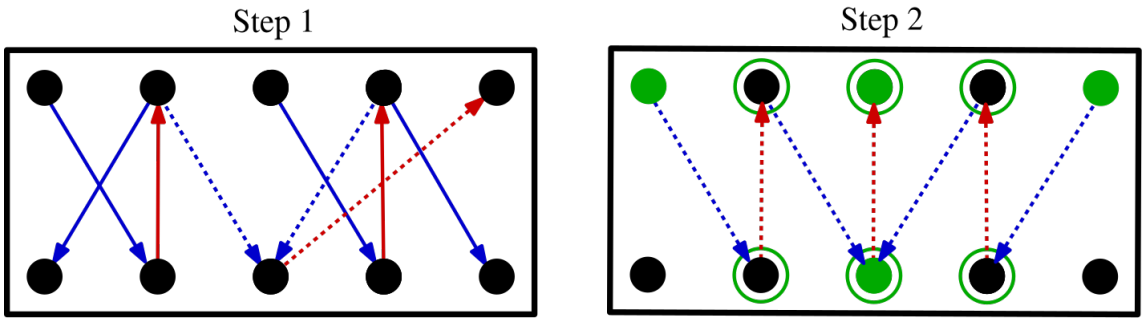

Figure 1: We color the edges of M in red and the edges of $T \backslash M$ in blue. On the left we have an example of step 1. Solid edges represent paths in the QMSNDAP $\mathcal{P}$ that we augment M along in step 1. On the right we have an example of step 2. All vertices colored or circled in green belong to the QMF $\mathcal{F}$. Circles help us visualize the implementation of step 2, described in Section 5. In Algorithm 3 $\mathcal{F}$ is built sequentially, where each iteration (lines 1-5) adds some edges to $\mathcal{F}$. At first, only the non-circled green vertices belong to $\mathcal{F}$. The first step adds the green-circled black edges, and the second step adds the green-circled green edges.

Proof Roadmap. The proof of Theorem 8 goes as follows. We prove that, after T iterations, all free vertices in F_0 have potential T. On the other hand, the majority of free vertices in F_1 have potential 0. We call *spurious* the free vertices in F_1 with non-zero potential and we show there are only few of them. Then, (roughly) we look at the final matching M generated by Algorithm 1 and a perfect matching M' and consider the graph G' having $M \oplus M'$ as its set of edges. G' can be partitioned into cycles and augmenting paths. Each augmenting path starts in a free vertex in F_0 and ends in a free vertex in F_1. If the 1-feasibility conditions are satisfied by all edges, then computing a certain function of potentials along an augmenting path and combining the results for

all augmenting paths yields an upper bound on the total number of free vertices. Unfortunately, not all edges satisfy the 1-feasibility constraints. We fix this by finding a small vertex cover of the 1-unfeasible edges. We say that such cover a suitable set of *broken* vertices. Ignoring spurious and broken vertices is sufficient to make our argument work.

Lemma 4.1. *After $t \in [T+1]$ iterations we have $\varphi(u) = t$ for each $u \in F_0$.*

Proof. After $t = 0$ iterations, we have $\varphi(u) = 0$ for each $u \in V$. First, we notice that the set of unmatched (or free) vertices F only shrinks over time, and so does F_0. Moreover, at each iteration we increase the potential of free vertices in F_0 by 1. □

Define the set S of $v \in F_1$ such that $\varphi(v) \neq 0$ as the set of *spurious* vertices.

Lemma 4.2. *After T iterations we have at most γn* spurious *vertices.*

Proof. We prove that at each iteration we increase the number of spurious vertices by at most $\gamma n/T$. A vertex cannot become spurious in Step 1. Indeed, in Step 1 we only decrease the potential of matched vertices. If a vertex $v \in F_1$ becomes spurious in Step 2, it means that there exists an augmenting path P from some $u \in F_0$ to v contained in a connected component of $\mathcal{F}$. Let $\mathcal{Q}$ be such that $\mathcal{Q} \cup \mathcal{P}$ is a maximal set of node-disjoint augmenting paths of length $\leqslant k$. By Definition 3.5 we have $|\mathcal{Q}| \leqslant \xi n$. Define the set of *forgotten* vertices as $\bigcup_{Q \in \mathcal{Q}} Q$. Thanks to item 3 in Definition 3.6, the path from u to w has length $\leqslant k$, thus P has length at most k. Recall that P is an augmenting path w.r.t. the graph obtained augmenting M along $\mathcal{P}$ at the end of Step 1. Therefore, P intersects a path in $\mathcal{Q} \cup \mathcal{P}$.

We now argue that that P cannot intersect $P' \in \mathcal{P}$. Suppose by contradiction that it does. Let $P = (P_0 \dots P_\ell)$ and $P' = (P'_0 \dots P'_{\ell'})$. Let P_s be the first (w.r.t. the order induced by P) node where P and P' intersect. We first rule out the case that s is even: for $s = 0$, $P_0 = u \in F_0$ implies that u did not belong to an augmenting path P' in Step 1. Moreover, for $s = 2i > 0$ if $P_{2i} = P'_j$ then $P_{2i-1} = \texttt{mate}_M(P_{2i}) \in \{P'_{j-1}, P'_{j+1}\}$, where M is the matching obtained at the end of Step 1. Now suppose that s is odd, and hence $P_s \in V_1 \cap P'$. Then $\varphi(P_s)$ is decreased by 1 at the end of Step 1, hence no edge outside of M incident to P_s is eligible in Step 2.

Thus, P must intersect a path in $\mathcal{Q}$. On the other hand $\bigcup_{Q \in \mathcal{Q}} Q$ contains at most $k\xi n$ vertices, so at most $k\xi n$ connected component of $\mathcal{F}$ contain a forgotten edge. Moreover, by item 4 of Definition 3.6 every connected component of $\mathcal{F}$ has size at most 2^k, thus at most $k\xi n 2^k = \gamma/T$ vertices become spurious. □

We say that $B \subseteq V$ is a suitable set of *broken* vertices if all $(u, v) \in (V_0 \backslash B) \times (V_1 \backslash B)$ are 1-feasible.

Lemma 4.3. *After T iterations, there exists a suitable set of broken vertices of size at most γn.*

Proof. First, we prove that every edge $(u, v) \in V_0 \times V_1$, which is 1-feasible at the beginning of Step 1, is also 1-feasible at the end of Step 1. Suppose that (u, v) becomes 1-unfeasible in Step 1. Let M and M' be the matching at the beginning and at the end of Step 1 respectively. Potentials only decrease in Step 1, so in order for (u, v) to become 1-unfeasible w.r.t. M' we must have $(u, v) \in M'$. Moreover, we decrease the potential of v only if $(u, v) \in P$, for some augmenting path P. Thus, at the beginning of Step 1 we had $\varphi(u) + \varphi(v) = c(u, v) + 1$, which implies $\varphi(u) + \varphi(v) = c(u, v)$ at the end of Step 1, thus (u, v) is 1-feasible w.r.t. M', contradiction.

Now, we grow a set of suitable broken vertices B. We initialize $B = \varnothing$ and show that each iteration Step 2 increases the size of B by at most $\gamma n/T$. If $(u, v) \in V_0 \times V_1$ is 1-feasible at the

beginning of Step 2 and becomes 1-unfeasible in Step 2, then we must have $u \in \mathcal{F}$ and $v \notin \mathcal{F}$. Indeed, by item 2 in Definition 3.6 if $(u,v) \in M'$ then either both u and v belong to $\mathcal{F}$ or neither of them does. This ensures that the sum of their potentials is unchanged. Else, if $(u,v) \notin M'$ then in order for it to violate 1-feasibility we must increase $\varphi(u)$ by one and not decrease $\varphi(v)$, and this happens only if $u \in \mathcal{F}$ and $v \notin \mathcal{F}$. Item 5 in Definition 3.6 ensures that there exists a vertex cover $U \subseteq V$ for the set of new 1-unfeasible edges with $|U| \leqslant \delta n = \gamma n/T$. We update $B \leftarrow B \cup U$. Thus, after T iterations we have $|B| \leqslant \gamma n$. □

Lemma 4.4. *After T iterations of template algorithm we have have $|F_0| = |F_1| \leqslant 4\gamma n$.*

Proof. Denote with M the final matching obtained by Algorithm 1. Let B be a suitable set of broken vertices with $|B| \leqslant \gamma n$, as in Lemma 4.3. Partition $B = B_M \cup B_F$, where $B_F := B \cap F$ is the set of unmatched vertices in B and B_M is the set of matched vertices in B. Consider the set B'_M of vertices currently matched to vertices in B_M, $B'_M = \{\mathtt{mate}_M(b) \mid b \in B_M\}$. We have $|(B_M \cup B'_M) \cap V_0| = |(B_M \cup B'_M) \cap V_1| \leqslant \gamma n$. Let S be the set of spurious vertices and recall that $|S| \leqslant \gamma n$ by Lemma 4.2. Let $S' \subseteq F_0 \backslash B_F$ such that $|S' \cup (V_0 \cap B_F)| = |S \cup (V_1 \cap B_F)|$. This implies that $|S' \cup (V_0 \cap B_F)| \leqslant |S| + |B| \leqslant 2\gamma n$. Define $A_0 := V_0 \backslash (B \cup B'_M \cup S')$ and $A_1 := V_1 \backslash (B \cup B'_M \cup S)$ and notice that they have the same size. Define $A = A_0 \cup A_1$. Let M' be a perfect matching over A.

The graph $G_A = (A, M \oplus M')$ contains exactly $\ell := |F_0 \cap A_0| = |F_1 \cap A_1|$ node-disjoint paths $P_1 \dots P_\ell$ where P_i starts in $f_0^{(i)} \in F_0 \cap A_0$ and ends in $f_1^{(i)} \in F_1 \cap A_1$. We define the *value* of a path P as

$$\mathcal{V}(P) = \sum_{(u,v) \in M' \cap P} (c(u,v) + 1) - \sum_{(u,v) \in M \cap P} c(u,v).$$

By 1-feasibility of φ we have

$$\mathcal{V}(P_i) \geqslant \sum_{(u,v) \in M' \cap P} (\varphi(u) + \varphi(v)) - \sum_{(u,v) \in M \cap P} (\varphi(u) + \varphi(v)) = \varphi\left(f_0^{(i)}\right) - \varphi\left(f_1^{(i)}\right) = T$$

where the last equality holds by definition of (non-)spurious vertices and Lemma 4.1. Then, we have $Cn \geqslant n + c(M') \geqslant \sum_i^\ell \mathcal{V}(P_i) \geqslant \ell T$. Thus, $\ell \leqslant Cn/T = \gamma n$ and

$$|F_1| = |F_0| \leqslant |F_0 \cap A| + |(B_M \cup B'_M) \cap V_0| + |S' \cup (V_0 \cap B_F)| \leqslant 4\gamma n.$$

□

Let φ be the potential at the end of the execution of Algorithm 1. Denote with $M^{\mathtt{ALG}}$ the final matching obtained by Algorithm 1 and with $M^{\mathtt{OPT}}$ a min-weight perfect matching. Given a matching M, we denote with $M_{[\alpha]}$ the matching obtained from M by removing the αn edges with highest cost.

Lemma 4.5. *We have $c(M^{ALG}_{[2\gamma]}) \leqslant c(M^{OPT})$.*

Proof. Let $M^{\mathtt{ALG}}_{\setminus B}$ be the matching obtained from $M^{\mathtt{ALG}}$ by removing all edges incident to vertices in B. Since $|B| \leqslant \gamma n$ we have $c(M^{\mathtt{ALG}}_{[\gamma]}) \leqslant c(M^{\mathtt{ALG}}_{\setminus B})$. Notice that all edges in $M^{\mathtt{ALG}}_{\setminus B}$ are 1-feasible. For each $(u,v) \in M^{\mathtt{ALG}}_{\setminus B}$ we have $c(u,v) = \varphi(u) + \varphi(v)$ and for each $(u,v) \in M^{\mathtt{OPT}}$ we have $\varphi(u) + \varphi(v) \leqslant c(u,v) + 1$. Thus,

$$c(M^{\mathtt{ALG}}_{[\gamma]}) \leqslant c(M^{\mathtt{ALG}}_{\setminus B}) \leqslant \sum_{u \in V} \varphi(u) = \sum_{(u,v) \in M^{\mathtt{OPT}}} \varphi(u) + \varphi(v) \leqslant \sum_{(u,v) \in M^{\mathtt{OPT}}} c(u,v) + 1 = n + c(M^{\mathtt{OPT}}).$$

Now, it is sufficient to notice that, since all edges have costs in $[1/\gamma, C/\gamma - 1]$, removing any γn edges from $M^{\mathtt{ALG}}_{[\gamma]}$ decreases its cost by n. Thus, $c(M^{\mathtt{ALG}}_{[2\gamma]}) \leqslant c(M^{\mathtt{OPT}})$. □

Now, we are ready to prove Theorem 8.

Proof of Theorem 8. Thanks to Lemma 4.5, we know that $c(M^{\mathtt{ALG}}_{[2\gamma]}) \leqslant c(M^{\mathtt{OPT}})$. Moreover, by Lemma 4.4 we have $|M^{\mathtt{ALG}}| = n - |F_0| \geqslant (1-4\gamma)n$, thus defining $M^{1-8\gamma}$ as the min-weight matching of size $(1-8\gamma)n$, we have $c(M^{1-8\gamma}) \leqslant c(M^{\mathtt{ALG}}_{[4\gamma]})$. We are left to prove that the estimate $\hat{c} = \frac{n}{|S|}\Sigma$ returned by Algorithm 1 satisfies $c(M^{\mathtt{ALG}}_{[4\gamma]}) \leqslant \hat{c} \leqslant c(M^{\mathtt{ALG}}_{[2\gamma]})$. Let S and Σ be defined as in Algorithm 1 and let w be maximum such that $3\gamma|S|$ edges in S have cost $\geqslant w$. If $\alpha_w \cdot n$ is the number of edges in $M^{\mathtt{ALG}}$ that cost $\geqslant w$, then using standard Chernoff Bounds arguments we have that, whp, $|\alpha_w - 3\gamma| \leqslant \gamma^2/C$. From now on we condition on this event. Notice that $\frac{(1-\alpha_w)n}{(1-3\gamma)|S|}\Sigma$ is an unbiased estimator of $c(M^{\mathtt{ALG}}_{[\alpha_w]})$. Moreover, since all costs are in $[1/\gamma, C/\gamma]$, then $O_{\gamma,C}(\log n)$ samples are sufficient to have $\frac{(1-\alpha_w)n}{(1-3\gamma)|S|}\Sigma$ concentrated, up to a factor $(1 \pm \frac{\gamma^2}{C})$, around $c(M^{\mathtt{ALG}}_{[\alpha_w]})$. Hence, assuming that γ is sufficiently small, we have

$$\frac{n}{|S|}\Sigma \in (1 \pm 3\gamma^2/C) \cdot c(M^{\mathtt{ALG}}_{[\alpha_w]}) \subseteq c(M^{\mathtt{ALG}}_{[\alpha_w]}) \pm 3\gamma n$$

where the last containment relation holds because all costs are $\leqslant C/\gamma$ and so $c(M^{\mathtt{ALG}}_{[\alpha_w]}) \leqslant Cn/\gamma$. Since all costs are $\geqslant 1/\gamma$ we have $c(M^{\mathtt{ALG}}_{[\alpha_w+3\gamma^2]}) \leqslant c(M^{\mathtt{ALG}}_{[\alpha_w]}) - 3\gamma n$ and $c(M^{\mathtt{ALG}}_{[\alpha_w-3\gamma^2]}) \geqslant c(M^{\mathtt{ALG}}_{[\alpha_w]}) + 3\gamma n$. Thus, picking γ small enough to have $\alpha_w \pm 3\gamma^2 \subseteq [2\gamma, 4\gamma]$ we have

$$c(M^{\mathtt{ALG}}_{[4\gamma]}) \leqslant \frac{n}{|S|}\Sigma \leqslant c(M^{\mathtt{ALG}}_{[2\gamma]}).$$

Therefore, we have $c(M^{1-8\gamma}) \leqslant \hat{c} \leqslant c(M^{\mathtt{OPT}})$ and rescaling γ gives exactly the desired result. □

Observation 9. As in the proof of Theorem 8, define w as the maximum value such that there are at least $3\gamma|S|$ edges with cost $\geqslant w$ in S and define α_w such that exactly $\alpha_w \cdot n$ edges in M have cost $\geqslant w$. We have, whp, $|\alpha_w - 3\gamma| \leqslant \gamma^2/C$, thus for γ small enough $c(M^{\mathtt{ALG}}_{[\alpha_w]}) \leqslant c(M^{\mathtt{ALG}}_{[2\gamma]}) \leqslant c(M^{\mathtt{OPT}})$ and (up to rescaling γ) $|M^{\mathtt{ALG}}_{[\alpha_w]}| \geqslant (1-\gamma)n$. Moreover, given an edge $e \in M^{\mathtt{ALG}}$ we can decide whether $e \in M^{\mathtt{ALG}}_{[\alpha_w]}$ simply by checking $c(e) \leqslant w$.

5 Implementing the Template in Sublinear Time

In this section we explain how to implement *Step 1* and *Step 2* from the template algorithm in sublinear time.

5.1 From Potential Oracles to Membership Oracles

Throughout this section, we would like to apply Theorem 6 and Theorem 7 on the eligibility graph $G_{\mathcal{E}} = (V, E_{\mathcal{E}})$ and forward graph $G_F = (V, E_F)$. However, we do not have random access to the adjacency matrix of these graphs. Indeed, to establish if $(u, v) \in V_0 \times V_1$ is eligible we need to check the condition $\varphi(u) + \varphi(v) = c(u, v) + 1$ (or $\varphi(u) + \varphi(v) = c(u, v)$). However, we will see that the potential $\varphi(\cdot)$ requires more than a single query to be evaluated. Formally, we assume that we have

a potential oracle $\texttt{eval}_\varphi(\cdot)$ that returns the value of $\varphi(\cdot)$ in time t_φ. Whenever checking whether (u,v) is an edge of G_F ($G_\mathcal{E}$) requires to evaluate a condition of the form $\varphi(u)+\varphi(v)=c(u,v)+1$ (or $\varphi(u)+\varphi(v)=c(u,v)$) we say that we have *potential oracle access* to the adjacency matrix of G_F ($G_\mathcal{E}$) with potential oracle time t_φ. We can think of t_φ as $\tilde{O}(n^{1+\varepsilon})$ and we will later prove that this is (roughly) the case.

Potential functions with constant-size range. If our potential function $\varphi : V \to \mathcal{R}$ has range size $|\mathcal{R}| \leqslant R$ then we say that it is an R-potential. If the eligibility (forward) graph is induced by R-potentials for $R = O(1)$ we can rephrase Theorem 6 and Theorem 7 to work with potential oracle access, without any asymptotic overhead. The following theorem is an analog of Theorem 6 for forward graphs.

Lemma 5.1. *Let $G_F = (V, E_F)$ be a forward graph w.r.t the R-potential φ, let $A \subseteq V$ be a vertex set. Suppose we have a potential oracle $\texttt{eval}_\varphi$ with oracle time t_φ and an membership oracle $\texttt{mem}_A$ with t_A query time. We are given as input constants $0 < \varepsilon \leqslant 0.2$ and $\delta_{\text{in}} > 0$.*

There exists an algorithm $\texttt{LargeMatchingForward}(\varphi, A, \delta_{\text{in}})$ that preprocesses G_F in $\tilde{O}_R((t_A + t_\varphi + n) \cdot n^{1-\varepsilon})$ time and either returns $\perp$ or constructs a matching oracle $\texttt{match}_M(\cdot)$ for a matching $M \subset G_F[A]$ of size at least $\delta_{\text{out}} n$ where $\delta_{\text{out}} = \frac{1}{2000 \cdot R^{10}} \delta_{\text{in}}^5 = \Theta_{\delta_{\text{in}},R}(1)$ that has $\tilde{O}_R((t_A + t_\varphi + n)n^{4\varepsilon})$ worst-case query time. If $\mu(G_F[A]) \geqslant \delta_{\text{in}} n$, then $\perp$ is not returned. The guarantee holds with high probability.

Proof. Without loss of generality, we assume that φ takes values in $[R]$. Suppose that $G_F[A]$ has a matching of size $\delta_{\text{in}} n$. We partition the edges $E_F[A] = E_F \cap (A \times A)$ into R^2 sets $E_{0,0} \dots E_{R-1,R-1}$ such that $(u,v) \in E_{i,j}$ iff $\varphi(u) = i$ and $\varphi(v) = j$. Then, there exist $i, j \in [R]$ such that $G_{i,j} = (V, E_{i,j})$ has a matching of size $\delta_{\text{in}} n / R^2$. Moreover, once we restrict ourselves to $G_{i,j}$, each edge query $(u,v) \in E_{i,j}$ becomes much easier. Indeed, we just need to establish if $i + j = c(u,v) + 1$. In order to restrict ourselves to $G_{i,j}$ it suffices to set $A' = A \cap (\varphi^{-1}(\{i\}) \times \varphi^{-1}(\{j\}))$. Then the membership oracle $\texttt{mem}_{A'}$ runs in time $O(t_A + t_\varphi)$. Hence, using Theorem 6 we can find a matching of size $\delta_{\text{out}} n$, where $\delta_{\text{out}} = \frac{1}{2000 \cdot R^{10}} \delta_{\text{in}}^5$. Algorithmically, we run the algorithm from Theorem 6 R^2 times (once for each pair (i,j)) and halt as soon as the algorithm does not return $\perp$. □

The following is an analog of Theorem 7 for eligibility graphs.

Lemma 5.2. *Let $\varepsilon_{in} > 0$ be a sufficiently small constant. Let $\alpha_{k,\gamma}$ and $\beta_{k,\gamma}$ be constants that depend on k and γ and set $\varepsilon_{out} := \alpha_{k,\gamma} \cdot \varepsilon_{in}$. We have an R-potential oracle $\texttt{eval}_\varphi$ with running time $t_\varphi = \tilde{O}(n^{1+\varepsilon_{in}})$, a matching oracle $\texttt{match}_{M^{in}}$ with running time $t_{M^{in}} = \tilde{O}(n^{1+\varepsilon_{in}})$ and an eligibility graph $G_\mathcal{E} = (V, E_\mathcal{E})$ w.r.t. φ and M^{in}.*

There exists an algorithm $\texttt{AugmentEligible}(\varphi, M^{in}, k, \gamma, \varepsilon_{in})$ that runs in $\tilde{O}_{k,\gamma,R}\left(n^{2-\varepsilon_{in}}\right)$ time. Further, either it returns an oracle $\texttt{match}_{M^{out}}(\cdot)$ with query time $\tilde{O}_{k,\gamma,R}(n^{1+\varepsilon_{out}})$, for some matching M^{out} in $G_\mathcal{E}$ of size $|M^{out}| \geqslant |M^{in}| + \beta_{k,\gamma} \cdot n$ (we say that it "succeeds" in this case), or it returns $\perp$. Finally, if the matching M^{in} admits a collection of $\gamma \cdot n$ many node-disjoint augmenting paths with length $\leqslant k$ in $G_\mathcal{E}$, then the algorithm succeeds whp.

Proof. We derive Lemma 5.2 combining Theorem 7 and Lemma 5.1. First, we notice that Theorem 7 says that the algorithm succeeds (whp) whenever there are $\gamma' n$ node-disjoint augmenting paths (NDAP) with length *exactly* $2k' + 1$, while Lemma 5.2 has the weaker requirement that there are at least γn NDAP of length $\leqslant k$. A simple reduction is obtained invoking Theorem 7 with $\gamma' = \gamma / k$ for all k' such that $2k' + 1 \leqslant k$ (notice that all augmenting paths have odd length). In this way,

if there exists a collection of γn NDAP of length $\leqslant k$ then there exists a $k' \leqslant (k-1)/2$ such that we have $\gamma' n$ NDAP of length *exactly* $2k'+1$. All guarantees are preserved since we consider both γ and k constants. Now, we are left to address the fact that we do not have random access to the adjacency matrix of $G_{\mathcal{E}}$, but rather potential oracle access.

We notice that, according to Theorem 7, the implementation of `Augment` from [BKS23a] never makes any query to the adjacency matrix besides those performed inside `LargeMatching`. Moreover, Lemma 5.1 implies that `LargeMatchingForward` is not asymptotically slower than `LargeMatching` as long as $R = O(1)$. □

Finally, we observe that in Algorithm 1 each potential is increased (or decreased) at most $T = O_{C,\gamma}(1)$ times. Hence, φ is a R-potential for $R = 2T + 1 = O_{C,\gamma}(1)$. Thus, we can consider R a constant when applying Lemma 5.1 or Lemma 5.2.

5.2 Implementing Step 1

In this subsection we implement Step 1 from the template algorithm in sublinear time. Here we assume that we have at our disposal a potential oracle $\texttt{eval}_{\varphi^{\texttt{in}}}$ running in time $t_{\varphi^{\texttt{in}}} = \tilde{O}(n^{1+\varepsilon_{\texttt{in}}})$ and a matching oracle $\texttt{match}_{M^{\texttt{in}}}$ with running time $t_{M^{\texttt{in}}} = \tilde{O}(n^{1+\varepsilon_{\texttt{in}}})$. We will output a potential oracle $\texttt{eval}_{\varphi^{\texttt{out}}}$ running in time $t_{\varphi^{\texttt{out}}} = \tilde{O}(n^{1+\varepsilon_{\texttt{out}}})$ and a matching oracle $\texttt{match}_{M^{\texttt{out}}}$ with running time $t_{M^{\texttt{out}}} = \tilde{O}(n^{1+\varepsilon_{\texttt{out}}})$. We show that there exists a (k,ξ)-QMSNDAP $\mathcal{A}$ such that: the matching $M^{\texttt{out}}$ is obtained from $M^{\texttt{in}}$ by augmenting it along all paths in $\mathcal{A}$; $\varphi^{\texttt{out}}$ is obtained from $\varphi^{\texttt{in}}$ by subtracting 1 to $\varphi^{\texttt{in}}(v)$ for each $v \in V_1 \cap \bigcup_{P \in \mathcal{A}} P$.

Algorithm 2 Implementation of Step 1.

Set k and γ as in Algorithm 1.
Initialize $M \leftarrow M^{\texttt{in}}$ and $\varepsilon \leftarrow \varepsilon_{\texttt{in}}$.
Repeat until `AugmentEligible` returns $\perp$:

1. Let $\texttt{AugmentEligible}(G_{\mathcal{E}}, M, k, \gamma, \varepsilon)$ return $\texttt{match}_{M'}$ with running time $t_{M'} = \tilde{O}(n^{1+\varepsilon'})$.
2. Update $M \leftarrow M'$ and $\varepsilon \leftarrow \varepsilon'$.

Set $M^{\texttt{out}} \leftarrow M$ and $\varepsilon_{\texttt{out}} \leftarrow \varepsilon$.

Implement $\texttt{eval}_{\varphi^{\texttt{out}}}(u)$ as follows:

- If $u \in V_0$, return $\texttt{eval}_{\varphi^{\texttt{in}}}$.
- Else, $u \in V_1$, set $v \leftarrow \texttt{match}_{M^{\texttt{out}}}(u)$.
- If v `==` $\perp$, return $\texttt{eval}_{\varphi^{\texttt{in}}}(u)$.
- Else, we have $(u,v) \in M^{\texttt{out}}$:
  - If $c(u,v) + 1$ `==` $\texttt{eval}_{\varphi^{\texttt{in}}}(u) + \texttt{eval}_{\varphi^{\texttt{in}}}(v)$, return $\texttt{eval}_{\varphi^{\texttt{in}}}(u) - 1$.
  - Else, return $\texttt{eval}_{\varphi^{\texttt{in}}}(u)$.

Analysis. First, we observe that the algorithm above correctly implements the template, with high probability (all our statements henceforth hold whp). Initialize $\mathcal{A} \leftarrow \varnothing$. For each run of $\texttt{AugmentEligible}(G, M, k, \gamma, \varepsilon)$ we decompose $M \oplus M'$ into a set of augmenting paths $\mathcal{P}$ and a set of alternating cycles $\mathcal{C}$ and we set $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{P}$. When $\texttt{AugmentEligible}(G, M, k, \gamma, \varepsilon)$ returns $\perp$ it means (by Lemma 5.2) that there are at most γn node-disjoint augmenting paths of length $\leqslant k$ that do not intersect $\bigcup \mathcal{A}$. Hence, $\mathcal{A}$ is a (k, ξ)-QMSNDAP . Clearly, $\texttt{match}_{M^{\texttt{out}}}$ implements the matching obtained from $M^{\texttt{in}}$ by augmenting along the paths in $\mathcal{A}$.

To see that the implementation of $\texttt{eval}_{\varphi^{\texttt{out}}}(u)$ is correct it is sufficient to notice that in the template algorithm we decrement $\varphi(u)$ iff: (i) $u \in V_1$, and (ii) there exists an augmenting path $P \in \mathcal{A}$ intersecting u. Since every node belongs to at most one path in $\mathcal{A}$ then u is matched in $M^{\texttt{out}}$ and $(u, v) \in M^{\texttt{out}}$ is an $M^{\texttt{in}}$-eligible edge. Thus, (ii) is equivalent to: (iii) $v = \texttt{match}_{M^{\texttt{out}}}$ satisfies $\varphi^{\texttt{in}}(v) + \varphi^{\texttt{in}}(u) = c(u, v) + 1$. Finally, we bound $\varepsilon_{\texttt{out}}$ as a function of $\varepsilon_{\texttt{in}}$.

Lemma 5.3. *Step 1 can be implemented in $\tilde{O}(n^{2-\varepsilon})$ time for some constant $\varepsilon > 0$. Moreover, the oracle $\texttt{match}_{M^{\texttt{out}}}$ has running time $t_{M^{\texttt{out}}}$ and the oracle $\texttt{eval}_{\varphi^{\texttt{out}}}$ has running time $t_{\varphi^{\texttt{out}}}$ such that $t_{M^{\texttt{out}}}, t_{\varphi^{\texttt{out}}} = \tilde{O}(n^{1+\varepsilon_{\texttt{out}}})$ and $\varepsilon_{\texttt{out}} = O_{\gamma,k}(\varepsilon_{\texttt{in}})$.*

Proof. Let $\varepsilon > 0$ and $\beta_{k,\gamma}$ as in Lemma 5.2. Algorithm 2 runs $\texttt{AugmentEligible}$ at most $1/\beta_{k,\gamma} + 1 = O_{k,\gamma}(1)$ times because the set $\mathcal{A}$ increases by $\beta_{k,\gamma} \cdot n$ after each successful run of $\texttt{AugmentEligible}$. Thus, there can be at most $1/\beta_{k,\gamma}$ successful runs. It is apparent that, by Lemma 5.2, Step 1 can be implemented in $\tilde{O}_{k,\gamma}(n^{2-\varepsilon})$ time.

Now we prove the bound on oracles time. First, we observe that $t_{\varphi^{\texttt{out}}} = O(t_{M^{\texttt{out}}} + t_{\varphi^{\texttt{in}}}) = \tilde{O}(n^{1+\varepsilon_{\texttt{out}}})$. Moreover, at every iteration we have $\varepsilon' \leqslant \alpha_{k,\gamma} \cdot \varepsilon$, hence $\varepsilon_{\texttt{out}} \leqslant \alpha_{k,\gamma}^{1/\beta_{k,\gamma}+1} \varepsilon_{\texttt{in}} = O_{k,\gamma}(\varepsilon_{\texttt{in}})$. □

5.3 Implementing Step 2

In this subsection we implement Step 2 from Algorithm 1 in sublinear time. Once again, we assume that we have at our disposal a potential oracle $\texttt{eval}_{\varphi^{\texttt{in}}}$ running in time $t_{\varphi^{\texttt{in}}} = \tilde{O}(n^{1+\varepsilon_{\texttt{in}}})$ and a matching oracle $\texttt{match}_{M^{\texttt{in}}}$ with running time $t_{M^{\texttt{in}}} = \tilde{O}(n^{1+\varepsilon_{\texttt{in}}})$. We will output a potential oracle $\texttt{eval}_{\varphi^{\texttt{out}}}$ running in time $t_{\varphi^{\texttt{out}}} = \tilde{O}(n^{1+\varepsilon_{\texttt{out}}})$. We show that there exists a (k, δ)-QMF$\mathcal{F}$ with respect to $M^{\texttt{in}}$ such that $\varphi^{\texttt{out}}(u) = \varphi^{\texttt{in}}(u) + 1$ for each $u \in \mathcal{F} \cap V_0$ and $\varphi^{\texttt{out}}(v) = \varphi^{\texttt{in}}(v) - 1$ for each $v \in \mathcal{F} \cap V_1$.

The execution of Algorithm 3 is represented in Figure 1, where vertices colored in the same way are added to $\mathcal{F}$ during the same iteration.

Analysis. First, we prove that Algorithm 3 implements the template (all guarantees hold whp). Namely, that $\mathcal{F}$ is a (k, δ)-QMF, where k and δ are defined as in Algorithm 1. With a slight abuse of notation, in Algorithm 3 we used $\mathcal{F}$ to denote the set of nodes in the forest. Here, we understand that for each $u \in \mathcal{F}'_t \backslash \mathcal{F}_t$ we have an edge $(u, \texttt{match}_{M_{t+1}}(u))$ and for each $v \in \mathcal{F}''_t \backslash \mathcal{F}'_t$ we have an edge $(v, \texttt{match}_{M^{\texttt{in}}}(v))$. Let τ be the total number of times $\texttt{LargeMatchingForward}$ runs successfully in Algorithm 3. We will see that $\tau \leqslant k/2$. Notice that $\mathcal{F}_\tau$ is the last forest produced by Algorithm 3 and for each $u \in \mathcal{F}_\tau \backslash F_0$ we add an edge incident to u, thus $\mathcal{F}_\tau$ is a forest with $|F_0|$ connected components, one for each $u \in F_0$. Now we show that $\mathcal{F}_\tau$ is a (k, δ)-QMFw.r.t. $M^{\texttt{in}}$. We refer to the notation of Definition 3.6. Item 1 is clearly satisfied. Item 2 is satisfied because of line 4 in Algorithm 3.

Algorithm 3 Implementation of Step 2.

Set δ as in Algorithm 1.
Initialize $t \leftarrow 0$, $\mathcal{F}_0 \leftarrow F_0$, where F_0 is the set of $M^{\texttt{in}}$-unmatched vertices in V_0.
Implement $\texttt{mem}_{\mathcal{F}_0}(u)$ as: $\texttt{match}_{M^{\texttt{in}}}(u)$ == $\perp$.
Repeat until `LargeMatchingForward` returns $\perp$:

1. $A_t \leftarrow (\mathcal{F}_t \cap V_0) \cup (V_1 \backslash \mathcal{F}_t)$.
2. Let $\texttt{LargeMatchingForward}(\varphi, A_t, \delta)$ return $\texttt{match}_{M_t}$.
3. $\mathcal{F}'_t \leftarrow \mathcal{F}_t \cup \{\texttt{match}_{M_t(u)} \mid u \in \mathcal{F}_t\}$
   Implement $\texttt{mem}_{\mathcal{F}'_t}(u)$ as: $\texttt{mem}_{\mathcal{F}_t}(u)$ or $\texttt{match}_{M_t}(u) \in \mathcal{F}_t$.
4. $\mathcal{F}''_t \leftarrow \mathcal{F}'_t \cup \{\texttt{match}_{M^{\texttt{in}}}(u) \mid u \in \mathcal{F}'_t\}$
   Implement $\texttt{mem}_{F''_t}(u)$ as: $\texttt{mem}_{\mathcal{F}'_t}(u)$ or $\texttt{match}_{M^{\texttt{in}}}(u) \in \mathcal{F}'_t$.
5. $\mathcal{F}_{t+1} \leftarrow \mathcal{F}''_t$; $t \leftarrow t+1$.

Implement $\texttt{eval}_{\varphi^{\texttt{out}}}(u)$ as $\texttt{eval}_{\varphi^{\texttt{in}}}(u) + \texttt{mem}_{\mathcal{F}_t}(u) \cdot (-1)^{\mathbb{1}_{u \in V_1}}$

Now we show that Item 3 is satisfied. Define $k = 6000(2T+1)^{10}/\delta^5$ as in Algorithm 1 and recall that φ is a $(2T+1)$-potential. Thanks to Lemma 5.1, at each step we increment $|\mathcal{F}|$ by at least $\frac{1}{2000(2T+1)^{10}} \cdot \delta^5 n$. Thus, no more than $\lceil 2000(2T+1)^{10}/\delta^5 \rceil \leqslant k/2$ iterations are performed and we cannot have more than k hops between $u \in \mathcal{F}$ and $v \in F_0$ if u belongs to the connected component of v.

Now we prove that Item 4 is satisfied. At each iteration, the size of each connected component of $\mathcal{F}_t$ at most triples. Indeed, let C be a connected component of $\mathcal{F}$. In step 3 we add to C at most $|C|$ vertices (because we add a vertex for each edge in a matching incident to C) and in step 4 we add to C at most one more vertex for each new vertex added in step 3.

Now we prove that Item 5 is satisfied. Algorithm 3 halts when $\texttt{LargeMatchingForward}(M^{\texttt{in}}, (\mathcal{F} \cap V_0) \cup (V_1 \backslash \mathcal{F}), \delta)$ returns $\perp$. This may only happen when there is no matching between of $\mathcal{F} \cap V_0$ and $V_1 \backslash \mathcal{F}$ of size δn. This implies that there exists a vertex cover of size $\leqslant \delta n$. Moreover, this is a vertex cover for the whole $E_{\mathcal{E}} \cap (\mathcal{F} \times V \backslash \mathcal{F})$ because all edges in $E_{\mathcal{E}} \cap V_1 \times V_0$ are in $M^{\texttt{in}}$ and by Item 2 have both endpoints either in $\mathcal{F}$ or in $V \backslash \mathcal{F}$.

It is easy to check that $\varphi^{\texttt{out}}(u) = \varphi^{\texttt{in}}(u) + 1$ for each $u \in \mathcal{F} \cap V_0$ and $\varphi^{\texttt{out}}(v) = \varphi^{\texttt{in}}(v) - 1$ for each $v \in \mathcal{F} \cap V_1$.

Lemma 5.4. *Step 2 can be implemented in $\tilde{O}(n^{2-\varepsilon})$ time for some constant $\varepsilon > 0$. Moreover, the oracle $\texttt{eval}_{\varphi_{out}}$ has running time $t_{\varphi^{out}} = \tilde{O}(n^{1+\varepsilon_{out}})$ and $\varepsilon_{out} = O_{k,\delta}(\varepsilon_{in})$.*

Proof. For $s = 0 \dots \tau$ denote with $\varepsilon_s > 0$ a constant such that $t_{A_s} = \tilde{O}(n^{1+\varepsilon_s})$, where t_{A_s} is the running time of $\texttt{mem}_{A_s}$. Notice that $\varepsilon_0 = \varepsilon_{\texttt{in}}$. At step s we choose $\hat{\varepsilon}_s := 2\varepsilon_s$ as the ε parameter in Lemma 5.1. This implies that `LargeMatchingForward` runs in $\tilde{O}(n^{1+\varepsilon_s} \cdot n^{1-\hat{\varepsilon}_s}) = \tilde{O}(n^{2-\varepsilon_s})$ time. We have already proved that $\tau \leqslant k$, thus Algorithm 2 takes $\tilde{O}(n^{2-\varepsilon})$ time in total, where $\varepsilon := \min_{s \in [0,\tau]} \varepsilon_s = \varepsilon_{\texttt{in}}$.

Denote with $t_{\mathcal{F}}$ the query time of $\texttt{mem}_{\mathcal{F}}$. For each s, we have $t_{\mathcal{F}_{s+1}} = t_{M_{s+1}} + t_{M^{\texttt{in}}} + t_{\mathcal{F}_s}$. Thanks to Lemma 5.1 we have $t_{M_{s+1}} = \tilde{O}((t_{A_s} + t_{\varphi^{\texttt{in}}} + n)n^{4\hat{\varepsilon}_s})$. Moreover, $t_{A_s} = t_{\mathcal{F}_s}$. Thus,

$t_{\mathcal{F}_{s+1}} = (t_{\mathcal{F}_s} + t_{\varphi^{\mathtt{in}}} + n)n^{8\varepsilon_s} + t_{M^{\mathtt{in}}} + t_{\mathcal{F}_s}$. Since, $t_{\mathcal{F}_0} = t_{\varphi^{\mathtt{in}}} = t_{M^{\mathtt{in}}} = \tilde{O}(n^{1+\varepsilon_{\mathtt{in}}})$ we have $t_{\varphi^{\mathtt{out}}} = t_{\varphi^{\mathtt{in}}} + t_{\mathcal{F}_\tau} = \tilde{O}(n^{1+9^\tau\cdot\varepsilon_{\mathtt{in}}}) = \tilde{O}_k(n^{1+O_{k,\delta}(\varepsilon_{\mathtt{in}})})$. □

5.4 Implementing the Template Algorithm

We can put together the results proved in the previous subsections and show that Algorithm 1 can be implemented in sublinear time.

Theorem 10. *There exists a constant $\varepsilon > 0$ such that Algorithm 1 can be implemented in time $O(n^{2-\varepsilon})$. Moreover, using the notation in Observation 9, we can return a matching oracle $\mathtt{match}_{M^{\mathit{ALG}}_{[\alpha_w]}}$ running in time $O(n^{1-\varepsilon})$ such that $M^{\mathit{ALG}}_{[\alpha_w]}$ satisfies $|M^{\mathit{ALG}}_{[\alpha_w]}| \geqslant (1-\gamma)n$ and $c(M^{\mathit{ALG}}_{[\alpha_w]}) \leqslant c(M^{\mathit{OPT}})$ and $\mathtt{match}_{M^{\mathit{ALG}}_{[\alpha_w]}}$.*

Proof. Algorithm 1 runs $T = O_{C,\gamma}(1)$ iterations, and a single iterations consists of Step 1 and Step 2. At iteration s denote with $\varepsilon^{(s)}_{\mathtt{in}}$ the value of $\varepsilon_{\mathtt{in}}$ for Step 1 input (or, equivalently, the value of $\varepsilon_{\mathtt{out}}$ for Step 2 output at iteration $s-1$) and with $\varepsilon^{(s)}_{\mathtt{out}}$ the value of $\varepsilon_{\mathtt{out}}$ for Step 1 output (or, equivalently, the value of $\varepsilon_{\mathtt{in}}$ for Step 2 input at iteration s). Every time we run either Step 1 or Step 2, the value of $\varepsilon_{\mathtt{out}}$ is at most some constant factor larger than $\varepsilon_{\mathtt{in}}$. This translates into $\varepsilon^{(s)}_{\mathtt{in}} = O_{C,k,\gamma}(\varepsilon^{(s-1)}_{\mathtt{out}})$ and $\varepsilon^{(s)}_{\mathtt{out}} = O_{C,k,\gamma}(\varepsilon^{(s)}_{\mathtt{in}})$. Thus, after T iterations $\varepsilon^{(T)}_{\mathtt{out}}$ is arbitrarily small, provided that $\varepsilon^{(0)}_{\mathtt{in}}$ is small enough. To conclude, we notice that the initial matching is empty and the initial potential is identically 0, so the first membership oracle and potential oracle run in linear linear, thus we can set $\varepsilon^{(0)}_{\mathtt{in}}$ arbitrarily small. Finally, let $M^{\mathtt{ALG}}$ be the last matching computed by Algorithm 1. We have at our disposal a matching oracle $\mathtt{match}_{M^{\mathtt{ALG}}}$ running in time $\tilde{O}(n^{1+\varepsilon^{(T+1)}_{\mathtt{in}}})$, so we can easily sample the a S of $O(\log n)$ edges from $M^{\mathtt{ALG}}$ in time $\tilde{O}(n^{1+\varepsilon^{(T)}_{\mathtt{out}}})$. This conclude the implementation of Algorithm 1.

Moreover, we compute w as the largest value such that at least $3\gamma|S|$ edges in S have cost $\geqslant w$ and define α_w such that exactly $\alpha_w \cdot n$ edges in $M^{\mathtt{ALG}}$ have cost $\geqslant w$. Then, we implement a matching oracle $\mathtt{match}_{M^{\mathtt{ALG}}_{[\alpha_w]}}$ for $M^{\mathtt{ALG}}_{[\alpha_w]}$ running in time $\tilde{O}(n^{1+\varepsilon^{(T)}_{\mathtt{out}}})$ as follows: given $u \in V_0 \cup V_1$ we set $v \leftarrow \mathtt{match}_{M^{\mathtt{ALG}}}(u)$; if $c(u,v) < w$ then we return v, else we return $\perp$. Thanks to Observation 9, we have $|M^{\mathtt{ALG}}_{[\alpha_w]}| \geqslant (1-\gamma)n$ and $c(M^{\mathtt{ALG}}_{[\alpha_w]}) \leqslant c(M^{\mathtt{OPT}})$. □

6 Proof of our Main Theorems

In this section we piece things together and prove Theorem 4. Then, we use Theorem 4 to prove Theorem 1, Theorem 2 and Theorem 3.

6.1 Proof of Theorem 4

In this subsection we strengthen Theorem 8, extend its scope to arbitrary costs and combine it with Theorem 10 to obtain Theorem 4. We restate the latter for convenience.

Theorem 4. *For each constants $0 \leqslant \alpha < \beta \leqslant 1$ there exists a constant $\varepsilon > 0$ and an algorithm running in time $O(n^{2-\varepsilon})$ with the following guarantees.*

The algorithm has adjacency-matrix access to an undirected, bipartite graph $G = (V_0 \cup V_1, E)$ and random access to the edge-cost function $c : E \to \mathbb{R}^+$. The algorithm returns $\hat{c}$ such that, whp,

$$c(M^\alpha) \leqslant \hat{c} \leqslant c(M^\beta)$$

where M^α is a minimum-weight matching of size αn and M^β is a minimum-weight matching of size βn.

Moreover, the algorithm returns a matching oracle data structure that, given a vertex u returns, in $n^{1+f(\varepsilon)}$ time, an edge $(u, v) \in \hat{M}$ or $\perp$ if $u \notin V(\hat{M})$, where $f(\varepsilon) \to 0$ when $\varepsilon \to 0$. The matching $\hat{M}$ satisfies $\alpha n \leqslant |\hat{M}| \leqslant \beta n$ and $c(M^\alpha) \leqslant c(\hat{M}) \leqslant c(M^\beta)$.

Roadmap of the proof. Theorem 8 works only for weights in $[1, C]$. In order to reduce to that case, we need to find a *characteristic cost* $\bar{w}$ of min-weight matchings with size in $[\alpha n, \beta n]$. Then, we round every cost to a multiple of $\frac{1}{2}\gamma^2\bar{w}$, where γ is a small constant. We show that, thanks to certain properties of the characteristic cost $\bar{w}$, the approximation error induced by rounding the costs is negligible. Finally, we pad each size of the bipartition with dummy vertices to reduce the problem of finding a matching of approximate size βn to that of finding an *approximate perfect matching*, which is addressed in Theorem 8.

Notation. Similarly to Theorem 4, we denote with M^ξ the min-weight matching of size ξn in G. Likewise, we will define a graph $\bar{G}$ and denote with $\overline{M}^\xi$ the min-weight matching of size ξn in $\bar{G}$. As in Section 5, given a matching M, we denote with $M_{[\delta]}$ the matching obtained from M by removing the δn most expensive edges. We denote with $\mu(M)$ the cost of the most expensive edge in M. Given $w \geqslant 0$, we denote with $G_{\leqslant w}$ the graph of edges which cost $\leqslant w$. Throughout this subsection, fix a constant $0 < \gamma < (\beta - \alpha)/4$.

Reduction from arbitrary weights to $[1, C]$. The next technical lemma shows that, if we can can solve an easier version of the problem in Theorem 4 where we allow an additive error $\gamma^2\bar{w}n$ on an instance where $\bar{w}$ is an upper bound for the cost function; then we can also solve the problem in Theorem 4. This reduction is achieved by finding a suitable characteristic cost $\bar{w}$ in sublinear time and running the aforementioned algorithm on $G_{\leqslant \bar{w}}$.

Lemma 6.1. *Suppose that there exists an algorithm that takes as input a bipartite graph $\bar{G} = (\bar{V}_0 \cup \bar{V}_1, \bar{E})$ endowed with a cost function $\bar{c} : \bar{E} \to [0, \bar{w}]$, outputs an estimate $\hat{c}$ and a matching oracle $\texttt{match}_{\hat{M}}$ such that (whp) $\hat{c}$ satisfies*

$$\bar{c}(\overline{M}^\alpha) \leqslant \hat{c} \leqslant \bar{c}(\overline{M}^{\alpha+\gamma}) + \gamma^2\bar{w}n$$

while $\hat{M}$ satisfies $|\hat{M}| \geqslant \alpha n$ and $c(\hat{M}) \leqslant \bar{c}(\overline{M}^{\alpha+\gamma}) + \gamma^2\bar{w}n$. Suppose also that such algorithm runs in time $O(\bar{n}^{2-\varepsilon})$ and $\texttt{match}_{\hat{M}}$ runs in time $O(\bar{n}^{1+\varepsilon})$ for some $\varepsilon > 0$, where $\bar{n} = |\bar{V}_0| = |\bar{V}_1|$.

Then, there exists an algorithm that takes as input a bipartite graph $G = (V_0 \cup V_1, E)$ endowed with a cost function $c : E \to \mathbb{R}^+$, outputs an estimate $\hat{c}$ and a matching oracle $\texttt{match}_{\hat{M}}$ such that (whp) $\hat{c}$ satisfies

$$C(M^\alpha) \leqslant \hat{c} \leqslant c(M^\beta)$$

while $\hat{M}$ satisfies $|\hat{M}| \geqslant \alpha n$ and $c(\hat{M}) \leqslant c(M^\beta)$. Moreover, such algorithm runs in time $O(n^{2-\varepsilon})$ and $\texttt{match}_{\hat{M}}$ runs in time $O(n^{1+\varepsilon})$ for some $\varepsilon > 0$, where $n = |V_0| = |V_1|$.

Proof. First, we show how to compute, in time $O(n^{2-\varepsilon})$, a value $\bar{w}$ such that:

(i) $\gamma \cdot \bar{w} \leqslant \mu(M^{\beta}_{[\gamma]})$

(ii) $\bar{w} \geqslant \mu(M^{\alpha+3\gamma}_{[2\gamma]})$.

We sample $s = \Theta(n \log n/\gamma)$ edges from G uniformly at random. Let $w_1 \leqslant w_2 \leqslant \cdots \leqslant w_s$ be their costs. Recall that Theorem 5 allows us to compute the size of a maximal-cardinality matching (MCM) of the graph G, up to a γn-additive approximation, in time $O(n^{2-\Omega_\gamma(1)})$. Denote that algorithm with $\texttt{ApproximateMatching}(G, \gamma)$. Using binary search, we find the largest cost w_i such that $\texttt{ApproximateMatching}(G_{\leqslant \gamma w_i}, \gamma)$ returns an estimated MCM size $< (\beta - 2\gamma)n$. Then, we set $\bar{w} := w_i$.

We prove that property (i) holds. Suppose that $\mu(M^{\beta}_{[\gamma]}) < \gamma w_i$. Then, in $G_{\leqslant \gamma w_i}$ there exists a matching of size $(\beta - \gamma)n$, therefore $\texttt{ApproximateMatching}(G_{\leqslant \gamma w_i}, \gamma)$ finds a matching of size $\geqslant (\beta - 2\gamma)n$ whp, contradiction.

We prove that property (ii) holds. First, we prove that $w_{i+1} \geqslant \mu(M^{\alpha+3\gamma}_{[\gamma]})$. Indeed, suppose the reverse (strict) inequality holds. Then, for each matching M of size $(\beta - 2\gamma)n$ we have

$$c(M) \geqslant c(M^{\beta-2\gamma}) \geqslant c(M^{\alpha+3\gamma}) > \gamma \cdot w_{i+1} n$$

which implies that there exists $e \in M$ such that $c(e) > \gamma \cdot w_{i+1}$. However, this cannot hold for each matching M of size $(\beta - 2\gamma)n$ because $\texttt{ApproximateMatching}(G_{\leqslant \gamma w_{i+1}}, \gamma)$ returned (whp) a matching (oracle) of size $\geqslant (\beta - 2\gamma)n$. Contradiction. Therefore, we have $w_{i+1} \geqslant \mu(M^{\alpha+3\gamma}_{[\gamma]})$. However, since we sampled $\Theta(n \log n/\gamma)$ edges, then for each i there are, whp, at most $\gamma \cdot n$ edges which cost w satisfies $w_i < w < w_{i+1}$. Hence, removing γn more edges from $M^{\alpha+3\gamma}_{[\gamma]}$ we obtain $w_i \geqslant \mu(M^{\alpha+3\gamma}_{[2\gamma]})$.

We define $\bar{G} := G_{\leqslant \bar{w}}$ and $\bar{c} = c|_{\bar{E}}$, run the algorithm in the premise of the lemma on $\bar{G}$ and $\bar{c}$ and let $\hat{c}$, $\texttt{match}_{\hat{M}}$ be its outputs. It is apparent that this reduction takes $O(n^{2-\varepsilon})$ for some $\varepsilon > 0$.

Since $\bar{G}$ is a subgraph of G, we have $c(M^{\alpha}) \leqslant c(\overline{M}^{\alpha})$. Moreover, conditions (i) and (ii), together with $5\gamma \leqslant \beta - \alpha$ imply

$$c(\overline{M}^{\alpha+\gamma}) \leqslant c(M^{\alpha+3\gamma}_{[2\gamma]}) \leqslant c(M^{\alpha+3\gamma}) \leqslant c(M^{\beta-\gamma}) \leqslant c(M^{\beta}_{[\gamma]}) \leqslant c(M^{\beta}) - \gamma n \cdot \mu(M^{\beta}_{[\gamma]}) \leqslant c(M^{\beta}) - \gamma^2 \bar{w} n.$$

Thus, $c(\overline{M}^{\alpha}) \leqslant \hat{c} \leqslant c(\overline{M}^{\alpha+\gamma}) + \gamma^2 \bar{w} n$ implies $c(M^{\alpha}) \leqslant \hat{c} \leqslant c(M^{\beta})$ and $c(\hat{M}) \leqslant c(\overline{M}^{\alpha+\gamma}) + \gamma^2 \bar{w} n$ implies $\hat{c} \leqslant c(M^{\beta})$. □

The following lemma shows how to reduce from real-values costs in $[0, w]$ (where possibly $w = \omega(1)$) to the more tame case where costs are integers in $[1, C]$. This reduction is achieved via rounding.

Lemma 6.2. *Suppose that there exists an algorithm that takes as input a bipartite graph $\bar{G} = (\bar{V}_0 \cup \bar{V}_1, \bar{E})$ endowed a cost function $\bar{c} : E \to [1, C]$, with $C = O(1)$, returns an estimate $\hat{c}$ and a matching oracle $\texttt{match}_{\hat{M}}$ such that (whp) $\hat{c}$ satisfies*

$$\bar{c}(\overline{M}^{\alpha}) \leqslant \hat{c} \leqslant \bar{c}(\overline{M}^{\beta})$$

while $\hat{M}$ satisfies $|\hat{M}| \geqslant \alpha n$ and $\bar{c}(\hat{M}) \leqslant \bar{c}(\overline{M}^{\beta})$. Suppose also that such algorithm runs in time $O(\bar{n}^{2-\varepsilon})$ and $\texttt{match}_{\hat{M}}$ runs in time $O(\bar{n}^{1+\varepsilon})$ for some $\varepsilon > 0$, where $\bar{n} = |\bar{V}_0| = |\bar{V}_1|$.

Then, there exists an algorithm that takes as input a bipartite graph $G = (V_0 \cup V_1, E)$ endowed a cost function $c : E \to [0, w]$ (possibly $w = \omega(1)$), returns an estimate $\tilde{c}$ and a matching oracle $\texttt{match}_{\tilde{M}}$ such that (whp) $\tilde{c}$ satisfies

$$c(M^\alpha) \leqslant \tilde{c} \leqslant c(M^\beta) + \gamma^2 wn$$

while $\tilde{M}$ satisfies $|\tilde{M}| \geqslant \alpha n$ and $c(\tilde{M}) \leqslant c(M^\beta) + \gamma^2 wn$. Moreover, such algorithm runs in time $O(n^{2-\varepsilon})$ and $\texttt{match}_{\hat{M}}$ runs in time $O(n^{1+\varepsilon})$ for some $\varepsilon > 0$, where $n = |V_0| = |V_1|$.

Proof. We define $\bar{c}(e) = \left\lceil \frac{2c(e)}{\gamma^2 w} \right\rceil + 1$. Then, the maximum value of $\bar{c}$ on $\bar{G}$ is $C := 2/\gamma^2 + 2 = O(1)$. We set $\bar{G} = G$ and run the algorithm in the premise of the lemma on $\bar{G}$ and $\bar{c}$. Let $\hat{c}$, $\texttt{match}_{\hat{M}}$ be its outputs. We define $\tilde{c} := \frac{1}{2}\gamma^2 w \cdot \hat{c}$ and $\tilde{M} := \hat{M}$. The definition of $\bar{c}$ implies that, for each edge e in $\bar{G}$, $c(e) \leqslant \frac{1}{2}\gamma^2 w \cdot \bar{c}(e) \leqslant c(e) + \gamma^2 w$. Hence,

$$\frac{1}{2}\gamma^2 w \cdot \bar{c}(\overline{M}^\alpha) \geqslant c(\overline{M}^\alpha) \geqslant c(M^\alpha)$$

and

$$\frac{1}{2}\gamma^2 w \cdot \bar{c}(\overline{M}^\beta) \leqslant \frac{1}{2}\gamma^2 w \cdot \bar{c}(M^\beta) \leqslant c(M^\beta) + \gamma^2 wn.$$

Therefore, $\bar{c}(\overline{M}^\alpha) \leqslant \hat{c} \leqslant \bar{c}(\overline{M}^\beta)$ implies $c(M^\alpha) \leqslant \tilde{c} \leqslant c(M^\beta) + \gamma^2 wn$ and $\bar{c}(\hat{M}) \leqslant \bar{c}(\overline{M}^\beta)$ implies $c(\hat{M}) \leqslant \frac{1}{2}\gamma^2 w \cdot \bar{c}(\hat{M}) \leqslant \frac{1}{2}\gamma^2 w \cdot \bar{c}(\overline{M}^\beta) \leqslant c(M^\beta) + \gamma^2 wn$. □

Reduction from size-βn matching to perfect matching. The following lemma shows that, if we can approximate the min-weight of a perfect matching (allowing δn outliers), then we can approximate the min-weight of a size-βn matching (allowing $(\beta - \alpha)n$ outliers). This reduction is achieved by padding the original graph with dummy vertices.

Lemma 6.3. *Suppose that, for each $\delta > 0$, there exists an algorithm that takes as input a bipartite graph $\bar{G} = (\bar{V}_0 \cup \bar{V}_1, \bar{E})$ endowed a cost function $\bar{c} : E \to [1, C]$, with $C = O(1)$, returns an estimate $\hat{c}$ and a matching oracle $\texttt{match}_{\hat{M}}$ such that (whp) $\hat{c}$ satisfies*

$$\bar{c}(\overline{M}^{1-\delta}) \leqslant \hat{c} \leqslant \bar{c}(\overline{M}^1)$$

while $\hat{M}$ satisfies $|\hat{M}| \geqslant (1 - \delta)n$ and $c(\hat{M}) \leqslant \bar{c}(\overline{M}^1)$. Suppose also that such algorithm runs in time $O(\bar{n}^{2-\varepsilon})$ and $\texttt{match}_{\hat{M}}$ runs in time $O(\bar{n}^{1+\varepsilon})$ for some $\varepsilon > 0$, where $\bar{n} = |\bar{V}_0| = |\bar{V}_1|$.

Then, for each $0 \leqslant \alpha < \beta \leqslant 1$ there exists an algorithm that takes as input a bipartite graph $G = (V_0 \cup V_1, E)$ and $c : E \to [1, C]$, returns an estimate $\tilde{c}$ and a matching oracle $\texttt{match}_{\tilde{M}}$ such that (whp) $\tilde{c}$ satisfies

$$c(M^\alpha) \leqslant \tilde{c} \leqslant c(M^\beta)$$

while $\tilde{M}$ satisfies $|\tilde{M}| \geqslant \alpha n$ and $c(\tilde{M}) \leqslant c(M^\beta)$. Moreover, such algorithm runs in time $O(n^{2-\varepsilon})$ and $\texttt{match}_{\tilde{M}}$ runs in time $O(n^{1+\varepsilon})$ for some $\varepsilon > 0$, where $n = |V_0| = |V_1|$.

Proof. Fix a constant $0 < \xi \leqslant (\alpha - \beta)/2$. We construct $\bar{G}$, starting from $\bar{G} = G$ and $\bar{c} = c$, as follows. We add a set of $(1 - \beta + \xi)n$ dummy vertices on each side of the bipartition: $\bar{V}_0 = V_0 \cup D_0$ and $\bar{V}_1 = V_1 \cup D_1$. Add an edge (d_0, v_1) to E for each $(d_0, v_1) \in D_0 \times V_1$ and set $\bar{c}(d_0, v_1) = 1$. Do the same for each $(v_0, d_1) \in V_0 \times D_1$. Notice that we construct both the adjacency matrix

and the cost function implicitly, because an explicit construction would take $\Omega(n^2)$ time. We have $\bar{n} := |\bar{V}_0| = |\bar{V}_1| = (2 - \beta + \xi)n$.

Set $\delta = \frac{\xi n}{2\bar{n}}$. Run the algorithm in hypothesis on $\bar{G}$ and let $\hat{c}$ and $\texttt{match}_{\hat{M}}$ be its outputs. We set $\tilde{c} = \hat{c} - (2 - 2\beta + \xi)n$. We set $\tilde{M} := \hat{M} \cap (V_0 \times V_1)$ and implement $\texttt{match}_{\tilde{M}}(u)$ as follows. Let $v \leftarrow \texttt{match}_{\hat{M}}(u)$. If $v = \perp$, return $\perp$. If either u or v is dummy, return $\perp$; else return v. It is easy to see that $\overline{M}^1 \cap (V_0 \times V_1)$ is a min-weight matching of size $(\beta - \xi)n$ in G, hence

$$\bar{c}(\overline{M}^1) = c(M^{\beta-\xi}) + |D_0| + |D_1| = c(M^{\beta-\xi}) + 2(1 - \beta + \xi)n \leqslant c(M^\beta) + (2 - 2\beta + \xi)n.$$

On the other hand, in $\overline{M}^{1-\delta}$ at most $2\delta\bar{n} = \xi n$ dummy vertices are left unmathced, so at least $(2 - 2\beta + \xi)n$ dummy vertices are matched in $\overline{M}^{1-\delta}$. Moreover, $\overline{M}^{1-\delta} \cap (V_0 \times V_1)$ is a matching in G of size $\geqslant n - (1 - \beta + \xi)n - \delta\bar{n} = (\beta - 2\xi)n$. Hence,

$$\bar{c}(\overline{M}^{1-\delta}) \geqslant c(M^{\beta-2\xi}) + (2 - 2\beta + \xi)n \geqslant c(M^\alpha) + (2 - 2\beta + \xi)n.$$

Thus, $\bar{c}(\overline{M}^{1-\delta}) \leqslant \hat{c} \leqslant \bar{c}(\overline{M}^1)$ implies $c(M^\alpha) \leqslant \tilde{c} \leqslant c(M^\beta)$.

Now we prove the bounds on $\tilde{M}$. We have that since $|\hat{M}| \geqslant (1 - \delta)\bar{n}$, then at most $2\delta\bar{n} = \xi n$ dummy vertices are left unmatched in $\hat{M}$ and so

$$\begin{aligned} c(\tilde{M}) &\leqslant \bar{c}(\hat{M}) - (2(1 - \beta + \xi)n - 2\delta\bar{n}) \\ &\leqslant \bar{c}(\overline{M}^1) - (2 - 2\beta + \xi)n \\ &= c(M^{\beta-\xi}) + 2(1 - \beta - \xi)n - (2 - 2\beta + \xi)n \\ &= c(M^{\beta-\xi}) + \xi n \leqslant c(M^\beta). \end{aligned}$$

Moreover, $|\tilde{M}| \geqslant (1 - \delta)\bar{n} - 2(1 - \beta + \xi)n = (\beta - \frac{3}{2}\xi)n \geqslant \alpha n$. □

Finally, we can prove Theorem 4.

Proof of Theorem 4. We notice that combining Theorem 8 and Theorem 10 we have a sublinear implementation of Algorithm 1 that takes a graph bipartite graph $G = (V_0 \cup V_1, E)$ and a cost function $c : E \to [1, C]$ as input, outputs an estimate $\hat{c}$ and a matching oracle $\texttt{match}_{\hat{M}}$. The estimate $\hat{c}$ satisfies $c(M^{1-\delta}) \leqslant \hat{c} \leqslant c(M^1)$, $\hat{M}$ satisfies $|\hat{M}| \geqslant (1 - \delta)n$ and $c(\hat{M}) \leqslant c(M^{\texttt{OPT}})$. Moreover, such algorithm runs in time $O(n^{2-\varepsilon})$ and $\texttt{match}_{\hat{M}}$ runs in time $O(n^{1+\varepsilon})$ for some $\varepsilon > 0$.

Then, combining Lemma 6.3, Lemma 6.2 and Lemma 6.1 we obtain an algorithm that takes as input a bipartite graph $G = (V_0 \cup V_1, E)$ endowed with a cost function $c : E \to \mathbb{R}^+$, outputs an estimate $\hat{c}$ and a matching oracle $\texttt{match}_{\hat{M}}$ such that (whp) $c(M^\alpha) \leqslant \hat{c} \leqslant c(M^\beta)$, $|\hat{M}| \geqslant \alpha n$, and $c(\hat{M}) \leqslant c(M^\beta)$. Moreover such algorithm runs in time $O(n^{2-\varepsilon})$ and $\texttt{match}_{\hat{M}}$ runs in time $O(n^{1+\varepsilon})$ for some $\varepsilon > 0$. □

6.2 Proof of Theorem 1 and Theorem 2

Since Theorem 2 is more general than Theorem 1 we simply prove the former.

Theorem 2. *Suppose we have sample access to two distributions μ, ν over metric space $(\mathcal{M}, d_{\mathcal{M}})$ satisfying $d(\cdot, \cdot) \in [0, 1]$ and query access to d. Suppose further that there exist μ', ν' with support size n such that $\mathrm{EMD}(\mu, \mu'), \mathrm{EMD}(\nu, \nu') \leqslant \xi$, for some $\xi > 0$.*

For each constant $\gamma > 0$ there exists a constant $\varepsilon > 0$ and an algorithm running in time $O(n^{2-\varepsilon})$ that outputs $\widehat{\mathrm{EMD}}$ such that

$$\widehat{\mathrm{EMD}} \in [\mathrm{EMD}(\mu, \nu) \pm (4\xi + \gamma)].$$

Moreover, such algorithm takes $\tilde{O}(n)$ samples from μ and ν.

Fix a constant $\gamma > 0$. From each probability distribution μ, ν we sample (with replacement) a multi-set of $m = \Theta(n \log(n))$ points. We use V_μ, V_ν to denote the respective multi-sets, and $\hat{\mu}, \hat{\nu}$ to denote the empirical distributions of sampling a random point from V_μ, V_ν. Let $\mathcal{T}_\mu, \mathcal{T}_\nu$ be the transport plans realizing $\mathrm{EMD}(\mu, \mu')$ and $\mathrm{EMD}(\nu, \nu')$ respectively. Namely, $\mathcal{T}_\mu$ is a coupling between μ and μ' such that $\mathrm{EMD}(\mu, \mu') = E_{(x,y)\sim\mathcal{T}_\mu}[d(x, y)]$ and likewise for $\mathcal{T}_\nu$. For each sample x in $\hat{\mu}$ we sample $x' \sim \mathcal{T}(x, \cdot)$ and let V'_μ be the multi-set of samples x' for $x \in V_\mu$. Define V'_ν similarly. Let $\hat{\mu}'$ and $\hat{\nu}'$ be the empirical distributions of sampling a random point from V'_μ and V'_ν.

Lemma 6.4. $\mathrm{EMD}(\hat{\mu}, \hat{\mu}') \leqslant \xi + \gamma$ *with high probability.*

Proof. $\mathrm{E}[\mathrm{EMD}(\hat{\mu}, \hat{\mu}')] \leqslant \mathrm{E}\left[\frac{1}{m} \cdot \sum_{x\in V_\mu} d(x, x')\right] = \mathrm{E}_{(x,x')\sim\mathcal{T}_\mu}[d(x, x')] = \mathrm{EMD}(\mu, \mu') \leqslant \xi$. Moreover, $\mathrm{Var}_{(x,x')\sim\mathcal{T}_\mu}[d(x, x')] \leqslant 1$, thus $m = \Theta(n \log n)$ ensures $\mathrm{EMD}(\hat{\mu}, \hat{\mu}') \leqslant \xi + \gamma$ whp. □

Lemma 6.5. $\mathrm{EMD}(\hat{\mu}', \mu') \leqslant \mathrm{TV}(\hat{\mu}', \mu') \leqslant \gamma$ *with high probability.*

Proof. First, we observe that V'_μ is distributed as a multi-set of m samples from μ'. For any point x' with at least $(\gamma/4n)$-mass in μ', we expect $\Omega(\log n)$ samples of x' in V'_μ, so by Chernoff bound we have that with high probability the number of samples of x' concentrates to within $(1 \pm \gamma/4)$-factor of its expectation. Furthermore, with high probability at most $(\gamma/2)$-fraction of the samples correspond to points with less than $(\gamma/4n)$-mass in the original distribution. Thus overall, the empirical distribution $\hat{\mu}'$ is within γ TV distance of μ'. Finally, $\mathrm{EMD}(\hat{\mu}', \mu') \leqslant \mathrm{TV}(\hat{\mu}', \mu')$ beacuse $d(\cdot, \cdot) \in [0, 1]$. □

Lemma 6.6. $\mathrm{EMD}(\mu, \hat{\mu}) \leqslant \xi + 2\gamma$ *with high probability.*

Proof. Combining Lemma 6.4, Lemma 6.5 we obtain the following

$$\mathrm{EMD}(\mu, \hat{\mu}) \leqslant \mathrm{EMD}(\mu, \mu') + \mathrm{EMD}(\mu', \hat{\mu}') + \mathrm{EMD}(\hat{\mu}', \hat{\mu}) \leqslant 2\xi + 2\gamma.$$

□

Proof of Theorem 2. We consider the bipartite graph with a vertex for each point in V_μ, V_ν and edge costs induced by $d(\cdot, \cdot)$. We apply the algorithm guaranteed by Theorem 4 to find an estimate of the min-weight matching over between $(1-\gamma)m$ and m vertices. We return the cost estimate $\widehat{\mathrm{EMD}}$ on the bipartite graph (normalized by dividing by m). Theorem 4 guarantees that $\widehat{\mathrm{EMD}} \in [\mathrm{EMD}(\hat{\mu}, \hat{\nu}) \pm \gamma]$. Then using triangle inequality on EMD, as well as Lemma 6.6 on both μ and ν we obtain

$$\left|\widehat{\mathrm{EMD}} - \mathrm{EMD}(\mu, \nu)\right| \leqslant \left|\widehat{\mathrm{EMD}} - \mathrm{EMD}(\hat{\mu}, \hat{\nu})\right| + \mathrm{EMD}(\mu, \hat{\mu}) + \mathrm{EMD}(\nu, \hat{\nu}) \leqslant 4\xi + 5\gamma.$$

Scaling γ down of a factor 5 we retireve Theorem 2. □

6.3 Proof of Theorem 3

Theorem 3 (Main theorem, graph interpretation)**.** *For each constant $\gamma > 0$, there exists a constant $\varepsilon > 0$, and an algorithm running in time $O(n^{2-\varepsilon})$ with the following guarantees. The algorithm takes as input a budget B, and query access to the edge-cost matrix of an undirected, bipartite graph G over n vertices. The algorithm returns an estimate $\widehat{M}$ that is within $\pm\gamma n$ of the size of the maximum matching in G with total cost at most B.*

Proof. Let M be the maximum matching in G with total cost at most B, and let $|M| = \xi n$. We perform a binary search for ξ using the algorithm from Theorem 4 as a subroutine. This loses only a factor $\log(n)$ in query complexity, which gets absorbed in $O(n^{2-\varepsilon})$ by choosing a suitable constant ε. □

Acknowledgments. We thank Tal Herman for pointing out the sample complexity lower bound implied by [VV10].

References

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks". In: *International conference on machine learning*. PMLR. 2017, pp. 214–223.

[Aga+22] Pankaj K Agarwal, Hsien-Chih Chang, Sharath Raghvendra, and Allen Xiao. "Deterministic, near-linear $(1+\epsilon)$-approximation algorithm for geometric bipartite matching". In: *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*. 2022, pp. 1052–1065.

[AIK08] Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. "Earth mover distance over high-dimensional spaces." In: *SODA*. Vol. 8. 2008, pp. 343–352.

[ALT21] Tenindra Abeywickrama, Victor Liang, and Kian-Lee Tan. "Optimizing bipartite matching in real-world applications by incremental cost computation". In: *Proceedings of the VLDB Endowment* 14.7 (2021), pp. 1150–1158.

[And+09] Alexandr Andoni, Khanh Do Ba, Piotr Indyk, and David Woodruff. "Efficient sketches for earth-mover distance, with applications". In: *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE. 2009, pp. 324–330.

[And+14] Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. "Parallel algorithms for geometric graph problems". In: *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. 2014, pp. 574–583.

[ANR17] Jason Altschuler, Jonathan Niles-Weed, and Philippe Rigollet. "Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration". In: *Advances in neural information processing systems* 30 (2017).

[AS14] Pankaj K Agarwal and R Sharathkumar. "Approximation algorithms for bipartite matching with metric and geometric costs". In: *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. 2014, pp. 555–564.

[AZ23] Alexandr Andoni and Hengjie Zhang. "Sub-quadratic (1+\eps)-approximate Euclidean Spanners, with Applications". In: *arXiv preprint arXiv:2310.05315* (2023).

[Ba+11] Khanh Do Ba, Huy L Nguyen, Huy N Nguyen, and Ronitt Rubinfeld. "Sublinear time algorithms for earth mover's distance". In: *Theory of Computing Systems* 48 (2011), pp. 428–442.

[Bac+20] Arturs Backurs, Yihe Dong, Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. "Scalable nearest neighbor search for optimal transport". In: *International Conference on machine learning*. PMLR. 2020, pp. 497–506.

[Băd+05] Mihai Bădoiu, Artur Czumaj, Piotr Indyk, and Christian Sohler. "Facility location in sublinear time". In: *Automata, Languages and Programming: 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005. Proceedings 32*. Springer. 2005, pp. 866–877.

[Beh+23] Soheil Behnezhad, Mohammad Roghani, Aviad Rubinstein, and Amin Saberi. "Beating greedy matching in sublinear time". In: *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2023, pp. 3900–3945.

[Beh22] Soheil Behnezhad. "Time-optimal sublinear algorithms for matching and vertex cover". In: *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2022, pp. 873–884.

[BKS23a] Sayan Bhattacharya, Peter Kiss, and Thatchaphol Saranurak. "Dynamic $(1+\epsilon)$-Approximate Matching Size in Truly Sublinear Update Time". In: *arXiv preprint arXiv:2302.05030* (2023).

[BKS23b] Sayan Bhattacharya, Peter Kiss, and Thatchaphol Saranurak. "Sublinear Algorithms for $(1.5+\epsilon)$-Approximate Matching". In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*. Ed. by Barna Saha and Rocco A. Servedio. ACM, 2023, pp. 254–266. DOI: 10.1145/3564246.3585252. URL: https://doi.org/10.1145/3564246.3585252.

[Bla+18] Jose Blanchet, Arun Jambulapati, Carson Kent, and Aaron Sidford. "Towards optimal running times for optimal transport". In: *arXiv preprint arXiv:1810.07717* (2018).

[BRR23] Soheil Behnezhad, Mohammad Roghani, and Aviad Rubinstein. "Sublinear time algorithms and complexity of approximate maximum matching". In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. 2023, pp. 267–280.

[Can20] Clément L Canonne. "A survey on distribution testing: Your data is big. But is it blue?" In: *Theory of Computing* (2020), pp. 1–100.

[Cha+23] Moses Charikar, Beidi Chen, Christopher Ré, and Erik Waingarten. "Fast Algorithms for a New Relaxation of Optimal Transport". In: *The Thirty Sixth Annual Conference on Learning Theory*. PMLR. 2023, pp. 4831–4862.

[Cha02] Moses S Charikar. "Similarity estimation techniques from rounding algorithms". In: *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 2002, pp. 380–388.

[Che+22a] Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. "Maximum flow and minimum-cost flow in almost-linear time". In: *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2022, pp. 612–623.

[Che+22b] Xi Chen, Rajesh Jayaram, Amit Levi, and Erik Waingarten. "New streaming algorithms for high dimensional EMD and MST". In: *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*. 2022, pp. 222–233.

[CS09] Artur Czumaj and Christian Sohler. "Estimating the weight of metric minimum spanning trees in sublinear time". In: *SIAM Journal on Computing* 39.3 (2009), pp. 904–922.

[Cut13] Marco Cuturi. "Sinkhorn distances: Lightspeed computation of optimal transport". In: *Advances in neural information processing systems* 26 (2013).

[DGK18] Pavel Dvurechensky, Alexander Gasnikov, and Alexey Kroshnin. "Computational optimal transport: Complexity by accelerated gradient descent is better than by Sinkhorn's algorithm". In: *International conference on machine learning*. PMLR. 2018, pp. 1367–1376.

[FL22] Kyle Fox and Jiashuai Lu. "A deterministic near-linear time approximation scheme for geometric transportation". In: *arXiv preprint arXiv:2211.03891* (2022).

[GT89] Harold N Gabow and Robert E Tarjan. "Faster scaling algorithms for network problems". In: *SIAM Journal on Computing* 18.5 (1989), pp. 1013–1036.

[HIS13] Sariel Har-Peled, Piotr Indyk, and Anastasios Sidiropoulos. "Euclidean spanners in high dimensions". In: *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2013, pp. 804–809.

[Ind03] Piotr Indyk. "Fast color image retrieval via embeddings". In: *Workshop on Statistical and Computational Theories of Vision (at ICCV), 2003*. 2003.

[Ind04] Piotr Indyk. "Algorithms for dynamic geometric problems over data streams". In: *Proceedings of the thirty-sixth annual ACM Symposium on Theory of Computing*. 2004, pp. 373–380.

[Kuh55] Harold W Kuhn. "The Hungarian method for the assignment problem". In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[Kus+15] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. "From word embeddings to document distances". In: *International conference on machine learning*. PMLR. 2015, pp. 957–966.

[Le+21] Khang Le, Huy Nguyen, Quang M Nguyen, Tung Pham, Hung Bui, and Nhat Ho. "On robust optimal transport: Computational complexity and barycenter computation". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 21947–21959.

[LMR19] Nathaniel Lahn, Deepika Mulchandani, and Sharath Raghvendra. "A graph theoretic additive approximation of optimal transport". In: *Advances in Neural Information Processing Systems* 32 (2019).

[LXH23] Yiling Luo, Yiling Xie, and Xiaoming Huo. "Improved Rate of First Order Algorithms for Entropic Optimal Transport". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2023, pp. 2723–2750.

[McG05] Andrew McGregor. "Finding graph matchings in data streams". In: *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer. 2005, pp. 170–181.

[PC+19] Gabriel Peyré, Marco Cuturi, et al. "Computational optimal transport: With applications to data science". In: *Foundations and Trends® in Machine Learning* 11.5-6 (2019), pp. 355–607.

[Pha+20] Khiem Pham, Khang Le, Nhat Ho, Tung Pham, and Hung Bui. "On unbalanced optimal transport: An analysis of sinkhorn algorithm". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7673–7682.

[Roh19] Dhruv Rohatgi. "Conditional hardness of earth mover distance". In: *arXiv preprint arXiv:1909.11068* (2019).

[RTG00] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. "The earth mover's distance as a metric for image retrieval". In: *International journal of computer vision* 40 (2000), pp. 99–121.

[SA12] R Sharathkumar and Pankaj K Agarwal. "A near-linear time ε-approximation algorithm for geometric bipartite matching". In: *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. 2012, pp. 385–394.

[San15] Filippo Santambrogio. "Optimal transport for applied mathematicians". In: *Birkäuser, NY* 55.58-63 (2015), p. 94.

[Sol+15] Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. "Convolutional wasserstein distances: Efficient optimal transportation on geometric domains". In: *ACM Transactions on Graphics (ToG)* 34.4 (2015), pp. 1–11.

[Vil+09] Cédric Villani et al. *Optimal transport: old and new*. Vol. 338. Springer, 2009.

[VV10] Gregory Valiant and Paul Valiant. "A CLT and tight lower bounds for estimating entropy." In: *Electron. Colloquium Comput. Complex.* Vol. 17. 2010, p. 179.

[Yur+19] Mikhail Yurochkin, Sebastian Claici, Edward Chien, Farzaneh Mirzazadeh, and Justin M Solomon. "Hierarchical optimal transport for document representation". In: *Advances in neural information processing systems* 32 (2019).