Highlights

On computing Discretized Ricci curvatures of graphs: local algorithms and (localized) fine-grained reductions

Bhaskar DasGupta, Elena Grigorescu, Tamalika Mukherjee

- We relate our curvature computation problem to minimum weight perfect matching problem on complete bipartite graphs via fine-grained reduction.
- We formalize the computational aspects of the curvature computation problems in suitable frameworks so that they can be studied by researchers in local algorithms.
- We provide the first known lower and upper bounds on queries for query-based algorithms for the curvature computation problems in our local algorithms framework. *En route*, we also illustrate a localized version of our fine-grained reduction.

On computing Discretized Ricci curvatures of graphs: local algorithms and (localized) fine-grained reductions

Bhaskar DasGupta[a,1,*], Elena Grigorescu[b,2], Tamalika Mukherjee[b,2]

[a]*Department of Computer Science, University of Illinois Chicago, Chicago, 60607, IL, USA*
[b]*Department of Computer Science, Purdue University, West Lafayette, 47907, IN, USA*

Abstract

Characterizing shapes of high-dimensional objects via Ricci curvatures plays a critical role in many research areas in mathematics and physics. However, even though several discretizations of Ricci curvatures for discrete combinatorial objects such as networks have been proposed and studied by mathematicians, the computational complexity aspects of these discretizations have escaped the attention of theoretical computer scientists to a large extent. In this paper, we study one such discretization, namely the Ollivier-Ricci curvature, from the perspective of efficient computation by fine-grained reductions and local query-based algorithms. Our main contributions are the following.

- ▷ We relate our curvature computation problem to minimum weight perfect matching problem on complete bipartite graphs via fine-grained reduction.
- ▷ We formalize the computational aspects of the curvature computation problems in suitable frameworks so that they can be studied by researchers in local algorithms.
- ▷ We provide the first known lower and upper bounds on queries for query-based algorithms for the curvature computation problems in our

*Corresponding author.
Email addresses: bdasgup@uic.edu (Bhaskar DasGupta), elena-g@purdue.edu (Elena Grigorescu), tmukherj@purdue.edu (Tamalika Mukherjee)

[1]Supported by NSF grant IIS-1814931.
[2]Supported in part by NSF grants CCF-1910659 and CCF-1910411.

local algorithms framework. *En route*, we also illustrate a localized version of our fine-grained reduction.

We believe that our results bring forth an intriguing set of research questions, motivated both in theory and practice, regarding designing efficient algorithms for curvatures of geometrical objects.

Keywords: Network shape, discrete Ricci curvature, query-based local algorithms
2000 MSC: 68Q25, 68Q17, 68W25, 68W20, 68W40

---

1. Introduction

A suitable notion of "shape" plays a critical role in investigating objects in mathematics, mathematical physics and other research areas. Various kinds of curvatures are very natural measures of shapes of higher dimensional objects in mainstream physics and mathematics [1, 2]. To quantify the shape of a higher-dimensional geometric object, one often fixes shapes of objects with specific properties as the "baseline shape" and then quantifies the shape of a given object *with respect to* these baseline shapes. For example, consider the case of the two-dimensional metric space. For this space, a baseline could be selected as the standard *Euclidean plane* in which the three angles of a triangle sum up to *exactly* 180°, and then one can quantify the shape of the given two-dimensional space by the *deviations* of the sum of the three angles of triangles in this space from the baseline of 180°. An alternative approach is to avoid selecting baseline shapes *explicitly* and instead *directly* quantify the shape of a given geometric object. Quantification of shape is often referred to as the *curvature* of the corresponding object. Quantification of shapes can be either *local* or *global*. A local shape of the object is usually computed for a specific *local neighborhood* of the object (*e.g.*, the Ricci curvature). In contrast, a global shape of the object is usually computed over the entire object (*e.g.*, the Gromov-hyperbolicity measure). Any attempt to extend notions of curvature measures from non-network domains to networks[3] (and other discrete combinatorial structures) need to overcome at least three key challenges, namely that (*a*) networks are *discrete* (non-continuous) combinatorial objects, (*b*) networks may *not* necessarily have an associated natural

[3]In this paper, we will use the two terms "graph" and "network" *interchangeably*.

geometric embedding, and (c) the extension need to be useful and non-trivial, *i.e.*, a network curvature measure should saliently encode non-trivial higher-order correlations among nodes and edges that *cannot* be obtained by other popular network measures.

1.1. Motivations behind studying shapes of networks

Although studying measures of shapes of networks (and hypergraphs) is mathematically intriguing, it is natural to ask if there are other valid reasons for such studies. Network shape measures *can* encode non-trivial topological properties that are *not* expressed by more established network-theoretic measures such as degree distributions, clustering coefficients or betweenness centralities (*e.g.*, see [3, 4]). Moreover, these shape measures can explain many phenomena one frequently encounters in real network-theoretic applications, such as **(*i*)** paths mediating up- or down-regulation of a target node starting from the same regulator node in *biological regulatory networks* often have many small crosstalk paths [3] and **(*ii*)** existence of congestions in a node that is not a hub in *traffic networks* [3, 5], that are *not* easily explained by other non-shape measures. Recently, shape measures have also found applications in traditional social networks applications such as community finding [6], and in neuroscience applications such as comparing brain networks to study slowly progressing brain diseases such as *attention deficit hyperactivity disorder* [4] and *autism spectrum disorder* [7, 8].

1.2. Brief history of existing notions of shapes for networks

There are several ways previous researchers have attempted to formulate notions of shapes of networks. Below we discuss *three* major directions in this regard. For further details and other approaches, the reader is referred to papers and books such as [9, 1, 10, 11, 12, 13, 14, 15, 16, 3, 17, 18, 19, 20, 21, 22, 23, 4].

One notion of network shapes, first suggested by Gromov in a non-network group theoretic context [24], is via the *Gromov-hyperbolicity* of networks. First defined for infinite continuous metric space [1], the measure was later adopted for finite graphs. Usually this measure is defined via properties of *geodesic triangles* or equivalently via 4-node conditions, though Gromov originally defined the measure using Gromov-product nodes in [24]. Informally, any infinite metric space has a finite Gromov-hyperbolicity measure if it behaves metrically in the large scale as a *negatively curved* Riemannian manifold, and thus the value of this measure can be correlated to the standard

scalar curvature of a hyperbolic manifold. For a finite network the measure is related to the properties of the set of exact and approximate geodesics of the network. There is a *large* body of research works dealing with theoretical and empirical aspects of this measure, *e.g.*, see [14, 15, 17, 16, 18, 25] for theoretical aspects, and see [3, 5, 26] for applications to real-world networks (such as traffic congestions in a road network). Gromov-hyperbolicity is a *global* measure in the sense that it assigns one scalar value to the entire network.

A second notion of shape of a network can be obtained by extending Forman's discretization of Ricci curvature for (polyhedral or CW) complexes (the "Forman-Ricci curvature") [19] to networks. Informally, the Forman-Ricci curvature is applied to networks by *topologically associating* components (sub-networks) of a given network with higher-dimensional objects. The topological association itself can be carried out several ways. Although formulated relatively recently, there are already a number of papers investigating properties of these measures [20, 21, 22, 14, 23, 4].

In contrast to both of the above approaches, the network curvature considered in this paper is obtained via a discretization of curvatures from Riemannian manifolds to the network domain to capture metric properties of the manifold that are different from those captured by the Forman-Ricci curvature. More concretely, the network curvature studied in this paper is Ollivier's earth-mover's distances based discretization of Ricci curvature (the "*Ollivier-Ricci curvature*") [10, 11, 12, 13]. For some theoretical comparison between Ollivier-Ricci curvature and Forman-Ricci curvature over graphs, see [4].

1.3. Basic definitions and notations

Let $G = (V, E)$ be a given undirected unweighted graph. The following notations related to a graph G will be used subsequently:

- ▷ $\mathsf{Nbr}_G(x) = \{\, y \,|\, \{x, y\} \in E\}$ and $\deg_G(x) = |\, \mathsf{Nbr}_G(x) \,|$ are the set of neighbors and the degree, respectively, of a node x.
- ▷ $\mathrm{dist}_G(x, y)$ is the *distance* (*i.e.*, number of edges in a shortest path) between the nodes x and y in G.

The following standard notations and terminologies from the field of approximation algorithms are used to facilitate further discussions:

- ▷ OPT is the *value* of the objective of an optimal solution of the problem under discussion.
- ▷ A (α, ε)-estimate for a minimization problem under discussion is a polynomial-time algorithm that produces a solution whose objective value β satisfies $\mathsf{OPT} \leq \beta \leq \alpha\,\mathsf{OPT} + \varepsilon$. A $(1, \varepsilon)$-estimate is also called an *additive* ε-approximation.

2. Ollivier-Ricci curvatures: intuition, definitions and simple bounds

To define the Ollivier-Ricci curvatures for the components of a graph, we first need to use the following standard definition of the *earth mover's distance* (also called the L_1 *Wasserstein distance*) in the specific context of a *edge-weighted complete bipartite* graph.

Definition 1 (Earth mover's distance (EMD) over a edge-weighted complete bipartite graph). *Let $H = (V_L, V_R, w)$ be an* edge-weighted complete bipartite *graph with $w : V_L \times V_R \mapsto \mathbb{R}^+ \cup \{0\}$ being the edge-weight function, and let $\mathbb{P}_L : V_L \mapsto \mathbb{R}^+$ and $\mathbb{P}_R : V_R \mapsto \mathbb{R}^+$ be two arbitrary distributions over the nodes in V_L and V_R, respectively. The* earth mover's distance *corresponding to the distributions $\mathbb{P}_L$ and $\mathbb{P}_R$, denoted by $\textsc{Emd}_H(\mathbb{P}_L, \mathbb{P}_R)$ (or simply* EMD*), is the value of the objective function of an optimal solution of the following linear program that has a variable $z_{x,y}$ for every pair of nodes $x \in V_L$ and $y \in V_R$:*

$$\boxed{\begin{array}{ll} \textit{minimize} & \sum_{x \in V_L} \sum_{y \in V_R} w(x,y)\, z_{x,y} \\ \textit{subject to} & \sum_{y \in V_R} z_{x,y} = \mathbb{P}_L(x), \;\; \text{for all } x \in V_L \\ & \sum_{x \in V_L} z_{x,y} = \mathbb{P}_R(y), \;\; \text{for all } y \in V_R \\ & z_{x,y} \geq 0, \;\; \text{for all } x \in V_L \text{ and } y \in V_R \end{array}} \tag{1}$$

Let $G = (V, E)$ be an undirected unweighted graph. Consider an edge $e = \{u, v\} \in E$. Define the **edge-weighted complete bipartite** graph $G_{u,v} = (L^G_{u,v}, R^G_{u,v}, w^G_{u,v})$ as follows:

- ▷ $L^G_{u,v} = \{u\} \cup \mathsf{Nbr}_G(u)$,
- ▷ $R^G_{u,v} = \{v\} \cup \mathsf{Nbr}_G(v)$, and

▷ the edge-weight function $w^G_{u,v}$ is given by $w^G_{u,v}(u', v') = \text{dist}_G(u', v')$ for all $u' \in L^G_{u,v}, v' \in R^G_{u,v}$.

Let $\mathbb{P}^G_u$ and $\mathbb{P}^G_v$ denote the two uniform distributions over the nodes in $L^G_{u,v}$ and $R^G_{u,v}$, respectively, *i.e.*,

$$\forall\, x \in L^G_{u,v} : \mathbb{P}^G_u(x) = \frac{1}{1 + \deg_G(u)}$$
$$\forall\, x \in R^G_{u,v} : \mathbb{P}^G_v(x) = \frac{1}{1 + \deg_G(v)}$$

We can now state the precise definitions of the curvatures used in this paper.

▷ The **Ollivier-Ricci curvature of the edge $e = \{u, v\}$ of** G is defined as [10][4]

$$\boxed{\mathfrak{C}_G(e) \stackrel{\text{def}}{=} \mathfrak{C}_G(u, v) = 1 - \ \textsc{Emd}_{G_{u,v}}(\mathbb{P}^G_u, \mathbb{P}^G_v)} \tag{2}$$

▷ The **Ollivier-Ricci curvature of a node v** is calculated by taking the average of the Ollivier-Ricci curvatures of all the edges incident on v, *i.e.*,

$$\boxed{\mathfrak{C}_G(v) = \frac{1}{\deg_G(v)} \sum_{e=\{u,v\} \in E} \mathfrak{C}_G(e)} \tag{3}$$

▷ Finally, the **average Ollivier-Ricci curvature of a graph G** is calculated by taking the average of the Ollivier-Ricci curvatures of all the edges in G, *i.e.*,

$$\boxed{\mathfrak{C}_{\text{avg}}(G) = \frac{1}{|E|} \sum_{e \in E} \mathfrak{C}_G(e)} \tag{4}$$

[4]For this paper, it is crucial to note that the computation of $\mathfrak{C}_G(e)$ requires *only* the value of $\textsc{Emd}_{G_{u,v}}(\mathbb{P}^G_u, \mathbb{P}^G_v)$ and does *not* require an explicit enumeration of the solution (variable values) of the linear program (1). This distinction is important in the context of designing efficient local algorithms. For example, given a graph G with n nodes in which the maximum degree of any node is $O(1)$ and a constant $\varepsilon > 0$, one can compute a number that is an additive εn-approximation of the size of maximum matching of G in $O(1)$ time in expectation [27], but of course if we were required to output an actual maximum matching we would take at least $\Omega(n)$ time.

For easy quick reference, we explicitly write below the version of the linear program in (1) as used in the calculation of $\mathfrak{C}_G(u,v)$:

$$\boxed{\begin{array}{l} \textit{minimize } \sum_{x\in\{u\}\cup\mathsf{Nbr}_G(u)} \sum_{y\in\{v\}\cup\mathsf{Nbr}_G(v)} \mathrm{dist}_G(x,y)\, z_{x,y} \\ \textit{subject to} \\ \quad \sum_{y\in\{v\}\cup\mathsf{Nbr}_G(v)} z_{x,y} = \frac{1}{1+\deg_G(u)}, \ \text{ for all } x \in \{u\}\cup\mathsf{Nbr}_G(u) \\ \quad \sum_{x\in\{u\}\cup\mathsf{Nbr}_G(u)} z_{x,y} = \frac{1}{1+\deg_G(u)}, \ \text{ for all } y \in \{v\}\cup\mathsf{Nbr}_G(v) \\ \quad z_{x,y} \geq 0, \ \text{ for all } x \in \{v\}\cup\mathsf{Nbr}_G(v) \text{ and } y \in \{u\}\cup\mathsf{Nbr}_G(u) \end{array}} \qquad (\mathsf{LP}\text{-}\mathfrak{C}_G)$$

Assuming $\deg_G(u) \leq \deg_G(v)$, the linear program in $(\mathsf{LP}\text{-}\mathfrak{C}_G)$ has $\deg_G(u) \times \deg_G(v) = O((\deg_G(v))^2)$ variables and $\deg_G(u) + \deg_G(v) \leq 2\deg_G(v)$ constraints. The best time-complexity for solving the linear program in $(\mathsf{LP}\text{-}\mathfrak{C}_G)$ can be estimated as follows:

- ▷ Based on the state-of-the-art algorithms for solving linear program for this situation [28], an exact solution of $(\mathsf{LP}\text{-}\mathfrak{C}_G)$ can be found in $O((\deg_G(v))^{5/2})$ time.
- ▷ Based on the results in publications such as [29, 30], an additive ε-approximation of $(\mathsf{LP}\text{-}\mathfrak{C}_G)$ can be obtained in $\tilde{O}\big(\frac{1}{\varepsilon^2}\deg_G(u)\deg_G(v)\big) = \tilde{O}\big(\frac{1}{\varepsilon^2}(\deg_G(v))^2\big)$ time[5].

The following observation is crucial for this paper.

Observation 1. *The values* $\mathrm{dist}_G(x,y)$ *in the linear program in* $(\mathsf{LP}\text{-}\mathfrak{C}_G)$ *satisfy the property that* $\mathrm{dist}_G(x,y) \in \{0,1,2,3\}$.

It is not difficult to see that Observation 1 implies $0 \leq \textsc{Emd}_{G_{u,v}}(\mathbb{P}_u^G, \mathbb{P}_v^G) \leq 3$ and therefore $-2 \leq \mathfrak{C}_G(e) \leq 1$. For computing $\mathfrak{C}_G(u,v)$ and related quantities, **we assume that $\deg_G(u) \leq \deg_G(v)$ without any loss of generality throughout the rest of the paper**. Moreover, **we also assume without loss of generality that $\deg_G(v) = \omega(1)$ since otherwise $\textsc{Emd}_{G_{u,v}}(\mathbb{P}_u^G, \mathbb{P}_v^G)$ can be computed in $O(1)$ time**.

[5]The standard $\tilde{O}$ notation in algorithmic analysis hides poly-logarithmic terms, *e.g.*, terms like $\log^{4/3}\deg_G(v)$.

2.1. Intuition behind the discretization resulting in definition of $\mathfrak{C}_G(e)$

For an intuitive understanding of the definition of $\mathfrak{C}_G(e)$, we recall the notion of Ricci curvature for a smooth Riemannian manifold. The Ricci curvature at a point x in the manifold along a direction can be thought of transporting a small ball centered at x along that direction and measuring the "distortion" of that ball due to the shape of the surface by comparing the distance between the two small balls with the distance between their centers. In the definition of $\mathfrak{C}_G(e)$, the role of the direction is captured by the edge $e = \{u, v\}$, the roles of the balls at the two points are played by the two closed neighborhoods $L^G_{u,v}$ and $R^G_{u,v}$, and the role of the distance between the two balls is captured by the earth mover's distance between the two distributions $\mathbb{P}^G_u$ and $\mathbb{P}^G_v$ over the nodes in $L^G_{u,v}$ and $R^G_{u,v}$ on the metric space of shortest paths in G. For further intuition, see publications such as [10]. The Forman-Ricci curvature also assigns a number to each edge of the given graph, but the numbers are calculated in quite a different way from that in the Ollivier-Ricci curvature to capture different metric properties of the manifold.

2.2. Equivalent reformulation of linear program $(\mathsf{LP}\text{-}\mathfrak{C}_G)$ *when* $\deg_G(u) = \deg_G(v)$

The following claim holds based on results in prior publications such as [31, 32]. For the convenience of the reader, we provide a self-contained proof in the appendix.

Fact 1. [31, 32] *If* $\deg_G(u) = \deg_G(v)$ *then the following claims are true regarding some optimal solution of the linear program* $(\mathsf{LP}\text{-}\mathfrak{C}_G)$*:*

- **(*i*)** *The values of the variables* $z_{u',v'}$ *are either* 0 *or* $\frac{1}{\deg_G(v)}$.
- **(*ii*)** *The edges in* $\Big\{ \{u', v'\} \,|\, z_{u',v'} = \frac{1}{\deg_G(v)} \Big\}$ *form a minimum-weight perfect matching in* $G_{u,v}$ *that uses the zero-weight edges* $\{u', u'\}$ *for all* $u' \in \{u, v\} \cup \big(\mathsf{Nbr}_G(u) \cap \mathsf{Nbr}_G(v) \big)$.

Based on Fact 1, for the case when when $\deg_G(u) = \deg_G(v)$ an optimal solution of the linear program $(\mathsf{LP}\text{-}\mathfrak{C}_G)$ can be obtained by finding a *minimum-weight perfect matching* for a *complete edge-weighted bipartite* graph $H = (L, R, w)$ where

- $L = \mathsf{Nbr}_G(u) \setminus \big(\mathsf{Nbr}_G(v) \cup \{v\}\big)$,

- ▷ $R = \mathsf{Nbr}_G(v) \setminus \big(\mathsf{Nbr}_G(u) \cup \{u\}\big)$, and
- ▷ the edge-weight function $w : L \times R \mapsto \{1, 2, 3\}$ is given by $w(x, y) = \mathrm{dist}_G(x, y)$.

Note that $|L| = |R| = \deg_G(v) - 1 - |\mathsf{Nbr}_G(u) \cap \mathsf{Nbr}_G(v)|$. Letting $\mathcal{M}(H) \in \{|R|, |R| + 1, \ldots, 3\,|R|\}$ denote the total weight of a minimum-weight perfect matching of H, we have

$$\mathfrak{C}_G(e) = 1 - \frac{\mathcal{M}(H)}{1 + \deg_G(v)}$$

Proposition 1. *An additive $\varepsilon|R|$-approximation of $\mathcal{M}(H)$ implies an additive ε-approximation of $\mathfrak{C}_G(e)$, and vice versa.*

Proof. This follows from the facts that $|R| \leq \mathcal{M}(H) \leq 3\,|R|$ and $\deg_G(v) > |R|$. □

2.3. Some simple bounds for $\textsc{Emd}_{G_{u,v}}(\mathbb{P}_u, \mathbb{P}_v)$ *and* $\mathfrak{C}_G(e)$

We use a calculation similar to the one used in [31]. Extend the distributions $\mathbb{P}_u^G$ and $\mathbb{P}_v^G$ to $\mathbb{P}_u^{G'}$ and $\mathbb{P}_v^{G'}$ over $L_{u,v}^G \cup R_{u,v}^G$ by letting $\mathbb{P}_u^{G'}(x) = 0$ for $x \in R_{u,v} \setminus L_{u,v}$ and $\mathbb{P}_v^{G'}(x) = 0$ for $x \in L_{u,v} \setminus R_{u,v}$. For notational simplicity, let $k = \mathsf{Nbr}_G(u) \setminus \mathsf{Nbr}_G(v)$, $\ell = \mathsf{Nbr}_G(u) \cap \mathsf{Nbr}_G(v)$, and $m = \mathsf{Nbr}_G(v) \setminus \mathsf{Nbr}_G(u)$, thus $\deg_G(u) = k + \ell$ and $\deg_G(v) = m + \ell$. By straightforward calculation, the total variation distance (TVD) between $\mathbb{P}'_u$ and $\mathbb{P}'_v$ is

$$\begin{aligned} ||\,\mathbb{P}'_u - \mathbb{P}'_v\,||_{\text{TVD}} &= \tfrac{1}{2} \times \big(\tfrac{k-1}{k+\ell+1} + \tfrac{m-1}{m+\ell+1} + (\ell+2) \times \big(\tfrac{1}{k+\ell+1} - \tfrac{1}{m+\ell+1}\big)\big) \\ &= 1 - \tfrac{\ell+2}{\deg(v)+1} \end{aligned}$$

Since $1 \leq \mathrm{dist}_G(u', v') \leq 3$ for all $u', v' \in L_{u,v}^G \cup R_{u,v}^G, u' \neq v'$, by standard relationships between Emd and TVD (*e.g.*, see [33]) it follows that $||\,\mathbb{P}'_u - \mathbb{P}'_v\,||_{\text{TVD}} \leq \textsc{Emd}_{G_{u,v}}(\mathbb{P}_u, \mathbb{P}_v) \leq 3 \times ||\,\mathbb{P}'_u - \mathbb{P}'_v\,||_{\text{TVD}}$, thereby giving

$$-2 + \frac{3\ell + 6}{\deg_G(v) + 1} \leq \mathfrak{C}_G(e) \leq \frac{\ell + 2}{\deg_G(v) + 1}$$

Furthermore, if G has no cycles of length 5 or less containing e then $\mathrm{dist}_G(u', v') = 3$ for all $u', v' \in L_{u,v}^G \cup R_{u,v}^G$ and $\ell = 0$ giving $\mathfrak{C}_G(e) = \frac{2}{\deg_G(v)+1}$.

3. Synopsis of our results

The main goal of this paper is to study algorithmic complexities of efficient computation of our network curvature measures. To this effect, our main contributions are threefold:

- ▷ We relate various cases of our curvature computation problems via fine-grained reduction.
- ▷ We formalize the computational aspects of the curvature computation problems in suitable frameworks so that they can be studied by researchers in local algorithms.
- ▷ We provide the first known lower and upper bounds on queries for query-based algorithms for the curvature computation problems in our local algorithms framework. *En route*, we also illustrate a localized version of our fine-grained reduction.

A summary of our contribution in the rest of this paper is the following.

- ❑ In Section 4 we relate via Theorem 3 the minimum weight perfect matching problem on complete bipartite graphs with ternary weights to computing $\mathfrak{C}_G(e)$ via fine-grained reduction.
- ❑ In Section 5 we present our results for computing $\mathfrak{C}_G(e)$ in the framework of local algorithms.
  - ○ In Sections 5.1 – 5.2 we provide details of the query models relevant to our case and prior related works on these query models.
  - ○ In Sections 5.4 – 5.6 Theorem 4, Theorem 5 and Theorem 6 provide query bounds for exact or approximate calculations of the curvature using the query models. The bounds are succinctly summarized in Section 5.3 via Table 1.
- ❑ In Section 6 Lemma 8 provides our results for computing the Ollivier-Ricci curvature $\mathfrak{C}_G(v)$ for nodes and for computing the average Ollivier-Ricci curvature $\mathfrak{C}_{\text{avg}}(G)$ for graphs using "black box" additive approximation algorithms for $\mathfrak{C}_G(e)$ and neighbor queries.
- ❑ We conclude in Section 7 with some possible future research problems.

4. Fine-grained reduction: relating minimum weight perfect matching on complete bipartite graphs to computing $\mathfrak{C}_G(e)$

Frameworks for characterizing polynomial-time solvable problems via *fine-grained reduction* have garnered considerable attention in recent years (*e.g.*, see [34] for a survey and [35, 36, 37] for a few well-known results in this direction). Essentially these fine-grained reductions are used to show that, given two problems $\mathcal{A}$ and $\mathcal{B}$ and two constants $a, b > 0$, if an instance $\mathcal{I}_\mathcal{B}$ of size $|\mathcal{I}_\mathcal{B}|$ of problem $\mathcal{B}$ can be solved in $O(|\mathcal{I}_\mathcal{B}|^b)$ time then an instance $\mathcal{I}_\mathcal{A}$ of size $|\mathcal{I}_\mathcal{A}|$ of problem $\mathcal{A}$ can be solved in $O(|\mathcal{I}_\mathcal{A}|^a)$ time.

To begin, we first formally state the *minimum weight perfect* matching problem on *complete bipartite* graphs with *ternary* edge weights.

Definition 2 (**minimum weight perfect matching on complete bipartite graphs with ternary weights (**MPMCT**)).** *Given a complete edge-weighted bipartite graph $H = (A, B, w)$ where $|A| = |B|$ and $w : A \times B \mapsto \{1, 2, 3\}$ is the edge-weight function, find the value of $\frac{|\mathcal{M}|}{|A|}$ where $|\mathcal{M}|$ is the value (sum of weights of edges) in a minimum-weight perfect matching $\mathcal{M}$ of H.*

For MPMCT, exact solution takes $O(|A|^{5/2})$ time [28], and an ε-additive approximation takes $\tilde{O}\big(\frac{1}{\varepsilon^2}|A|^2\big)$ time. The following theorem related MPMCT to the problem of computing a solution of the linear program in (LP-$\mathfrak{C}_G$) via a fine-grained reduction.

Theorem 3. *Suppose that we have an algorithm $\mathfrak{A}$ that provides (α, ε)-estimate for* MPMCT *in $O(|A|^{2+\mu})$ time for some $\mu \geq 0$ for a given input instance $H = (A, B, w)$.*

Then, there exists an algorithm $\mathfrak{A}_<$ that provides the following estimates for the linear program in (LP-$\mathfrak{C}_G$) *in $O(\deg_G(v)^{2+\mu})$ time:*

- *(i)* *(α, ε)-estimate if $\deg_G(v)+1$ is an integral multiple of $\deg_G(u)+1$, and*
- *(ii)* *$(\alpha, \varepsilon + \delta)$-estimate (for $\delta > 0$) provided δ satisfies at least one of the following conditions:*
  - *(a)* $\deg_G(u) \leq (\delta/3) \times \deg_G(v)$, *or*
  - *(b)* $\deg_G(u) \geq (1 - (\delta/3)) \times \deg_G(v)$.

Remark 1. *An illustration of the result in Theorem* 3 *is as follows. Suppose that we can solve* MPMCT *exactly in* $O(|A|^{2.4})$ *time (implying* $\alpha = 1$ *and* $\varepsilon = 0$*). Then, such an algorithm can be used to obtain a* $\deg_G(v)^{-1/2}$*-additive approximation of* $(\mathsf{LP}\text{-}\mathfrak{C}_G)$ *(i.e.,* $\delta = \deg_G(v)^{-1/2}$*) in* $O((\deg_G(v))^{2.4})$ *time provided at least one of the following conditions hold:* (a) $\deg_G(u) \leq \frac{\sqrt{\deg_G(v)}}{3}$, (b) $\deg_G(u) \geq \deg_G(v) - \frac{\sqrt{\deg_G(v)}}{3}$*, or* (c) $\deg_G(v)+1$ *is an integral multiple of* $\deg_G(u)+1$*. Such a result will improve the best possible running time for a* $\deg_G(v)^{-1/2}$*-additive approximation of* $(\mathsf{LP}\text{-}\mathfrak{C}_G)$*.*

Proof. Let $\mathsf{Nbr}_G(u) \cup \{u\} = \{x_1, \ldots, x_{\deg(u)+1}\}$, and $\mathsf{Nbr}_G(v) \cup \{v\} = \{y_1, \ldots, y_{\deg(v)+1}\}$, where $x_{\deg_G(u)} = y_{\deg_G(v)} = u$ and $x_{\deg_G(u)+1} = y_{\deg_G(v)+1} = v$. Let $\deg_G(v) + 1 = a\,(\deg_G(u)+1) + b$ for two integers $a \geq 1$ and $0 \leq b < \deg_G(u)+1$. We construct a new graph $G'_{u,v} = (L^{G'}_{u,v}, R^{G'}_{u,v}, w^{G'}_{u,v})$ from $G_{u,v}$ in the following manner:

- ▷ We set $R^{G'}_{u,v} = R^{G}_{u,v}$.
- ▷ Every node x_i is replaced by a nodes $x_i^1, \ldots, x_i^a$ in $L^{G'}_{u,v}$. Moreover, we have b additional "special" nodes $r_1, \ldots, r_b$ in $L^{G'}_{u,v}$. Note that after these modifications $|L^{G'}_{u,v}| = |R^{G'}_{u,v}| = 1 + \deg_G(v)$.
- ▷ We set the new weights $w^{G'}_{u,v}$ as follows:

$$\begin{aligned} w^{G'}_{u,v}(x_i^j, y_\ell) &= \operatorname{dist}_G(x_i, y_\ell) \text{ for } i \in \{1, \ldots, \deg_G(u)+1\}, \\ &\qquad\qquad j \in \{1, \ldots, a\}, \text{ and } \ell \in \{1, \ldots, \deg_G(v)+1\} \\ w^{G'}_{u,v}(r_i, y_\ell) &= 3 \text{ for } i \in \{1, \ldots, b\}, \text{ and } \ell \in \{1, \ldots, \deg_G(v)+1\} \end{aligned}$$

- ▷ The two *new* probability distributions $\mathbb{P}_u^{G'_{u,v}}$ and $\mathbb{P}_v^{G'_{u,v}}$ over the nodes in $L^{G'}_{u,v}$ and $R^{G'}_{u,v}$ are as follows: $\mathbb{P}_v^{G'_{u,v}}(x) = \mathbb{P}_v^{G}(x)$ for all $x \in \{y_1, \ldots, y_{\deg_G(v)+1}\}$, and $\mathbb{P}_u^{G'_{u,v}}(x) = \frac{1}{1+\deg_G(v)}$ for all $x \in \bigcup_{i,j}\{x_i^j\} \cup \{r_1, \ldots, r_b\}$.

Since $|L^{G'}_{u,v}| = |R^{G'}_{u,v}| = 1 + \deg_G(v)$, using the reformulations as discussed in Section 2.2 it follows that $G'_{u,v}$ is a valid instance $H = (A, B, w)$ of MPMCT with $|A| = \deg_G(v)+1$ and $w(p,q) = w^{G'}_{u,v}(p,q)$. Note that building the graph $G'_{u,v}$ takes $O((\deg_G(v))^2)$ time, and algorithm $\mathfrak{A}$ provides a (α, ε)-estimate for $\textsc{Emd}_{G'_{u,v}}(\mathbb{P}_u^{G'_{u,v}}, \mathbb{P}_v^{G'_{u,v}})$ in $O((\deg_G(v))^{2+\mu})$ time. Thus, to complete the proof it suffices to show that

$$\textsc{Emd}_{G_{u,v}}(\mathbb{P}_u^{G}, \mathbb{P}_v^{G}) \leq \textsc{Emd}_{G'_{u,v}}(\mathbb{P}_u^{G'_{u,v}}, \mathbb{P}_v^{G'_{u,v}}) \leq \textsc{Emd}_{G_{u,v}}(\mathbb{P}_u^{G}, \mathbb{P}_v^{G}) + \delta$$

The linear program for $\textsc{Emd}_{G'_{u,v}}(\mathbb{P}_u^{G'_{u,v}}, \mathbb{P}_v^{G'_{u,v}})$ is a straightforward modified version of $(\mathsf{LP}\text{-}\mathfrak{C}_G)$ with appropriate change of subscripts of the variables. We will refer to this modified version by $(\mathsf{LP}\text{-}\mathfrak{C}_G)'$.

We can show $\textsc{Emd}_{G'_{u,v}}(\mathbb{P}_u^{G'_{u,v}}, \mathbb{P}_v^{G'_{u,v}}) \leq \textsc{Emd}_{G_{u,v}}(\mathbb{P}_u^{G}, \mathbb{P}_v^{G}) + \delta$ as follows. Consider an *optimal* solution of the linear program $(\mathsf{LP}\text{-}\mathfrak{C}_G)$ of value $\textsc{Emd}_{G_{u,v}}(\mathbb{P}_u^{G}, \mathbb{P}_v^{G})$. From this solution we can create a *feasible solution* of the linear program $(\mathsf{LP}\text{-}\mathfrak{C}_G)'$ in the following manner.

- ▷ For $i = 1, \ldots, \deg_G(u) + 1$ and $j = 1, \ldots, \deg_G(v) + 1$, if $z_{x_i,y_j} > 0$ then distribute the value of z_{x_i,y_j} among the corresponding variables of $(\mathsf{LP}\text{-}\mathfrak{C}_G)'$ as follows:
  - Repeatedly select a variable from $\{x_i^1, \ldots, x_i^a\}$, say x_i^ℓ, such that $x_i^\ell < \frac{1}{1+\deg_G(v)}$. Increase x_i^ℓ to $\min\left\{\frac{1}{(1+\deg_G(v))}, z_{x_i,y_j}\right\}$, and decrease z_{x_i,y_j} by the amount by which x_i^ℓ was increased. Note that $w_{u,v}^{G'}(x_i, y_j) = \text{dist}_G(x_i, y_j)$. Repeat this step until z_{x_i,y_j} becomes zero or no such variable x_i^ℓ exists.
  - If $z_{x_i,y_j} > 0$ after the previous step ends then execute the following steps. Repeatedly select a variable from $\{r_1, \ldots, r_b\}$, say r_ℓ, such that $r_\ell < \frac{1}{1+\deg_G(v)}$. Increase r_ℓ to $\min\left\{\frac{1}{(1+\deg_G(v))}, z_{x_i,y_j}\right\}$, and decrease z_{x_i,y_j} by the amount by which r_ℓ was increased. Note that $w_{u,v}^{G'}(x_i, y_j) \leq \text{dist}_G(x_i, y_j) + 3$. Repeat this step until z_{x_i,y_j} becomes zero.

A straightforward calculation show that $\textsc{Emd}_{G'_{u,v}}(\mathbb{P}_u^{G'_{u,v}}, \mathbb{P}_v^{G'_{u,v}}) \leq \textsc{Emd}_{G_{u,v}}(\mathbb{P}_u^{G}, \mathbb{P}_v^{G}) + \frac{3b}{\deg_G(v)+1}$. Therefore it suffices if we have $\frac{3b}{\deg(v)+1} \leq \delta$. If $\deg_G(u) + 1$ is an integral multiple of $\deg_G(v) + 1$ then $b = 0$ and this proves the claim in (i). Otherwise, since $b < \deg_G(u) + 1 \leq \deg_G(v) + 1$ and $b \leq (\deg_G(v) + 1) - (\deg_G(u) + 1) = \deg_G(v) - \deg_G(u)$ we get

$$\deg_G(u) \leq (\delta/3) \times \deg_G(v) \Rightarrow b < \deg_G(u) + 1 \leq (\delta/3) \times \deg_G(v) + 1 \Rightarrow \frac{3b}{\deg(v)+1} < \frac{\delta \times \deg_G(v)+1}{\deg(v)+1} \leq \delta$$

$$\deg_G(u) \geq (1 - (\delta/3)) \times \deg_G(v) \Rightarrow \deg_G(v) - \deg_G(u) \leq (\delta/3) \times \deg_G(v) \Rightarrow \frac{3b}{\deg(v)+1} \leq \frac{\delta \times \deg_G(v)}{\deg(v)+1} < \delta$$

The proof of $\textsc{Emd}_{G_{u,v}}(\mathbb{P}_u^{G}, \mathbb{P}_v^{G}) \leq \textsc{Emd}_{G'_{u,v}}(\mathbb{P}_u^{G'_{u,v}}, \mathbb{P}_v^{G'_{u,v}})$ is similar. □

5. Computing $\mathfrak{C}_G(e)$ in the framework of local algorithms

By now designing local algorithms for efficient solution of graph-theoretic problems has become a well-established research area in theoretical computer science and data mining with a large body of publications (*e.g.*, see [38, 27, 39]). A basic idea behind many of these algorithms is to suitably sample a small "local" neighborhood of the graph to infer the value of some non-local property of a graph. Frameworks for graph-theoretic applications of local algorithms hinges on the following two premises:

- ▷ We assume that our algorithm has a list of all nodes in the graph in a suitable format that allows for sampling a node based on some distribution.
- ▷ The edges and their weights are *not* known to our algorithm *a priori*. Instead, the algorithm uses a "query" on a node or a pair of nodes to discover an edge and its weight. Different query models for local algorithms arise based on what kind of queries are allowed. Later in Section 5.2 we will provide details of query models that are applicable to our problems.
- ▷ The performance of the algorithm is measured by the *number* of queries used.

Additional notations and conventions

For the case when $\deg_G(u) = \deg_G(v)$, we will use the reformulations of the linear program (LP-$\mathfrak{C}_G$) as discussed in Section 2.2 and the associated notations contained therein. We will use the following additional notations and conventions related to the graph $H = (L, R, w)$ mentioned in Section 2.2:

- ▷ $|L| = |R| = n$, $L = \{u_1, \ldots, u_n\}$ and $R = \{v_1, \ldots, v_n\}$.
- ▷ For $j \in \{1, 2\}$ $\deg_{H,j}(x)$ denotes the number of edges of weight j incident on node x in a graph H.

Note that H has $2n$ nodes. Moreover, for any edge-weighted graph $F = (V, E, w)$ with $w : E \mapsto \mathbb{R}$, wt-$\deg_F(u) = \sum_{v:\{u,v\}\in E} w(u, v)$ denotes the *weighted degree* of node u in F, and $\mathcal{M}(F)$ denotes the total weight of a minimum-weight perfect matching of F.

5.1. Prior related works

Designing sublinear time and sketching algorithms for the general earth mover's distance on the shortest path metric for arbitrary graphs have been investigated in prior research papers such as [40, 41]. In particular, for an edge-weighted tree with W being the maximum weight of any edge and for any two unknown probability distributions on the nodes, the authors in [40] show that an estimate of the EMD with ε-additive error can be achieved by using $\tilde{O}(\frac{W^2n^2}{\varepsilon^2})$ samples from the two distributions and observes that their algorithm is optimal up to polylog factors. To the best of our knowledge, local algorithms for computing the Ollivier-Ricci curvatures of a graph have *not* been investigated explicitly before.

5.2. Query models for edge-weighted complete bipartite graphs

Two standard query models that appear in the local algorithms literature for unweighted graphs (*e.g.*, see [38]) are as follows: the *node-pair* query model (the query is a pair of nodes and the answer is whether an edge between them exists or not), and the *neighbor* query model (the query is a node and the answer is a random not-yet-explored adjacent node if it exists). Since our given graph is an edge-weighted complete bipartite graph $H = (L, R, w)$ via the reformulation described in Section 2.2, natural extensions lead to the following query models for our case:

- ▷ **weighted node-pair query model**: the query is a pair of nodes x, y and the answer is the *weight* $w(x, y)$.
- ▷ **neighbor query model**: the query is a node x and the answer is a random "not-yet-explored" node adjacent to x (if no such node exists then the query returns a special symbol to indicate that). *Note that such a query does not give any useful information* (*beyond simply picking a node uniformly at random*) *for the graph H since it is a complete graph.* We will only use this type of query for the entire given graph G for computing $\mathfrak{C}_G(v)$ and $\mathfrak{C}_{\text{avg}}(G)$ in Section 6.
- ▷ **weighted neighbor query model**: the query is (x, y) where x is a node and y is a number, and the answer is a random "not-yet-explored" node z such that $w(x, z) = y$ (if no such node exists then the query returns a special symbol to indicate that).

▷ **weighted selective degree query model**: the query is (x, y) where x is a node and y is a number, and the answer is the number of edges of weight y that are incident on x.

5.3. Summary of our query bounds on computing $\mathfrak{C}_G(e)$

For the convenience of the reader, we summarize our query bounds for computing $\mathfrak{C}_G(e)$ in Table 1. Subsequent sub-sections in this section provide proofs of these bounds.

	query types	additive approx.	expected # of queries	result(s)	additional remark(s)
lower bounds	weighted node-pair	exact computation	$> \frac{(\deg_G(v)-1)^2}{6}$	Theorem 4(a)-(i)	①
	weighted neighbor	exact computation	$> \frac{\deg_G(v)-1}{2}$	Theorem 4(a)-(ii)	
	weighted node-pair	$2-\varepsilon_1$	$> \frac{\deg_G(v)-1}{6}$	Theorem 4(b)	
upper bounds	weighted neighbor	$1+\varepsilon_2$	$O(1)$	Theorem 5(a)	②
	weighted neighbor	$\frac{1}{2}+\varepsilon_2$	$O(1)$	Theorem 5(b)	③
	weighted neighbor	$1+\varepsilon_2+\delta$	$O(1)$	Corollary 7(i)	④
	weighted neighbor	$\frac{1}{2}+\varepsilon_2+\delta$	$O(1)$	Corollary 7(ii)	⑤

① *even if* $\deg_{H,1}(x) \leq 1$, $\deg_{H,2}(x) = 0$ for every node x, and *any* number of weighted selective degree queries are allowed.

② if $\deg_{H,1}(x) = O(1)$ for every node x, *and* $\deg_G(u) = \deg_G(v)$.

③ if *both* $\deg_{H,1}(x) = O(1)$ and $\deg_{H,2}(x) = O(1)$ for every node x, and $\deg_G(u) = \deg_G(v)$.

④ if $\deg_{H,1}(x) = O(1)$ for every node x, and $\deg_G(u) \geq (1 - (\delta/3)) \times \deg_G(v)$.

⑤ if *both* $\deg_{H,1}(x) = O(1)$ and $\deg_{H,2}(x) = O(1)$ for every node x, and $\deg_G(u) \geq (1 - (\delta/3)) \times \deg_G(v)$.

Table 1: A summary of query bounds for computing $\mathfrak{C}_G(e)$; $\varepsilon_1, \varepsilon_2, \delta$ are arbitrary constants satisfying $0 < \varepsilon_1 \leq 2$ and $\varepsilon_2, \delta > 0$.

5.4. Lower bounds on number of queries for computing $\mathfrak{C}_G(e)$

Note that for query lower bounds it suffices to prove the lower bound for complete edge-weighted bipartite graph reformulations of the problem as discussed in Section 2.2. Any complete bipartite graph $H = (L, R, w)$ used in our lower bound proofs will satisfy $L \cap R = \emptyset$, thereby implying $\boldsymbol{n = \deg_G(v) - 1}$. Since we provide our inputs in the form

of such graphs H, we first need to show that there exists a graph G with the edge $\{u, v\}$ such that $G_{u,v} = H$ in the notations used in Section 2.2.

Proposition 2. *Given any complete edge-weighted bipartite graph $H = (L, R, w)$ where $w : L \times R \mapsto \{1, 2, 3\}$ there exists a graph $G = (V, E)$ such that $G_{u,v} = H$.*

Proof. Start with the edge $\{u, v\}$ in G, connect the nodes $u_1, \ldots, u_n$ to u, and connect the nodes $v_1, \ldots, v_n$ to v. For every pair of nodes $(u_i, v_j) \in L \times R$, if $w(u_j, v_j) = 1$ then add the edge $\{u_i, v_j\}$ to G. Otherwise if $w(u_j, v_j) = 2$ then add a new node $x_{i,j}$ to G and add the two edges $\{u_i, x_{i,j}\}$ and $\{x_{i,j}, v_j\}$ to G. □

A common thread in our lower bound proofs is the following easy but crucial observation.

Observation 2. *Suppose that we have two separate classes of (complete edge-weighted bipartite, as described in Section 2.2) graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, two numbers $1 \leq \alpha < \beta \leq 3$, and an algorithm $\mathcal{A}$ such that the following holds:*

- ▷ *Every graph $H \in \mathcal{G}_1$ satisfies $n \leq \mathcal{M}(H) \leq \alpha\, n$.*
- ▷ *Every graph $H \in \mathcal{G}_2$ satisfies $\beta\, n \leq \mathcal{M}(H) \leq 3n$.*
- ▷ *Given a graph from $\mathcal{G}_1 \cup G_2$, algorithm $\mathcal{A}$ cannot determine in which class the given graph belongs.*

Then, using Proposition 1, it follows that algorithm $\mathcal{A}$ cannot provide an additive $(\beta - \alpha - \varepsilon)$-approximation of $\mathfrak{C}_G(e)$ for any constant $\varepsilon > 0$.

Our proofs in Theorem 4 for lower bounds on the number of queries will use the well-known *Yao's minimax principle* for randomized algorithms [42]. Namely, we will construct two separate classes $\mathcal{G}_1$ and $\mathcal{G}_2$ of graphs and show that any *deterministic* algorithm that picks graphs uniformly at random from these two classes will need *at least* a certain number of queries, say q, to be able to decide from which class the graph was selected with at least a certain probability, say p. Then, the expected number of queries performed by any deterministic algorithms on inputs drawn from the aforementioned distribution is at least pq, and thus by the minimax principle the expected number of queries for any randomized algorithm over all possible inputs is also at least pq. Note that since our input instances are complete bipartite graphs, two graphs are differentiated based on the assignments of weights to all possible edges (see [38] for further elaborations on this point).

Theorem 4. *Consider any local algorithm that is allowed to make an unlimited number of weighted selective degree queries. Let Q be the expected number of queries,* ***excluding*** *all weighted selective degree queries, performed by the algorithm for computing $\mathfrak{C}_G(e)$. Then the following claims hold.*

(*a*) *Suppose that we want to compute $\mathfrak{C}_G(e)$ exactly. Then the following bounds hold.*

- **(*i*)** *$Q > n^2/6$ if the queries used are weighted node-pair queries.*
- **(*ii*)** *$Q > n/6$ if the queries used are weighted neighbor queries.*

(*b*) *For every $0 < \varepsilon < 2$, any randomized algorithm computing an additive $(2-\varepsilon)$-approximation of $\mathfrak{C}_G(e)$ requires $Q > n/6$ weighted node-pair queries.*

Proof. All the bipartite graphs $H = (L, R, w)$ in our proofs will satisfy that $\deg_{H,1}(x) = 1$ and $\deg_{H,2}(x) = 0$ for every node $x \in L \cup R$, and therefore any number of weighted selective degree queries will provide *no* information about the value of $\mathfrak{C}_G(e)$.

Proof of (*a*)

Corresponding to every node pair (u_i, v_j) with $u_i \in L$ and $v_j \in R$, the class $\mathcal{G}_1$ contains a graph in which $w(u_i, v_j) = 1$ and all other edges have weight 3. The class $\mathcal{G}_2$ contains just one graph in which all edge weights are set to 3. Note that the minimum weight of a perfect matching for each graph in $\mathcal{G}_1$ is $3n - 2$, whereas the minimum weight of a perfect matching for the graph in $\mathcal{G}_2$ is $3n$.

Proof of (*a*)-(*i*)

Suppose that our algorithm has already made t queries (edges) $e_1, \ldots, e_t$ for $t < n^2 - 1$ with $w(e_1) = \cdots = w(e_t) = 3$ and let e_{t+1} be the next query. Consider a graph $G_1 \in \mathcal{G}_1$ that is consistent with the first t queries with $w(e_{t+1}) = 1$. Note that there is exactly one such graph in $\mathcal{G}_1$. Since there are at least $n^2 - (t+1)$ node pairs (edges) that have not been queried after the $(t+1)^{\text{th}}$ query, we have at least $n^2 - (t+1)$ distinct graphs in $\mathcal{G}_1$ with $w(e_{t+1}) = 3$ that is consistent with the first t queries (set the weight of exactly one of the $n^2 - (t+1)$ edges to 1 and the weight of the remaining edges to 3). Since graphs are selected uniformly at random from $\mathcal{G}_1$ it follows that $\Pr[w(e_{t+1}) = 1 \,|\, w(e_1) = \cdots = w(e_t) = 3] \leq \frac{1}{n^2-(t+1)}$. Summing over all t, we get

$$\begin{aligned}&\Pr[\text{ number of queries needed is at least } t+1\ ]\\ &\quad = 1 - \Pr[\text{ one of the } t \text{ queries contain an edge of weight 1 }]\\ &\quad \geq 1 - \tfrac{t}{n^2-(t+1)}\end{aligned}$$

Putting $t = {}^{n^2}\!/_3$, the probability that "the number of queries is at least $1 + {}^{n^2}\!/_3$" is at least ${}^1\!/_2$.

Proof of (*a*)-(*ii*)

Suppose that our algorithm has already made t queries (nodes, weights) $(x_1, y_1), \ldots, (x_t, y_t) \in (L \cup R) \times \{1, 2, 3\}$ for $t < n-1$. Let $e_1 = \{x_1, x'_1\}, \ldots, e_t = \{x_t, x'_t\}$ be the answers (edges) to these queries with $w(e_1) = \cdots = w(e_t) = 3$ and let (x_{t+1}, y_{t+1}) be the next query that reveals the weight of an edge $e_{t+1} = \{x_{t+1}, x'_{t+1}\}$. Consider a graph $G_1 \in \mathcal{G}_1$ that is consistent with the first t queries with $w(e_{t+1}) = y_{t+1} = 1$. Note that there is exactly one such graph in $\mathcal{G}_1$. Since there are at least $n-(t+1)$ nodes in each of L and R that have not been queried after the $(t+1)^{\text{th}}$ query, we have at least $(n-(t+1))^2$ distinct graphs in $\mathcal{G}_1$ with $w(e_{t+1}) = 3$ that is consistent with the first t queries (set the weight of exactly one edge among these nodes to 1 and the weights of all remaining edges to 3). Since graphs are selected uniformly at random from $\mathcal{G}_1$ it follows that $\Pr[w(e_{t+1}) = 1 \,|\, w(e_1) = \cdots = w(e_t) = 3] \leq \frac{1}{(n-(t+1))^2}$. Summing over all t, we get

$$\begin{aligned}&\Pr[\text{ number of queries needed is at least } t+1\ ]\\ &\quad = 1 - \Pr[\text{ one of the } t \text{ queries contain an edge of weight 1 }]\\ &\quad \geq 1 - \tfrac{t}{(n-(t+1))^2}\end{aligned}$$

Putting $t = {}^n\!/_2$, the probability that "the number of queries is at least $1 + {}^n\!/_2$" is at least $1 - \frac{2}{n}$.

Proof of (*b*)

Corresponding to each of the possible $n!$ perfect matchings, the class $\mathcal{G}_1$ contains a graph in which the edges in the matching have weight 1 and all other non-matching edges have weight 3. The class $\mathcal{G}_2$ contains just one graph in which all edge weights are set to 3. Note that the minimum weight of a perfect matching for each graph in $\mathcal{G}_1$ is n, whereas the minimum weight of a perfect matching for the graph in $\mathcal{G}_2$ is $3n$. Suppose that our algorithm has already made t (edge) queries $e_1, \ldots, e_t$ for $t < n-1$ with $w(e_1) = \cdots = w(e_t) = 3$ and let e_{t+1} be the next (edge) query.

We first show that as long as $t < n$ there exists at least one graph in $\mathcal{G}_1$ that is consistent with the weight assignments of the first t queries. Consider a random perfect matching $M = \{\{u_1, v_{\pi(1)}\}, \ldots, \{u_n, v_{\pi(n)}\}\}$ given by a random permutation π of $1, \ldots, n$. The probability of the event $\mathcal{E}_j$ that the j^{th} query e_j is in M is $\frac{(n-1)!}{n!} = 1/n$. It follows that $\Pr[\wedge_{j=1}^{t} \overline{\mathcal{E}_j}] = 1 - \Pr[\vee_{j=1}^{t} \mathcal{E}_j] \geq 1 - \sum_{j=1}^{t} \Pr[\mathcal{E}_j] \geq 1 - \frac{t}{n} > 0$ and therefore $\mathcal{G}_1$ contains at least one such graph.

Assume without loss of generality that $e_{t+1} = (u_n, v_n)$ and let M be a perfect matching of the nodes in L and R, say $M = \{\{u_1, v_1\}, \ldots, \{u_n, v_n\}\}$, that is consistent with the first t queries, and includes e_{t+1} as a matched edges (note that $w(u_1, v_1) = \cdots = w(u_n, v_n) = 1$). If such a matching M does not exist then $\Pr[w(u_n, v_n) = 1 \,|\, w(e_1) = \cdots = w(e_t) = 3] = 0$. Otherwise, note that there are at least $n-t$ nodes in each of L and R, say $u_1, \ldots, u_{n-t} \in L$ and $v_1, \ldots, v_{n-t} \in R$, such that the edges (u_n, v_j) and (u_j, v_n) for $j = 1, \ldots, n-t$ have not been queried yet. For every such perfect matching M, we can then construct a set S_M of at least $n-t$ *distinct* perfect matchings with $w(e_{t+1}) = 3$ that is consistent with the first t queries as follows: in the ℓ^{th} perfect matching set $w(u_\ell, v_\ell) = w(u_n, v_n) = 3$ and set $w(u_n, v_\ell) = w(u_\ell, v_n) = 1$. It is also easy to see that any two matchings from two different sets S_M and $S_{M'}$ differ in at least one edge. Since graphs are selected uniformly at random from $\mathcal{G}_1$ it follows that $\Pr[w(u_n, v_n) = 1 \,|\, w(e_1) = \cdots = w(e_t) = 3] \leq \frac{1}{n-t}$. Summing over all t, we get

$$\begin{aligned}\Pr[\text{ number of queries needed is at least } t+1\ ] \\ = 1 - \Pr[\text{ any of the } t \text{ queries contain an edge of weight 1 }] \\ > 1 - \tfrac{t}{n-(t-1)}\end{aligned}$$

Putting $t = n/3$, the probability that "the number of queries is at least $1 + n/3$" is at least $1/2$. □

5.5. Upper bounds on number of queries for computing $\mathfrak{C}_G(e)$ when $\deg(u) = \deg(v)$

The proofs in Theorem 4 do *not* use any edge of weight 2 and have *at most* one edge of weight 1 incident on any node with the additional restriction that these edges of weight 1 provide a unique matching for the nodes that are end-points of these edges. In this section we show that if weighted neighbor queries are allowed then $O(1)$ expected number of queries will suffice for a non-trivial additive approximation for a class of weighted complete

bipartite graphs that properly includes the instances generated by the proofs in Theorem 4 (note that for the instances (graphs) generated by the proofs in Theorem 4 we have $\deg_{H,1}(x) \leq 1$ and $\deg_{H,2}(x) = 0$ for *every* node x).

Theorem 5. *Assume that* $\deg_G(u) = \deg_G(v)$*, and let* $d, \varepsilon > 0$ *be two fixed constants. Then, using* $O(1)$ *expected number of weighted neighbor queries*[6] *we can obtain the following type of approximations for* $\mathfrak{C}_G(e)$*:*

(*a*) *an additive* $(1+\varepsilon)$*-approximation when* $\max_x\{\deg_{H,1}(x)\} \leq d$*, and*

(*b*) *an additive* $\left(\frac{1}{2}+\varepsilon\right)$*-approximation when* $\max_x\{\deg_{H,1}(x)\} \leq d$ *and* $\max_x\{\deg_{H,2}(x)\} \leq d$*.*

Remark 2. *Let* m_1*,* m_2 *and* m_{12} *be as defined in the proof of this theorem. The bounds in Theorem* 5 *are tight in the sense that no algorithm that knows only estimates of* m_1 *(resp. estimates of* m_1, m_2, m_{12}*) can provide better additive ratios for parts* (a) *(resp.* (b)*); see* Fig. 1 (a)–(b)*. The example in* Fig. 1 (c) *shows that no algorithm can provide better than additive* $\frac{2}{3}$*-approximation for the case in Theorem* $5(b)$ *if the estimate for* m_{12} *is not used.*

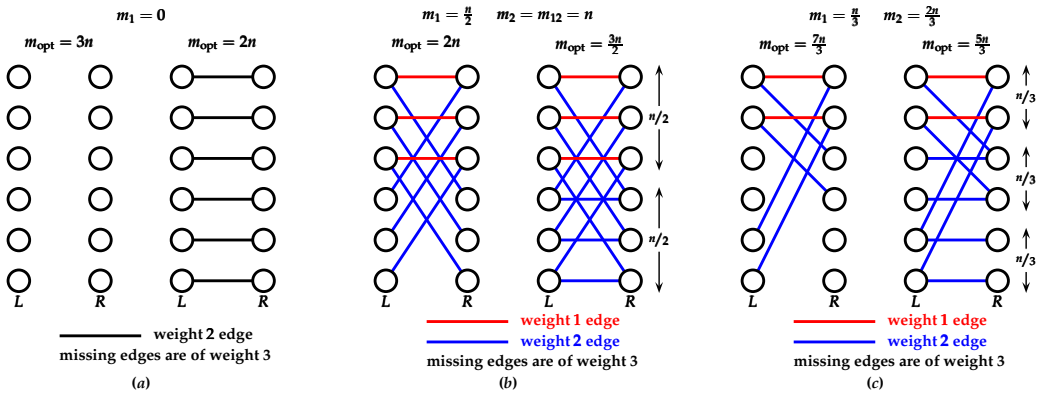

Figure 1: **(*a*)** Example showing tightness of bounds in Theorem 5 when only estimate for m_1 is known. **(*b*)** Example showing tightness of bounds in Theorem 5 when estimates for m_1, m_2, m_{12} are known; **(*c*)** Example showing that better than additive $\frac{2}{3}$-approximation is not possible if only estimates for m_1 and m_2 are used for the case in Theorem $5(b)$.

[6]The constant in $O(1)$ depends on d and ε.

Proof. Since $\deg_G(u) = \deg_G(v)$ we can use the reformulations of the linear program (LP-$\mathfrak{C}_G$) outlined in Section 2.2. Let $\delta > 0$ be a *constant* to be fixed later. Let H_1, H_2 and H_{12} be the subgraphs of H induced by the edges in H of weight 1, edges in H of weight 2, and edges in H of weights 1 and 2, respectively. Fix *maximum-cardinality* matchings $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_{12}$ of H_1, H_2 and H_{12} having $m_1\,n$, $m_2\,n$ and $m_{12}\,n$ edges, respectively. Also, fix a minimum-weight perfect matching M_{opt} of H of *total* weight $m_{\text{opt}}\,n$, and let $m_{\text{opt},\ell}\,n$ be the number of edges of weight $\ell \in \{1,2,3\}$ in M_{opt}. The following inequalities will be useful during the rest of the proof:

$$m_{\text{opt},1} \leq m_1, \quad m_{\text{opt},2} \leq m_2, \quad m_{\text{opt},3} \geq 1 - m_{12}, \quad m_{12} \geq \max\{m_1, m_2\},$$

$$\begin{aligned} m_{\text{opt}} = m_{\text{opt},1} + 2\,m_{\text{opt},2} + 3(1 - m_{\text{opt},1} - m_{\text{opt},2}) &= 3 - 2\,m_{\text{opt},1} - m_{\text{opt},2} \\ &\geq 2 - 2\,m_{\text{opt},1} \geq 2 - 2\,m_1 \end{aligned}$$

Let $\mathcal{M}_s$ be a perfect matching of H generated by taking all the edges (of weight 1) in $\mathcal{M}_1$ and pairing the remaining nodes from L and R arbitrarily. Note that the total weight $m_s\,n$ of the edges in $\mathcal{M}_s$ satisfies $m_{\text{opt}} \leq m_s$ and $m_s \leq m_1 + 3(1 - m_1) = 3 - 2\,m_1$; thus it follows that $3 - 2\,m_1 \geq m_{\text{opt}}$. Similarly, taking $\mathcal{M}_s$ to be a perfect matching of H of total weight $m_s\,n$ generated by taking all the edges (of weight 2) in $\mathcal{M}_2$ and pairing the remaining nodes from L and R arbitrarily we get $m_{\text{opt}} \leq m_s$ and $m_s \leq 2\,m_2 + 3(1 - m_2) = 3 - m_2$; thus it follows that $3 - m_2 \geq m_{\text{opt}}$.

Our algorithm proceeds in two main steps. The first common step in our algorithm for both **(*a*)** and **(*b*)** is to determine the set of nodes in L and R from $\mathsf{Nbr}_G(u)$ and $\mathsf{Nbr}_G(v)$. This can be done by comparing the list of nodes in $\mathsf{Nbr}_G(u)$ and $\mathsf{Nbr}_G(v)$ to identify all nodes in $\mathsf{Nbr}_G(u) \cap \mathsf{Nbr}_G(v)$ and setting $L = \mathsf{Nbr}_G(u) \setminus \big(\mathsf{Nbr}_G(u) \cap \mathsf{Nbr}_G(v)\big)$, $R = \mathsf{Nbr}_G(v) \setminus \big(\mathsf{Nbr}_G(u) \cap \mathsf{Nbr}_G(v)\big)$. Note that this step does *not* use any query at all. The remaining parts of our algorithms will only use weighted neighbor queries (x, s) for $x \in L \cup R$ and $s \in \{1, 2\}$.

Proof of (*a*)

Since $\max_{v \in L \cup R}\{\deg_{H_1}(v)\} \leq d$, then using the results of Yoshida *et al.* [27] we can compute a number $\widetilde{m}_1$ using $d^{O(1/\delta^2)}(1/\delta)^{O(1/\delta)} = O(1)$ expected number of queries such that $m_1\,n - \delta\,n \leq \widetilde{m}_1\,n \leq m_1\,n$. It is straightforward to see that each query in Yoshida *et al.* [27] can be implemented by a weighted

neighbor query $(x, 1)$ for some appropriate $x \in L \cup R$. After using $O(1)$ expected number of weighted neighbor queries to compute $\widetilde{m}_1\, n$ we output the number $\Delta = (3 - 2\,\widetilde{m}_1)$ as our estimate for m_{opt}. Note that $\Delta \geq (3 - 2\,m_1) \geq m_{\text{opt}}$, and $\Delta - m_{\text{opt}} = (3 - 2\,\widetilde{m}_1) - m_{\text{opt}} \leq (3 - 2\,m_1) + 2\,\delta - m_{\text{opt}} \leq 1 + 2\,\delta$. Our proof is completed by taking $\delta = \varepsilon/2$.

Proof of (*b*)

Since $\max_{v \in L \cup R}\{\deg_{H_1}(v)\} \leq d$ and $\max_{v \in L \cup R}\{\deg_{H_2}(v)\} \leq d$, using the results of Yoshida *et al.* [27] we can compute numbers $\widetilde{m}_1$, $\widetilde{m}_2$, and $\widetilde{m}_{12}$ using $(2d)^{O(1/\delta^2)}(1/\delta)^{O(1/\delta)} = O(1)$ expected number of queries such that $m_\ell\, n - \delta\, n \leq \widetilde{m}_\ell\, n \leq m_\ell\, n$ for $\ell \in \{1, 2, 12\}$. It is straightforward to see that each query in Yoshida *et al.* [27] can be implemented by a weighted neighbor query (x, s) for some appropriate $x \in L \cup R$ and $s \in \{1, 2\}$. We perform the following case analysis to provide our estimate Δ of m_{opt}.

Case 1: $\widetilde{m}_1 \leq 1/4$. Our estimate for m_{opt} is $\Delta = 3 - \widetilde{m}_2$. Note that $\Delta \geq 3 - m_2 \geq m_{\text{opt}}$. For the additive error estimation, we have

$$\begin{aligned}\Delta - m_{\text{opt}} = 3 - \widetilde{m}_2 - m_{\text{opt}} &\leq (3 - m_2 - 2m_1) + 2m_1 + \delta - m_{\text{opt}} \\ &\leq (3 - m_{\text{opt},2} - 2\,m_{\text{opt},1}) + 2\big(\tfrac{1}{4} + \delta\big) + \delta - m_{\text{opt}} = \tfrac{1}{2} + 3\,\delta\end{aligned}$$

Case 2: $\widetilde{m}_2 \leq 1/2$ or $\widetilde{m}_1 \geq 1/2$ or $\widetilde{m}_{12} \leq 3/4$. Our estimate for m_{opt} is $\Delta = 3 - 2\,\widetilde{m}_1$. Note that $\Delta \geq 3 - 2\,m_1 \geq m_{\text{opt}}$. For the additive error bounds, we have the following:

- ▷ If $\widetilde{m}_2 \leq 1/2$ then $\Delta - m_{\text{opt}} = 3 - 2\,\widetilde{m}_1 - m_{\text{opt}} = (3 - m_2 - 2\,m_1) + m_2 + 2\,\delta - m_{\text{opt}} \leq (3 - m_{\text{opt},2} - 2\,m_{\text{opt},1}) + 1/2 + 3\,\delta - m_{\text{opt}} \leq \frac{1}{2} + 3\,\delta$.
- ▷ If $\widetilde{m}_1 \geq 1/2$ then since the *smallest possible* total weight that any perfect matching of H could have is achieved by taking all the $m_1\, n$ edges of weight 1 and the remaining $(1 - m_1)\, n$ edges of weight 2 we get $m_{\text{opt}} \geq m_1 + 2(1 - m_1) = 2 - m_1$. Consequently,

$$\Delta - m_{\text{opt}} \leq (3 - 2\,\widetilde{m}_1) - (2 - m_1) \leq 1 - m_1 + 2\,\delta \leq \tfrac{1}{2} + 2\,\delta$$

- ▷ If $\widetilde{m}_{12} \leq 3/4$ then since $m_{\text{opt},3} \geq 1 - m_{12}$ the *smallest possible* total weight that any perfect matching of H could achieve is by taking $m_1\, n$ edges of weight 1, $(1 - m_{12})n$ edges of weight 3, and the remaining $(m_{12} - m_1)n$ edges of weight 2 we get $m_{\text{opt}} \geq m_1 + 2(m_{12} - m_1) + 3(1 - m_{12}) = 3 - m_1 - m_{12}$. Consequently,

$$\Delta - m_{\text{opt}} \leq (3 - 2\,\widetilde{m}_1) - (3 - m_1 - m_{12}) \leq (m_{12} - m_1) + 2\,\delta \\ \leq (\tfrac{3}{4} + \delta - \tfrac{1}{4}) + 2\,\delta = \tfrac{1}{2} + 3\,\delta$$

Case 3: when Cases 1 and Case 2 do not apply. For this case the following inequalities hold:

$$1/4 < \widetilde{m}_1 < 1/2 \;\Rightarrow\; 1/4 < m_1 < 1/2 + \delta,\;\; \widetilde{m}_2 > 1/2 \;\Rightarrow\; m_2 > 1/2 + \delta \\ \widetilde{m}_{12} > 3/4 \;\Rightarrow\; m_{12} > 3/4$$

For this case, we use the following lower bound for m_{opt}. Since $m_{\text{opt},3} \geq 1 - m_{12}$ the *smallest possible* total weight that any perfect matching of H could have is achieved by taking $m_1\,n$ edges of weight 1, $(1 - m_{12})n$ edges of weight 3, and the remaining $(m_{12} - m_1)n$ edges of weight 2. This implies $m_{\text{opt}} \geq m_1 + 2(m_{12} - m_1) + 3(1 - m_{12}) = 3 - m_1 - m_{12}$.

Let $\alpha = \max\{m_{12} - 2\,m_1,\, 0\}$. Suppose that $\mathcal{M}_{12}$ contains $m'_1 \leq m_1$ edges of weight 1. Consider the following process: we start with the edges in $\mathcal{M}_{12}$, remove m'_1 edges of weight 1 from it, add m_1 edges of weight 1 from $\mathcal{M}_1$ to it and finally remove ("knock out") the edges of weight 2 that share an end-point with the edges of $\mathcal{M}_1$ added to our collection. Since m_1 edges of weight 1 can knock out *at most* $2\,m_1$ edges of weight 2, it follows that there are at least α "surviving" edges of weight 2 that do *not* share any end-point with the edges in $\mathcal{M}_1$. We now have the following two sub-cases.

Case 3.1: $\widetilde{m}_{1,2} \leq 2\,\widetilde{m}_1 + \delta$**.** Note that $\widetilde{m}_{1,2} \leq 2\,\widetilde{m}_1 + \delta$ implies $m_{1,2} \leq 2\,m_1 + 2\,\delta$. Our estimate for m_{opt} is $\Delta = 3 - 2\,\widetilde{m}_1$. Note that $\Delta \geq 3 - 2\,m_1 \geq m_{\text{opt}}$. For the additive error estimation, note that

$$\Delta - m_{\text{opt}} \leq (3 - 2\,\widetilde{m}_1) - (3 - m_1 - m_{12}) \leq m_{12} - m_1 + 2\,\delta \\ \leq m_1 + 4\,\delta < \tfrac{1}{2} + 5\,\delta$$

Case 3.2: $\widetilde{m}_{1,2} > 2\,\widetilde{m}_1 + \delta$**.** Our estimate for m_{opt} is $\Delta = 3 - \widetilde{m}_{1,2}$. Note that $\widetilde{m}_{1,2} > 2\,\widetilde{m}_1 + \delta$ implies $m_{1,2} > 2\,m_1 + \delta$. Thus, for this case, $\alpha = m_{12} - 2\,m_1 > \delta > 0$. Let $\mathcal{M}'$ be a perfect matching of H generated by taking all the edges (of weight 1) in $\mathcal{M}_1$, the α surviving edges of weight 2, and pairing the remaining nodes from L and R arbitrarily. Then, the total weight $m'\,n$ of the edges in $\mathcal{M}'$ satisfies $m_1 + 2(m_{12} - 2\,m_1) + 3(1 - (m_1 + (m_{12} - 2\,m_1))) = 3 - m_{12} \geq m' \geq m_{\text{opt}}$, and it

follows that $\Delta \geq 3 - m_{1,2} \geq m_{\text{opt}}$. For the additive error estimation, note that

$$\Delta - m_{\text{opt}} \leq (3 - \widetilde{m}_{1,2}) - (3 - m_1 - m_{12}) \leq m_1 + \delta < \tfrac{1}{2} + 2\,\delta$$

In all cases, setting $\delta = \varepsilon/5$ provides an additive $\left(\frac{1}{2} + \varepsilon\right)$-approximation. □

5.6. Upper bounds on number of queries for computing $\mathfrak{C}_G(e)$ when $\deg(u) \neq \deg(v)$ using "localized" fine-grained reduction

Theorem 5 provides non-trivial approximation of $\mathfrak{C}_G(e)$ when $\deg_G(u) = \deg_G(v)$. In this section, we show that a "localized" version of the fine-grained reduction used in Theorem 3 can be applied to extend these local approximation algorithms to some cases when $\deg_G(u)$ and $\deg_G(v)$ are *not* necessarily equal. Such a localized version of the fine-grained reduction is *not* allowed to construct the reduction explicitly, but instead the details of the reduction need to be revealed incrementally to the local algorithm on a "need-to-know" basis to simulate the queries of the local algorithm on the graph constructed by the fine-grained reduction. The overall simulation is summarized in Theorem 6.

Theorem 6 (Computing $\mathfrak{C}_G(e)$ via localized fine-grained reduction). *Suppose that we have an algorithm $\mathfrak{B}_=$ that provides an (α, ε)-estimate for $\mathfrak{C}_G(e)$ when $\deg_G(u) = \deg_G(v)$ using t queries $q'_1, \ldots, q'_t$ when each query q'_i is either a weighted node-pair query, a weighted neighbor query or a weighted selective degree query.*

Then, letting $\delta > 0$ denote any constant, we can design an algorithm $\mathfrak{B}_<$ for the case when $\deg_G(u) \neq \deg_G(v)$ using $\mathfrak{B}_=$ with the following properties:

(a) *Corresponding to each query q'_i, $\mathfrak{B}_<$ performs at most one weighted selective degree query and at most one additional query of the same type as q'_i on $G_{u,v}$.*

(b) *$\mathfrak{B}_<$ provides an (α, ε)-estimate for $\mathfrak{C}_G(e)$ if $\deg_G(u) + 1$ is an integral multiple of $\deg_G(v) + 1$.*

(c) *$\mathfrak{B}_<$ provides an $(\alpha, \varepsilon+\delta)$-estimate for $\mathfrak{C}_G(e)$ if either $\deg_G(u) \leq (\delta/3) \times \deg_G(v)$ or $\deg_G(u) \geq (1 - (\delta/3)\,) \times \deg_G(v)$.*

Corollary 7. *If* $\deg_G(u) \geq (1-(\delta/3)) \times \deg_G(v)$ *for some constant* $\delta > 0$ *then* $\max_x\{\deg_{G_{u,v},1}(x)\} = O(1)$ (*respectively,* $\max_x\{\deg_{G_{u,v},2}(x)\} = O(1)$) *implies* $\max_x\{\deg_{G'_{u,v},1}(x)\} = O(1)$ (*respectively,* $\max_x\{\deg_{G'_{u,v},2}(x)\} = O(1)$), *and thus each weighted selective degree query for the weight* 1 (*respectively, for the weight* 2) *can be trivially simulated by* $O(1)$ *weighted neighbor queries for the weight* 1 (*respectively, for the weight* 2) *on* $G'_{u,v}$. *Thus, combining Theorem 6 with the approximations in Theorem 5 gives us algorithms of the following types for the case when* $\deg_G(u) \neq \deg_G(v)$*:*

(*i*) *additive* $(1+\varepsilon+\delta)$*-approximation using* $O(1)$ *weighted neighbor queries*[7] *if* $\max_x\{\deg_{H,1}(x)\} = O(1)$ *and* $\deg_G(u) \geq (1-(\delta/3)) \times \deg_G(v)$,

(*ii*) *additive* $\left(\frac{1}{2}+\varepsilon+\delta\right)$*-approximation using* $O(1)$ *weighted neighbor queries*[7] *if* $\max_x\{\deg_{H,1}(x)\} = O(1)$, $\max_x\{\deg_{H,2}(x)\} = O(1)$, *and* $\deg_G(u) \geq (1-(\delta/3)) \times \deg_G(v)$.

Proof. We will *reuse* the notations and the reduction used in the proof of Theorem 3; in particular in those notations the graph H is also the graph $G_{u,v}$. Our algorithm $\mathfrak{B}_<$ has a list of nodes in the graph $G'_{u,v}$ and also the numbers a and b. We show next how the value of a query q'_i on $G'_{u,v}$ can be obtained from the values of a collection $\mathcal{Q}_i$ of (at most two) queries on $G_{u,v}$ by $\mathfrak{B}_<$.

▷ **Case 1: q'_i is a weighted node-pair query**. If q'_i is of the form (x_i^j, y_ℓ) then $\mathcal{Q}_i = \{(x_i, y_\ell)\}$ and $\mathfrak{B}_<$ returns the value of the query (x_i, y_ℓ) on $G_{u,v}$ as the value of q'_i. If q'_i is of the form (r_i, y_ℓ) then $\mathcal{Q}_i = \emptyset$ and $\mathfrak{B}_<$ returns 3 as the value of q'_i.

▷ **Case 2: q'_i is a weighted selective degree query**. Let s be a number from the set $\{1, 2, 3\}$.

- If q'_i is of the form (x_i^j, s) then $\mathcal{Q}_i = \{(x_i, s)\}$ and $\mathfrak{B}_<$ returns the value of the weighted selective degree query (x_i, s) on $G_{u,v}$ as the value of q'_i.
- If q'_i is of the form (r_i, s) then $\mathcal{Q}_i = \emptyset$ and $\mathfrak{B}_<$ returns $3\deg_G(v) + 3$ if $s = 3$ and 0 otherwise as the value of q'_i.

[7]The constant in $O(1)$ depends on the value of $\frac{1}{1-(\delta/3)}$.

- If q'_i is of the form (y_ℓ, s) then $\mathcal{Q}_i = \{(y_\ell, s)\}$, and $\mathfrak{B}_<$ returns the following as the value of q'_i:
  - the value of the weighted selective degree query (y_ℓ, s) on $G_{u,v}$ times a if $s \in \{1, 2\}$, and
  - the value of the weighted selective degree query (y_ℓ, s) on $G_{u,v}$ times a plus b otherwise.

▷ **Case 3: q'_i is a weighted neighbor query**. Let s be a number from the set $\{1, 2, 3\}$.

▷ **Case 3.1: q'_i is of the form (x_i^j, s)**. The following example illustrates the subtlety of this case. Suppose that x_1 is connected to four nodes y_1, y_2, y_3, y_4 via edges of weight s in $G_{u,v}$. Then each of the nodes $x_1^1, \ldots, x_1^a$ is connected to y_1, y_2, y_3, y_4 via edges of weight s in $G'_{u,v}$.

- As a first attempt, one may simulate the answer to the query (x_1^j, s) by performing a query (x_1, s) on $G_{u,v}$. However, this will not provide new nodes with the correct probabilities required for random uniform selection among not-yet-explored nodes. For example, suppose that $\mathfrak{B}_<$ already made the query (x_1^1, s) giving the node y_1. If now $\mathfrak{B}_<$ makes another query (x_1^2, s) then such a simulation will return a node uniformly randomly from the set of nodes $\{y_2, y_3, y_4\}$ but the correct simulation would have been to select a node uniformly randomly from the set of nodes $\{y_1, y_2, y_3, y_4\}$. Moreover, if $\mathfrak{B}_<$ has already made the queries $(x_1^1, s), (x_1^2, s), (x_1^3, s), (x_1^4, s)$ using such a simulation then this simulation of a new query (x_1^5, s) will simply return the special symbol.
- As a second attempt, to simulate the answer to a query (x_1^j, s) one may first check if the answer to a query $(x_1^{j'}, s)$ for some $j' \neq j$ is already available, and if so simply return that answer. But, in this case, the answers to the queries (x_1^j, s) and $(x_1^{j'}, s)$ will *not* be statistically independent.

To address these and other subtleties we design Algorithm $\mathfrak{B}_<$ to handle all queries of the form (x_i^j, s) **for each specific** i and s in the following manner. Let $\mathcal{S}_{x_i,s}$ be the set of (not initially known to $\mathfrak{B}_<$) $\sigma_{i,s} = |\mathcal{S}_{x_i,s}|$ nodes connected to x_i in $G_{u,v}$ via edges of weight s.

(*i*) If *not* already done before, we make one new weighted selective degree query (x_i, s) on $G_{u,v}$ giving us the value of $\sigma_{i,s}$ (if the value

of $\sigma_{i,s}$ *is* already available we simply use it without making a query).

(*ii*) For each x_i^j, we keep a count $\kappa_{x_i^j,s}$ of how many times the query (x_i^j, s) has been asked involving the node x_i^j before the current query and store the answers to these queries in a set $\mathcal{T}_{x_i^j,s}$. We also maintain $\mathcal{T}_{i,s} = \cup_{j=1}^{a} \mathcal{T}_{x_i^j,s}$ and $\kappa_{i,s} = |\mathcal{T}_{i,s}|$. Note the following:

- If $\kappa_{x_i^j,s} < \sigma_{i,s}$ then performing a new weighted neighbor query $\boldsymbol{(x_i^j, s)}$ on $\boldsymbol{G'_{u,v}}$ *must* return a node uniformly at random from the set of nodes $\Lambda_{x_i^j,s} = \mathcal{S}_{x_i,s} \setminus \mathcal{T}_{x_i^j,s}$ with probability $1/\lambda_{x_i^j,s}$ where $\lambda_{x_i^j,s} = |\Lambda_{x_i^j,s}| = \sigma_{i,s} - \kappa_{x_i^j,s}$.
- If $\kappa_{i,s} < \sigma_{i,s}$ then performing a new weighted neighbor query (x_i, s) on $\boldsymbol{G_{u,v}}$ returns a node uniformly at random from the set of nodes $\Lambda_{i,s} = \mathcal{S}_{x_i,s} \setminus \mathcal{T}_{i,s}$ with probability $1/\lambda_{i,s}$ where $\lambda_{i,s} = |\Lambda_{i,s}| = \sigma_{i,s} - \kappa_{i,s}$.
- Note that we know all the elements of $\mathcal{T}_{i,s}$; in particular, this means that **we *can* sample a node from a subset of $\boldsymbol{\mathcal{T}_{i,s}}$ uniformly at random**.

(*iii*) For a query (x_i^j, s), we have the following cases.

- **Case I: $\boldsymbol{\kappa_{i,s} = \sigma_{i,s}}$.** In this case $\mathcal{T}_{i,s} = \mathcal{S}_{x_i,s}$.
  - **Case I-a: $\boldsymbol{\kappa_{x_i^j,s} < \kappa_{i,s}}$.** We select a node uniformly at random from the set $\mathcal{T}_{i,s} \setminus \mathcal{T}_{x_i^j,s} = \mathcal{S}_{x_i,s} \setminus \mathcal{T}_{x_i^j,s}$ and return it as the answer to the query.
  - **Case I-b: $\boldsymbol{\kappa_{x_i^j,s} = \kappa_{i,s}}$.** We return an invalid entry as the answer to the query.
- **Case II: $\boldsymbol{\kappa_{i,s} < \sigma_{i,s}}$.** We make a new query (x_i, s) on $G_{u,v}$ giving us a node $y_p \in \Lambda_{i,s} = \mathcal{S}_{x_i,s} \setminus \mathcal{T}_{i,s}$ with the property that $\Pr[y_p \in \Lambda_{i,s} \text{ is returned}] = \frac{1}{\lambda_{i,s}}$.
  - **Case II-a: $\boldsymbol{\kappa_{x_i^j,s} = \kappa_{i,s}}$.** For this case, $\mathcal{T}_{i,s} = \mathcal{T}_{x_i^j,s}$ and $\lambda_{i,s} = \Lambda_{x_i^j,s}$. We return the node y_p as the answer to the query and update all relevant sets and counters appropriately.
  - **Case II-b: $\boldsymbol{\kappa_{x_i^j,s} < \kappa_{i,s}}$.** For this case $\mathcal{T}_{x_i^j,s} \subset \mathcal{T}_{i,s} \subset \mathcal{S}_{x_i,s}$, $\lambda_{i,s} = |\mathcal{S}_{x_i,s} \setminus \mathcal{T}_{i,s}| > 0$, and $\lambda_{x_i^j,s} = |\mathcal{S}_{x_i,s} \setminus \mathcal{T}_{x_i^j,s}| > \lambda_{i,s}$. We sample the nodes in $\{y_p\} \bigcup \left(\mathcal{T}_{i,s} \setminus \mathcal{T}_{x_i^j,s}\right)$ based on the following probability distribution and update all relevant sets and

counters appropriately:

$$\Pr[y_p \text{ is selected}] = \frac{\lambda_{i,s}}{\lambda_{x_i^j,s}}$$
$$\forall\, y_\ell \in \mathcal{T}_{i,s} \setminus \mathcal{T}_{x_i^j,s} : \Pr[y_\ell \text{ is selected}] = \frac{1}{\lambda_{x_i^j,s}}$$

Thus, the answer to the query (x_i^j, s) is selected *uniformly at random* from the set $\Lambda_{x_i^j,s} = \mathcal{S}_{x_i,s} \setminus \mathcal{T}_{x_i^j,s}$ since

$$\forall\, y_\ell \in \mathcal{S}_{x_i,s} \setminus \mathcal{T}_{i,s} : \Pr[y_\ell \text{ is selected}] = \frac{1}{\lambda_{i,s}} \times \frac{\lambda_{i,s}}{\lambda_{x_i^j,s}} == \frac{1}{\lambda_{x_i^j,s}}$$
$$\forall\, y_\ell \in \mathcal{T}_{i,s} \setminus \mathcal{T}_{x_i^j,s} : \Pr[y_\ell \text{ is selected}] = \frac{1}{\lambda_{x_i^j,s}}$$

- ▷ **Case 3.2: q_i' is of the form (r_i, s)**. We keep a count $\nu(r_i)$ of how many times the query $(r_i, 3)$ has been asked involving the node r_i before the current query, and store the answers to these queries in the set $\mathcal{S}_{r_i}$. If $s \neq 3$ or $\nu(r_i) = \deg_G(v) + 1$ we return the special symbol. Otherwise, we return a node selected uniformly at random from the set of nodes $\{y_1, \ldots, y_{\deg_G(v)+1}\} \setminus \mathcal{S}_{r_i}$ as the answer and update all relevant sets and counters appropriately.
- ▷ **Case 3.3: q_i' is of the form (y_ℓ, s)**. This case is similar in spirit to Case 3.1. We show how to handle all queries of the form (y_ℓ, s) **for each specific** ℓ and s.
  - **(*i*)** Assume without loss of generality that y_ℓ is connected, via edges of weight s, to (not initially known to $\mathfrak{B}_<$) a set $\mathcal{S}_{\nu_1} = \{x_1, \ldots, x_{\nu_1}\} \subseteq \{x_1, \ldots, x_{\deg_G(u)+1}\}$ of $\nu_1 = |\mathcal{S}_{\nu_1}|$ nodes. If *not* already done before, we make one new weighted selective degree query (y_ℓ, s) on $G_{u,v}$ giving us the value of ν_1 (if ν_1 *is* already known we simply use it without making a query).
  - **(*ii*)** Define the set $\mathcal{S}_{\nu_2}$ of $\nu_2 = |\mathcal{S}_{\nu_2}| \in \{0, b\}$ nodes as $\mathcal{S}_{\nu_2} = \{r_1, \ldots, r_b\}$ if $s = 3$ and $\mathcal{S}_{\nu_2} = \emptyset$ otherwise. Note that we know the value of ν_2 since we know the value of s.
  - **(*iii*)** We keep a count κ of how many times the query (y_ℓ, s) has been asked involving the node y_ℓ before the current query, and let $\mathcal{T}_\kappa$ be the set of those $\kappa = |\mathcal{T}_\kappa|$ nodes **of $G'_{u,v}$** that have been returned because of these prior queries. **Note that if $\kappa < a\,\nu_1 + \nu_2$ then performing a new weighted neighbor query (y_ℓ, s) on**

$G'_{u,v}$ *must* return a node uniformly at random from the set of nodes $\Lambda_\kappa = \left(\cup_{i=1}^{\nu_1} \cup_{j=1}^{a} \{x_i^j\} \cup \mathcal{S}_{\nu_2}\right) \setminus \mathcal{T}_\kappa$ with probability $1/\lambda_\kappa$ where $\lambda_\kappa = |\Lambda_\kappa| = (a\,\nu_1 + \nu_2) - \kappa$.

(*iv*) Assume without loss of generality that $\mathcal{S}'_{\nu_1} = \{x_1, x_2, \ldots, x_{\nu'_1}\} \subseteq \mathcal{S}_{\nu_1}$ be the set of $\nu'_1 = |\mathcal{S}'_{\nu_1}| \leq \min\{\kappa, \nu_1\}\}$ nodes **in $G_{u,v}$** that have been returned as a result of the queries on $G_{u,v}$ due to the simulation of prior κ queries on $G'_{u,v}$. Note that **if $\nu'_1 < \nu_1$ then performing a new weighted neighbor query (y_ℓ, s) on $G_{u,v}$ returns a *new* node uniformly at random from the set of nodes $\Phi = \mathcal{S}_{\nu_1} \setminus \mathcal{S}'_{\nu_1}$ with probability $1/\varphi$ where $\varphi = |\Phi| = \nu_1 - \nu'_1$.**

(*v*) Define the subset $\Lambda'_\kappa \subseteq \Lambda_\kappa$ of nodes of $G'_{u,v}$ as $\Lambda'_\kappa = \left(\cup_{i=1}^{\nu'_1} \cup_{j=1}^{a} \{x_i^j\} \cup \mathcal{S}_{\nu_2}\right) \setminus \mathcal{T}_\kappa$. Note that we know all the elements of Λ'_κ and $\lambda'_\kappa = |\Lambda'_\kappa| = (a\nu'_1 + \nu_2) - \kappa$. In particular, this means that **we *can* sample a node from Λ'_κ uniformly at random**.

(*vi*) For a new query (y_ℓ, s), we have the following cases.

- **Case I: $\kappa \geq \nu_1$**. In this case, $\nu'_1 = \nu_1$, $\Lambda'_\kappa = \Lambda_\kappa$ and $\lambda'_\kappa = \lambda_\kappa$. We select as our answer to the query a node uniformly at random from Λ'_κ, and update all relevant sets and counters appropriately.
- **Case II: $\kappa < \nu_1$**. In this case, $\nu'_1 < \nu_1$, and $\varphi > 0$. We simulate the query as follows.
  - We make a new query (y_ℓ, s) on $G_{u,v}$ giving us a node $x_p \in \Phi$ for $p \in \{\nu'_1 + 1, \ldots, \nu_1\}$ with probability $1/\varphi$. We select $j \in \{1, \ldots, a\}$ uniformly at random giving us a node x_p^j.
  - We *sample* a node from $\{x_p^j\} \cup \Lambda'_\kappa$ based on the following probability distribution and update all relevant sets and counters appropriately:

$$\Pr[x_p^j \text{ is selected }] = \tfrac{a\varphi}{\lambda_\kappa}$$
$$\forall\, x_i^j \in \Lambda'_\kappa : \Pr[x_i^j \text{ is selected }] = \tfrac{1}{\lambda_\kappa}$$

Note that $\Pr[x_i^j \in \Lambda_\kappa \setminus \Lambda'_\kappa \text{ is selected }] = \frac{1}{a\varphi} \times \frac{a\varphi}{\lambda_\kappa} = \frac{1}{\lambda_\kappa}$, as desired. To verify that all the probabilities add up to 1, note that $\Pr[x_p^j \text{ is selected }] + \sum_{x_i^j \in \Lambda'_\kappa} \Pr[x_i^j \in \Lambda'_\kappa \text{ is selected }] = \frac{a(\nu_1 - \nu'_1)}{a\nu_1 + \nu_2 - \kappa} + (a\nu'_1 + \nu_2 - \kappa) \times \frac{1}{a\nu_1 + \nu_2 - \kappa} = 1$.

□

6. Computing $\mathfrak{C}_G(v)$ and $\mathfrak{C}_{\mathbf{avg}}(G)$ using "black box" additive approximation algorithms for $\mathfrak{C}_G(e)$

In this section we provide efficient local algorithms to compute $\mathfrak{C}_G(v)$ and $\mathfrak{C}_{\text{avg}}(G)$. The following assumptions are used by our algorithms:

- ▷ For a fixed $\mathfrak{r}$, we have an efficient local algorithm $\mathfrak{B}$ for an additive $\mathfrak{r}$-approximation, say $\mathfrak{C}^{\mathfrak{r}}_G(e)$, of $\mathfrak{C}_G(e)$ for an edge e.
- ▷ We have access to the *neighbor query model* mentioned in Section 5.2.

Lemma 8. *With probability at least 2/3 the following two claims hold.*

(*a*) *We can compute an additive $2\mathfrak{r}$-approximation of $\mathfrak{C}_G(v)$ using $O(1/\mathfrak{r}^2)$ neighbor queries and $O(1/\mathfrak{r}^2)$ invocations of algorithm $\mathfrak{B}$ on the edges incident on v, and*

(*b*) *If the degrees of all the nodes of G are know then we can compute an additive $2\mathfrak{r}$-approximation of $\mathfrak{C}_{\text{avg}}(G)$ using $O(1/\mathfrak{r}^2)$ neighbor queries and $O(1/\mathfrak{r}^2)$ invocations of algorithm $\mathfrak{B}$ over all edges in G.*

Proof.

(*a*) Let k be a parameter to be specified later. We use $k' = \min\{k, \deg_G(v)\}$ neighbor queries to get k' nodes adjacent to v, say $u_1, \ldots, u_{k'}$, compute $\mathfrak{C}^{\mathfrak{r}}_G(v, u_1), \ldots, \mathfrak{C}^{\mathfrak{r}}_G(v, u_{k'})$ using algorithm $\mathfrak{B}$, and return $\widetilde{\mathfrak{C}_G}(v) = \frac{1}{k'} \sum_{j=1}^{k'} \mathfrak{C}^{\mathfrak{r}}_G(v, u_j)$ as our answer.

If $k > \deg_G(v)$ then $\mathfrak{C}_G(v) = \frac{1}{k'} \sum_{j=1}^{k'} \mathfrak{C}_G(v, u_j)$ and thus $\widetilde{\mathfrak{C}_G}(v)$ is in fact an additive $\mathfrak{r}$-approximation of $\mathfrak{C}_G(v)$. Otherwise, assume that $k \leq \deg_G(v)$ and therefore $k' = k$. For any number x, we use the notation $x \boxplus \rho$ to indicate a number y that satisfies $x \leq y \leq x + \rho$. Observe that

$$\mathbb{E}\left[\widetilde{\mathfrak{C}_G}(v)\right] = \frac{1}{k} \sum_{j=1}^{k} \mathbb{E}\left[\mathfrak{C}^{\mathfrak{r}}_G(v, u_j)\right] = \frac{1}{k} \sum_{j=1}^{k} \sum_{u \in \mathsf{Nbr}_G(u)} (\mathfrak{C}_G((u, v)) \boxplus \mathfrak{r}) \times \frac{1}{\deg_G(v)}$$

$$= \frac{1}{k} \sum_{j=1}^{k} (\mathfrak{C}_G(v) \boxplus \mathfrak{r}) = \mathfrak{C}_G(v) \boxplus \mathfrak{r}$$

Since the $\frac{1}{k}\mathfrak{C}^{\mathfrak{r}}_G(v, u_j)$'s are mutually independent for $j = 1, \ldots, k$, and each $\frac{1}{k}\mathfrak{C}^{\mathfrak{r}}_G(v, u_j)$ lies in the interval $[-2/k, 1/k]$ (*cf.* see Section 2.3), applying Hoeffding's inequality [43, Theorem 2] we get

$$\Pr[\widetilde{\mathfrak{C}_G}(v) > \mathfrak{C}_G(v) + 2\mathfrak{r}] \leq \Pr[\widetilde{\mathfrak{C}_G}(v) > \mathbb{E}[\widetilde{\mathfrak{C}_G}(v)] + \mathfrak{r}]$$

$$< \exp\left(-\frac{2\mathfrak{r}^2}{\sum_{i=1}^{k}(2/k-(-1/k))^2}\right) = \exp\left(-\frac{2}{9}k^2\mathfrak{r}^2\right)$$

Setting $k = \Theta(\mathfrak{r}^{-2})$ we get $\Pr[\widetilde{\mathfrak{C}_G}(v) > \mathfrak{C}_G(v) + 2\mathfrak{r}] < 1/3$.

(*b*) The algorithm and its proof is very similar to those in **(*a*)**. For this case, we need to randomly sample $k' = \min\{O(1/\mathfrak{r}^2), , |E|\}$ edges $e_1, \ldots, e_{k'}$ from E, compute $\mathfrak{C}^{\mathfrak{r}}_G(e_1), \ldots, \mathfrak{C}^{\mathfrak{r}}_G(e_{k'})$ using algorithm $\mathfrak{B}$, and return $\frac{1}{k'}\sum_{j=1}^{k'}\mathfrak{C}^{\mathfrak{r}}_G(e_j)$ as our answer. The only remaining part of the proof is to show how to sample an edge uniformly at random from the set of edges E of G. Since the degrees of all nodes are known, the following procedure can be used. We first select a node $x \in V$ with probability $\frac{\deg_G(x)}{\sum_{z\in V}\deg_G(z)}$, then we select a random neighbor of x, say y, using one neighbor query, and finally we select the edge $\{x, y\}$. The proof is completed by observing that

$$\begin{aligned}
&\Pr[\{u,v\} \in E \text{ is selected }] \\
&\quad = \Pr[x \in V \text{ is selected }] \times \Pr[y \in \mathsf{Nbr}_G(x) \text{ is selected }] \\
&\qquad + \Pr[y \in V \text{ is selected }] \times \Pr[x \in \mathsf{Nbr}_G(y) \text{ is selected }] \\
&\qquad\quad = \frac{\deg_G(x)}{\sum_{z\in V}\deg_G(z)} \times \frac{1}{\deg_G(x)} + \frac{\deg_G(y)}{\sum_{z\in V}\deg_G(z)} \times \frac{1}{\deg_G(y)} = \frac{1}{|E|}
\end{aligned}$$

□

7. Concluding remarks

We hope that this paper will stimulate further attention from computer scientists concerning the exciting interplay between notions of curvatures from network and non-network domains. An obvious candidate for future research is improvement of the query complexities for local algorithms for computing the Ollivier-Ricci curvature for networks. Another possible future research direction is to investigate computational complexity issues of other discretizations of Ricci curvatures. For example, another discretization of Ricci curvature for networks proposed by Ollivier and Villani [44] is guided by the observation that the infinite-dimensional version of the well-known Brunn-Minkowski inequality over $\mathbb{R}^n$ [45] can be tightened in the presence of a positive curvature for a smooth Riemannian manifold [46, 47]. To our knowledge, these discretizations have largely escaped computational complexity considerations.

Appendix A. A self-contained proof of Fact 1

Let $\deg_G(u) = \deg_G(v) = \alpha$. Build a directed single-source single-sink flow network [48] $G^f_{u,v}$ from $G_{u,v}$ in the following manner: add a new source node s and a new sink node t, add an arc (directed edge) from s to every node of $L^G_{u,v}$ of weight zero and capacity 1, add an arc from every node of $R^G_{u,v}$ to t of weight zero and capacity 1, orient every edge $\{u', v'\}$ of $G_{u,v}$ from u' to v' and set its capacity to 1. Since $|L^G_{u,v}| = |R^G_{u,v}| = \alpha + 1$, we have $\mathbb{P}^G_u(u') = \mathbb{P}^G_v(v') = \frac{1}{\alpha+1}$ for all $u' \in \mathsf{Nbr}^G(u) \cup \{u\}$ and $v' \in \mathsf{Nbr}^G(v) \cup \{v\}$. Thus, since $G_{u,v}$ is a complete bipartite graph, by a simple scaling it follows that $\textsc{Emd}_{G_{u,v}}(\mathbb{P}^G_u, \mathbb{P}^G_v) = \frac{\mathcal{M}}{\alpha+1}$ where $\mathcal{M}$ is the total weight of a minimum-weight maximum s-t flow on $G^f_{u,v}$. Since the node-arc incidence matrix of a directed graph is totally unimodular, the flow value of every arc of any extreme-point optimal solution of the minimum-weight maximum s-t flow on $G^f_{u,v}$ is integral and therefore 0 or 1 (see Theorem 13.3 and its corollary in [48]). This integrality of flow values and the fact that $G_{u,v}$ is a complete bipartite graph imply $\mathcal{M}$ is also the total weight of a minimum-weight *perfect* matching of $G_{u,v}$.

We now show that there is such a minimum-weight perfect matching that uses all the zero-weight edges $\{u', u'\}$ for all $u' \in \{u, v\} \cup \big(\, \mathsf{Nbr}_G(u) \cap \mathsf{Nbr}_G(v) \,\big)$. For a contradiction, suppose that the edge $\{u', u'\}$ is not used for some $u' \in \{u, v\} \cup (\mathsf{Nbr}_G(u) \cap \mathsf{Nbr}_G(v))\}$. Since our solution is a perfect matching, the nodes $u' \in L^G_{u,v}$ and $u' \in R^G_{u,v}$ must be matched to some other nodes, say to nodes $v'' \in R^G_{u,v}$ and $u'' \in L^G_{u,v}$, respectively. Then, if we instead use the edges $\{u', u'\}$ and $\{u'', v''\}$ then using the triangle inequality it follows that the total weight of this modified perfect matching is no more than that of the original perfect matching since:

$$\begin{aligned} w^G_{u,v}(u', u') + w^G_{u,v}(u'', v'') = w^G_{u,v}(u'', v'') &\leq w^G_{u,v}(u'', u') + w^G_{u,v}(u', u') + w^G_{u,v}(u', v'') \\ &= w^G_{u,v}(u'', u') + w^G_{u,v}(u', v'') \end{aligned}$$

References

[1] M. R. Bridson, A. Häfliger, Metric Spaces of Non-Positive Curvature, 1st Edition, Springer-Verlag Berlin Heidelberg, 1999. doi:10.1007/978-3-662-12494-9.

[2] M. Berger, A Panoramic View of Riemannian Geometry, 1st Edition, Springer-Verlag Berlin Heidelberg, 2003. doi:10.1007/978-3-642-18245-7.

[3] R. Albert, B. DasGupta, N. Mobasheri, Topological implications of negative curvature for biological and social networks, Physical Review E 89 (2014) 032811. doi:10.1103/PhysRevE.89.032811.
URL https://link.aps.org/doi/10.1103/PhysRevE.89.032811

[4] T. Chatterjee, R. Albert, S. Thapliyal, N. Azarhooshang, B. DasGupta, Detecting network anomalies using forman-ricci curvature and a case study for human brain networks, Scientific Reports 11 (2021). doi:10.1038/s41598-021-87587-z.

[5] E. Jonckheere, M. Lou, F. Bonahon, Y. Baryshnikov, Euclidean versus hyperbolic congestion in idealized versus experimental networks, Internet Mathematics 71 (2011) 1–27. doi:10.1080/15427951.2010.554320.
URL https://doi.org/10.1080/15427951.2010.554320

[6] J. Sia, E. Jonckheere, P. Bogdan, Ollivier-ricci curvature-based method to community detection in complex networks, Scientific Reports 9 (2019) 9800. doi:10.1038/s41598-019-46079-x.

[7] A. K. Simhal, K. L. H. Carpenter, S. Nadeem, J. Kurtzberg, A. Song, A. Tannenbaum, G. Sapiro, G. Dawson, Measuring robustness of brain networks in autism spectrum disorder with Ricci curvature, Scientific Reports 10 (2020) 10819. doi:10.1038/s41598-020-67474-9.

[8] P. Elumalai, Y. Yadav, N. Williams, E. Saucan, J. Jost, A. Samal, Graph ricci curvatures reveal atypical functional connectivity in autism spectrum disorder, bioRxiv (2021). doi:10.1101/2021.11.28.470231.
URL https://www.biorxiv.org/content/early/2021/12/21/2021.11.28.470231

[9] B. Chow, F. Luo, Combinatorial ricci flows on surfaces, Journal of Differential Geometry 63 (1) (2003) 97–129. doi:10.4310/jdg/1080835659.

[10] Y. Ollivier, A visual introduction to Riemannian curvatures and some discrete generalizations, in: G. Dafni, R. J. McCann, A. Stancu (Eds.), Analysis and Geometry of Metric Measure Spaces: Lecture Notes of the

50th Séminaire de Mathématiques Supérieures (SMS), Montréal, 2011, Vol. 56, American Mathematical Society, Providence, RI, USA, 2013, pp. 197–219. doi:10.1090/crmp/056/08.
URL https://hal.archives-ouvertes.fr/hal-00858008

[11] Y. Ollivier, Ricci curvature of markov chains on metric spaces, Journal of Functional Analysis 256 (2009) 810–864. doi:10.1016/j.jfa.2008.11.001.

[12] Y. Ollivier, A survey of ricci curvature for metric spaces and markov chains, in: M. Kotani, M. Hino, T. Kumagai (Eds.), Advanced Studies in Pure Mathematics, Vol. 57, Mathematical Society of Japan, 2010, pp. 343–381. doi:10.2969/aspm/05710343.

[13] Y. Ollivier, Ricci curvature of metric spaces, Comptes Rendus Mathematique 345 (11) (2007) 643–646. doi:10.1016/j.crma.2007.10.041.
URL https://www.sciencedirect.com/science/article/pii/S1631073X07004414

[14] B. DasGupta, M. V. Janardhanan, F. Yahyanejad, Why did the shape of your network change? (on detecting network anomalies via non-local curvatures), Algorithmica 82 (7) (2020) 1741–1783. doi:10.1007/s00453-019-00665-7.

[15] B. DasGupta, M. Karpinski, N. Mobasheri, F. Yahyanejad, Effect of gromov-hyperbolicity parameter on cuts and expansions in graphs and some algorithmic implications, Algorithmica 80 (2) (2018) 772–800. doi:10.1007/s00453-017-0291-7.

[16] I. Benjamini, Expanders are not hyperbolic, Israel Journal of Mathematics 108 (1998) 33–36. doi:10.1007/BF02783040.

[17] J. Chalopin, V. Chepoi, F. F. Dragan, G. Ducoffe, A. M. A., Y. Vaxès, Fast approximation and exact computation of negative curvature parameters of graphs., Discrete and Computational Geometry 65 (2021) 856–892. doi:10.1007/s00454-019-00107-9.

[18] H. Fournier, A. Ismail, A. Vigneron, Computing the gromov hyperbolicity of a discrete metric space, Information Processing Letters 115 (6) (2015) 576–579. doi:10.1016/j.ipl.2015.02.002.
URL https://doi.org/10.1016/j.ipl.2015.02.002

[19] R. Forman, Bochner's method for cell complexes and combinatorial ricci curvature, Discrete and Computational Geometry 29 (3) (2003) 323–374. doi:10.1007/s00454-002-0743-x.

[20] R. P. Sreejith, K. Mohanraj, J. Jost, E. Saucan, A. Samal, Forman curvature for complex networks, Journal of Statistical Mechanics: Theory and Experiment 2016 (6) (2016) 063206. doi:10.1088/1742-5468/2016/06/063206.

[21] R. P. Sreejith, J. Jost, E. Saucan, A. Samal, Systematic evaluation of a new combinatorial curvature for complex networks, Chaos, Solitons and Fractals 101 (2017) 50–67. doi:10.1016/j.chaos.2017.05.021.
URL https://www.sciencedirect.com/science/article/pii/S0960077917302102

[22] M. Weber, E. Saucan, J. Jost, Characterizing complex networks with forman-ricci curvature and associated geometric flows, Journal of Complex Networks 5 (4) (2017) 527–550. doi:10.1093/comnet/cnw030.

[23] A. Samal, R. P. Sreejith, J. Gu, S. Liu, E. Saucan, J. Jost, Comparative analysis of two discretizations of ricci curvature for complex networks, Scientific Reports 8 (2018) 8650. doi:10.1038/s41598-018-27001-3.

[24] M. Gromov, Hyperbolic groups, in: S. M. Gersten (Ed.), Essays in Group Theory, Vol. 8, Springer, New York, NY, 1987, pp. 75–263. doi:10.1007/978-1-4613-9586-7_3.

[25] V. Chepoi, F. Dragan, B. Estellon, M. Habib, Y. Vaxès, Diameters, centers, and approximating trees of delta-hyperbolicgeodesic spaces and graphs, in: Proceedings of the Twenty-Fourth Annual Symposium on Computational Geometry, SCG '08, Association for Computing Machinery, New York, NY, USA, 2008, pp. 59–68. doi:10.1145/1377676.1377687.
URL https://doi.org/10.1145/1377676.1377687

[26] F. Papadopoulos, D. Krioukov, M. Boguna, A. Vahdat, Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces, in: 2010 Proceedings IEEE INFOCOM, 2010, pp. 1–9. doi:10.1109/INFCOM.2010.5462131.

[27] Y. Yoshida, M. Yamamoto, H. Ito, Improved Constant-Time Approximation Algorithms for Maximum Matchings and Other Optimization Problems, SIAM Journal on Computing 41 (4) (2012) 1074–1093. doi:10.1137/110828691.

[28] Y. T. Lee, A. Sidford, Efficient inverse maintenance and faster algorithms for linear programming, in: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, 2015, pp. 230–249. doi:10.1109/FOCS.2015.23.

[29] K. Quanrud, Approximating Optimal Transport With Linear Programs, in: J. T. Fineman, M. Mitzenmacher (Eds.), 2nd Symposium on Simplicity in Algorithms (SOSA 2019), Vol. 69 of OpenAccess Series in Informatics (OASIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2018, pp. 6:1—6:9. doi:10.4230/OASIcs.SOSA.2019.6.
URL http://drops.dagstuhl.de/opus/volltexte/2018/10032

[30] P. Dvurechensky, A. Gasnikov, A. Kroshnin, Computational optimal transport: Complexity by accelerated gradient descent is better than by sinkhorn's algorithm, in: J. Dy, A. Kraus (Eds.), Proceedings of the 35th International Conference on Machine Learning, Vol. 80 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 1367—1376.
URL https://proceedings.mlr.press/v80/dvurechensky18a.html

[31] N. Azarhooshang, P. Sengupta, B. DasGupta, A review of and some results for ollivier-ricci network curvature, Mathematics 8 (1416) (2020). doi:10.3390/math8091416.

[32] G. Peyré, M. Cuturi, Computational optimal transport: With applications to data science, Foundations and Trends in Machine Learning 11 (5–6) (2019) 355–607. doi:10.1561/2200000073.
URL http://dx.doi.org/10.1561/2200000073

[33] A. L. Gibbs, F. E. Su, On choosing and bounding probability metrics, International Statistical Review / Revue Internationale de Statistique 70 (3) (2002) 419–435. doi:10.2307/1403865.
URL http://www.jstor.org/stable/1403865

[34] V. V. Williams, On some fine-grained questions in algorithms and complexity, in: Proceedings of the International Congress of Mathematicians (ICM 2018), 2019, pp. 3447–3487. doi:10.1142/9789813272880_0188.

[35] A. Abboud, F. Grandoni, V. V. Williams, Subcubic equivalences between graph centrality problems, apsp and diameter, in: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '15, Society for Industrial and Applied Mathematics, USA, 2015, pp. 1681–1697.

[36] M. Patrascu, Towards polynomial lower bounds for dynamic problems, in: Proceedings of the Forty-Second ACM Symposium on Theory of Computing, STOC '10, Association for Computing Machinery, New York, NY, USA, 2010, pp. 603–610. doi:10.1145/1806689.1806772.
URL https://doi.org/10.1145/1806689.1806772

[37] L. Lee, Fast context-free grammar parsing requires fast boolean matrix multiplication, Journal of the ACM 49 (1) (2002) 1–15. doi:10.1145/505241.505242.
URL https://doi.org/10.1145/505241.505242

[38] M. Parnas, D. Ron, Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms, Theoretical Computer Science 381 (1) (2007) 183–196. doi:https://doi.org/10.1016/j.tcs.2007.04.040.
URL https://www.sciencedirect.com/science/article/pii/S0304397507003696

[39] K. Onak, D. Ron, M. Rosen, R. Rubinfeld, A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size, in: Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms, SIAM, 2012, pp. 1123–1131.

[40] K. D. Ba, H. L. Nguyen, H. N. Nguyen, R. Rubinfeld, Sublinear time algorithms for earth mover's distance, Theory of Computing Systems 48 (2) (2011) 428–442. doi:10.1007/s00224-010-9265-8.

[41] A. McGregor, D. Stubbs, Sketching earth-mover distance on graph metrics, in: P. Raghavendra, S. Raskhodnikova, K. Jansen, J. D. P.

Rolim (Eds.), Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, Lecture Notes in Computer Science, Vol. 8096, Springer, Berlin, Heidelberg, 2013, pp. 274–286. doi:10.1007/978-3-642-40328-6_20.

[42] A. C.-C. Yao, Probabilistic computations: Toward a unified measure of complexity, in: 18th Annual Symposium on Foundations of Computer Science, 1977, pp. 222–227. doi:10.1109/SFCS.1977.24.

[43] W. Hoeffding, Probability inequalities for sums of bounded random variables, Journal of the American Statistical Association 58 (301) (1963) 13–30.
URL http://www.jstor.org/stable/2282952

[44] Y. Ollivier, C. Villani, A curved brunn–minkowski inequality on the discrete hypercube, or: What is the ricci curvature of the discrete hypercube?, SIAM Journal on Discrete Mathematics 26 (3) (2012) 983–996. arXiv:https://doi.org/10.1137/11085966X, doi:10.1137/11085966X.
URL https://doi.org/10.1137/11085966X

[45] R. J. Gardner, The Brunn-Minkowski inequality, Bulletin of American Mathematical Society 39 (3) (2002) 355–405. doi:10.1090/S0273-0979-02-00941-2.

[46] D. Cordero-Erausquin, R. J. McCann, M. Schmuckenschläger, A riemannian interpolation inequality à la borell, brascamp and lieb, Inventiones Mathematicae 146 (2001) 219–257. doi:10.1007/s002220100160.

[47] D. Cordero-Erausquin, R. J. McCann, M. Schmuckenschläger, Prékopa–leindler type inequalities on Riemannian manifolds, Jacobi fields, and optimal transport, Annales de la Faculté des sciences de Toulouse : Mathématiques Ser. 6, 15 (4) (2006) 613–635. doi:10.5802/afst.1132.
URL https://afst.centre-mersenne.org/articles/10.5802/afst.1132/

[48] C. H. Papadimitriou, K. Steiglitz, Combinatorial optimization: algorithms and complexity, Prentice-Hall, Inc., NJ, USA, 1982.