ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Panagiotis Kyriakis

Metric Approximations and Clustering

Master Thesis

Computer Engineering and Networks Laboratory
Swiss Federal Institute of Technology (ETH) Zurich

Supervision

Yuyi Wang
Prof. Dr. Roger Wattenhofer

April 2017

Preface

In this thesis culminates a tumultuous yet revelatory journey through the labyrinth of knowledge; a journey plagued with eternal questioning, continuous self-doubt and everlasting disenchantments; a journey of uneven struggle with the colossal achievements of science, in an idealistic and quixotic endeavor to master them all. Only to be left with the debilitating felling of grief and sorrow in the realization of my vanity; in the realization of the infinitesimality of my masteries; in the realization of the triviality of our beings when juxtaposed to the models conceived to describe them. A feeling that, ironically, impels me to surge through my ashes, reborn and keep struggling with the Leviathan of science; keep walking this futile road to the sempiternal quest for knowledge and drawing satisfaction in every tiny step. The end of this journey is marked by insurmountable gratification but signals no future consumed in sloth and self-indulgence. It rather harbingers the inception of a new journey; a harder journey of a continuous, life-long struggle in the vain hope of satisfying my greed for knowledge; the greed that has always kept me wondering in this labyrinth; the greed that clarifies, cuts through and captures the essence of my inquisitive spirit; the greed that - for the lack of a better word - is good.

Panagiotis Kyriakis
Zürich, April 2017

Contents

Abstract

The emergence of complex networks appearing in engineered, physical, biological and socio-economic systems and the higher-order structure of the observed interactions necessitate the development of novel techniques as, traditionally, simple pairwise relations are postulated. Axiomatically accepting that every networked system admits such representation may lead to loss of information and sub-optimal solutions when the interactions deviate from the pairwise model. In order for this pitfall to be circumvented, higher-order structures need to be considered and adopted. Not surprisingly, this reinforcement of the model comes with a caveat, as the complexity of the problem increases; an issue that needs to be addressed.

In this thesis, we address this issue by embedding and approximating the higher-order structure with a simpler one. More specifically, we consider the hypergraph model of n nodes and $|E|$ hyperedges in order to capture higher-order interactions among elements in a network and equip it with an appropriately chosen k-order metric. Then, we design a centralized probabilistic clustering algorithm that produces a hierarchical partitioning of the hypergraph. Constitutively, we create a rooted tree using the produced clusters, define a shortest-path based tree-metric and prove that the distortion associated with this embedding is $\mathcal{O}(\frac{k^2 \log n}{\log k})$, tightly bounded on the number of points. Additionally, departing from the centralized approach and confining ourselves to the 3-order case, we present a distributed structure that encodes the clustering procedure and develop an algorithm, tailored to the $\mathcal{CONGEST}$ model, intended for its computation. We prove the correctness of this distributed algorithm and provide linear on the shortest path diameter bounds of its message and round complexity, namely $\mathcal{O}(SPD_3|E| \log n)$ and $\mathcal{O}(SPD_3 \log n)$. Constitutively, we introduce a representation of the tree resulting from this distributed embedding and construct a distortion-invariant algorithm which reduces its height to $\mathcal{O}(\log n)$. Finally, we conclude by proving that the message and round complexity of the last algorithm are bounded, in terms of the weighted (Δ) and the hop (h_D) diameter, by $\mathcal{O}(|E|h_D \log \Delta)$ and $\mathcal{O}(h_D \log \Delta)$, respectively.

Keywords

hypergraphs, generalized metric spaces, tree metric, embedding, probabilistic approximation, clustering algorithms, distributed tree embedding, height reduction

Chapter 1

Introduction

1.1 Metric Embeddings

Loosely speaking, a metric space is a set of points equipped with a distance function ("metric") that defines how far apart two points of the space are. What formally defines a metric space is a set of *axioms*, which every candidate function must satisfy in order to be considered a metric. These axioms, arissing from the natural intuition of distance are commonly called *identity*, *symmetry* and *triangle inequality*.

Even though the study of metric spaces is a topic of theoretical mathematics, many researchers in computer science, motived by their wide applicability to practically useful problems, have put extensive effort in using them into their field. From this perspective, computer scientists are more interested in designing effective and efficient algorithms that can provide optimal solutions to the problems represented by metric spaces, and less interested in understanding their induced topological properties, even though the latter may sometimes aid in the former.

In computer science, metric spaces arise in certain classes of several important graph theoretical problems. When a problem can be represented by a graph, weights can be added to the edges and a distance function satisfying the aforementioned axioms can be induced, then the resulting set is a metric space. Even though virtually all problems can, in theory, be solved in the above setting, due to their computational intractability or our desire to obtain approximate but really fast solutions, it is often preferable to convert then to a simpler metric space, with trees being the most commonly chosen one. This "conversion" is called metric embedding and can be informally described as a mapping from the first space ("original") to the second ("host"). Trees are a special case of graphs, occurring when there exists only one way to reach any point from any other point and as one would expect. They are the most commonly chosen host space owing to all problems being easily if not trivially solvable on them.

Although this method appears as panacea, its utilization comes with a caveat: one cannot expect that the distances would remain invariant when embedding arbitrary graphs into trees, except in trivial cases. Thus, a certain amount of distortion shall be expected and ideally, with sagaciously chosen embedding methods, minimized. After the embedding the original space into the desired tree has been

obtained, we can tackle the problem by implementing a simper, a faster or more efficient algorithm and by converting the solution back to the former space. Whenever the algorithm on the tree is itself approximative, which is often the case, the distortion induced by the embedding only contributes a factor to the overall asymptotic approximation ratio. It should be noted that the embedding and the algorithm design procedure on the host space can be implemented completely independently from each other. In fact, owing the their simplicity, there exist already several algorithms for a wide range of problems on trees. Hence, purely by the embedding, one can solve the corresponding problems on arbitrary graphs or achieve better performance on existing solutions.

1.2 Motivation

Motivated by the theoretical as well as practical interest of the above technique, many researchers have proposed several[1] - initially centralized - algorithms for embedding and approximating metric spaces arising from the shortest path distance on graphs into trees. Additionally, the advent of large scale distributed and parallel systems, which makes centralized implementations primitive, costly or even intractable, reinforced by the fact that the simplicity of tree problems over graph problems is not particular to centralized systems, has led to the retrospective modernization and refinement of some of the original embedding methods.

Although the implicit assumption that simple pairwise interactions govern every problem chosen to be modeled by a graph may undermine the effectiveness of the solution and lead to substantial loss of information, the research community has not been enthralled by the possibility of devising methods to embed and approximate spaces arising from higher-order structures, such as hypergraphs. Dropping the naive hypothesis of pairwise relationships and adopting a hypergraph model can lead to substantial improvement of the applied methods, as it has been demonstrated by a plethora of examples in several fields, including spectral clustering techniques for machine learning [SPH07] and computer vision [ALZM+05], link prediction [LXLS13], music recommendation systems [BTC+10], community structure detection [LSC+09], matching and combinatorial auctions [KRTV13], multi-interface wireless networks [ACKP09], just to name a few. Additionally, many combinatorial optimization problems with a wide range of applications can be naturally expressed using the hypergraph model, including the full Steiner tree concatenation [BZ15], the minimum-flow maximum-cut [PM03], the perfect matching [AY05] and the multiway partitioning [EN14].

In order to improve the effectiveness of the solutions of such problems, it is imperative to consider approximating hypergraphs by a simpler structure. However, whenever the hypergraph model is chosen as a representation of the problem at hand, inevitably the notion of distance between three or more points arises. The intrinsic inability of the so far proposed algorithms to deal with such higher-order metrics necessitates the development of novel embedding methods, adapted to higher-order

[1] Look at the following section for a literature review.

structures. Such methods, if shrewdly chosen, are guaranteed to improve the current solutions of problems defined on hypergraphs, as they would only need to be solved in the simpler space. Motivated by this need to lessen the ostracism the appears to have plagued higher-order embeddings and in the idealistic hope of acting as quintessential exemplar for further research in this topic, we develop in this thesis a method that embeds and approximates higher-order metrics that arise on hypergraph models.

1.3 Previous Work

Finite metric embeddings have been a vibrant field of research. In the realm of normed host spaces, Burgain's famous theorem [Bou85] guarantees an $\mathcal{O}(\log n)$ distortion bound when embedding into l_2 of dimension exponential in n. The dimension was later improved to $\mathcal{O}(\log^2 n)$ and the distortion bound of Euclidean embeddings was proved to be tight [LLR95]. In [Mat96], a embedding into a $\mathcal{O}(Dn^{2/D}\log n)$-dimensional l_∞-space inducing distortion of D was proved to exist. In [Fei00], Feige strengthened the previous embeddings by accounting for the volume distortion of k-simplices and proved a $\mathcal{O}(\log n + \sqrt{k \log n \log k})$ distortion bound in l_2. As the literature on embedding finite metrics into normed spaces is vast, for a comprehensive review of state-of-the-art techniques the reader is directed to [Mat02, IM04].

Motivated by the algorithmic point of view, trees are a commonly chosen host metric space. On the backbone of proposed methods lie probabilistic algorithms, which embed the original space not into a single but rather into a distribution over different tree metrics. Their superiority stems from the fact that they provide acceptable distortion bounds even for classes of metric, for which deterministic methods fail to do so, as it can be easily inferred by constructing graphs that cannot be embedded into a *single* tree metric with lower than $\Omega(n)$ distortion [RR98]. This bound can be improved by using probabilistic methods, as outlined in the following.

One of the earliest probabilistic approaches, inducing a distortion bounded exponentially on $\mathcal{O}(\sqrt{\log n \log\log n})$, was presented in [AKPW95], where a graph was approximated by a distribution over spanning trees. In [Bar96], Bartal defined probabilistic embeddings over distributions trees and proved an $\mathcal{O}(\log^2 n)$ distortion bound. His trees had a special structure, which he termed hierarchically well separated, whereby the weights on successive levels decay by a constant factor. Additionally, he showed that the embedding of arbitrary graphs into distribution over[2] tree induces a distortion of at least $\Omega(\log n)$. Later [Bar98], he improved his previous result by providing a $\mathcal{O}(\log n \log\log n)$ upper bound. His latest method was derandomized and used in order to obtain deterministic approximation algorithms for several problems, such as the buy-at-bulk network design [CCG+98], and group Steiner tree [CCGG98]. In [KRS01], the distortion upper bound of Bartal's algorithm was shown to be $\mathcal{O}(\log n)$ for planar graphs. The relevant probabilistic approaches culminated in the FRT-method presented in [FRT04], which was proved

[2]Since we are only concerned with distributions over trees, the "distribution over" may be omitted.

to be existentially optimal as it provided an $\mathcal{O}(\log n)$ approximation, therefore closed the gap between the upper and the lower bound.[3]

Due to its proclaimed optimality, the FRT-method has evolved into a highly influential and archetypal technique and given a profound impetus to further improve and tailor it to different settings. A parallel implementation, requiring $\mathcal{O}(n^2 \log n)$ work and $\mathcal{O}(\log^2 n)$ depth, was presented in [BGT12] and used for obtaining RNC approximation algorithms for the k-median and buy-at-bulk network design problems. The work was latter reduced to $\mathcal{O}((|E| + n^{1+\epsilon}) \log n)$ in [FL15] at the expense of an increased stretch of ϵ^{-1} on the distortion bound. In [KKM+12], a distributed implementation of a round complexity bounded by $\mathcal{O}(SPD \log n)$[4] was presented and used for developing a distributed approximation algorithm for the Steiner forest problem. The round complexity was then improved to almost $\mathcal{O}((h_D + \sqrt{n}) \log n)$[5] [GL14] inducing only a constant factor on the distortion bound. Finally, in [BGS16], a time efficient implementation of $\mathcal{O}(|E| \log n)$ complexity was proposed and used for constructing approximate distance oracles based on the resulting FTR trees.

1.4 Contribution of the Thesis

The work presented in this thesis is of theoretical as well as practical interest, due to universality of the proposed methods and the wide range of applications of hypergraph models, as argued in Sec. 1.2. Its novelties and contributions are essentially threefold: the cornerstone of the originality is the first embedding algorithm that approximates higher-order metrics (also called generalized metrics) with a distribution over tree metrics. The algorithm is itself probabilistic and produces a hierarchical decomposition of the hypergraph. The second major contribution is its distributed implementation, achieved by constructing an appropriate structure to encode the decomposition, developing an efficient algorithm to compute it and by distributively representing the resulting tree. Finally, an original algorithm, which reduces the height of such distributed tree representations, was designed.

The rest of the text is organized as follows: In *Chapter 2*, we present the fundamental concepts of generalized metric spaces and hypergraphs. We define the probabilistic embedding and approximation problem and, finally, introduce some rudimentary properties of distributed systems. In *Chapter 3*, we present and analyze the probabilistic embedding algorithm. We prove that it solves the approximation problem and bound the expected distortion it induces. Also, we argue for the existential optimality of the embedding and manipulate the properties of the resulting tree. In *Chapter 4*, we present the distributed tree embedding and height reduction algorithms, prove their correctness and analyze their performance using the message and round complexity. In *Chapter 5*, we conclude by summarizing and discussing our results and by giving directions for future research.

[3] With some abuse of notation, we may use the term "lower bound" in place of "lower bound of upper bound". Look at Def. 2.9.

[4] SPD = shortest path diameter, look at Sec. 2.2.

[5] h_D=hop diameter, look at Sec. 2.2.

Chapter 2

Metrics, Graphs and Embeddings

In this chapter we formalize the concepts and give the pertinent mathematical background necessary for understanding the methods developed in this thesis. We start by reviewing the axiom systems of metric spaces and by examining their induced graph representations. Then we give the formal definition of the metric embedding problem and finally present the distributed computation model which is used for the decentralized implementation of our methods.

2.1 Generalized Metric Spaces

The notion of metric ("distance") between two points is one of the most known and widely understood concepts in mathematics and can by expressed by the following axioms.

Definition 2.1. Let X be a set of points, then *metric* is said to be any function $d : X \times X \to \mathbb{R}_{\geq 0}$ that satisfies

A 1. $d(x, y) = 0 \Leftrightarrow x = y$

A 2. $d(y, x) = d(y, x)$

A 3. $d(x, y) \leq d(x, z) + d(z, y)$

for all $x, y, z \in X$. The resulting tuple (X, d) is called *metric space*. ■

The axioms are called *identity*, *symmetry* and *triangle inequality*, respectively. When X is an inner product space, the Euclidean metric, defined as $d(x, y) = \sqrt{\langle x, y \rangle}$, where $\langle \cdot, \cdot \rangle$ denotes the inner product, is a commonly used metric and corresponds to our natural intuition of distance.

Even though such metrics between two points are a well studied topic in mathematics, generalized[1] metric spaces, which formalize the concept of distance between three or more points, have been ignored. Only in 1928 Menger in [Men28] introduced that notion by taking k points in a high-dimensional Euclidean space and assigning their distance to be the volume of the simplex that they define. Following

[1] When it is clear from the context that we refer to a generalized metric or the distinction makes no difference, we omit the word "generalized"

that, Gähler in [Gäh63, Gäh64] restricted himself to three points, defined an axiom system, which he named 2-metric and studied the properties of the resulting space. Motivated by the pitfalls of the 2-metric, namely the lack of necessity for continuity [Lin90] and the absence of an easy correspondence to a metric (which was proved by Gähler himself), Dhange slightly modified the axioms and defined the $\mathbb{D}$-metric [Dha92]. Both of these metrics encompass similar definitions, which we present here by following Menger's approach for arbitrary number of k points.

Definition 2.2. Let X be a set of points and $p(\cdot)$ be a function that returns the permutations of its arguments, then $\mathbb{2}_k$-*metric* ($\mathbb{D}_k$-*metric*) is said to be any function $d : X^k \to R_{\geq 0}$ that satisfies the following axioms

A 1. $d(x_1, x_2, \ldots, x_k) = 0 \Leftrightarrow$ at least two of $(x_1, x_2, \ldots, x_k)$ equal (all equal)

A 2. $d(x_1, x_2, \ldots, x_k) = d(p(x_1, x_2, \ldots, x_k))$

A 3. $d(x_1, x_2, \ldots, x_k) \leq \sum_{i=1}^{k} d(x_1, x_2, \ldots, x_{i-1}, x_{k+1}, x_{i+1}, \ldots, x_k)$

for all $x_1, x_2, \ldots, x_{k+1} \in X$. The resulting tuple (X, d_{2_k}) $((X, d_{D_k}))$[2] is called $\mathbb{2}_k$-*metric space* ($\mathbb{D}_k$-*metric space*) and the integer $k \geq 2$ is called *order* of the metric. ■

We observe that the only difference introduced by Dhange is the identity axiom, namely k points are identical if at least two of them are equal ($\mathbb{2}_k$-metric) or if all of them are equal ($\mathbb{D}_k$-metric). The third axiom is a generalization of the tetrahedral inequality for k-simplices. These metrics can be picturized as the volume ($\mathbb{2}_4$-metric) and the surface ($\mathbb{D}_4$-metric) of the hexahedron formed by 4 points in a three dimensional space. Dhange's modification was not however sufficient to provide the intended correspondence, as the metric induced by a given $\mathbb{D}_k$-metric need not satisfy the triangle inequality [MS04]. Based on that, Mustafa and Sims introduced in [MS06] a new generalized metric, which they named $\mathbb{G}$-metric

Definition 2.3. Let X be a set of points and $p(\cdot)$ be a function that returns the permutations of its arguments, then $\mathbb{G}_k$-*metric* is said to be any function $d : X^k \to \mathbb{R}_{\geq 0}$ that satisfies the following axioms

A 1. $d(x_1, \ldots, x_k) = 0 \Leftrightarrow$ all $(x_1, \ldots, x_k)$ equal

A 2. $d(a_i) \leq d(a_{i+1}), a_i = (\underbrace{x_1, \ldots, x_1}_{k-i}, x_2, \ldots, x_{i+1}) \Leftrightarrow |\{x_2, \ldots, x_k\}| = k - 1$

A 3. $d(x_1, \ldots, x_k) = d(p(x_1, \ldots, x_k))$

A 4. $d(x_1, \ldots, x_k) \leq d(x_1, x_{k+1}, \ldots, x_{k+1}) + d(x_{k+1}, x_2, \ldots, x_k)$

for all $x_1, \ldots, x_{k+1} \in X$ and $i = 1, \ldots, k$. The resulting tuple (X, d) is called $\mathbb{G}_k$-*metric space* and the integer $k \geq 2$ is called *order* of the metric. ■

The newly introduced second axiom accounts for the fact that the distance of a set of points increases as *new* points are added to the set. The $\mathbb{G}_k$-metric can be picturized as the perimeter of the polygon formed by the k points. As pointed out in [MS06], for the $\mathbb{G}_3$-metric there exists a simple correspondence to a metric (which

[2] We subscript the functions to avoid confusion.

can be extended to k points), hence guarantying also the continuity of the former. Therefore, the $\mathbb{G}_k$-metric alleviates the shortcomings of the other two and appears to be the most prudent choice of an axiom system for the k-point distance. Finally, the three new metric spaces simplify to the Def. 2.1 for $k = 2$, as intuitively expected.

2.2 Hypergraph Representations

From the computer science perspective we are interested in utilizing metric spaces to model "costs" arising from practically useful problems and in designing algorithms to efficiently solve them. As such, it is imperative that they are represented by a "computer-friendly" structure, with the most appropriate being that of a weighted hypergraph.

Definition 2.4. A *weighted hypergraph* $\mathcal{H}$ is a triple (V, E, w) consisting of a set of *vertices (or nodes)* V, a set of multisets[3] called *hyperedges* E, which is created by taking from elements of the powerset 2^V and a *weight* function $w : E \to \mathbb{R}$. The set $N(v) = \{u \in V : (v, u) \subseteq e \text{ for some } e \in E\}$ is called *neighbors* of the node v. If the cardinality of each hyperedge is equal to k then the hypergraph is called *k-uniform* and if each possible multiset of cardinality k belongs to E, then it is called *k-complete*. Finally, if for all $e \in E \Rightarrow p(e) \in E \wedge w(e) = w(p(e))$[4], then the hypergraph is called *undirected*. ■

It is easy to see that each metric space defined in the previous section admits a hypergraph representation. Specifically, given a *finite* $\mathbb{G}_k$-metric space (X, d) we construct a (possibly directed) k-regular, k-complete hypegraph consisting of the vertex set X and for each hyperedge e assign its weight to be $d(e)$. Naturally, the same observation holds for the $\mathbb{2}_k$ and $\mathbb{D}_k$-metric spaces and we also note that for the special case $k = 2$ the representation is the commonly understood notion of graphs[5]. Having established this correspondence of a generalized metric space to a hypergraph, the question that naturally arises is whether the converse can be deduced. In order to answer that, we need some additional definitions.

Definition 2.5. Given a k-regular hypergraph $\mathcal{H} = (V, E, w)$ and a set $S \in V^k$, a *chain* C_S *on* $\mathcal{H}$ is said to be any alternating sequence $\{v_0, e_1, v_1, e_2, v_2 \cdots e_l, v_l\}$ of distinct vertices v_i and distinct hyperedges e_i such that $v_{i-1}, v_i \in e_i$ and for all $s \in S \Rightarrow s \in C_S$. The *path* of the chain is defined as $P_S = \{e : e \in C_S\}$. If $\mathcal{H}$ is non-negatively weighted, then the *shortest path*, the *distance* of S and the *diameter* of $\mathcal{H}$ are defined respectively as

$$P_S^* = \arg\min_{P_S} \sum_{e \in P_S} w(e) \tag{2.1}$$

$$W_d(S) = \sum_{e \in P_S^*} w(e) \tag{2.2}$$

[3]Multiset is an unordered set that allows repeated elements.
[4]Function $p(\cdot)$ as previously defined
[5]The absence of the prefix "hyper" implies $k = 2$.

$$W_D(\mathcal{H}) = \max_{S \in V^k} W_d(S) \tag{2.3}$$

Similarly, we define the *hop distance* $h_d(S)$ and *hop diameter* $h_D(\mathcal{H})$ simply by setting $w(e) = 1$ for all $e \in E$. Finally, the *k-shortest path diameter* SPD_k of the hypergraph is the lowest positive integer h such that for all $S \in V^k$ there exists a least-weight path P_S of cardinality at most h. ■

We can easily deduce that the function $W_d(S)$ induces a $\mathbb{D}_k$ or $\mathbb{G}_k$ metric on the hypergragh, as it satisfies the respective axioms. However, the same need not hold for the $\mathbb{2}_k$ metric due to contradictory definitions of the identity axiom. Hence, we have established a duality between the $\mathbb{D}_k$ or $\mathbb{G}_k$-metric and a hypergraph, since the former gives rise to the latter and vice versa. As such, we are using the two terms interchangeably in the rest of the thesis.

Definition 2.6. A graph $\mathcal{G}$ is called *tree* $\mathcal{T}$ if there exist a unique path between any two nodes. When $\mathcal{T}$ has a node designated as *root* then it is called *rooted tree* and the set of terminal nodes of every maximal-weight path originating from the root is called *leaves*. The hop diameter of the tree is called *tree height*. Also, if the weights from the root to its children are equal and they decrease by a factor of l in each step on every root-to-leaf path, then the tree is called *l-hierarchically well separated* (*l-HST*). ■

Similarly to graphs, trees can also induce a metric. In the latter case the metric itself is quite simpler, as there is no need to find the shortest path between any two nodes. The l-HST is a pivotal type of tree in this thesis, as it appears in the solution of the embedding problem defined below.

2.3 Embedding and Approximation

Given a metric space[6] the procedure encompasses the construction of a mapping to a new space and a metric function in the latter.

Definition 2.7. Let (X, d_X) be an *original* k-th order metric space and (Y, d_Y) be a *host* l-th order metric space, then a *metric embedding* of X into Y is defined to be any mapping $\phi : X \to Y$. The *distortion* induced by ϕ is defined to be $\sigma_\phi = \inf_{\sigma \in S_\sigma} \sigma$, where

$$S_\sigma = \left\{\sigma \geq 1 : \exists r > 0 \text{ s.t } r \cdot d_X(e) \leq d_Y(\phi(e)) \leq r \cdot \sigma \cdot d_X(e), \forall e \in X^k\right\} \tag{2.4}$$

The number r is called *scaling factor*. ■

Note that r permits scaling of the metric without any effect on the distortion. Given this definition, we formalize the probabilistic approximation problem as follows.

[6] Or a hypergraph, remember that the concepts are used interchangeably.

Definition 2.8. Given an original k-th order metric space (X, d_X), a family of l-th order host metric spaces (Y, d_Y), an embedding $\phi : X \to Y$ and a distribution $\mathcal{P} : (Y, d_Y) \to [0, 1]$, we say that $(Y, d_Y, \mathcal{P})$ *a_U-probabilistically approximates* (X, d_X) and write

$$(X, d_X) \xhookrightarrow[\phi]{a_U} (Y, d_Y) \xrightarrow[\mathcal{P}]{} [0, 1]$$

if, for some $a_U > 0$, the following hold

$$d_Y(\phi(e)) \geq d_X(e) \tag{2.5}$$

$$E_{\mathcal{P}}\left[\frac{d_Y(\phi(e))}{d_X(e)}\right] \leq a_U \tag{2.6}$$

for all $e \in X^k$. The number a_U is called *upper bound* of the distortion. ■

We observe that the first requirement ensures that no distance gets contracted, that is, the host metrics *dominate* the original, and the second one bounds the *expected* distortion induced by the embedding, with the expected value taken with respect to the given probability distribution. It is obviously desired that the parameter a_U is as small as possible, ideally equal to its lowest possible value, which is given by the following definition.

Definition 2.9. Given the following embedding

$$(X, d_X) \xhookrightarrow[\phi]{a_U} (Y, d_Y) \xrightarrow[\mathcal{P}]{} [0, 1]$$

the *lower bound of the upper* of the distortion is defined as

$$a_L = \min_{S_H} \left\{ a : \forall e \in X^k \Rightarrow E_P\left[\frac{d_Y(\phi(e))}{d_X(e)}\right] \leq a \right\} \tag{2.7}$$

where S_H is the set of all hypergraphs, whose representation is the space (X, d_X). When an embedding induces distortion such that $a_U = \mathcal{O}(a_L)$, we say that it is *tightly bounded*. ■

The above definition essentially provides an optimality criterion for the embedding, as it is outlined in the following theorem.

Theorem 2.1. *Given the following embedding*

$$(X, d_X) \xhookrightarrow[\phi]{a_U} (Y, d_Y) \xrightarrow[\mathcal{P}]{} [0, 1]$$

assume that $(Y, d_Y, \mathcal{P})$ a_U-probabilistically approximates (X, d_X) and $a_U = \mathcal{O}(a_L)$, then there exists no other embedding such that (X, d_X) can be a-probabilistically approximated by $(Y, d_Y, \mathcal{P})$ for $a < a_U = \mathcal{O}(a_L)$.

Proof. Let $\mathcal{H}'$ be the hypergraph that achieves the minimum in Eq. 2.7 and assume that a lower-distortion embedding $\phi' \neq \phi$ exists, that is, the embedding

$$(X, d_X) \xhookrightarrow[\phi']{a_U} (Y, d_Y) \xrightarrow[\mathcal{P}]{} [0, 1]$$

a-probabilistically approximates (X, d_X) for some $a < \mathcal{O}(a_U)$. However, since $\mathcal{H}'$ can also induce a (X, d_X) space, we infer, based on Eq. 2.7, that previous embedding induces a distortion of at least $a_U = \mathcal{O}(a_L)$, which contradicts the initial assumption. Hence, no such ϕ' exists. □

It is evident from the above, that we can prove the optimality of an a_U-probabilistic embedding simply by constructing a hypergraph that cannot be embedded with distortion less than $\mathcal{O}(a_U)$. Such a tight bound is a really powerful tool in metric embeddings, because, in its presence, one need not search for lower-distortion embeddings.

2.4 Distributed Model

Even though the so far proposed methods are inherently centralized, the metric embedding and approximation problem defined above is not restricted to such a case. Given that the distributed algorithm design is often easier on trees than on general graphs, one should expect that there exists some merit in implementing such embedding techniques distributively.

In the distributed setting, each node v in a hypergraph $\mathcal{H} = (V, E, w)$ of n nodes knows only its neighbors $u \in N(v)$ as well as the respective weights $w(\cdot)$, while being completely unaware of the topology of H. The presence of a unique clock, that synchronizes the transmissions, is assumed and in each clock's time step ("round") each node can send an arbitrary message of size $\mathcal{O}(B)$ to all of its neighbors. Such a restriction in the transmission capacity is necessitated by natural limitations arising in every physically implementable communication channel. Moreover, note that in the absence of a restriction the distributed algorithm design is itself futile, as one can simply send all the information in a single convergecast to a *leader* node (set Th. 4.3 for leader election) and perform all the computation there. They above presented setting is a commonly used model in distributed computing called $\mathcal{CONGEST}(B)$ model [Pel00].

Taking into consideration that the round time interval is usually of several orders higher than the processor clock, we further refine the $\mathcal{CONGEST}(B)$ model by assuming that local computations performed in each node are free up to a polynomial factor. As such, the time complexity of local computations is of no relevance, as long as it remains within $\mathcal{O}(n^m)$ for some positive constant m. As it can be easily inferred, all the distributed methods proposed in Ch. 4 satisfy this requirement. Therefore, in the distributed setting we use different measures to assess and compare our algorithms, as outlined bellow.

Definition 2.10. Given a distributed, synchronous algorithm $\mathcal{A}$ over a hypergraph $\mathcal{H} = (V, E, w)$, let M_v be the set of all messages that node $v \in V$ needs to send during the worst-case execution of $\mathcal{A}$. For all messages $m \in M = \bigcup M_v$, define the *message hop* h_m as the number of nodes that m is expected to visit before reaching its destination. Also, let t_v be the *waiting hop count*, during which node v needs to

wait for other messages to be transmitted. Then we define

$$\mathcal{C}_M(\mathcal{A}) = \sum_{m \in M} h_m \tag{2.8}$$

$$\mathcal{C}_R(\mathcal{A}) = \max_{v \in V} \left\{ t_v + \sum_{m \in M_v} m \cdot h_m \right\} \tag{2.9}$$

as the *message* and *round complexity* of $\mathcal{A}$, respectively. ■

It is straight forward to see that, unless none or one message is transmitted per round, the message complexity is higher than the round complexity and that the former one is upper bounded by the latter, that is $\mathcal{C}_M(\mathcal{A}) \leq \mathcal{C}_R(\mathcal{A}) \sum_{e \in E} |e|$. Both of those measures are equally important in assessing the performance of distributed algorithms, as the first one accounts for storage and bandwidth and the second one for time requirements of the algorithm. Finally, as pointed out it Th. 2.1, the optimality of an algorithm with respect to $\mathcal{C}_M$ or $\mathcal{C}_R$ can be proved by providing the respective tight bound.

Chapter 3

Embedding $\mathbb{G}_k$-Metric Spaces

Along the lines of the exemplary FRT-method presented in [FRT04], we develop a centralized method that embeds a $\mathbb{G}_k$-metric space into a distribution over tree metrics. The approach is based on a probabilistic, hierarchical partitioning of the hypergraph into clusters of decreasing diameters. The $\mathbb{G}_k$-metric was chosen as an axiom system for the k-point distance due to its postulated superiority over the other two metrics, as commented in the previous chapter. Also, even though our ultimate goal is to embed the k-th order metric, presenting our method for arbitrary k would be a laborious and cumbersome task. Given that its extension from the special to a general case is intuitive, it appears to be prudent to restrict ourselves to the $k = 3$ case.

We start by describing how the our algorithm partitions the hypergraph into a set of clusters. Then we map them to the host metric space, define the host metric, prove its dominance over the original one and bound the induced expected distortion. After that we comment on the properties of the resulting host space and on how it can be further manipulated and optimized. Finally, we intuitively describe the extension to the k-order case.

3.1 Clustering Algorithm

Before we start describing the method we need to give some preliminary definitions of what a cluster and a laminar family are.

Definition 3.1. Given a set of points X a *cluster* C is defined to be any subset of the powerset 2^X. A set of clusters $\mathcal{F} \subseteq 2^X$ is said to be a *laminar family* of clusters if for any $A, B \subseteq \mathcal{F}$ it is the case that $A \subseteq B$ or $B \subseteq A$ or $A \cap B = \emptyset$. A laminar family can be easily mapped to a tree $\mathcal{T}$ as follows: Each set $C \subseteq \mathcal{F}$ is mapped to a node n_C on $\mathcal{T}$ and each maximal subset of C corresponds to a child of n_C. ■

Having been equipped with the above definition, we present in the following a clustering algorithm $\mathcal{A}_C$, which, given an original $\mathbb{G}_3$-metric space (X, d) of n points, decomposes the hypergraph recursively to a laminar family of clusters. We assume, without loss of generality, that each hyperedge has a distance of at least 1 and that

the diameter Δ can be written as $\Delta = 3^\delta$, for some positive integer δ. Note that such assumption poses no restriction as the weights of the hypergraph can be scaled at will. Such a scaling has no impact on the distortion due to multiplicative constants being of no importance, as it will become apparent later. Noting that the algorithm is recursive, we define the following.

Definition 3.2. Define $L = \{\delta, \delta - 1, \delta - 2 \dots 0\}$ as the *recursion level set* and for all $i \in L$ let S_{ij} be the j-th cluster created at the i-th level. Also, for all $v \in V$, define

$$a_i(v) = \begin{cases} 1 & \text{if and only if there exists } k : v \in S_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

When $a_i(v)$ has unit value we say that vertex v has been *assigned* to some cluster at level i. ■

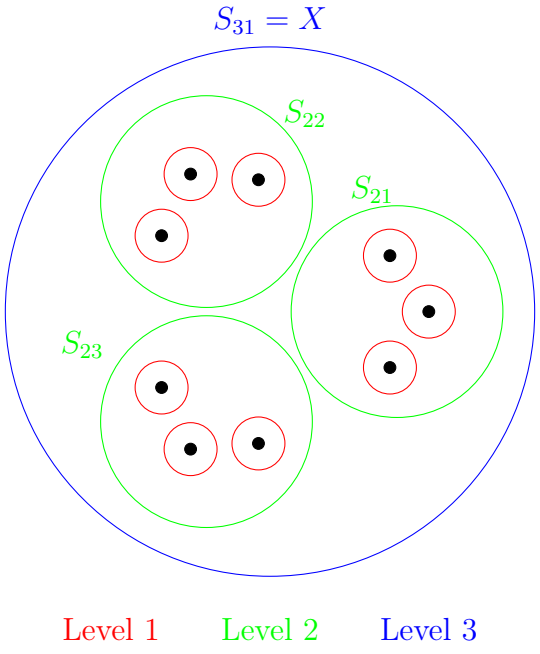


Figure 3.1: Laminar family and levels

The clustering algorithm, formally presented[1] in Algorithm 1, can be intuitively described as follows: Initially, we pick a random permutation π of all the vertices in X, which remains constant throughout the algorithm. Additionally, we sample a number β from the distribution

$$f_b(x) = \begin{cases} \frac{1}{x \ln 3} & x \in [1, 3] \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

At each iteration four parameters are given to the algorithm: a cluster S, the level i, the number β and the random permutation π. If the termination condition ($|S| = 1$)

[1] $\wedge$: logical AND, $\oplus$: logical XOR, $\downarrow$: logical NOR

is invalid, we set all $a_i(u), u \in S$ equal to 0 and compute $\beta_i = 3^{i-1}\beta$. Then, we loop over the random permutation and for each vertex $p_j \in \pi$ create a sub-cluster S_{ij} consisting of all edges[2] $(u, v) \in S$ that have not been assigned to a different cluster at the same level and whose distance from p_j is at most β_i. In this case, we call p_j the *origin vertex* of the cluster S_{ij}. Finally for each created sub-cluster we reduce the level by one and call the algorithm recursively. Initially, the algorithm is called

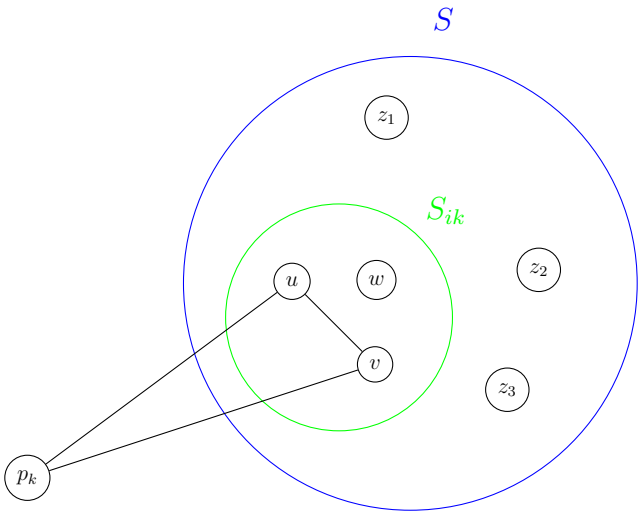


Figure 3.2: Cluster Construction

Algorithm 1 The $\mathcal{A}_C$ Clustering Algorithm

procedure ACALGORITHM(S, i, β, π)
  if $|S| = 1$ **return**
  For all $u \in S$ **set** $a_i(u) \leftarrow 0$
  $\beta_i \leftarrow 3^{i-1}\beta$
  for $j = 1, 2 \cdots n$ **do**
    $p_j \leftarrow \pi(j)$
    $S_{ij} \leftarrow \left\{(u, v) \in S \mid \left(i \oplus u \neq v\right) \wedge \left(d(u, v, p_j) \leq \beta_i\right) \wedge \left(a_i(u) \downarrow a_i(v)\right)\right\}$
    For all $v \in S_{ij}$ **set** $a_i(v) \leftarrow 1$
  end for
  For all non empty $S_{ij}, j = 1, 2 \cdots n$ **call** ACALGORITHM($S_{ij}, i - 1, \beta, \pi$)
end procedure

with $S = X, i = \delta$ and terminates when it reaches singleton clusters ($|S| = 1$). Since each edge can be assigned only to one cluster at each level and given that clusters at lower levels are subsets of clusters at higher level, the above procedure defines a laminar family. We need to note that p_j need not belong to the set S, since we loop over all vertices of the hypergraph. However, this creates no intersections between

[2] With some abuse of notation; a pair of nodes need not belong to the hyperedge set of a 3-regular hypergraph.

the created sub-clusters because p_j itself is not included in any of them. The vertex p_j will eventually be included in a cluster, but only when the condition on line **10** is satisfied.

3.2 Host Metric Space

As we mentioned above, the above decomposition defines a laminar family which, as outlined in Def. 3.1, corresponds to a rooted tree as follows: Each cluster S_{ij} corresponds to a node in the tree and each maximal sub-cluster in S_{ij} corresponds to a child of the node corresponding to S_{ij}. The tree distance between the parent node and its children is assigned to be $3^i\beta$ (which is equal to the maximum diameter of each cluster at level i, as it will be proved in the following). It should be obvious from the clustering algorithm that the root of the tree ($i = L = \delta$) is the set of all vertices in the hypergraph and the leafs ($i = 0$) are the vertices themselves.

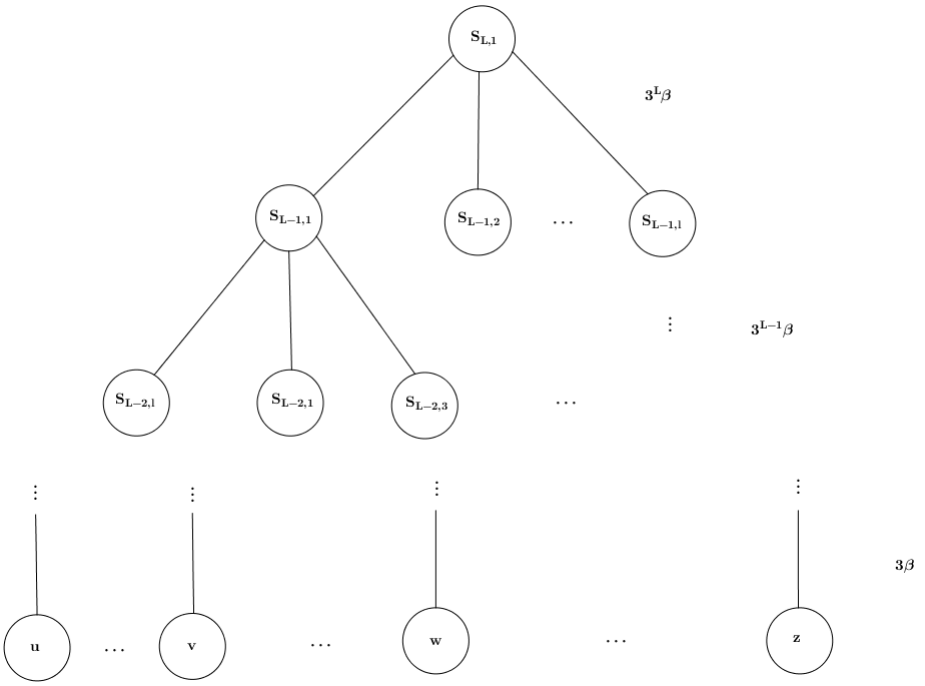


Figure 3.3: The resulting 3-HST

The above constructed tree is formally denoted as $\mathcal{T} = (V_T, E_T, w_T)$ where V_T is the set of nodes, E_T is the set of edges connecting these nodes and w_T are the weights of the edges. Following Def. 2.6, since the weights on any root-to-leaf path decrease by a factor of 3 in each step, $\mathcal{T}$ is a *3-hierarchically well separated* (*3-HST*) tree, with its height being $\mathcal{O}(\log \Delta)$.

Definition 3.3. Let X_T be the set of all leaves of $\mathcal{T}$ and for all $(u, v, w) \in X_T^3$ define the host metric as

$$d_T(u, v, w) = W_d(u, v) + W_d(v, w) + W_T(w, u) \tag{3.3}$$

where $W_d(\cdot,\cdot)$ is the weighted distance of the respective leaves, as defined in the Eq. 2.2. The tuple (X_T, d_T) is host metric space. The mapping from the original to the host space is

$$\phi \triangleq \mathbb{1} : u \mapsto u \text{ for all } u \in X \tag{3.4}$$

where $\mathbb{1}$ is the identity function. ■

3.3 Probabilistic Approximation

Having defined in the previous section the host metric space, we summarize the main result of our method in the following theorem.

Theorem 3.1. *For a given $\mathbb{G}_3$-metric space (X, d), the host metric space (X_T, d_T) and the mapping ϕ defined above, the following holds*

$$(X, d_X) \xrightarrow[\phi]{\mathcal{O}(\log n)} (Y, d_Y) \xrightarrow[\mathcal{P}]{} [0, 1]$$

where $\mathcal{P}$ is a probability distribution function.

The proof of this theorem is partitioned to several different lemmas. Initially, we upper bound the diameter of each created cluster, then prove that the host metric dominates the original and finally provide the postulated by the above theorem $\mathcal{O}(\log n)$ upper bound of the expected distortion. Although the distribution $\mathcal{P}$ is implicitly defined, its explicit definition is of no interest, thus no attempt is made to find it.

Lemma 3.2. *The diameter of each created cluster is at most $3^i\beta$.*

Proof. Assume that the vertex p is the origin of a cluster S . Then, by definition, for each $u, v \in S \Rightarrow d(u, v, p) \leq 3^{i-1}\beta$. Based on that, for all $(u, v, w) \in S$ we have

$$d(u, v, w) \leq d(u, p, p) + d(p, v, w) \leq d(u, v, p) + d(v, w, p) \leq 3^i\beta \tag{3.5}$$

where we used the the second and the fourth axioms of the G_3-metric. □

With the aid of the above lemma, we prove the dominance of the host metric.

Theorem 3.3. *The host metric dominates the $\mathbb{G}_3$-metric.*

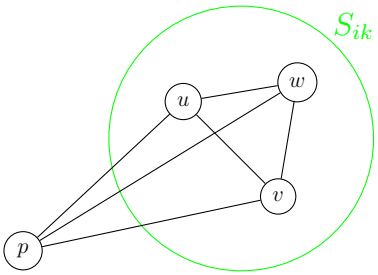
Proof. We say that an arbitrary hyperedge (u, v, w) is separated at level L if for each $e \in \{(u, v), (u, w), (v, w)\}$ it holds that L is the lowest level for which there exists a cluster in which e belongs. Based on that, there exist two separation levels L_1, L_2. Assuming, that $L_2 \geq L_1$, the host metric can be expressed as

$$d_T(u, v, w) = 4\sum_{i=1}^{L_2} 3^i\beta + 2\sum_{i=1}^{L_1} 3^i\beta > 3^{L_2}\beta \geq d(u, v, w) \tag{3.6}$$

The equality holds because the host metric is essentially the sum of three paths between three leaves. Two of these paths have one of their least common ancestors at level L_2 and the other at level L_1. Note that the hyperedge (u, v, w) appears in the right hand side of the equality only implicitly, through the separation levels. □

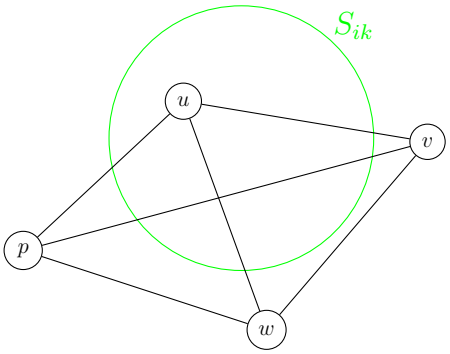
Theorem 3.4. *The original $\mathbb{G}_3$-metric is distorted in expectation by $\mathcal{O}(\log n)$, at most.*

Proof. Let us fix an arbitrary hyperedge (u, v, w). We say that the vertex p settles the hyperedge (u, v, w) at level i if at least one of the vertices u, v, w is include at level i in the cluster originating from p. We say that the vertex cuts the hyperedge if at least one and at most two of the vertices are included.



$$d(u, v, w) \leq 3^{i-1}\beta$$

Figure 3.4: Hyperedge settling



$$d(u, v, w) > 3^{i-1}\beta$$

Figure 3.5: Hyperedge cutting

Then, we define the following

$$d_T^p(u, v, w) \triangleq \sum_i \mathbb{1}\Big(p \text{ cuts } (u, v, w) \text{ at level } i\Big) 3^{i+2}\beta \tag{3.7}$$

where $\mathbb{1}(\cdot)$ is an indicator function[3]. Taking into account Eq. 3.6 and the separation levels as defined in Th. 3.3, we upper bound the host metric using the previous equation as follows

$$d_T(u, v, w) \leq \sum_p d_T^p(u, v, w) = 2\sum_{i=1}^{L_2} 3^{i+2}\beta + \sum_{j=1}^{L_1} 3^{j+2}\beta \tag{3.8}$$

[3]The indicator function is equal to 1 if the argument is true, otherwise it is 0.

The equality in the above equation holds because between levels L_2 and L_1 the hyperedge is being cut twice and between levels 1 and L_2 trice, while the inequality part is obvious from Eq. 3.6. Next, we define the distance of the vertex p from the hyperedge (u, v, w) as

$$d_p(u, v, w) \triangleq \min\{d(u, v, p), d(u, w, p), d(v, w, p)\} \tag{3.9}$$

Given this definition, we arrange every vertex on the hypergraph in order of increasing distance from the hyperedge (u, v, w) (breaking ties randomly) and pick the s-th vertex in the sequence. We assume without loss of generality that

$$d(u, v, p_s) \leq d(u, w, p_s) \leq d(v, w, p_s) \tag{3.10}$$

For p_s to cut (u, v, w) the following must hold

I $d(u, v, p_s) \leq \beta_i \leq d(u, w, p_s)$ or $d(u, w, p_s) \leq \beta_i \leq d(v, w, p_s)$

II p_s settles (u, v, w) at level i

The above conditions create two independent events and their probabilities can be computed as follows: When the first event occurs, there exists a contribution of $3^{i+2}\beta = 27\beta_i$ to $d_T^{p_s}$. Consider a particular $x \in [d(u, v, p_s), d(u, w, p_s)]$. The probability that β_i falls in the range $[x, x + dx]$ is at most $\frac{1}{x \ln 3} dx$. Conditioned on $\beta_i = x$, any of the vertices up to and including p_s in the above sequence can settle (u, v, w) at level i. The first one among these in the permutation π will settle (u, v, w) and thus the conditioned probability is $\frac{1}{s}$. Thus,

$$\begin{aligned} P\Big[p_s \text{ settles } (u, v, w) \text{ at level } i | d(u, v, p_s) \leq \beta_i \leq d(u, w, p_s)\Big] \\ \leq \frac{27}{s} \int_{d(u,v,p_s)}^{d(u,w,p_s)} \frac{x}{x \ln 3} dx = \frac{27}{s \ln 3}(d(u, w, p_s) - d(u, v, p_s)) \end{aligned} \tag{3.11}$$

We can compute the probability of the second event similarly and, given their independence, sum them up to get

$$\begin{aligned} E\big[d_T^{p_s}(u, v, w)\big] \leq \frac{27}{s \ln 3}(d(u, w, p_s) - d(u, v, p_s)) \leq \frac{27}{s \ln 2} d(w, v, v) \\ \leq \frac{27}{s \ln 3} d(u, v, w), u \neq w \end{aligned} \tag{3.12}$$

where in the last two steps we used the fourth and the second axioms of the $\mathbb{G}_3$-metric, respectively. Note that the requirement $u \neq w$ does not imply that the result does not hold when the hyperedge has a cardinality of 2; in this case we can simply skip the last step. Using the linearity of the expectation and Eq. 3.8 we have

$$E\big[d_T(u, v, w)\big] = \frac{27 d(u, v, w)}{\ln 3} \sum_{k=1}^{n} \frac{1}{s} < \frac{27 d(u, v, w)}{\ln 3} \ln n = \mathcal{O}(\log n) \cdot d(u, v, w) \tag{3.13}$$

where we use the the fact that $\sum_{k=1}^{n} \frac{1}{s} < \ln n + 1$. □

We observe that we are only interested in the *order* of the approximation, thus multiplicative and additive constants are of no relevance, as already mentioned in Sec. 3.1, and no attempt was made to optimize them. This lemma concludes the proof of Th. 3.1.

3.4 Distortion Lower Bound

It was stated in Th. 2.1 and in the discussion therein that it is desired to obtain tight bounds on the distortion induced by metric embeddings, as this proves the optimality of the method. The previously presented algorithm is indeed optimal, as outlined by the following theorem.

Theorem 3.5. *The expected distortion induced by the algorithm $\mathcal{A}_C$ and the subsequent procedure is tightly bounded by $\Theta(\log n)$.*

Proof. Following the previous notation, assume that (X, d) is the original $\mathbb{G}_3$-metric space, that (X_T, d_T) is the host metric space, as defined above, and that the embedding is the identity function. Let $\mathcal{G} = (V, E, w)$ be an expander graph and for all $u, v \in V$ let $W_d(u, v)$ be the weighted distance, as given in Eq. 2.2. As noted in [Bar96], the embedding of the $\mathbb{G}_2$-metric space $(X_1, d_1) = (V, W_d)$ induced by $\mathcal{G}$ into a tree metric space $(X_T^{'}, d_T^{'})$ incurs a distortion of at least $\Omega(\log n)$, that is

$$d_T^{'}(u, v) \leq a \cdot d_1(u, v), \forall u, v \in X_1 \Rightarrow a = \Omega(\log n) \tag{3.14}$$

Now, let $X = X_1$ and for all $u, v, w \in X$ define

$$d(u, v, w) = d_1(u, v) + d_1(v, w) + d_1(w, u) \tag{3.15}$$

We can easily verify that the above function is indeed a $\mathbb{G}_3$-metric. Hence, by means of simple summation we extract the lower bound, which is $\Omega(\log n)$, and by taking into account the Th. 3.1, we obtain the desired $\Theta(\log n)$ bound, which concludes the proof. □

The above proof strengthens our result as it guarantees the existential optimality of our method; one need not search for a lower-distortion embedding of $\mathbb{G}_3$-metrics into metrics arising from the shortest path distance on trees (or equivalent to that metrics) as its existence is precluded by the lower bound. Nonetheless, the possibility of a different embedding, which achieves a lower distortion, into a metric non-equivalent to the above, cannot be eliminated.

3.5 *l*-HST and Tree Height

As mentioned in Sec. 3.2, the constructed host metric space is the leaves of a *3-HST*, which has a height of $\mathcal{O}(\log \Delta)$. However, the properties of the tree, namely the decay factor of the weights and the height, can be further manipulated, as outline in the following theorems.

Theorem 3.6. *A $\mathbb{G}_3$-metric space can be embedded into a l-HST, for arbitrary $l \geq 3$, inducing a distortion of $\mathcal{O}(\frac{l \log n}{\log l})$, tightly bounded on $\log n$.*

Proof. Since the proof follows that of Th. 3.1, only major modifications are outlined here and the details are left to the reader. Assume that $\Delta = l^{\delta}, \delta \in \mathbb{N}_{\geq 0}$ and let $\mathcal{A}'_C = \mathcal{A}_C$ be a new algorithm with the Line **4** of $\mathcal{A}_C$ replaced by $\beta_i \leftarrow l^{i-1}\beta$. Also, set the probability distribution function, from with β is sampled, to

$$f_b(x) = \begin{cases} \frac{1}{x \ln l} & x \in [1, l] \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

Following that, we construct the l-*HST* and the host space, similarly to Sec. 3.2 and prove the $l^i\beta$-upper bound of the cluster diameter and the dominance of the host metric, similarly to Lem. 3.2 and Th. 3.3, respectively. Next, we define the following function, for some $\epsilon > 0$

$$d_T^p(u, v, w) = \sum_i \mathbb{1}\Big(p \text{ cuts } (u, v, w) \text{ at level } i\Big) l^{i+\epsilon}\beta \tag{3.17}$$

which, following similar reasoning as in Th. 3.4, leads to

$$\sum_p d_T^p(u, v, w) = 2\sum_{i=1}^{L_2} l^{i+\epsilon}\beta + \sum_{i=1}^{L_1} l^{i+\epsilon}\beta \tag{3.18}$$

Using trivial calculations we conclude that the condition $\epsilon \geq \frac{\ln 2}{\ln l}$ is sufficient for the last equation to upper bound the host metric and by letting it assume its lowest possible value, a contribution of $\mathbb{O}(l)$ on $\sum_p d_T^p(\cdot)$ arises when the hyperedge gets cut. The rest of the proof follows as in Th. 3.4 and the tight bound on $\log n$ arises from Th. 3.5. □

Additionally, it is desirable to keep the height of the resulting tree as low as possible and when the diameter is *higher-than-polynomial* in the number of nodes, improvement is guaranteed by the following theorem.

Theorem 3.7. *The height of the resulting l-HST can be reduced to $\mathcal{O}(\log n)$ incurring a constant multiplicative distortion.*

Proof. We say that a node is *balanced* if none of its children contains more than half of the node's nodes. A tree of n leaves and balanced nodes has a height of $\mathcal{O}(\log n)$. In order to make the previous tree balanced, we simply need to recurse from the root and contract the edges of unbalanced nodes. Since the induced distortion is a multiplicative constant, none of the previous asymptotic bounds changes. Further details about the algorithm and the complete proof can be looked up in [BBMN15]. □

It should be evident that the requirement of a diameter *higher-than-polynomial* in n is necessitated by the fact that, if $\Delta = \mathcal{O}(n^m)$ for some constant $m > 0$, then the height of the tree is $\mathcal{O}(\log n)$ and the best-case improvement that the above method can achieve is a constant factor.

3.6 Extension to G_k-Metric

It was stated at the beginning of the chapter that our ultimate goad is to embed $\mathbb{G}_k$-metric spaces, for arbitrary integer k. The obtained result for the $\mathbb{G}_3$-metric space can be easily generalized as follows.

Theorem 3.8. *A $\mathbb{G}_k$-metric space can be embedded into a k-HST, for arbitrary $k \geq 3$, inducing a distortion of $\mathcal{O}(\frac{k^2 \log n}{\log k})$, tightly bounded on $\log n$.*

Proof. As expected, there exist several similarities between this and Th. 3.1, so only a brief scetch of the proof is presented here. Initially, let $\mathcal{A}_c^k = \mathcal{A}_c$ be a copy of the original algorithm, modified as follows: Given a cluster S in each iteration and for each p_s in the random permutation of the vertices, we create a sub-cluster containing all of the cluster's S hyperedges of cardinality $k-1$, whose distance from p_s is bounded by $k^{i-1}\beta$. Also, the probability distribution of β is modified as in Th. 3.6.

The diameter of each cluster in the resulting laminar family is bounded by $k^i\beta$. The family is then is mapped to a k-*HST* and the host metric is defined as the sum of all $\binom{k}{2}$ combinations of the respective hyperedge's leaves. As in Th. 3.3, using the separation levels $\{L_1, L_2 ... L_\delta\}, \delta = \log \Delta$, assumed to be in ascending order, the host metric can be expressed as

$$d_T(e) = \sum_{j=1}^{k-1} \sum_{i=1}^{L_j} 2j \cdot k^i \beta \tag{3.19}$$

where $e \in X^k$ is a hyperedge. Then, we define $d_T^p(e)$ similarly to Eq. 3.17 and using the separation levels we have

$$\sum_p d_T^p(e) = \sum_{j=1}^{k-1} \sum_{i=1}^{L_j} j \cdot k^{i+\epsilon} \beta \tag{3.20}$$

It is straightforward to see that the condition $\epsilon \geq \frac{\ln 2}{\ln k}$ suffices for the last equation to upper bound the host metric, which gives rise to a $\mathcal{O}(k)$ term, as in Th. 3.6. Next, we define the distance of a vertex p from a hyperedge e as

$$d_p(e) = \min_{2 \leq i \leq k-1} d(e_1, \cdots, e_{i-1}, p, e_{i+1}, \cdots, e_k) \tag{3.21}$$

Given that, we arrange every vertex in order of increasing distance from e and choose a random one from this ordering. In order for the chosen vertex to cut e, one of k independent events must occur (look at Th. 3.1), the probability of each one being $\mathcal{O}(\frac{k \log n}{\log k})$, which concludes the upper bound proof. The lower bound arises from Th. 3.5. □

Chapter 4

Distributed Embedding

In this chapter we present the distributed implementation of our embedding method, which draws upon the algorithm presented by Khan et.al in [KKM+12]. For the sake of simplicity and better understanding of the procedure, we restrict ourselves to the embedding of the $\mathbb{G}_3$-metric space, as before. We start by presenting a structure, called least vertex lists, which is used for representing the distributed embedding. Then, we develop an algorithm for calculating these lists, prove its correctness and analyze its performance. Following that, we use the calculated least vertex lists in order to define tree embedding and finally we give a distributed algorithm which reduces its height.

4.1 Least Vertex Lists

In the distributed setting, each node of the hypergraph has limited information about the topology and therefore needs to represent and construct the embedding using information that is propagated to it by its neighbors. This is achieved by the least vertex lists, which implicitly encode the embedding method of the previous chapter. Before we describe it, we need to define some properties of the distributed network.

Definition 4.1. Let $\mathcal{H} = (V, E, w_H)$ be a 3-regular hypergraph of n nodes, diameter Δ and weights greater than 1. Denote with (X, d) the $\mathbb{G}_3$-metric space induced in $\mathcal{H}$ by the shortest path distance defined in Sec. 2.2. Also, assume that the $\mathcal{CONGEST}(\log n)$ model holds and in order to be able to transmit a weight in a single round let $w = \mathcal{O}(n^m)$ for some constant $m > 0$. Each node picks a unique $\mathcal{O}(\log n)-$bit *identifier (ID)* uniformly at random. Let $v \prec w$ denote the fact that that the *ID* of node v is less than the *ID* of w.. ■

Having outlined the distributed setting, we proceed to the definition of the least vertex lists, originally appeared in [Coh97, CK07] .

Definition 4.2. Let ρ be a non-negative number. Given two *adjacent* nodes $u, v \in V$ we say that $w \in V$ is the *least vertex* $\mathcal{L}_\rho(u, v)$ of u, v within a distance ρ if the

following holds

$$\mathcal{L}_\rho(u,v) = w \Leftrightarrow \nexists z : z \prec w \wedge d(u,w,v) \leq \rho \tag{4.1}$$

For all $\rho \in [0, \Delta]$ define

$$\mathcal{L}(u,v) = \Big\{(w,\rho) : d(u,v,w) = \rho \wedge w \in \mathcal{L}_\rho(u,v)\Big\} \tag{4.2}$$

as the *least vertex list* of (u,v). ■

We ought to note that not all possible values of ρ appear in $\mathcal{L}(u,v)$, as duplicate entries $\big((w,\rho) \in \mathcal{L}(u,v) \wedge (w,\rho-\epsilon) \in \mathcal{L}(u,v), \epsilon > 0\big)$ can be deleted. We also notice that there exists an intrinsic redundancy in the definition of the structure, as $\mathcal{L}_{\rho\neq 0}(u,v) = \mathcal{L}_{\rho\neq 0}(v,u)$, which, owing to the fact that each node u needs to compute $\mathcal{L}(u,v)$ for all of its neighbors $v \in N(u)$, may place additional stress to the network. However, as it will become clear, this does not affect the message and round complexity. In the following, we use the notation $\mathcal{L}_u(v) = \mathcal{L}(u,v)$ in order to explicitly clarify that each $u \in V$ stores a least vertex list for each of its neighbors $v \in N(u)$.

4.2 Distributed Algorithm

Drawing upon a similar distributed tree embedding method presented by Khan in [KKM+08], we present an algorithm, named $\mathcal{A}_D$, for computing the previously defined least vertex lists. At the termination of the algorithm, each node $u \in V$ has one list $\mathcal{L}_u(v)$ for each of its neighbors $v \in N(u)$. The algorithm, formally given in Algorithm 2, can be described as follows: Each node $u \in V$ creates one list $L_u^0(v)$ for each of its neighbors, initializes it and sends all of them to its neighbors. Then, using the lists received from neighbors, u updates its lists by adding all the elements of the former. Finally, the resulting lists are scanned is ascending order of distance and entries, whose *ID* is not less than all the previous *ID* encountered, are removed. If successive iterations produce no alterations in any of the lists of the network, the algorithm is terminated and the resulting lists are the desired least vertex lists. Its correctness is guaranteed by the following theorem.

Theorem 4.1. *The $\mathcal{A}_D$ algorithm returns the correct least vertex lists.*

Proof. We prove this theorem using an expansion technique and induction. Fix two $u, v \in V$ and let $\mathcal{H}_{uv}^i = \big(V_{uv}^i, E_{uv}^i, w_{uv}^i\big)$ be the sub-hypergraph induced from $\mathcal{H}$ by deleting all vertices of hop distance $h_d(u,v,\cdot)$ strictly higher than i, $N_{uv}^i(w)$ be the neighbors of a node $w \in \mathcal{H}_{uv}^i$ and $\mathcal{L}_u^i(v)$ be the least vertex lists on $\mathcal{H}_{uv}^i$. We shall prove that $\mathcal{L}_u^i(v) = L_u^i(v), i \in \mathbb{N}_{\geq 0}$, where $L_u^i(v)$ are the lists calculated at each iteration of $\mathcal{A}_D$ when $\mathcal{H}$ is given as input. This holds trivially true for $i = 0$. Assume that it is also true for arbitrary $i > 0$. Then, taking into account the induction hypothesis, the new lists created by $\mathcal{A}_D$ at *phase* $i+1$, *after* the **delete** step is completed, are

$$L_u^{i+1}(v) = \mathcal{L}_u^i(v) \cup \big\{(z, d + d(u,v,w))\big\} \tag{4.3}$$

Algorithm 2 The $\mathcal{A}_D$ Least Vertex Lists Algorithm

procedure ADALGORITHM(V, E, w_H)
  Initialize:
    $i \leftarrow 0$
    set $L_u^0(v) = \{(u, 0)\}$ for all $u \in V$ and $(v, \cdot) \in N(v)$
  All $u \in V$ **do** in parallel
    send $\bigcup_{v \in N(u)} L_u^i(v)$ to $w \in N(u)$
    set $L_u^{i+1}(v) \leftarrow L_u^i(v)$ for all v
    For all $v \in N(u), w \in N(v)$ and $(z, d) \in L_v^i(w)$
      set $L_u^{i+1}(v) \leftarrow L_u^{i+1}(v) \cup \{(z, d + d(u, v, w))\}$
    sort L^{i+1} in ascending distance
    delete entries with *ID* greater or equal to a previous *ID*
  $i \leftarrow i + 1$
  while $\exists u, v \in V$ such that $L_u^i(v) \neq L_u^{i-1}(v)$
  return $L_u^i(v)$ for all $u \in V$ and $v \in N(u)$
end procedure

for all $v \in N_{uv}^i(u), w \in N_{uv}^i(v), (z, d) \in \mathcal{I}_v^i(w)$, where $\mathcal{I}^i(v, w) \subseteq \mathcal{L}_(^i w)$ are the set of entries that survived the **delete** step. Assume that $L_u^{i+1}(v) \neq \mathcal{L}_u^{i+1}(v)$, which implies that

$$\exists t \in V_{uv}^{i+1} \wedge \rho_1, \rho_2 > 0 : (t, \rho_1) \in \mathcal{L}_u^{i+1}(v) \wedge (t, \rho_2) \notin L_u^{i+1}(v)$$

Given the previous equation and the induction hypothesis, the third statement is essentially equivalent to $(t, \rho_1) \notin \mathcal{I}^i(u, v)$ (if it belonged to $\mathcal{L}_u^i(v) \setminus \mathcal{I}_u^i(v)$ it would not have been deleted). Hence, the above imply the following, respectively

$$\begin{gathered} \nexists s \in V_{uv}^{i+1} : s \prec t \wedge d(u, v, s) \leq d(u, v, t) \\ \wedge \\ \exists p \in V_{uv}^{i+1} : p \prec t \wedge d(v, w, p) \leq d(v, w, t) \end{gathered}$$

Simply by setting $s = p$, we see that these two statements are contradictory. Hence $L_u^{i+1}(v) = \mathcal{L}_u^{i+1}(v)$, which completes the induction step. □

Having proved the correctness of the algorithm, we analyze in the following its performance using the round and message complexity.

Theorem 4.2. *The $\mathcal{A}_D$ algorithm terminates w.h.p*[1] *after $\mathcal{O}(SPD_3 \log n)$ rounds of the $\mathcal{CONGEST}(\log n)$ model, having sent $\mathcal{O}(SPD_3|E| \log n)$ messages, where SPD_3 is the 3-shortest path diameter.*

Proof. Using the notations of the previous proof, we fix $u, v \in V$, $i \in \mathbb{N}_{\geq 0}$ and order the N nodes in $\mathcal{H}_{uv}^i$ in ascending distance from u, v. Given that the *ID*s are chosen

[1] *With high probability (w.h.p)* implies that there exists $c > 0$ such that an event occurs with probability at least $1 - 1/n^c$.

at random, the j-th node in the above ordering has an independent from the other nodes probability of $1/j$ of being included in $L_u^i(v)$, which, as in Th. 3.4, implies that $E\big[|L_u^i(v)|\big] = \mathcal{O}(\log N)$. Applying the Chernoff's bound we conclude that for all $u, v \in V, i \in \mathbb{N}_{\geq 0}$

$$|L_u^i(v)| = \mathcal{O}(\log n), w.h.p \tag{4.4}$$

Thus, given that the distributed model allows the transmission of $\mathcal{O}(\log n)$ bits per round, we infer that, for each phase of $\mathcal{A}_D$ (each different i), $\mathcal{O}(\log n)$ rounds are required. In order to find the number of phases required, let P_{uvw} be the shortest path between vertices u, v, w and assume that for some $z \in N(u) \cap N(v) \cap P_{uvw}$ it holds true that $(w, d(u, v, w)) \in \mathcal{L}_u(v) \wedge (w, d(u, z, w)) \in L_u^i(z)$. Then, $(w, d(u, v, w)) \in L_u^{i+1}(v)$. Thus, in each successive phase (one hop), the correct information about the least vertex of u, v spreads on the shortest path. This implies that $\mathcal{O}(SPD_3)$ phases are required to reach the whole network, which, combined with the previous result, gives the $\mathcal{O}(SPD_3 \log n)$ round complexity. Finally, since all edges transmit messages in each round, the message complexity follows. □

We ought to note that the termination condition of the algorithm is only locally known, which necessitates the existence of global approach. Even though sophisticated methods can be developed, it is sufficient to restrict to a primitive algorithm: While a vertex detects alterations in its least vertex lists, it floods the leader (see Th. 4.3 for leader election) with an **alter** signal. If the leader receives no such signal, it waits $\mathcal{O}(h_D)$ rounds, where h_D is the hop diameter, and then floods the network with a **termination** signal. In order to make sure that this flooding produces no infinite cycles, an $\mathcal{O}(h_D)$ counter is attached to each message and reduced by one each time the message is re-transmitted by a node. This requires $\mathcal{O}(h_D)$ additional rounds and $\mathcal{O}(|E|h_D)$ additional messages. The fact that none of them affects the asymptotic complexities given above, renders an intricate algorithm needless.

4.3 Tree Embedding

One can easily understand the usefulness of the least vertex lists simply by assuming that the randomly chosen *ID*s represent the random permutation π defined in the previous chapter and looking at the proof of Th. 3.4: they represent the of i-th level the cluster, in which node u is included. Formally, the embedding procedure is described and proved in the following.

Theorem 4.3. *Let the hypergraph $\mathcal{H} = (V, E, w_H)$ be given as input to $\mathcal{A}_D$. The resulting least vertex lists $\mathcal{L}(u, v), u \in V, v \in N(u)$ define the embedding produced by the $\mathcal{A}_C$ algorithm.*

Proof. Assume that a different instance of $\mathcal{A}_D$ is executed on the modified hypergraph $\mathcal{H}' = (V, E, w_H = 1)$. Then, for the resulting lists, it holds true that $(v_L, \Delta) \in \mathcal{L}'_u(v), \forall u \in V, \wedge \forall v \in N(u)$, where v_L is the node of lowest *ID*. Thus, since each node knows the lowest *ID* node, v_L is elected as the *leader*, picks a number β from the distribution in Eq. 3.2 and sends it back to the network. These

messages are exchanged using the flooding method described at the end of the previous section. It can be easily seen that none of these procedures increases the asymptotic complexities of Th. 4.2. When the nodes receive β, they calculate $\beta_i = 3^{i-1}\beta, i = 1, 2 \dots \delta = \log_3 \Delta$ and then construct their β-*lists* as

$$\beta_u(v) = \left\{(c_i, \beta_i) : (c_i, \beta_i) \in \mathcal{L}_u(v), i = 1, 2 \dots \delta = \log_3 \Delta\right\} \tag{4.5}$$

for all $u \in V, v \in N(u)$. These lists imply that c_i is the lowest *ID* node within distance β_i of (u, v). Thus, c_i creates a i-level cluster, in which both u, v are included. From a node's perspective, each u could have been included in any i-level cluster created by nodes that belong in any of the $\beta_u(\cdot)$-lists, since for all $v \in N(u), (c_i, \beta_i) \in \beta_u(v)$ it holds $d(u, v, c_i) \leq \beta_i$. However, the node $c_u^i \in \beta_u(\cdot)$ with lowest *ID* (first in permutation π) is the one that creates the i-level cluster, in which u is included. Thus, the set $C_u = \left\{(c_u^i, 3^i\beta) : i = 1, 2 \dots \delta = \log_3 \Delta\right\}$ defines the *ancestor lists* of u and the edge heights of the desired tree $\mathcal{T}$ □

We observe that each node $u \in V$ can only reconstruct the root-to-leaf path ending on itself, and it is unaware of edges not contained in this path. Thus, no explicit representation of the $\mathcal{T}$ exists; the tree structure is encoded distributively by keeping track of the ancestor of each leaf at each level. Nonetheless, this poses no restriction in determining the host metric.

4.4 Distributed Height Reduction

In defining the distributed setting (Def. 4.1), in was necessary to assume that the weights are polynomial in n, due to the transmission capacity being limited by the $\mathcal{CONGEST}(\log n)$ to $\mathcal{O}(\log n)$ bits per round. This appears to be quite restrictive as it precludes the application of the developed method on hypergraphs with weights *higher-than-polynomial* in n. Dropping this assumption (thus increasing the transmission capacity) results in a tree height that is non-optimal and can be further reduced, as mentioned in Th. 3.7 and the surrounding discussion.

In the following we present the implementation of the tree height reduction algorithm for the distributed representation of the tree given by the method in the previous section. The algorithm $\mathcal{A}_T$, as presented in Algorithm 3, implements distributively the method briefly presented in Th. 3.7 and can be described as follows: Node u initializes the variable γ_u to 1 and sends it along with its $i+1$-level ancestor c_u^{i+1} to its i-level ancestor c_u^i. Then, c_u^i creates the set V of all the nodes, from which it received a message, calculates the sum of all the received γ's and asks a random node in V to increase its γ by one. Following that, it sends the calculated sum to c_u^{i+1}. The last node, in turn, sums up all the values it received and verifies whether a value grater than the half of the computed sum was sent by any node. If that holds true, the tree edge connecting the latter node to c_u^{i+1} needs to be contracted.

It should be obvious that the above algorithm implements the one described in Th. 3.7, because, essentially, each node of the tree, as it is encoded in the distributed representation, balances the sub-trees rooted at its children. However, strictly speaking, $\mathcal{A}_T$, as described, cannot be directly implemented in a distributed system, as it

Algorithm 3 The $\mathcal{A}_T$ Tree Height Reduction Algorithm

procedure ATALGORITHM($\mathcal{H}, C_u$)
  Initialize:
    set $\gamma_u = 1$ for all $u \in V$
  for $i = 1, 2 \cdots \delta = \log \Delta$ **do** in parallel
    u **sends** (γ_u, c_u^{i+1}) to c_u^i
    c_u^i **computes** $s(c_u^i) = 1 + \sum_{v \in V} \gamma_v$, V = nodes that sent $\gamma(\cdot)$ to c_u^i
    c_u^i **sends** a control message to a random node in $_r \in V$
    v_r **sets** $\gamma_{v_r} \leftarrow \gamma_{v_r} + 1$
    c_u^i **sends** $s(c_u^i)$ to c_u^{i+1}
    c_u^{i+1} **computes** $p = \sum_{w \in W} s(w)$, W = nodes that sent $s(\cdot)$ to c_u^{i+1}
    if there exists $w \in W$ such that $2s(w) > p$ **then**
      replace $(w, \beta_i) \in C_u$ with (c_u^{i+1}, β_{i+1})
    end if
  end for
  return *new* $C_u, u \in V$
end procedure

is not explicitly expressed with respect to a single node. It is presented in this way in order to make it intuitive to understand. Nonetheless, it is in fact distributed as it only requires knowledge of the neighbors and the ancestor lists. The modifications required for an real-life implementation are straightforward. The performance is analyzed in the following.

Theorem 4.4. *Assume that a tree $\mathcal{T}$ is represented by the ancestor lists $C_u, u \in V$, as given by the $\mathcal{A}_D$ algorithm. Then, $\mathcal{A}_T$ returns new ancestor lists C'_u, which represent a tree of $\mathcal{O}(\log n)$ height, without asymptotically increasing the induced distortion of the embedding. It requires $\mathcal{O}(h_D \log \Delta)$ rounds and sends $\mathcal{O}(|E| h_D \log \Delta)$ messages.*

Proof. The proof that the tree balancing gives a tree of height $\mathcal{O}(\log n)$ without increasing the distortion can be looked up in [BBMN15]. The algorithm requires $\mathcal{O}(\log \Delta)$ phases in order to verify that all of the levels of the initial tree representation are balanced. In each phase, the nodes of the graph exchange $\mathcal{O}(1)$ messages using the flooding method described at the end of Sec. 4.2, which requires $\mathcal{O}(h_D)$ rounds. Thus, we have an overall round complexity of $\mathcal{O}(h_D \log \Delta)$ and given that in general each edge transmits a message in each round, the message complexity is $\mathcal{O}(|E| h_D \log \Delta)$. □

In the above, only the bound on $\log \Delta$ is tight, as one can easily construct a distributed representation of a tree, which requires only $\mathcal{O}(1)$ rounds per phase (e.g when the graph is itself a tree). Finally, we should note that this algorithm is of wider interest, not necessarily tailored to embeddings, as it can be applied to any distributed tree representation given by its ancestor lists.

Chapter 5

Concluding Remarks

5.1 Summary of Results

This thesis dealt with the problem of approximating generalized distances between multiple points that arise on hypergraphs with distributions over tree metrics. After presenting the underlying concepts and definitions, we developed a probabilistic algorithm, named $\mathcal{A}_C^k$, which produces a hierarchical decomposition of the hypergraph to a laminar family of clusters. Then, we mapped those clusters into a hierarchical well separated tree and defined the tree metric. Consecutively, we provided the upper bound of the induced distortion and proved the existential optimality of the method with respect to the number of points. Following that, we presented, for the special case $k = 3$, a representation of our embedding method via a structure termed least vertex lists and provided a distributed algorithm, $\mathcal{A}_D$, which computes those lists. We proved the correctness of this algorithm and analyzed its performance using the message and round complexity. We also presented a distributed representation of the resulting tree via its ancestor lists and developed an algorithm, named $\mathcal{A}_T$, which reduces the height of this implicitly defined tree. We concluded by analyzing the number of messages and communication rounds that $\mathcal{A}_T$ requires.

From a practical point of view, the contribution of the thesis can be summarized in the tree developed methods: the clustering algorithm, $\mathcal{A}_C^k$, its distributed implementation, $\mathcal{A}_D$ and the distributed tree height reduction algorithm, $\mathcal{A}_T$. In the realms of theoretical interest lie the approximation, correctness and performance analyses presented and the optimality guarantees given. The novelty and originality stem from the fact that, to the best of our knowledge, no other centralized (let alone distributed) embedding method for higher-order metrics has ever been proposed. This is greatly amplified by the fact that the purposed height reduction algorithm is a novel attempt to apply that concept to distributed representations of trees. However, one limitation of our methods stems from the fact that we confined ourselves to the 3-order case when distributively implementing the embedding, which precludes us from understating the effect of the order k on the performance of the respective algorithms.

In the following we present the main conclusions of this thesis. Broadly speaking, this is a restatement of Th. 3.8, 4.2, 4.4. However, the definitions, the algorithms

and the respective proofs, upon which they are based, are spread in all three previous chapters. They are simply given here for the ease of reference and in order to grasp the essential contributions without having to plough through technicalities.

Conclusion 1. *Let $\mathcal{H}_k = (X, E, w)$ be a positively-weighted, undirected, k-regular hypergraph of n points and (X, d) be the $\mathbb{G}_k$-metric space that arises by the k-point shortest path distance. Then, using the algorithm $\mathcal{A}_C^k$, the metric space (X, d) can be embedded into a k-HST-metric space (X, d_T) with a distortion $\mathcal{O}(\frac{k^2 \log n}{\log k})$, tightly bounded on $\log n$.*

Conclusion 2. *Let $\mathcal{H}_3 = (X, E, w)$ be a positively-weighted, undirected, 3-regular hypergraph of n points and (X, d) be the $\mathbb{G}_3$-metric space that arises by the 3-point shortest path distance. Then, the algorithm $\mathcal{A}_D$ implements distributively the embedding produced by $\mathcal{A}_C^3$ in $\mathcal{O}(SPD_3 \log n)$ rounds and $\mathcal{O}(SPD_3|E| \log n)$ messages of the $\mathcal{CONGEST}(\log n)$ model, where SPD_3 is the shortest path diameter.*

Conclusion 3. *Let the tree $\mathcal{T}$ be represented by the ancestor lists, as given by $\mathcal{A}_D$. Then, the algorithm $\mathcal{A}_T$ returns new ancestor lists, which represent a tree of $\mathcal{O}(\log n)$ height, without asymptotically increasing the induced distortion of the embedding. It requires $\mathcal{O}(h_D \log \Delta)$ rounds and sends $\mathcal{O}(|E| h_D \log \Delta)$ messages, where Δ and h_D are the diameter and the hop diameter, respectively.*

5.2 Applications and Open Problems

Hypergraphs are an intricate structure and, with graphs being a special case of them, it is reasonably expected that whenever problems defined on the latter are extended to the former they become harder to solve. In fact, their time complexity may increase, such as the minimum cut problem [CX16], they may become computationally intractable (NP-hard), such as the minimum spanning tree [AF95], or they may even become hard to approximate, such as the set packing [HSS06]. It is therefore prudent to claim that hypergraph problems are in general harder and that our methods emancipates us from the burden of their intractability. In fact, as argued in [FRT04], graph embeddings into trees lead to substantial improvement on the approximation ratio of several underlying problems. Given that our method essentially embeds hypergraphs into similar metrics, we can expect that its benefit is higher. Any problem on a (metric inducing) hypergraph and be approximated with a factor $\mathcal{O}(\frac{k^2 \log n}{\log k})$, the only difference with [FRT04] being the term $\frac{k^2}{\log k}$, which can be considered acceptable given that we are dealing with k-th order metrics.

This thesis paves the way for further research in the field of hypergraphs and their embeddings. Since our focus was on the $\mathbb{G}$-metric, one topic that deserves further attention is the $\mathbb{2}$-metric [Gäh63, Gäh64] and the $\mathbb{D}$-metric [Dha92]. It would be worth to investigate whether there exists any merit in embedding and approximating these metrics. One possible approach for alleviating their oddities would be to refine the axiom systems. Additionally, when our method was extended to embed the $\mathbb{G}_3$-metric into an *l-HST* rather that a 3-*HST*, only the optimality with

respect to the number of points was guaranteed, thus it is also important to provide a lower bound on l. An expander graph similar to the one used in [Bar96] would be the first line of attack and a tight bound on l would also provide an optimality guarantee for the 2-*HST* to l-*HST* conversion method presented in [BCR01]. Naturally, the same observation holds for the $\mathbb{G}_k$-metric, where a lower bound on k would also be interesting. Also, since we refrained from non-probabilistic methods, following the paradigm of [CCG+98, CCGG98], where the probabilistic embedding is viewed as linear optimization problem, our method could be derandomized and used for obtaining deterministic approximation algorithms. The time complexity of our algorithm is prone to further improvements, as it was demonstrated in [BGS16], where the authors, using an approximate shortest path algorithm, purposed a time-efficient algorithm to construct probabilistic tree embeddings, for the method, upon which ours draws. This also suggests an additional research direction, as in real-world applications it is necessary to drop the assumption of *a priori* knowledge of all k-point distances. Thus, shortest path algorithms on hypergraphs, such as the one presented in [GZR+12], should be tailored to our method and ideally used for reducing the time-complexity. The performance of the distributed implementation may also be improved and ideally made similar to the near-optical presented in [GL14], where the least vertex list were initially computed for a small skeleton subgraph and then propagated to the rest of network. Additionally, since we confined ourselves to the distributed embedding of the $\mathbb{G}_3$-metric, it would be interesting to extend it to the $\mathbb{G}_k$-metric in order to unmask the effect of the k-point distance on the message and round complexity. A parallel implementation, drawing upon the archetype of [BGT12] and its subsequent improvement in [FL15], would also be of practical interest, as it could provide *RNC* approximation algorithm for problems defined on hypergraphs.

Bibliography

[ACKP09] Stavros Athanassopoulos, Ioannis Caragiannis, Christos Kaklamanis, and Evi Papaioannou. *Energy-Efficient Communication in Multi-interface Wireless Networks*, pages 102–111. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[AF95] Lars Dovling Andersen and Herbert Fleischner. The np-completeness of finding a-trails in eulerian graphs and of finding spanning trees in hypergraphs. *Discrete Applied Mathematics*, 59(3):203 – 214, 1995.

[AKPW95] Noga Alon, Richard M. Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1):78–100, February 1995.

[ALZM+05] Sammer Agarwal, Jongwoo Lim, Lihi Zelnik-Manor, Pietro Perona, David Kriegman, and Serge Belongie. Beyond pairwise clustering. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 838–845 vol. 2, June 2005.

[AY05] Noga Alon and Raphael Yuster. On a hypergraph matching problem. *Graphs and Combinatorics*, 21(4):377–384, 2005.

[Bar96] Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *In 37th Annual Symposium on Foundations of Computer Science*, pages 184–193, 1996.

[Bar98] Yair Bartal. On approximating arbitrary metrices by tree metrics. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 161–168, New York, NY, USA, 1998. ACM.

[BBMN15] Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph (Seffi) Naor. A polylogarithmic-competitive algorithm for the k-server problem. *J. ACM*, 62(5):40:1–40:49, November 2015.

[BCR01] Yair Bartal, Moses Charikar, and Danny Raz. Approximating min-sum k-clustering in metric spaces. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, STOC '01, pages 11–20, New York, NY, USA, 2001. ACM.

[BGS16] Guy E. Blelloch, Yan Gu, and Yihan Sun. A new efficient construction on probabilistic tree embeddings. *CoRR*, abs/1605.04651, 2016.

[BGT12] Guy E. Blelloch, Anupam Gupta, and Kanat Tangwongsan. Parallel probabilistic tree embeddings, k-median, and buy-at-bulk network design. In *Proceedings of the Twenty-fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '12, pages 205–213, New York, NY, USA, 2012. ACM.

[Bou85] Jean Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1):46–52, 1985.

[BTC+10] Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. Music recommendation by unified hypergraph: Combining social media information and music content. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pages 391–400, New York, NY, USA, 2010. ACM.

[BZ15] Marcus Brazil and Martin Zachariasen. *Steiner Trees in Graphs and Hypergraphs*, pages 301–317. Springer International Publishing, Cham, 2015.

[CCG+98] Moses Charikar, Chandra Chekuri, Ashish Goel, Sudipto Guha, and Serge Plotkin. Approximating a finite metric by a small number of tree metrics. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280)*, pages 379–388, Nov 1998.

[CCGG98] Moses Charikar, Chandra Chekuri, Ashish Goel, and Sudipto Guha. Rounding via trees: Deterministic approximation algorithms for group steiner trees and k-median. *Unknown Journal*, pages 114–123, 1998.

[CK07] Edith Cohen and Haim Kaplan. Spatially-decaying aggregation over a network. *Journal of Computer and System Sciences*, 73(3):265 – 288, 2007.

[Coh97] Edith Cohen. Size-estimation framework with applications to transitive closure and reachability. *Journal of Computer and System Sciences*, 55(3):441 – 453, 1997.

[CX16] Chandra Chekuri and Chao Xu. Computing minimum cuts in hypergraphs. *CoRR*, abs/1607.08682, 2016.

[Dha92] B.C. Dhage. Generalised metric spaces and mappings with fixed point. *Bulletin of Cal. Math. Soc.*, 84(4):329–336, 1992.

[EN14] Alina Ene and Huy L. Nguyen. From graph to hypergraph multiway partition : is the single threshold the only route? In *Algorithms - ESA 2014 : 22th Annual European Symposium, Wroclaw, Poland, September*

8-10, 2014 : proceedings, number Number 8737 in Lecture notes in computer science, pages 382–393. Springer, Berlin, 2014.

[Fei00] Uriel Feige. Approximating the bandwidth via volume respecting embeddings. *Journal of Computer and System Sciences*, 60(3):510 – 539, 2000.

[FL15] Stephan Friedrichs and Christoph Lenzen. Parallel metric tree embedding based on an algebraic view on moore-bellman-ford. *CoRR*, abs/1509.09047, 2015.

[FRT04] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485 – 497, 2004. Special Issue on {STOC} 2003.

[Gäh63] Siegfried Gähler. 2-metrische räume und ihre topologische struktur. *Mathematische Nachrichten*, 26(1-4):115–148, 1963.

[Gäh64] Siegfried Gähler. Lineare 2-normierte räume. *Mathematische Nachrichten*, 28(1-2):1–43, 1964.

[GL14] Mohsen Ghaffari and Christoph Lenzen. *Near-Optimal Distributed Tree Embedding*, pages 197–211. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[GZR+12] Jianhang Gao, Qing Zhao, Wei Ren, Ananthram Swami, Ram Ramanathan, and Amotz Bar-Noy. Dynamic shortest path algorithms for hypergraphs. In *2012 10th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, pages 238–245, May 2012.

[HSS06] Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating k-set packing. *Comput. Complex.*, 15(1):20–39, May 2006.

[IM04] Piotr Indyk and Jiri Matousek. Low-distortion embeddings of finite metric spaces. In *in Handbook of Discrete and Computational Geometry*, pages 177–196. CRC Press, 2004.

[KKM+08] Maleq Khan, Fabian Kuhn, Dahlia Malkhi, Gopal Pandurangan, and Kunal Talwar. Efficient distributed approximation algorithms via probabilistic tree embeddings. In *Proceedings of the Twenty-seventh ACM Symposium on Principles of Distributed Computing*, PODC '08, pages 263–272, New York, NY, USA, 2008. ACM.

[KKM+12] Maleq Khan, Fabian Kuhn, Dahlia Malkhi, Gopal Pandurangan, and Kunal Talwar. Efficient distributed approximation algorithms via probabilistic tree embeddings. *Distributed Computing*, 25(3):189–205, 2012.

[KRS01] Goran Konjevod, R. Ravi, and F.Sibel Salman. On approximating planar metrics by tree metrics. *Information Processing Letters*, 80(4):213 – 219, 2001.

[KRTV13] Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. *An Optimal Online Algorithm for Weighted Bipartite Matching and Extensions to Combinatorial Auctions*, pages 589–600. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[Lin90] C.-S. Lin. On strictly convex and strictly 2-convex 2-normed spaces. *Mathematische Nachrichten*, 149(1):149–154, 1990.

[LLR95] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[LSC+09] Yu-Ru Lin, Jimeng Sun, Paul Castro, Ravi Konuru, Hari Sundaram, and Aisling Kelliher. Metafac: Community discovery via relational hypergraph factorization. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 527–536, New York, NY, USA, 2009. ACM.

[LXLS13] Dong Li, Zhiming Xu, Sheng Li, and Xin Sun. Link prediction in social networks based on hypergraph. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13 Companion, pages 41–42, New York, NY, USA, 2013. ACM.

[Mat96] Jiří Matoušek. On the distortion required for embedding finite metric spaces into normed spaces. *Israel Journal of Mathematics*, 93(1):333–344, 1996.

[Mat02] Jiří Matoušek. *Embedding Finite Metric Spaces into Normed Spaces*, pages 355–400. Springer New York, New York, NY, 2002.

[Men28] Karl. Menger. Untersuchungen über allgemeine metrik. *Mathematische Annalen*, 100:75–163, 1928.

[MS04] Zead Mustafa and Brailey Sims. Some remarks concerning d-metric spaces. 2004.

[MS06] Zead Mustafa and Brailey Sims. A new approach to generalized metric spaces. *Journal of Nonlinear and convex Analysis*, 7(2):289–297, 2006.

[Pel00] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000.

[PM03] Joachim Pistorius and Michel Minoux. An improved direct labeling method for the max-flow min-cut computation in large hypergraphs and applications. *International Transactions in Operational Research*, 10(1):1–11, 2003.

[RR98] Yuri Rabinovich and Ran Raz. Lower bounds on the distortion of embedding finite metric spaces in graphs. *Discrete & Computational Geometry*, 19(1):79–94, 1998.

[SPH07] Bernhard Schölkopf, John Platt, and Thomas Hofmann. *Learning with Hypergraphs: Clustering, Classification, and Embedding*, pages 1601–1608. MIT Press, 2007.