Massively Parallel Algorithms and Hardness for Single-Linkage Clustering under ℓ_p Distances

Grigory Yaroslavtsev [1] **Adithya Vadapalli** [1]

Abstract

We present first massively parallel (MPC) algorithms and hardness of approximation results for computing Single-Linkage Clustering of n input d-dimensional vectors under Hamming, ℓ_1, ℓ_2 and ℓ_∞ distances. All our algorithms run in $O(\log n)$ rounds of MPC for any fixed d and achieve $(1+\epsilon)$-approximation for all distances (except Hamming for which we show an exact algorithm). We also show constant-factor inapproximability results for $o(\log n)$-round algorithms under standard MPC hardness assumptions (for sufficiently large dimension depending on the distance used). Efficiency of implementation of our algorithms in Apache Spark is demonstrated through experiments on the largest available vector datasets from the UCI machine learning repository exhibiting speedups of several orders of magnitude.

1. Introduction

1.1. Single-linkage clustering

Single-Linkage Clustering is one of the oldest methods for clustering multi-dimensional vectors based on the nearest-neighbor rule and has been studied since 1951, see e.g. (Zahn, 1971). It can be used for hierarchical clustering and is one of the cornerstone techniques in data mining (see e.g. Chapter 17 of a classic text on information retrieval by Manning, Raghavan and Schütze (Manning et al., 2008)). Applications of Single-Linkage Clustering include reconstruction of semantic relationships from word embeddings such as Word2Vec (Malak & East, 2016), phylogenetic tree reconstruction (Gower & Ross, 1969), etc.

[1]Department of Computer Science, Indiana University, Bloomington, Indiana, United States. Correspondence to: Grigory Yaroslavtsev <grigory@grigory.us>, Adithya Vadapalli <avadapal@iu.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

We consider the problem of constructing a Single-Linkage Clustering for large-scale data. Given a dataset consisting of n real-valued d-dimensional vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ the goal of Single-Linkage Clustering is to construct a partition of these vectors into k clusters $C_1, \ldots, C_k$ such that the smallest distance between two vectors in different clusters is maximized. Formally, for $i \neq j$ let the single-linkage distance between two clusters C_i and C_j under ℓ_p distance be $d_p(C_i, C_j) = \min_{v_a \in C_i, v_b \in C_j} \|v_a - v_b\|_p$ where $\|x\|_p = (\sum_i |x_i|^p)^{1/p}$ is the standard p-norm. Then in the k-Single-Linkage Clustering (k-SLC) problem under ℓ_p distance we aim to find a partition into k clusters that maximizes $\min_{i \neq j} d_p(C_i, C_j)$. It is well-known that k-SLC can be constructed from the Minimum Spanning Tree (MST) of the underlying metric by taking as clusters connected components resulting from removal of $k-1$ longest MST edges (see Figure 1 for an example).

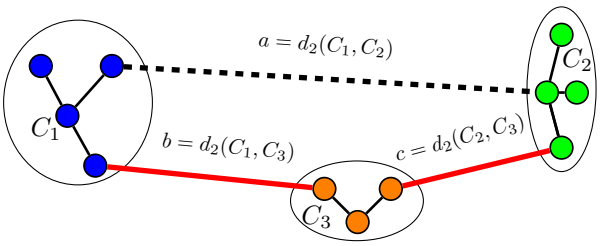

Figure 1. 3-SLC objective is $\min(a, b, c)$, MST shown in solid.

Note that with this approach once the MST is constructed it can be used to compute k-SLC for any value of k. Furthermore, it induces a hierarchical clustering structure that is often desirable in practice. According to Manning, Raghavan and Schütze (Manning et al., 2008) the main impediment to this approach in practice that motivates the use of various heuristics is that for large-scale data no practically feasible techniques are currently known for constructing an exact MST. Our work overcomes this challenge by leveraging two observations: 1) inexact but close to optimum solutions can suffice in practice due to the fact that real-valued data always contains rounding errors, 2) while exact MST algorithms are very sequential, approximate solutions can be computed in parallel on a distributed cluster.

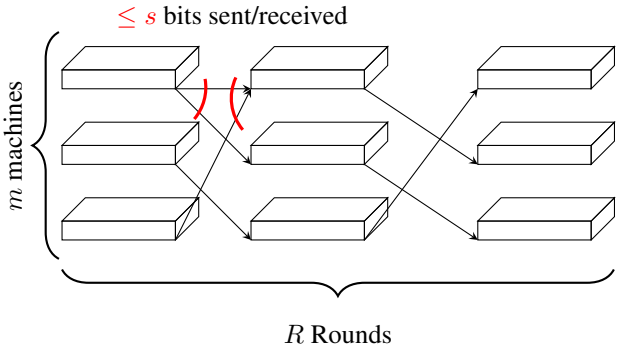

Figure 2. MPC model of computation

1.2. Massively parallel computation

We present analysis of performance of our algorithms in the Massively Parallel Computation model (MPC) which is the most commonly used theoretical model of computation on synchronous large-scale data processing platforms such as MapReduce and Spark. As we demonstrate through experiments in Spark this model accurately reflects performance of our algorithms on real data. MPC model has attracted a lot of interest recently. It has emerged through a sequence of papers (Feldman et al., 2008; Karloff et al., 2010; Goodrich et al., 2011; Beame et al., 2013; Andoni et al., 2014) and has been analyzed extensively (Fish et al., 2015; Roughgarden et al., 2016). While several variations of this basic model exist here we follow the strictest known version of the model used in (Andoni et al., 2014) and hence our algorithmic results hold in other versions as well.

In the MPC model we are given access to m identical processors with local RAM space s on each. For an input of size n the total space available to all processors is $m \cdot s = \tilde{O}(n)$. The computation is performed in synchronous rounds. In each round each machine: 1) performs a local computation on its data (under its local space restriction of s), 2) sends and receives messages of total length at most s to other machines which are received before the next round begins[1] (see Figure 2). Furthermore, we assume that the most time/space-efficient known algorithm for local subproblems (in our case almost linear-time and space) is used on each machine during the round.

In this setup the key complexity measure of performance in such computation is the number of rounds it takes to complete it as other characteristics such as time and communication depend directly on it. The parameter s is set to n^α for some fixed constant $\alpha < 1$, see (Karloff et al., 2010; Andoni et al., 2014) for more details. In this setting of parameters sorting can be done in $O(1)$ rounds (Goodrich et al., 2011) while sparse graph connectivity takes $O(\log n)$(Rastogi et al., 2013; Kiveris et al., 2014) which is conjectured to be optimal (Karloff et al., 2010; Beame et al., 2013; Rastogi et al., 2013; Roughgarden et al., 2016). It is folklore that an $O(\log n)$-round algorithm for MST in sparse graphs can be obtained via a simulation of Boruvka's algorithm in MPC. We use these facts extensively in this paper.

[1]Note that restriction of s on the total length of received messages follows from the local space constraint assuming there is no computation performed on the fly on incoming data.

1.3. Our results and previous work

While scalable algorithms with provable guarantees for other popular clustering methods such as k-means and k-median are known (Bahmani et al., 2012; Balcan et al., 2013) we are not aware of any such algorithms for Single-Linkage Clustering [2]. Also despite the fact that scalable heuristics exist for k-SLC and MST computation for vector data, e.g. (Jin et al., 2015), the only MPC algorithm with provable guarantees in this area that we are aware of is (Andoni et al., 2014)[3]. For other recent work on geometric data structures and algorithms in the MPC model see (Agarwal et al., 2016; Nath et al., 2016) and results on distributed constructions of coresets (Agarwal et al., 2005; Indyk et al., 2014; Bateni et al., 2014).

In (Andoni et al., 2014) it is shown that a $(1+\epsilon)$-approximate MST under ℓ_2 can be constructed in $O(1)$ rounds of MPC for constant dimension. However, while the overall cost of the MST is a good approximation to the optimum the length of any given edge can be arbitrarily distorted. This makes it impossible to directly use this algorithm of for the Single-Linkage Clustering problem. For example, consider an input corresponding to a set of points on the line shown in Figure 3 and $k = 2$. In this case a $(1+\epsilon)$-approximate MST would not necessarily lead to a $(1+\epsilon)$-approximate clustering as any such clustering would have to have clusters $\{1, \ldots, n-1\}$ and $\{n\}$ which are at distance 100 from each other. Moreover, the algorithm of (Andoni et al., 2014) will indeed introduce edges of length $\Omega(\epsilon n)$ into its approximate MST between the first $n-1$ points if run on this example. Hence for the MST constructed this way the basic approach of removing the longest edge to obtain a 2-SLC will result in two clusters which are at distance 1 with a very large probability.

[2]With the exception of recent work of (Derakhshan et al., 2017) who consider a more general graph metric setting and hence get results which are inherently different from our work as representation of the metric requires $\Theta(n^2)$ space

[3]For general graph metrics an MST algorithm in MPC is given in (Karloff et al., 2010). In our case using this algorithm directly would imply a quadratic increase in space since our graph is implicitly given by n^2 distances between the vectors and hence constructing the graph explicitly is infeasible under the overall space restriction.

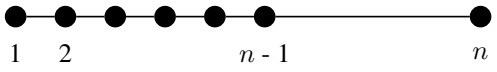

Figure 3. $\forall i \leq n-1\colon \|v_{i-1} - v_i\|_2 = 1, \|v_{n-1} - v_n\|_2 = 100$.

In this paper we show how to overcome this difficulty and give a different family of algorithms which allow to compute an approximate Single-Linkage Clustering under various distance metrics. While in (Andoni et al., 2014) only ℓ_2 metric is considered here we further extend this framework so that it also applies to ℓ_1 and ℓ_∞ with similar performance guarantees. Perhaps most interestingly, while an arbitrarily good MST approximation can be computed in $O(1)$ rounds of MPC (for fixed dimension) our algorithms for k-SLC run in $O(\log n)$ rounds. As it turns out, such an increase is likely to be necessary. We justify it through a number of hardness results. Our results show that even for $k = 2$ assuming two most popular conjectures in the MPC literature regarding complexity of sparse connectivity no $o(\log n)$-round algorithm can compute k-SLC for sufficiently large dimension of the data with better than some fixed constant-factor approximation that depends on the distance metric used. See Table 1 for a summary of these results[4].

In order to complete the picture of approximability of k-SLC under the most frequently used ℓ_p distances we also give algorithms and hardness results under Hamming distance (commonly referred to as ℓ_0). In contrast to other distances studied in this paper we are able to completely resolve approximability of the k-SLC problem for constant-dimensional data in this case. As we show, there exists an exact algorithm for $d = O(1)$ that runs in $O(\log n)$ rounds of MPC while under Conjecture 3.1 no algorithm running in $o(\log n)$ rounds can obtain better than 2-approximation even for $d = 2$. See Table 1 for details.

1.4. Our techniques

$\boldsymbol{\ell_1, \ell_2, \ell_\infty}$ Our algorithms under ℓ_1, ℓ_2 and ℓ_∞ all share the same high-level structure: we tackle the problem of the input having $O(n^2)$ edges by first constructing a sparsifier that only has $O(n \log n)$ edges and then run an MST algorithm on this sparsifier. In order to construct a sparsifier we execute a $(1+\epsilon)$-approximate MST algorithm $O(\log n)$ times and collect all edges of the MSTs constructed in these executions. We then run an exact $O(\log n)$-round exact MST algorithm on this set of $O(n \log n)$ edges and output clusters resulting from removing $k-1$ longest edges of the resulting MST. Note that the executions of the $(1+\epsilon)$-approximate MST algorithm can be done in parallel and hence it is the second step that introduces $O(\log n)$ rounds into the overall complexity of the algorithm. Our algorithms under ℓ_1, ℓ_2 and ℓ_∞ are given in Section 2. Assuming the same high-level structure this approach is unlikely to be improved as there are no known algorithms for solving MST in sparse graphs in $o(\log n)$ rounds.

Hardness In fact, we make the above observation formal by giving reductions from two most popular problems conjectured to require $\Omega(\log n)$ rounds in the MPC model: sparse connectivity (Conjecture 3.1) and a stronger "one cycle vs. two cycles" problem (Conjecture 3.2). Our reductions follow the same general strategy – we introduce a vector $v_i \in \mathbb{R}^n$ for each vertex in the input graph. This vector is initially set to be e_i, the i-th standard unit vector. Then for each edge (i, j) adjacent to the vertex i we update the coordinate j of the vector by adding a carefully chosen value ξ. This ensures that the for pairs of points which are connected by an edge the distance between their correponding vectors is different from the distance between points which are not connected by an edge. The parameter ξ is then chosen to maximize the ratio of distances in these two cases. Details are given in Section 3.

$\boldsymbol{\ell_0}$ Under ℓ_0 (Hamming distance) we can't construct a $(1+\epsilon)$-approximate MST using (Andoni et al., 2014) and hence our algorithms and hardness results are quite different. Using sorting as a primitive we construct an auxiliary graph and then run an $O(\log n)$-round connectivity algorithm on it d times. This way we obtain an exact MST and hence an exact k-SLC for any value of k. Details are given in Section A. Our hardness reduction in this case is also quite different as we construct a hard instance by creating a set of points in 2D instead of using high-dimensional vectors. Hence our result rules out an $o(\log n)$-round 2-approximation even for $d = 2$. See Section 3.2 for details.

1.5. Experimental results

We implemented our algorithm (for ℓ_2 distances) in Java on Apache Spark and empirically evaluated the performance. The largest datasets we used were the SIFT10M and HIGGS datasets from the UCI ML repository which has been used widely in literature ($\approx 11 \times 10^7$). Note that storage of the n^2 adjacency would take nearly 960TB of memory and hence building a complete graph locally is infeasible. We observed speedups of several orders of magnitude compared to our benchmark sequential Prim's algorithm when using 200 reducers. We remark that the speedup is not just due to the parallelism in our algorithm but also due to the use of approximation which is helpful even if the algorithm is executed locally. See Section 4.

[4]While our algorithms work in the most restricted known version of MPC model, our hardness results also hold in more relaxed versions for which hardness of sparse connectivity is conjectured, see (Roughgarden et al., 2016) for further details. Furthermore, in hardness results for ℓ_0 and ℓ_1 that require dimension $d = \Omega(n)$ the result holds for $O(1)$-sparse vectors, i.e. the overall input size is still $O(n)$ words.

Table 1. Approximation and hardness of k-Single Linkage Clustering in MPC under ℓ_p distances.

	Approximation in $O(\log n)$ rounds	Hardness of approximation in $o(\log n)$ rounds
ℓ_0	Exact for $d = O(1)$, Thm. A.1	2 for $d = 2$ under Conj. 3.1, Thm. 3.5 3 for $d = \Omega(n)$ under Conj. 3.2, Thm. 3.3
ℓ_1	$(1+\epsilon)$ for $d = O(1)$, Thm. 2.1	3 for $d = \Omega(n)$ under Conj. 3.2, Thm. 3.3 2 for $d = \Omega(n)$ under Conj. 3.1, Thm. 3.3
ℓ_2	$(1+\epsilon)$ for $d = O(1)$, Thm. 2.1	$1.84 - \epsilon$ for $d = \Omega(\frac{\log n}{\epsilon^2})$ under Conj. 3.2, Thm. 3.3 $1.41 - \epsilon$ for $d = \Omega(\frac{\log n}{\epsilon^2})$ under Conj. 3.1, Thm. 3.3
ℓ_∞	$(1+\epsilon)$ for $d = O(1)$, Thm. 2.1	2 for $d = \Omega(\log n)$ under Conj. 3.1, Thm. 7.1 (Andoni et al., 2014)

2. Algorithms

At a high level our k-SLC algorithm for ℓ_2 is very simple and can be described as follows:

Algorithm 1 Simplified k-SLC Algorithm for ℓ_2

Input: vectors $v_1, \ldots, v_n \in \mathbb{R}^d$
$E' = \emptyset$
Repeat $O(\log n)$ times sequentially:
  $E =$ set of edges of a $(1+\epsilon)$-approximate MST
  $E' = E \cup E'$
Run Boruvka's MST algorithm on E' and remove $k-1$ longest edges to obtain the clustering.

In order for the above algorithm to produce an approximate k-SLC it is important however that the MST constructed during sequential repetitions obeys certain properties. As we show below, for ℓ_2 these properties hold for the algorithm of (Andoni et al., 2014). Furthermore, in order to extend this approach to ℓ_1, ℓ_∞ a more detailed analysis is required.

2.1. Partition-based algorithm for $\ell_1, \ell_2, \ell_\infty$

Theorem 2.1. *For each of the three metrics ℓ_1, ℓ_2 and ℓ_∞ for any constants $0 < \eta \leq 3$, $0 < \alpha < 1/2$ such that $\eta = \Omega(s^{\frac{2\alpha-1}{2d}})$ there exists an $O(\log n)$-round MPC algorithm that computes $(1+\eta)$-approximate k-Single-Linkage Clustering for any constant dimension d given as an input set of vectors $v_1, \ldots, v_n \in \mathbb{R}^d$. The algorithm works simultaneously for all values of k under these metrics. The algorithm is randomized and produces correct result with high probability. Given access to machines with RAM space s it uses $\tilde{O}(n/s)$ machines and time at most $\tilde{O}(s)$ per round on each machine.*

In this section we describe a generic partition-based algorithm, Algorithm 2.1, that is used to prove the above theorem. We also give analysis of its approximation guarantee. Algorithm 2.1 relies on (a, b, c)-distance-preserving partitions and uses Algorithm 3 which we describe in Section B.

We start by recalling standard definitions of distance preserving hierarchical partitions. Let $M(S, \rho)$ be a metric space with distance function ρ. For $S' \subseteq S$ we denote its diameter as $\Delta(S') = \sup_{x,y \in S'} \rho(x, y)$. A *deterministic hierarchical partition* P with L *levels* is defined as a sequence $P = (P_0, \ldots, P_L)$ where $P_L = \{S\}$ and each level P_ℓ is a subdivision of $P_{\ell+1}$. For a partition P_i we call its parts *cells*. The *diameter* at level i is defined as $\Delta(P_i) = \max_{C \in P_i} \Delta(C)$. The *degree of a cell* $C \in P_\ell$ is $deg(C) = |\{C' \in P_{\ell-1} : C' \subseteq C\}|$. The *degree of a hierarchical partition* is the maximum degree of any of its cells. The unique cell at level ℓ containing a point x is denoted as $C_\ell(x)$. We say that a partition is indexable if this cell can be computed based on x and ℓ. A *randomized hierarchical partition* is a distribution over deterministic hierarchical partitions.

Definition 2.1 (Distance-preserving partition)**.** *For parameters $a \in (0, 1)$, $b, c \in \mathbb{R}^+$ and $\gamma > 1$ a randomized hierarchical partition $\mathcal{P}$ of a metric space with L levels is (a, b, c)-distance-preserving with approximation γ if the degree of all deterministic partitions in its support is at most c and the following properties are satisfied for $\Delta_\ell = \gamma a^{L-\ell} \Delta(S)$:*

1. *(Bounded diameter) For every deterministic partition $P = (P_0, \ldots, P_L)$ in the support of $\mathcal{P}$ and for all $\ell \in \{0, \ldots, L\}$ it holds that $\Delta(P_\ell) \leq \Delta_\ell$.*
2. *(Probability of cutting an edge) For every $x, y \in S$ and for all $\ell \in \{0, \ldots, L\}$:*

$$\Pr_{P \sim \mathcal{P}}[C_\ell(x) \neq C_\ell(y)] \leq b \frac{\rho(x, y)}{\Delta_\ell}.$$

Let $M(S, \rho)$ be a metric space and $w \colon S \times S \to \mathbb{R}^+$ be a weight function $w(x, y) = \rho(x, y)$. We think of w as representing weights of edges in a complete graph. Let $MST_i(w)$ denote the weight of the i-th Minimum Spanning Tree edge of this graph sorted in non-decreasing order.

Algorithm 2 Partition-based Distributed k-SLC Algorithm

Input: vectors $v_1, \ldots, v_n \in \mathbb{R}^d$, parameters η, α, p
$E = \emptyset$
Set $a = s^{-\alpha/d}, b = poly(d), c = s^{\alpha}, L = O(\log_{1/a} n)$
Set $\epsilon = \min\left(\frac{\eta}{6c_1 Lb}, \frac{\eta}{3c_2}\right)$
Repeat $O(\log n)$ times sequentially:
  Sample partition P with L levels from (a, b, c)-distance-preserving family w_P
  Execute unit step Algorithm 3 for each cell in P with parameter ϵ
  $E' =$ set of edges output in the previous step
  $E = E \cup E'$
Run Boruvka's MST algorithm on E and remove $k-1$ longest edges to obtain the clustering.

Let $w^+ : S \times S \to \mathbb{R}^+$ be a random family of functions that satisfies that for each x, y it holds that $w(x,y) \leq w^+(x,y)$ and $\mathbb{E}[w^+(x,y)] \leq (1+\gamma)w(x,y)$ for some fixed $\gamma > 0$. Note that the weights given by this random family to different pairs might be correlated with each other.

Definition 2.2 (Crossing edge)**.** *For a partition $(C_1, \ldots, C_t)$ of S we say that a pair of points (x, y) crosses this partition if $x \in C_i$ and $y \in C_j$ for $i \neq j$.*

Definition 2.3 (Cut-preserving spanning tree)**.** *We say that $\mathcal{T}$ is an α-cut-preserving spanning tree for $w : S \times S \to \mathbb{R}^+$ if for every partition (C_1, C_2) of S there exists an edge in $\mathcal{T}$ that crosses this partition and is at most α times longer than the shortest such edge with respect to w.*

As we show below Algorithm 2.1 can be seen as performing the following experiment: draw k functions $w_1, \ldots, w_k$ i.i.d at random from the family w^+. Compute a $(1+\delta)$-cut-preserving spanning tree $\mathcal{T}_i$ for each w_i. Then for each $(x,y) \in S \times S$ define $w'_i(x,y) = w(x,y)$ if (x,y) is in this spanning tree and $w'_i(x,y) = +\infty$ otherwise. Then for all $(x,y) \in S \times S$ define $\bar{w}^k(x,y) = \min_{i=1}^k w'_i(x,y)$. The final run of Boruvka's MST algorithm is then executed on $\bar{w}^k$.

Indeed, random family of functions w^+ satisfying the properties described above is constructed by Algoirthm 2.1 as follows from a result (Andoni et al., 2014) given. It is important to note that cut-preserving spanning tree computations for random function samples from this family required above can be also performed as guaranteed by the following lemma:

Lemma 2.2 ((Andoni et al., 2014), Lemmas 3.4 and 3.13)**.** *Given access to an (a, b, c)-distance-preserving partition with L levels and approximation γ for $M(S, \rho)$ there exists an MPC algorithm that runs in $O(1)$ rounds and constructs a random family of weight functions w_P which satisfies:*

$$\rho(i,j) \leq w_P(i,j) \text{ and } \mathbb{E}[w_P(i,j)] \leq (1 + c_1 \epsilon L b)\, \rho(i,j).$$

Furthermore, execution of unit step Algorithm 3 for all cells in this partition for a random function w^* sampled from w_P produces a $(1 + c_2\epsilon)$-cut-preserving spanning tree $\mathcal{T}$ for w^*.

Let $w(i,j) = \|v_i - v_j\|_2$, $w^+ = w_P$ and let $\gamma = c_1 \epsilon d$ and $\delta = c_2 \epsilon$.

Lemma 2.3. *Let $n = |S|$. There is a large enough constant $c > 0$ such that if $k = c \log n$ then for all i it holds that:*

$$\Pr_{w_1, \ldots, w_k}[MST_i(\bar{w}_k) \geq (1+2\gamma)(1+\delta) MST_i(w)] \leq n^{-\Omega(1)}.$$

Proof. Fix $(x,y) \in S \times S$ and let $\Delta(x,y) = w^+(x,y) - w(x,y)$. Because $\Delta(x,y) \geq 0$ and $\mathbb{E}[\Delta(x,y)] \leq \gamma w(x,y)$ with probability at least $1/2$ it holds that $\Delta(x,y) \leq 2\gamma w(x,y)$ by Markov inequality. If $k = c \log n$ then with probability $1 - 1/n^c$ there exists i such that $w_i(x,y) - w(x,y) \leq 2\gamma w(x,y)$. By a union bound over all n^2 pairs (x,y) with probability $1 - 1/n^{c-2}$ for each such pair a corresponding index exists. Below we refer to this event as $\mathcal{E}$ and condition on it.

Proposition 2.4. *Let $(C_1, \ldots, C_t)$ be an arbitrary partition of S. Let $(x^*, y^*) \in S \times S$ be the closest w.r.t w pair of points that belong to different parts of this partition. Then conditioned on the event $\mathcal{E}$ there exists a pair of points (x', y') that crosses this partition and:*

$$w(x^*, y^*) \leq \bar{w}^k(x', y') \leq (1+2\gamma)(1+\delta) w(x^*, y^*).$$

Proof. First, consider the case when $t = 2$ and consider any partition (C_1, C_2) of S. Let (x^*, y^*) be the shortest edge that crosses this partition, i.e. $(x^*, y^*) := arg\min_{x \in C_1, y \in C_2} w(x,y)$. Conditioned on $\mathcal{E}$ there exists i such that $w_i(x^*, y^*) \leq (1+2\gamma) w(x^*, y^*)$. Furthermore, there exists an edge (x', y') in the $(1+\delta)$-cut-preserving spanning tree $\mathcal{T}_i$ constructed for w_i that has length $w'_i(x', y') = w_i(x', y') \leq (1+\delta) w_i(x^*, y^*) \leq (1+2\gamma)(1+\delta) w(x^*, y^*)$. On the other hand, because $w_i \geq w$ for every pair (x,y) that crosses the partition (C_1, C_2) it holds that $w_i(x,y) \geq w(x^*, y^*)$. Combining these two facts we conclude that in $\mathcal{T}_i$ there exists some edge (x', y') that crosses the cut and satisfies $w(x^*, y^*) \leq w'_i(x', y') \leq (1+2\gamma)(1+\delta) w(x^*, y^*)$. By definition of $\bar{w}^k$ the same holds for it as well, i.e. $w(x^*, y^*) \leq \bar{w}^k(x', y') \leq (1+2\gamma)(1+\delta) w(x^*, y^*)$.

Now suppose $t > 2$. For $i = 1, \ldots, t$ define a family of cuts (S_i, T_i) where $S_i = C_i$ and $T_i = \cup_{j \neq i} C_j$. Let (x_i^*, y_i^*) be the shortest pair crossing the cut (S_i, T_i). If (x^*, y^*) is the shortest edge that crosses $(C_1, \ldots, C_t)$ then we have $w(x^*, y^*) = \min_i w(x_i^*, y_i^*)$. Let $i^* = arg\min_i w(x_i^*, y_i^*)$. Then using the argument above for $t = 2$ there exists (x', y') such that $x' \in S_{i^*}, y \in T_{i^*}$ and:

$$\begin{aligned} w(x^*, y^*) &= w(x^*_{i^*}, y^*_{i^*}) \\ &\le \bar{w}^k(x', y') \\ &\le (1+2\gamma)(1+\delta) w(x^*_{i^*}, y^*_{i^*}) \\ &= (1+2\gamma)(1+\delta) w(x^*, y^*). \square \end{aligned}$$

Given Proposition 2.4 the rest of the proof is the same as analysis of approximate Kruskal's algorithm in (Indyk, 2000), we give the proof here for completeness. Since edges output by Kruskal's algorithm are produced in the order of non-decreasing weight MST_i is the i-th edge that is output. Consider executions of Kruskal's algorithm on weights w and $\bar{w}^k$. Let the edges output by the former execution be $e_1, \ldots, e_{n-1}$ in order. Let the edges output by the latter execution be $e'_1, \ldots, e'_{n-1}$.

To prove Lemma 2.3 it suffices to show that conditioned on $\mathcal{E}$ it holds that $w(e_i) \le w(e'_i) \le (1+2\gamma) w(e_i)$ for all i. The first inequality here essentially follows from the fact that the weight of the i-th MST edge is a monotone function of the weights and $w \le \bar{w}^k$.

The i-th edge in Kruskal's algorithm is constructed by joining two closest clusters among $n - i + 1$ clusters constructed so far. Let these clusters in the execution of Kruskal's algorithm on $\bar{w}^k$ be denoted as $C_1, \ldots, C_{n-i+1}$. The key observation is that there exists an index $i^* \le i$ such that endpoints of the edge e_{i^*} belong to different parts of the partition $C_1, \ldots, C_{n-i+1}$. Indeed, edges $e_1, \ldots, e_i$ form a forest and thus having all such edges be inside $C_1, \ldots, C_{n-i+1}$ would be a contradiction.

Let (x^*, y^*) be the closest w.r.t to w pair of points in different parts of the partition $C_1, \ldots, C_{n-i+1}$. By applying Proposition 2.4 to e_{i^*} there exists a pair of points (x', y') whose endpoints belong to different parts of the partition $C_1, \ldots, C_{n-i+1}$ and $\bar{w}^k(x', y') \le (1+2\gamma) w(x^*, y^*)$. Putting everything together we have:

$$\begin{aligned} w(e'_i) &\le \bar{w}^k(e'_i) && w \le \bar{w}^k \\ &\le \bar{w}^k(x', y') \\ &\le (1+2\gamma)(1+\delta) w(x^*, y^*) && \text{Proposition 2.4} \\ &\le (1+2\gamma)(1+\delta) w(e_{i^*}) \\ &\le (1+2\gamma)(1+\delta) w(e_i) && \square \end{aligned}$$

The second inequality follows because e'_i is shortest edge w.r.t $\bar{w}^k$ that crosses $(C_1, \ldots, C_{n-i+1})$. The last inequality follows because $i^* \le i$, edge weights are non-decreasing.

Putting everything together we obtain analysis of approximation guaranteed by Algorithm 2.1.

Theorem 2.5. *For $\eta \le 3$ and $p = 1, 2, \infty$ Algorithm 2.1 constructs a spanning tree for $w(i,j) = \|v_i - v_j\|_p$ for each t its t-th longest edge (x, y) has weight $w(x, y) \le (1+\eta) MST_k(w)$. This guarantee holds with high probability over the randomness used in Algorithm 2.1.*

Proof. Note that taking $w^+ = w_P$ for $w(i,j) = \|v_i - v_j\|_p$ where $p = 1, 2, \infty$ satisfies conditions of Lemma 2.3 by Lemma 2.2. Hence our algorithm constructs a function $\bar{w}_k$ with properties required for Lemma 2.3. Since $c_1 \epsilon Lb \le \eta/6$ and $c_2 \epsilon \le \eta/3$ we can set $\delta = \eta/6$ and $\gamma = \eta/3$ in Lemma 2.3 and hence for $\eta \le 3$:

$$\Pr[\mathcal{E}_1] \ge \Pr[\mathcal{E}_2] \ge 1 - \frac{1}{poly(n)}.$$

where $\mathcal{E}_1$ is the event that $MST_i(\bar{w}_k) \le (1+\eta) MST_i(w)$ and $\mathcal{E}_2$ is the event that $MST_i(\bar{w}_k) \le (1+2\gamma)(1+\delta) MST_i(w)$.

After $\bar{w}_k$ is constructed by running Boruvka's algorithm on it we find an MST exactly and hence the approximation guarantee for each of the MST edges follows. $\square$

2.2. Solve-and-Sketch framework and unit step

We use Solve-and-Sketch (SAS) framework of (Andoni et al., 2014) for computing an approximate minimum spanning tree. SAS framework works with a partition $P = (P_0, \ldots, P_L)$ of the input $M(S, \rho)$, sampled from a randomized (a, b, c)-partition $\mathcal{P}$. Then SAS algorithm proceeds through L levels, and in level ℓ a *unit step* algorithm $\mathcal{A}_u$ is executed in each cell C of the partition P_ℓ, with input the union of the outputs of the unit steps applied to the children of C. The unit step also outputs a subset of the edges of a spanning tree in addition to the input for the next level. Once the unit step has been executed for the root cell of partition at level P_L (and hence also for all other cells) the computation is complete. We give the description of the unit step algorithm below (Algorithm 3).

Definition 2.4 (δ-covering)**.** *Let $M = (S, \rho)$ be a metric space and let $\delta > 0$. A set $S' \subseteq S$ is a* δ-covering *if for any point $x \in S$, there is a point $y \in S'$ such that $\rho(x, y) \le \delta$.*

3. Hardness of k-SLC

3.1. Hardness under ℓ_1 and ℓ_2

The following two conjectures are widely used in the MPC literature (Karloff et al., 2010; Beame et al., 2013; Rastogi et al., 2013; Roughgarden et al., 2016). Note that the second conjecture is stronger and hence can potentially be used to get stronger hardness results.

Conjecture 3.1 (Sparse connectivity hardness)**.** *If $s = n^\alpha$ for a constant $\alpha < 1$ then solving connectivity on an input*

Algorithm 3 Unit Step at Level ℓ,

Input: Cell $C \in P_\ell$, a collection $V(C)$ of points in C, and a partition $Q = \{Q_1, \dots Q_k\}$ of $V(C)$ into previously computed connected components.
Output: $V' \subseteq V$, an $\epsilon^2\Delta_\ell$-covering for C, the partition $Q(V')$ induced by Q on V'.
$\theta := 0$
while $k > 1$ and $\theta \leq \epsilon\Delta_\ell$ **do**
  Let $\tau = \min_{\substack{i,j \\ i \neq j}} \min_{u \in Q_i, v \in Q_j} \rho(u, v)$
  Find $u \in Q_i$ and $v \in Q_j$ for some i and j such that $i \neq j$ and $\rho(u, v) \leq (1 + \epsilon)\tau$.
  $\theta := \rho(u, v)$
  if $\theta \leq \epsilon\Delta_\ell$ **then**
    Output tree edge (u, v).
    Merge Q_i and Q_j and update Q and k.
  end if
end while

graph with n vertices and $O(n)$ edges requires $\Omega(\log n)$ rounds of MPC.

Conjecture 3.2 (One cycle vs. two cycles hardness)**.** *If $s = n^\alpha$ for a constant $\alpha < 1$ then distinguishing the following two instances requires $\Omega(\log n)$ rounds of MPC: 1) a cycle on n vertices, 2) two cycles on $n/2$ vertices each.*

Theorem 3.3. *No $o(\log n)$-round MPC algorithm can achieve approximation for 2-SLC:*

1. *Better than $(\sqrt{2+\sqrt{2}} - \epsilon)$ under ℓ_2 for $d = \Omega(\log n/\epsilon^2)$ under Conjecture 3.2.*
2. *Better than 3 under ℓ_1 for $O(1)$-sparse vectors and $d = \Omega(n)$ under Conjecture 3.2.*
3. *Better than $(\sqrt{2} - \epsilon)$ under ℓ_2 for $d = \Omega(\log n/\epsilon^2)$ under Conjecture 3.1.*
4. *Better than 2 under ℓ_1 for $O(1)$-sparse vectors and $d = \Omega(n)$ under Conjecture 3.1.*

Proof. We give proof of Part 1 here, other proofs are similar and are deferred to Appendix E. Given an instance of the "one cycle vs. two cycles problem" we reduce it to the 2-SLC problem as follows:

1. Create a vector $v'_i \in \mathbb{R}^n$ for each vertex where $v'_i = e_i$ and e_i is the i-th standard unit vector.
2. For each edge (a, b) in the input graph update the corresponding vectors as $v'_a = v'_a + \xi e_b$ and $v'_b = v'_b + \xi e_a$ where $\xi = \frac{1}{\sqrt{2}}$.
3. Apply Johnson-Lindenstrauss transform to $v'_1, \dots, v'_n$ to construct $v_1, \dots, v_n \in \mathbb{R}^d$ where $d = O(\log n/\epsilon^2)$.

Note that the above reduction can be performed in only a constant number of MPC rounds. Indeed, Step 1 can be done locally by partitioning vectors between machines and to perform Step 2 we can send each edge (a, b) to the machines holding vectors v_a and v_b. For Step 3 note that for each i we have $v_i = Mv'_i$ where M is the Johnson-Lindenstrauss matrix and each v'_i has at most 3 non-zero entries. Hence, all v_i can be computed in one round of MPC with $O(\log n/\epsilon^2)$ communication per vector.

Proposition 3.4. *If (i, j) is an edge in the input graph then $\|v'_i - v'_j\|_2 = \sqrt{2}(\sqrt{2 - \sqrt{2}})$, otherwise $\|v'_i - v'_j\|_2 = 2$.*

Proof. Indeed, if there is an edge (i, j) in the input then there exist two other edges (i, i') and (j, j') and hence, the non-zero entries of v'_i and v'_j are as follows: $v'_{ii} = 1, v'_{ii'} = \xi, v'_{ij} = \xi, v'_{jj} = 1, v'_{jj'} = \xi, v'_{ji} = \xi$. Hence $\|v'_i - v'_j\|_2 = \sqrt{2(1-\xi)^2 + 2\xi^2}$. On the other hand, if there is no edge (i, j) then there exist four edges $(i, i'), (i, i''), (j, j')$ and (j, j'') and non-zero entries of v'_i and v'_j are: $v'_{ii} = 1, v'_{ii'} = \xi, v'_{ii''} = \xi, v'_{jj} = 1, v'_{jj'} = \xi, v'_{jj''} = \xi$. Hence $\|v'_i - v'_j\|_2 = \sqrt{2 + 4\xi^2}$. Maximum of the ratio $\frac{\sqrt{2+4\xi^2}}{\sqrt{2(1-\xi)^2+2\xi^2}}$ is achieved when $\xi = 1/\sqrt{2}$ and equals $\sqrt{2 + \sqrt{2}}$. □

By Proposition 3.4, if the input graph is one cycle then the cost of 2-SLC of $v'_1, \dots, v'_n$ equals $\sqrt{2}\sqrt{2 - \sqrt{2}}$, otherwise it is 2. As Johnson-Lindenstrauss transform preserves all pairwise distances up to a multiplicative $(1 \pm \epsilon)$ factor with high probability the same is true for the cost of 2-SLC of $v_1, \dots, v_n$ up to $\pm\epsilon$ error. This completes the proof.

□

3.2. Hardness of Hamming k-SLC

Theorem 3.5. *No algorithm for computing Hamming k-SLC cost for $d = 2$ in $o(\log n)$ rounds of MPC can achieve better than 2-approximation under Conjecture 3.1.*

Proof. Let $G(V, E)$ be an instance of sparse connectivity. Our reduction to Hamming 2-SLC constructs an input set of 2-dimensional vectors as follows: 1) for each vertex $i \in V$ create a vector (i, i), 2) or each edge $(i, j) \in E$ create a vector (i, j). Clearly this reduction can be performed in a constant number of rounds of MPC and the resulting instance has $|V| + |E| = O(n)$ many vectors. We will show that if the input graph is connected the cost of Hamming 2-SLC of the input equals 1 and the cost is 2 otherwise. Indeed, note that the distances between resulting vectors are always either 1 or 2. If G is connected then it is easy to construct a connected spanning subgraph in the resulting Hamming graph where each edge has cost 1. Indeed, consider a subgraph that for each edge (i, j) in the input graph contains two edges: one between vectors (i, i) and (i, j) and another between vectors (j, j) and (i, j). Clearly, if the input graph is connected then this is a connected spanning subgraph. Hence the Hamming MST cost of the con-

Table 2. Scalability experiments.

DATA SET	n POINTS	n^2 EDGES	d	TIME (S)	ϵ
SIFT10M	1.1×10^7	1.2×10^{14}	3	1.2×10^5	3
HIGGS	1.1×10^7	1.2×10^{14}	3	8.4×10^4	10

structed point set equals $|V| + |E| - 1$ and the Hamming 2-SLC cost equals 1. On the other hand, if G is disconnected then consider any partitioning (S, T) of G into connected components. Clearly, any two vectors representing vertices belonging to different parts of this partition in our reduction are at distance 2 from each other. This implies that the Hamming MST cost is at least $|V| + |E|$ and the Hamming 2-SLC cost is 2. □

4. Experiments

Small datasets Four standard clustering datasets used in the literature were taken for experimental evaluation: 1) Image dataset, $d = 3$, $n = 34112$ (house images, `https://cs.joensuu.fi/sipu/datasets/`), 2) KDDCUP04Bio dataset , $d = 10$, $n = 145751$ (preprocessed to select 10 numerical dimensions out of 74, accessed via the link above), 3) Shuttle data set from the UCI ML repository, $d = 9$, $n = 43500$. 4) US Census dataset from the UCI ML repository, $d = 8$, $n = 2548285$.

Due to page limitations here, we only show plots for the largest Census dataset. Other plots are deferred to Appendix D. Figure 4 shows dependence of speedup as a function of approximation. We observe a dramatic increase in the speedup at around approximation 1.26 due to the fact the local inputs start to fit in L2-cache. Figure 5 shows dependence of approximation on k for the census data.

Large datasets In order to test scalability, we took the largest real-valued vector datasets from the UCI ML repository: SIFT10M and HIGGS. Both the datasets have approximately 11 million entries. Thus, constructing the full matrix of distances in memory is clearly infeasible as the size of this matrix would be roughly 960TB in both cases[5]. Dimension reduction for this data was done using PCA for $d = 3$. Results are given in Table 2.

4.1. Experimental setup

We implemented Algorithm 2.1 in Java on Apache Spark 2.0.2 for Hadoop 2.7.3. Experiments were performed on two different setups:

Google Cloud Dataproc (GCD) platform on two cluster configurations: 1) single-core 1 master / 7 worker (1m/7w) cluster, 2) dual-core 1 master / 3 worker (1m/3w) cluster. Each core had an Intel Xeon E5 processor at 2.2–2.6 GHz and 3.75GB RAM + 10GB HDD space. Due to the limitations of the free tier access on GCD the total number of cores in a cluster is limited to 8, which is still sufficient to demonstrate at least an order of magnitude speedup over the benchmark sequential algorithm. This setup was used for the *small datasets*.

Local Simulation with 200 reducers on a Dell XPS13 Laptop with an Intel core I5 processor and 8GB RAM. This setup was used for the *large datasets*.

[5]Assuming 8-byte double-precision arithmetic.

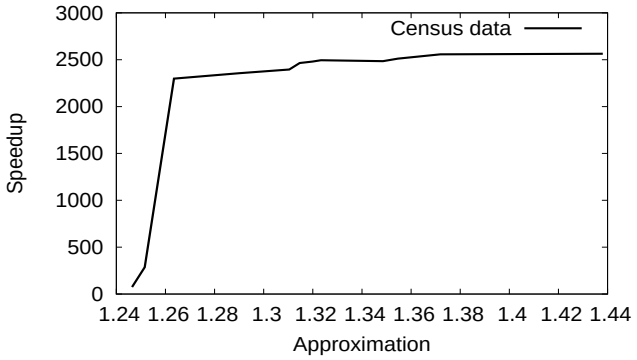

Figure 4. Speedup vs approximation

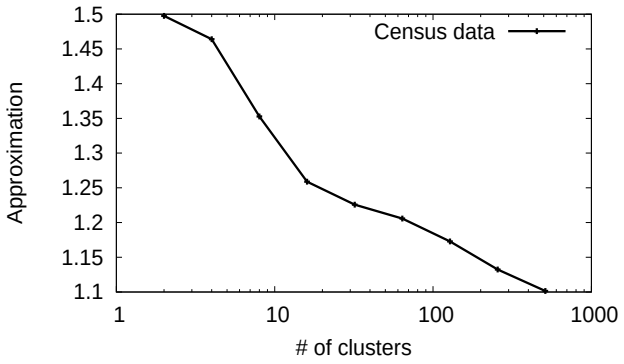

Figure 5. Approximation vs number of clusters

5. Acknowledgements

This research was supported by NSF Award 1657477. The authors would like to thank Alexandr Andoni, Aleksandar Nikolov and Krzysztof Onak for multiple discussions of (Andoni et al., 2014) and its relationship to the single-linkage clustering problem which led to this work.

References

Agarwal, Pankaj K., Har-Peled, Sariel, and Varadarajan, Kasturi R. Geometric approximation via coresets. *Combinatorial and Computational Geometry (MSRI publication)*, 52, 2005.

Agarwal, Pankaj K., Fox, Kyle, Munagala, Kamesh, and Nath, Abhinandan. Parallel algorithms for constructing range and nearest-neighbor searching data structures. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pp. 429–440, 2016.

Andoni, Alexandr, Nikolov, Aleksandar, Onak, Krzysztof, and Yaroslavtsev, Grigory. Parallel algorithms for geometric graph problems. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pp. 574–583, 2014. URL `http://arxiv.org/abs/1401.0042`.

Arya, Sunil, Mount, David M., Netanyahu, Nathan S., Silverman, Ruth, and Wu, Angela Y. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998.

Bahmani, Bahman, Moseley, Benjamin, Vattani, Andrea, Kumar, Ravi, and Vassilvitskii, Sergei. Scalable k-means++. *PVLDB*, 5(7):622–633, 2012.

Balcan, Maria-Florina, Ehrlich, Steven, and Liang, Yingyu. Distributed k-means and k-median clustering on general communication topologies. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pp. 1995–2003, 2013.

Bateni, Mohammadhossein, Bhaskara, Aditya, Lattanzi, Silvio, and Mirrokni, Vahab. Distributed balanced clustering via mapping coresets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2591–2599. Curran Associates, Inc., 2014.

Beame, Paul, Koutris, Paraschos, and Suciu, Dan. Communication steps for parallel query processing. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, pp. 273–284, 2013.

Derakhshan, Mahsa, Behnezhad, Soheil, Bateni, Mohammadhossein, Mirrokni, Vahab, Hajiaghayi, Mohammad-Taghi, Lattanzi, Silvio, and Kiveris, Raimondas. On distributed hierarchical clustering. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, 2017.

Feldman, Jon, Muthukrishnan, S., Sidiropoulos, Anastasios, Stein, Clifford, and Svitkina, Zoya. On distributing symmetric streaming computations. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pp. 710–719, 2008.

Fish, Benjamin, Kun, Jeremy, Lelkes, Ádám Dániel, Reyzin, Lev, and Turán, György. On the computational complexity of mapreduce. In *Distributed Computing - 29th International Symposium, DISC 2015, Tokyo, Japan, October 7-9, 2015, Proceedings*, pp. 1–15, 2015.

Goodrich, Michael T., Sitchinava, Nodari, and Zhang, Qin. Sorting, searching, and simulation in the mapreduce framework. In *Algorithms and Computation - 22nd International Symposium, ISAAC 2011, Yokohama, Japan, December 5-8, 2011. Proceedings*, pp. 374–383, 2011.

Gower, John C and Ross, GJS. Minimum spanning trees and single linkage cluster analysis. *Applied statistics*, pp. 54–64, 1969.

Indyk, Piotr. *High-dimensional Computational Geometry*. PhD thesis, Stanford University, 2000.

Indyk, Piotr, Mahabadi, Sepideh, Mahdian, Mohammad, and Mirrokni, Vahab S. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pp. 100–108, 2014.

Jin, Chen, Liu, Ruoqian, Chen, Zhengzhang, Hendrix, William, Agrawal, Ankit, and Choudhary, Alok. A scalable hierarchical clustering algorithm using spark. In *Big Data Computing Service and Applications (BigDataService), 2015 IEEE First International Conference on*, pp. 418–426. IEEE, 2015.

Karloff, Howard J., Suri, Siddharth, and Vassilvitskii, Sergei. A model of computation for mapreduce. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pp. 938–948, 2010.

Kiveris, Raimondas, Lattanzi, Silvio, Mirrokni, Vahab S., Rastogi, Vibhor, and Vassilvitskii, Sergei. Connected components in mapreduce and beyond. In *Proceedings of the ACM Symposium on Cloud Computing, Seattle, WA, USA, November 03 - 05, 2014*, pp. 18:1–18:13, 2014.

Malak, Michael S and East, Robin. *Spark GraphX in action*. Manning Publ., 2016.

Manning, Christopher D., Raghavan, Prabhakar, and Schütze, Hinrich. *Introduction to information retrieval*. Cambridge University Press, 2008. ISBN 978-0-521-86571-5.

Nath, Abhinandan, Fox, Kyle, Munagala, Kamesh, and Agarwal, Pankaj K. Massively parallel algorithms for computing TIN dems and contour trees for large terrains. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2016, Burlingame, California, USA, October 31 - November 3, 2016*, pp. 25:1–25:10, 2016.

Rastogi, Vibhor, Machanavajjhala, Ashwin, Chitnis, Laukik, and Sarma, Anish Das. Finding connected components in map-reduce in logarithmic rounds. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pp. 50–61, 2013.

Roughgarden, Tim, Vassilvitskii, Sergei, and Wang, Joshua R. Shuffles and circuits: (on lower bounds for modern parallel computation). In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2016, Asilomar State Beach/Pacific Grove, CA, USA, July 11-13, 2016*, pp. 1–12, 2016.

Zahn, Charles T. Graph-theoretical methods for detecting and describing gestalt clusters. *Computers, IEEE Transactions on*, 100(1):68–86, 1971.