Brief Announcement: Parallel Network Mapping Algorithms

Ramtin Afshar
Univ. of California, Irvine
USA
afsharr@uci.edu

Michael T. Goodrich
Univ. of California, Irvine
USA
goodrich@uci.edu

Pedro Matias
Univ. of California, Irvine
USA
pmatias@uci.edu

Martha C. Osegueda
Univ. of California, Irvine
USA
mosegued@uci.edu

ABSTRACT

Motivated from parallel network mapping, we provide efficient query complexity and round complexity bounds for graph reconstruction using distance queries, including a bound that improves a previous sequential complexity bound. Our methods use a high-probability parametric parallelization of a graph clustering technique of Thorup and Zwick, which may be of independent interest.

CCS CONCEPTS

• **Theory of computation → Parallel algorithms**.

KEYWORDS

Network mapping; graph reconstruction; parallel algorithms

ACM Reference Format:
Ramtin Afshar, Michael T. Goodrich, Pedro Matias, and Martha C. Osegueda. 2021. Brief Announcement: Parallel Network Mapping Algorithms. In *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '21), July 6–8, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3409964.3461822

1 INTRODUCTION

Suppose we are given access to a connected, undirected graph, $G = (V, E)$, where the set of n vertices, V, is known but the set of edges, E, is initially unknown. Motivated by network mapping, the goal of the ***graph reconstruction*** problem is to learn E by issuing queries to an oracle that knows E in a small number of rounds. In this paper, we focus on distance queries:

- distance(u, v): return the number of edges on a shortest path from u to v in G.

Because of the inherently distributed nature of the network mapping motivation, we are interested in algorithms that work in parallel. Thus, we measure the efficiency of a parallel network mapping algorithm, **A**, using the following measures, which we define in terms of n, the number of vertices in G, $|V|$.

- $Q(n)$: the ***query complexity*** of **A**. This is the total number of queries issued to the oracle.
- $R(n)$: the ***round complexity*** of **A**. This is the number of rounds of querying performed by **A**, where the queries issued in a round are given in a batch such that any query issued in a round may not depend on the response to any other query in that round (but each query may depend on results of queries from previous rounds).

Prior Related Work. There is considerable prior work on ***graph reconstruction***, e.g., see [1–6, 6–12, 14–17, 19, 20, 22–24], but most of this work has focused on how to sequentially reconstruct the graph, G. The most relevant prior work is by Kannan, Mathieu, and Zhou [19], who give algorithms for reconstructing Δ-degree connected graphs using $O(\Delta^3 n^{3/2} \log^2 n \log\log n)$ distance queries in expectation, and they prove that any algorithm takes at least $\Omega(\Delta n \log n / \log(\log n / \log \Delta))$ queries for the reconstruction.

Our Results. We first provide a parallel implementation of a graph clustering technique of Thorup and Zwick [25] that allows us to tune the number of rounds for graph clustering to be any value between $O(1)$ and $O(\log n)$, while their original implementation implies an expected round complexity of $O(\log n)$. Moreover, we show that our complexity bounds hold with high probability [1]. Then, we provide parallel graph reconstruction algorithms with query complexities and round complexities that improve bounds that can be derived from previous bounds for graph reconstruction problems. One of our parallel constructions improves the current best sequential complexity for reconstructing a degree-Δ graph from distance queries by [19]. Also, we give a constant-round reconstruction algorithm with query complexity better than the trivial brute-force algorithm.

2 PARALLEL GRAPH CLUSTERING

We begin by describing our parallel graph clustering algorithm, which may be of independent interest, as it provides a parameterized parallel extension of a well-known graph clustering result of Thorup and Zwick [25]. Also, whereas Thorup and Zwick establish their bounds in expectation, we establish ours with high probability.

We begin with some review from Thorup and Zwick [25]. Let $G = (V, E)$ be a connected, undirected n-vertex graph, and let $\delta(u, v)$ denote the distance between vertices u and v in G. In this

[1] We say an event holds ***with high probability*** (w.h.p.) if it occurs with probability at least $1 - 1/n$.

SPAA '21, July 6–8, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8070-6/21/07.
https://doi.org/10.1145/3409964.3461822

Algorithm 1: parallel-centers(V, s, β):

```
A ← ∅, W ← V
while |W| > 0 do
    A′ ← Sample(W, s)
    A ← A ∪ A′
    for w ∈ W do in parallel
        C_A(w) ← {v ∈ V : δ(w, v) < δ(A, v)}
    W ← {w ∈ W : |C_A(w)| > βn/s}
return A
```

section, we allow G to be weighted, where $\delta(u, v)$ is the sum of weights on a shortest path from u to v, but in our algorithms for parallel network mapping, we assume G is unweighted, in which case $\delta(u, v)$ is the number of edges on a shortest path from u to v. For a subset $A \subseteq V$, let $\delta(A, v) = \min_{a \in A} \delta(a, v)$, and, for vertices $w, v \in V$, let $C_A(w)$ be the ***cluster*** of w and $B_A(v)$ be the ***bunch*** of v with respect to A, defined as follows:

$$C_A(w) = \{v \in V \mid \delta(w, v) < \delta(A, v)\}$$
$$B_A(v) = \{w \in V \mid \delta(w, v) < \delta(A, v)\}$$

Note that if $w \in A$, then $C_A(w) = \emptyset$. Also, observe that bunches and clusters are "inverses" of each other, in that $v \in C_A(w)$ if and only if $w \in B_A(v)$. In addition, notice that clusters and bunches can only shrink as we add vertices to A; that is, if $A' \subseteq A$, then $C_A(w) \subseteq C_{A'}(w)$ and $B_A(v) \subseteq B_{A'}(v)$, for all v and w in V.

Now, let $\beta \in [4, n)$, be a "parallelism" parameter and let $s \in [4 \ln n, n)$ be a "size" parameter. Define a subset, $A \subseteq V$, to be a set of (β, s)-***balanced centers*** if $|C_A(w)| \leq \beta n/s$, for all $w \in V$. Thorup and Zwick [25] give a sequential algorithm for finding a a set of $(4, s)$-balanced centers of expected size $O(s \log n)$. In Algorithm 1, we give a parallel algorithm for finding a set of (β, s)-balanced centers of size $O(s \log_\beta n)$ in $O(\log_\beta n)$ rounds w.h.p. Thus, the parameter β allows one to trade off parallel time and cluster size.

Our algorithm (Algorithm 1) takes a graph $G = (V, E)$ as input and initializes A, the eventual output of the algorithm, to be empty, and W, the set of nodes $v \in V$ where $|C_A(v)| > \beta n/s$, to be V. Then, we iteratively add Sample(W, s) to A, and replace W with vertices $w \in W$ such that $|C_A(w)| > \beta n/s$, in parallel. The function, Sample(W, s), returns W if $|W| \leq s$ and, otherwise, returns a set of elements from W such that each element in W is selected independently at random with probability $s/|W|$. We continue in this way until $W = \emptyset$.

Since the size of a cluster, $|C_A(w)|$, does not increase as we add more vertices to A, the set A returned by our algorithm is a set of (β, s)-balanced centers. Also, the Sample function returns a sample of size at most $2s$ with probability at least $1 - e^{-s/3}$, which holds with high probability across all iterations when $s \geq 4 \ln n$, by a standard Chernoff bound, e.g., see [21, p. 69]. Incidentally, Thorup and Zwick use the same Sample function, but don't bound its maximum size as we do. This high-probability upper bound for the sample size is not sufficient to achieve a high-probability bound, however, for the entire parallel graph clustering algorithm.

To that end, we define a parameter, α, as follows:

$$\alpha = \begin{cases} 2 & \text{if } \beta \leq ((4/3)e)^4 \\ (4/3)e\beta^{1/2} & \text{otherwise} \end{cases}$$

where $e \approx 2.71828$ is Euler's number. This definition of α is made so that we achieve high probability bounds for a range of β values.

Let W_i denote the set W at the beginning of iteration i, let A'_i denote the set A' that was added in iteration i, and let A_i denote the set A in this iteration, including the set, A'_i, i.e., $A_i = A_{i-1} \cup A'_i$, for $i = 1, 2, \ldots$, where $A_0 = \emptyset$. Say that iteration i is "bad" if the following inequality holds:

$$\sum_{w \in W_i} |C_{A'_i}(w)| > \frac{\alpha n |W_i|}{s},$$

and that otherwise it is "good". Note that, since W_i is a given for iteration i, whether iteration i is good or bad depends only on A'_i.

Lemma 2.1 (Thorup-Zwick [25], Lemma 3.2). *Let $W \subseteq V$, let $1 \leq s \leq n$, and let $A' \leftarrow$ Sample(W, s). Then, for every $v \in V$, $E[\,|B_{A'}(v) \cap W|\,] \leq |W|/s$.*

This implies the following:

$$E\left[\sum_{w \in W_i} |C_{A'_i}(w)|\right] = E\left[\sum_{v \in V} |B_{A'_i}(v) \cap W_i|\right] \leq \frac{n|W_i|}{s}.$$

Therefore, by Markov's inequality, an iteration is bad with probability at most $1/\alpha$.

Let W_{i+1} denote the set of vertices, W, whose clusters have size at least $\beta n/s$ at the end of a good iteration i. As $W_{i+1} \subseteq W_i$, and $C_{A_i}(w) \subseteq C_{A'_i}(w)$, for all $w \in V$, in a good iteration we have:

$$\frac{\beta n |W_{i+1}|}{s} \leq \sum_{w \in W_i} |C_{A_i}(w)| \leq \sum_{w \in W_i} |C_{A'_i}(w)| \leq \frac{\alpha n |W_i|}{s};$$

hence, $|W_{i+1}| \leq (\alpha/\beta)|W_i|$ in a good iteration. Thus, the number of good iterations of our algorithm is $O(\log_{(\beta/\alpha)} n)$, which is $O(\log_\beta n)$ for either choice of α. Moreover, because an iteration is good independent of whether any other iteration is good or bad, we may use Chernoff bounds to show that the number of bad iterations is also $O(\log_\beta n)$ w.h.p., for either value of α. This gives us the following:

Theorem 2.2. *Given an undirected, connected graph, $G = (V, E)$, we can find a set, A, of (β, s)-balanced centers of size $O(s \log_\beta n)$ in $O(\log_\beta n)$ parallel rounds w.h.p.*

For example, if $\beta = 4$, then we construct A to have size $O(s \log n)$ in $O(\log n)$ rounds, and if $\beta = n^\epsilon$, for some constant $0 < \epsilon < 1/2$, then we construct A to have size $O(s)$ in $O(1)$ rounds.

3 PARALLEL NETWORK MAPPING ALGORITHM FOR DEGREE-Δ GRAPHS

In this section, we provide an efficient parallel network mapping algorithm for an n-vertex connected, undirected, unweighted graph with maximum degree, Δ. Our parallel algorithm just uses distance queries, and we perform all the queries needed in our algorithm in a subroutine, Distances(v, V), which determines the distance,

Algorithm 2: Our parallel querying algorithm, estimated-parallel-centers(V, s, β), for finding a set of (β, s)-balanced centers A.

$A \leftarrow \emptyset$, $W \leftarrow V$
$T \leftarrow c_1(s/\beta)\log n$ // T = expected size of our estimation set; c_1 is a constant set in the analysis
while $|W| > 0$ **do**
  $A' \leftarrow$ Sample(W, s)
  $A \leftarrow A \cup A'$
  $R \leftarrow$ a random subset s.t. each $v \in V$ is chosen independently with probability T/n
  for *each* $r \in R$ **do in parallel**
    Distances(r, V)
  for $w \in W$ **do in parallel**
    $S(w) \leftarrow \{v \in R : \delta(w, v) < \delta(A, v)\}$ // $S(w) = C_A(w) \cap R$
  $W \leftarrow \{w \in W : |S(w)| > 2\beta T/s\}$ // that is, $|S(w)|(n/T) > 2\beta n/s$
return A

$\delta(v, w)$, for a given $v \in V$ and every other $w \in V$, by issuing a distance(v, w) query, for each $w \in V$ in parallel.

The key idea of our parallel network mapping algorithm is to first find a set, A, of (β, s)-balanced centers, using our parallel algorithm from the previous section, and then use this set of centers to compute a graph-theoretic Voronoi diagram [13, 18] for G, from which we may efficiently then perform a brute-force querying step for each Voronoi region. This initial center-finding step runs in $O(\log_\beta n)$ rounds and builds a set, A, of size $O(s \log_\beta n)$. One of the challenges in implementing this algorithm efficiently in parallel using distance queries is that we need to determine cluster sizes for all vertices in V in each iteration, which would take too many queries to compute exactly. So, rather than compute such sizes exactly, we instead build a global random set, R, which we use to approximate the size of each cluster. We can accurately estimate the size of the clusters for all the vertices in V with respect to a candidate set, A, of centers by appropriately choosing T, the size of set R. We give the details in Algorithm 2.

Lemma 3.1. *Our estimated-parallel-centers algorithm constructs a set, A, of $(3\beta, s)$-balanced centers of size $O(s \log_\beta n)$ in $O(\log_\beta n)$ parallel rounds, using $O(n(s/\beta)\log n \cdot \log_\beta n)$ distance queries w.h.p.*

This gives us the following corollaries.

Corollary 3.2. *If $\beta = n^\epsilon$, for some constant $0 < \epsilon < 1/2$, then estimated-parallel-centers(V, s, β) performs $O(n^{1-\epsilon} s \log n)$ distance queries in $O(1)$ rounds and outputs a set of $(O(n^\epsilon), s)$-balanced centers, A, of size $O(s)$ w.h.p.*

Corollary 3.3. *If $\beta = 4$, then estimated-parallel-centers(V, s, β) performs $O(ns \log^2 n)$ distance queries in $O(\log n)$ rounds and outputs a set of $(O(1), s)$-balanced centers, A, of size $O(s \log n)$ w.h.p.*

Now that we have defined and analyzed the function estimated-parallel-centers(V, s, β), let us next turn to our parallel algorithm for reconstructing a connected, unweighted graph, G, with maximum degree Δ. This algorithm (Algorithm 3, reconstruct(V)) takes as

Algorithm 3: Parallel network mapping

Function reconstuct(V):
  $A \leftarrow$ estimated-parallel-centers(V, s, β) // for each $a \in A$, we also get $N_2(a)$ and $\delta(a, v)$, for each $v \in V$
  for *each* $a \in A$ **do in parallel**
    Distances(a, V)
  for *each* $a \in A$ **do in parallel**
    $E_a \leftarrow$ Exhaustive-Query$(N_2(a))$
    for $b \in N_2(a)$ **do in parallel**
      Distances(b, V)
      $C_A(b) \leftarrow \{v \in V : \delta(b, v) < \delta(A, v)\}$
      $E_{a,b} \leftarrow$ Exhaustive-Query$(C_A(b))$
  return $\bigcup_{a \in A} \left(E_a \cup \bigcup_{b \in N_2(a)} E_{a,b} \right)$

input the vertex set V, and outputs, E, the set of edges of the graph $G = (V, E)$. Let $A \subseteq V$ be a set of ***centers***, which in our network mapping algorithm will come from a call to our estimated-parallel-centers(V, s, β) algorithm. For any center, $a \in A$, let $N_2(a)$ be the set of 2-***neighboring*** vertices of a in V, that is, $N_2(a) = \{v \in V : \delta(a, v) \leq 2\}$. Note that, since G has maximum degree, Δ, $|N_2(a)| \leq \Delta^2 + 1$. The following lemma shows that it is sufficient to consider the 2-neighbors and clusters defined by 2-neighbors, for each center $a \in A$, in order to cover all the edges in E.

Lemma 3.4. *Let (u, v) be an edge in E. Then there exists a center, $a \in A$, such that u and v are both in $N_2(a)$ or both in $C_A(b)$ for some $b \in N_2(a)$.*

Lemma 3.4 establishes the correctness for our parallel network mapping algorithm, which we give in detail in Algorithm 3.

Through a call to estimated-parallel-centers(V, s, β), we find a set of $(O(\beta), s)$-balanced centers, A. Next, we issue distance queries from each vertex $a \in A$ to be able to identify nodes in $N_2(a)$. Then, our mapping algorithm, reconstruct, constructs graph-theoretic Voronoi diagram for the centers in A, and then "branches out" from each center $a \in A$ by considering the nodes in $N_2(a)$ and the clusters defined by nodes in $N_2(a)$. Finally, after having done this Voronoi decomposition, our algorithm performs an exhaustive search in each cluster in parallel, in $O(1)$ rounds.

The following lemma characterizes the query complexity and round complexity.

Lemma 3.5. *Suppose estimated-parallel-centers(V, s, β) has query complexity, $Q(n)$, and round complexity, $R(n)$. Then Algorithm 3 reconstructs G using $O(Q(n) + \Delta^2(n + (\beta n/s)^2)s \log_\beta n)$ queries in $O(R(n))$ rounds, assuming Δ is $O(\sqrt{n})$.*

Proof. The round complexity of our reconstruction algorithm is dominated by the call to estimated-parallel-centers, since the rest of the algorithm can be done in $O(1)$ rounds. With respect to query complexity, since A has size $O(s \log_\beta n)$, we need at most $O(ns \log_\beta n)$ queries to find the distances from each $a \in A$ to each $v \in V$. Since each 2-neighbor set has size $O(\Delta^2)$, we need at most $|A|\Delta^2 n \in O(n\Delta^2 s \log_\beta n)$ queries to find the distances to each $v \in V$ from each vertex $b \in N_2(a)$ for each center $a \in A$. Further, for brute force querying in every cluster, we need at most $|A|(\Delta^4 + \Delta^2(\beta n/s)^2)$

queries, since each 2-neighbor set has size $O(\Delta^2)$ and each cluster has size $O(\beta n/s)$. Altogether, the total query complexity adds up to be $O(Q(n) + \Delta^2(n + (\beta n/s)^2)s \log_\beta n)$ queries, if Δ is $O(\sqrt{n})$. □

By plugging in our derived bounds for the estimated-parallel-centers algorithm, we get the following theorems.

Theorem 3.6. *One can reconstruct an n-vertex connected graph, $G = (V, E)$, with $O(\Delta^2 n^{3/2+\epsilon})$ distance queries in $O(1)$ rounds, for constant $0 < \epsilon < 1/2$, w.h.p.*

Proof. Set $\beta = n^\epsilon$ and $s = n^{1/2+\epsilon}$ in the estimated-parallel-centers(V, s, β) method. The claimed bounds follow from corollary 3.2 and lemma 3.5. □

The above theorem establishes an improvement over the brute-force querying algorithm for solving the parallel network mapping problem in $O(1)$ rounds. The next theorem establishes a bound for solving the parallel network mapping problem in $O(\log n)$ rounds.

Theorem 3.7. *One can reconstruct an n-vertex connected undirected graph, $G = (V, E)$, with $O(\Delta^2 n^{3/2} \log^{3/2} n)$ distance queries in $O(\log n)$ rounds w.h.p.*

Proof. Set $\beta = 4$ and $s = (n/\log n)^{1/2}$ in the estimated-parallel-centers(V, s, β) method. The claimed bounds then follow from corollary 3.3 and lemma 3.5. □

Even though this is a parallel algorithm, it improves the sequential query complexity of an algorithm in [19] for graph reconstruction via distance queries by an $O(\Delta \log^{1/2} n \cdot \log\log n)$ factor, while also improving the bounds to hold with high probability rather than in expectation. Also, while the algorithm by Kannan, Mathieu, and Zhou [19] needs to know the maximum degree, Δ, in order to reconstruct the graph, our algorithm is oblivious to Δ.

4 CONCLUSION

We have given algorithms for solving the parallel network mapping problem in parallel using $O(1)$ or $O(\log n)$ rounds. We have also given new, parallel implementations for graph clustering, which provide tradeoffs between the number of center vertices and the sizes of clusters. Even for sequential algorithms, this result may prove useful for applications where minimizing the number of center points is a primary optimization goal. For instance, one can apply our construction to the problems studied by Honiden, Houle, Sommer [18] for balancing graph-theoretic Voronoi diagrams to shave a $O(\log n)$ factor of the number of centers. It seems likely, therefore, that this result will have other applications as well.

ACKNOWLEDGMENTS

This article reports on work supported by NSF grant 1815073.

REFERENCES

[1] Mikkel Abrahamsen, Greg Bodwin, Eva Rotenberg, and Morten Stöckel. 2016. Graph Reconstruction with a Betweenness Oracle. In ***33rd Symp. on Theoretical Aspects of Computer Science (STACS) (LIPIcs, Vol. 47)***, Nicolas Ollinger and Heribert Vollmer (Eds.). Schloss Dagstuhl, 5:1–5:14. https://doi.org/10.4230/LIPIcs.STACS.2016.5

[2] Ramtin Afshar, Michael T. Goodrich, Pedro Matias, and Martha C. Osegueda. 2020. Reconstructing Binary Trees in Parallel. In ***32nd ACM Symp. on Parallelism in Algorithms and Architectures (SPAA)***, Christian Scheideler and Michael Spear (Eds.). ACM, 491–492. https://doi.org/10.1145/3350755.3400229

[3] Ramtin Afshar, Michael T. Goodrich, Pedro Matias, and Martha C. Osegueda. 2020. Reconstructing Biological and Digital Phylogenetic Trees in Parallel. In ***28th European Symposium on Algorithms (ESA) (LIPIcs, Vol. 173)***, Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders (Eds.). Schloss Dagstuhl, 3:1–3:24. https://doi.org/10.4230/LIPIcs.ESA.2020.3

[4] Noga Alon and Vera Asodi. 2005. Learning a Hidden Subgraph. ***SIAM J. Discrete Math.*** 18, 4 (2005), 697–712. https://doi.org/10.1137/S0895480103431071

[5] Noga Alon, Richard Beigel, Simon Kasif, Steven Rudich, and Benny Sudakov. 2004. Learning a Hidden Matching. ***SIAM J. Comput.*** 33, 2 (2004), 487–501. https://doi.org/10.1137/S0097539702420139

[6] Dana Angluin and Jiang Chen. 2006. Learning a Hidden Hypergraph. ***J. Mach. Learn. Res.*** 7 (2006), 2215–2236. http://jmlr.org/papers/v7/angluin06a.html

[7] Dana Angluin and Jiang Chen. 2008. Learning a hidden graph using O(log n) queries per edge. ***J. Comput. Syst. Sci.*** 74, 4 (2008), 546–556. https://doi.org/10.1016/j.jcss.2007.06.006

[8] Zuzana Beerliova, Felix Eberhard, Thomas Erlebach, Alexander Hall, Michael Hoffmann, Matús Mihalák, and L. Shankar Ram. 2006. Network Discovery and Verification. ***IEEE Journal on Selected Areas in Communications*** 24, 12 (2006), 2168–2181. https://doi.org/10.1109/JSAC.2006.884015

[9] Richard Beigel, Noga Alon, Simon Kasif, Mehmet Serkan Apaydin, and Lance Fortnow. 2001. An optimal procedure for gap closing in whole genome shotgun sequencing. In ***Fifth International Conference on Computational Biology (RECOMB)***, Thomas Lengauer (Ed.). ACM, 22–30. https://doi.org/10.1145/369133.369152

[10] Mathilde Bouvel, Vladimir Grebinski, and Gregory Kucherov. 2005. Combinatorial Search on Graphs Motivated by Bioinformatics Applications: A Brief Survey. In ***31st Int. Workshop on Graph-Theoretic Concepts in Computer Science (LNCS, Vol. 3787)***, Dieter Kratsch (Ed.). Springer, 16–27. https://doi.org/10.1007/11604686_2

[11] Sung-Soon Choi and Jeong Han Kim. 2010. Optimal query complexity bounds for finding graphs. ***Artificial Intelligence*** 174, 9 (2010), 551 – 569. https://doi.org/10.1016/j.artint.2010.02.003

[12] Luca Dall'Asta, Ignacio Alvarez-Hamelin, Alain Barrat, Alexei Vázquez, and Alessandro Vespignani. 2006. Exploring networks with traceroute-like probes: Theory and simulations. ***Theoretical Computer Science*** 355, 1 (2006), 6–24. https://doi.org/10.1016/j.tcs.2005.12.009

[13] Martin Erwig. 2000. The Graph Voronoi Diagram with Applications. ***Networks*** 36, 3 (2000), 156–163. https://doi.org/10.1002/1097-0037(200010)36:3<156::AID-NET2>3.0.CO;2-L

[14] Vladimir Grebinski. 1998. On the Power of Additive Combinatorial Search Model. In ***4th Int. Conf. on Computing and Combinatorics (COCOON) (LNCS, Vol. 1449)***, Wen-Lian Hsu and Ming-Yang Kao (Eds.). Springer, 194–203. https://doi.org/10.1007/3-540-68535-9_23

[15] Vladimir Grebinski and Gregory Kucherov. 1998. Reconstructing a Hamiltonian Cycle by Querying the Graph: Application to DNA Physical Mapping. ***Discret. Appl. Math.*** 88, 1-3 (1998), 147–165. https://doi.org/10.1016/S0166-218X(98)00070-5

[16] Vladimir Grebinski and Gregory Kucherov. 2000. Optimal Reconstruction of Graphs under the Additive Model. ***Algorithmica*** 28, 1 (2000), 104–124. https://doi.org/10.1007/s004530010033

[17] Jotun J Hein. 1989. An optimal algorithm to reconstruct trees from additive distance data. ***Bulletin of mathematical biology*** 51, 5 (1989), 597–603.

[18] S. Honiden, M. E. Houle, and C. Sommer. 2009. Balancing Graph Voronoi Diagrams. In ***Sixth International Symposium on Voronoi Diagrams***. 183–191.

[19] Sampath Kannan, Claire Mathieu, and Hang Zhou. 2018. Graph Reconstruction and Verification. ***ACM Trans. Algorithms*** 14, 4 (2018), 40:1–40:30. https://doi.org/10.1145/3199606

[20] Valerie King, Li Zhang, and Yunhong Zhou. 2003. On the complexity of distance-based evolutionary tree reconstruction. In ***14th ACM-SIAM Symposium on Discrete Algorithms (SODA)***. 444–453. http://dl.acm.org/citation.cfm?id=644108.644179

[21] Michael Mitzenmacher and Eli Upfal. 2017. ***Probability and Computing: Randomized Algorithms and Probabilistic Analysis*** (2/e ed.). Cambridge University Press.

[22] Lev Reyzin and Nikhil Srivastava. 2007. On the longest path algorithm for reconstructing trees from distance matrices. ***Inf. Process. Lett.*** 101, 3 (2007), 98–100. https://doi.org/10.1016/j.ipl.2006.08.013

[23] Sandeep Sen and V. N. Muralidhara. 2010. The Covert Set-Cover Problem with Application to Network Discovery. In ***4th Int. Workshop on Algorithms and Computation (WALCOM) (LNCS, Vol. 5942)***, Md. Saidur Rahman and Satoshi Fujita (Eds.). Springer, 228–239. https://doi.org/10.1007/978-3-642-11440-3_21

[24] Fabien Tarissan, Matthieu Latapy, and Christophe Prieur. 2009. Efficient Measurement of Complex Networks Using Link Queries. ***CoRR*** abs/0904.3222 (2009). arXiv:0904.3222 http://arxiv.org/abs/0904.3222

[25] Mikkel Thorup and Uri Zwick. 2001. Compact Routing Schemes. In ***13th ACM Symposium on Parallel Algorithms and Architectures (SPAA)***, Arnold L. Rosenberg (Ed.). 1–10. https://doi.org/10.1145/378580.378581