Dynamic algorithms for k-center on graphs[*]

Emilio Cruciani[†]   Sebastian Forster[†]   Gramoz Goranci[‡]   Yasamin Nazari[§]

Antonis Skarlatos[†]

Abstract

In this paper we give the first efficient algorithms for the k-center problem on dynamic graphs undergoing edge updates. In this problem, the goal is to partition the input into k sets by choosing k centers such that the maximum distance from any data point to its closest center is minimized. It is known that it is NP-hard to get a better than 2 approximation for this problem.

While in many applications the input may naturally be modeled as a graph, all prior works on k-center problem in dynamic settings are on point sets in arbitrary metric spaces. In this paper, we give a deterministic decremental $(2 + \epsilon)$-approximation algorithm and a randomized incremental $(4 + \epsilon)$-approximation algorithm, both with amortized update time $kn^{o(1)}$ for weighted graphs. Moreover, we show a reduction that leads to a fully dynamic $(2 + \epsilon)$-approximation algorithm for the k-center problem, with worst-case update time that is within a factor k of the state-of-the-art upper bound for maintaining $(1+\epsilon)$-approximate single-source distances in graphs. Matching this bound is a natural goalpost because the approximate distances of each vertex to its center can be used to maintain a $(2+\epsilon)$-approximation of the graph diameter and the fastest known algorithms for such a diameter approximation also rely on maintaining approximate single-source distances.

1   Introduction

Clustering is a key concept in data analysis that involves organizing 'similar' data into groups. One of the most fundamental and well-studied objectives is the *k-center objective*. Specifically, given a metric space with n points and a positive integer $k \leq n$, the goal of the *k-center problem* is to select k points, referred to as *centers*, such that the maximum distance of any point in the metric space to its closest center is minimized. It is known that k-center is NP-hard to approximate within a factor of $(2 - \epsilon)$ for any $\epsilon > 0$ [35]. Due to its popularity, k-center has been considered under several algorithmic frameworks, including approximation algorithms [35, 25, 34, 39, 22], parameterized complexity [23, 5], massive parallel computation (MPC) model [15, 8, 9], and beyond worst-case analysis [4], among others. This problem also serves as a testbed for developing fundamental algorithmic definitions and paradigms, which are then often applied to solving other variants of clustering objectives.

Clustering in the *dynamic setting* has received increasing attention in recent years. This line of work was initiated by Charikar et al. [17] who considered the problem of minimizing cluster diameters under the insertions of new points in an underlying metric space. Under both point insertions and deletions, the k-center problem was considered by Chan et al. [16], who achieved a $(2 + \epsilon)$-approximation in $O(k^2\epsilon^{-1} \log \Delta)$ amortized update time, where Δ is the aspect ratio of the metric space. Later on, the amortized update time was improved to $O(k\epsilon^{-1}\text{poly} \log(n, \Delta))$ by Bateni et al. [6]. This is almost optimal considering that even in the static setting, any algorithm for k-clustering problems (including k-center, k-median, and k-means) on point sets in arbitrary metric spaces that achieves any non-trivial approximation, must make at least $\Omega(nk)$ distance queries, and in turn must take $\Omega(nk)$ time [6]. These results spurred several follow-up works that studied other point sets-based clustering objectives such as k-means [20, 30], k-median [28], facility location [26, 29, 27, 12], and sum-of-radii [33] in the dynamic setting.

An important case of k-center clustering is when the input metric is induced by a graph G on n vertices and m edges. Naturally, any k-center algorithm that works with points in arbitrary metric spaces, can be applied on top of the graphical metric obtained by computing all-pairs shortest paths in G. However, the latter leads to slow

running times, especially since it could make a sparse graph G very dense. In the static setting, Thorup [43] gave a faster algorithm for the k-center problem in the graph setting, achieving a $(2 + \epsilon)$-approximation in $\tilde{O}(m\epsilon^{-1})$ time, where $\tilde{O}$ hides polylogarithmic factors in n and in the maximum edge-weight of the graph. This result was recently revisited by the work of Eppstein et al. [21] and even more recently by Abboud et al. [1] who gave a refined and simpler algorithm for k-center on graphs.

We note that graph clustering has also received attention in the machine learning community, albeit for the closely related objective of k-means [40]. They observe the computational challenges involving graphs (see also [2, Section 2.3]) and specifically the output sensitivity due to distance changes caused by edge updates.

Motivated by these developments, we study the fundamental problem of *dynamic k-center on graphs*. In comparison to the model with dynamic point sets in arbitrary metric spaces, we remark that the model with dynamic graphs is more challenging since (i) there is no guarantee of having oracle access to all-pairs shortest paths distances, and (ii) a single edge update may have a *global effect* on the underlying graph metric, forcing a large number of vertex pairs to change their shortest path distance. This is also why we cannot use other black-box approaches such as a distance oracle of the metric completion of the graph.

To that end, we ask the natural question of to what extent one can leverage the structure of graphical k-center in the context of obtaining faster algorithms for dynamic k-center on graphs:

Are there efficient algorithms for k-center on graphs undergoing edge updates?

1.1 Our Contribution In this paper, we answer the question in the affirmative. Our first contribution is a fully dynamic k-center algorithm that follows from prior work using a surprisingly simple trick.

THEOREM 1.1. *Given a weighted undirected graph $G = (V, E, w)$ subject to edge updates, an integer parameter $k \geq 1$, and a positive constant parameter $\epsilon \leq 1/2$, there are two fully dynamic algorithms for the k-center problem on graphs, that maintain a $(2 + \epsilon)$-approximation with the following guarantees (based on the current value of the matrix multiplication exponent):*
  1. *Deterministic algorithm with $O(kn^{1.529}\epsilon^{-2})$ worst-case update time, if G has uniform weights;*
  2. *Randomized algorithm, against an adaptive adversary, with $O(kn^{1.823}\epsilon^{-2})$ worst-case update time, if G has general weights.*

Both algorithms have preprocessing time $O(n^{2.373}\epsilon^{-2} \log \epsilon^{-1})$.

Note that our update time bounds match up to an $\tilde{O}(k)$ factor, those of the state-of-the-art fully-dynamic single-source distance approximation algorithms with multiplicative error $(1 + \epsilon)$ [14, 13]. Matching this bound is a natural goalpost for dynamic k-center algorithms maintaining the $(1 + \epsilon)$-approximate distance of each vertex to its closest center, because such distance approximations are sufficient to return a $(2 + \epsilon)$-approximation for graph diameter when $k = 1$ and the fastest known approach for this is to use a dynamic single-source distance approximation algorithm. Our algorithms—and to the best of our knowledge all related dynamic k-center algorithms on general metrics—do have this desirable property of maintaining the $(1 + \epsilon)$-approximate distance of each vertex to its closest center. Moreover, the previous result is a reduction to the problem of maintaining k-source approximate shortest paths in a fully dynamic setting; hence any improvement on the shortest paths algorithms directly improves the running time of our algorithms as well.

The above suggests that in order to achieve faster running times, we need to consider *partially dynamic algorithms* for the k-center problem on graphs, where edge updates are restricted to only edge insertions or edge deletions. In particular, the insertions-only algorithms (also known as the *incremental setting*) in the context of clustering are particularly well-motivated from a practical viewpoint. For example, real-world graphs such as co-authorship networks are incremental since the fact that two scientists co-authoring a research paper (almost) never changes over time. Our main result regarding the incremental setting is the following.

THEOREM 1.2. *Given a weighted undirected graph $G = (V, E, w)$ subject to edge insertions, an integer parameter $k \geq 1$, and a positive constant parameter $\epsilon < 1$, there is a randomized incremental $(4+\epsilon)$-approximation algorithm for the k-center problem on graphs. The algorithm is correct w.h.p. and has $kn^{o(1)}$ amortized update time over a sequence of $\Theta(m)$ updates.*

To complete the picture of partially dynamic algorithms, we also study the k-center problem on graphs undergoing edge deletions only, known as the *decremental* setting. Here, we obtain an algorithm that achieves a tight $(2 + \epsilon)$ approximation ratio.

THEOREM 1.3. *Given a weighted undirected graph $G = (V, E, w)$ subject to edge deletions, an integer parameter $k \geq 1$, and a positive constant parameter $\epsilon < 1$, there is a deterministic decremental $(2 + \epsilon)$-approximation algorithm for the k-center problem on graphs, with $kn^{o(1)}$ amortized update time over a sequence of $\Theta(m)$ updates.*

We note that the $n^{o(1)}$ factors in the running time are also due to using partially dynamic approximate single-source shortest paths (SSSP) algorithms, which is inherent in our bounds based on similar reasoning as in the fully dynamic setting.

Outline. In Section 2, we give an overview of our algorithm and also discuss the main challenges we face in dynamic graphs, as opposed to point sets. In Section 4, we review a well-known reduction that relates the k-center problem to finding a maximal independent set on a graph. This reduction is fundamental to our partially dynamic algorithms. Section 5 presents our primary technical contribution, showcasing the incremental algorithm of Theorem 1.2. Section 6 completes the partially dynamic picture by providing the decremental algorithm of Theorem 1.3. In Section 7, we explore the fully-dynamic setting, in which we use a different type of algorithm to prove Theorem 1.1. Unlike the reduction presented in Section 4, our approach here is based on the greedy algorithm of Gonzalez [25]. In addition to the set of centers, we can also answer other natural queries, such as the corresponding center for each vertex. We briefly discuss this in Appendix A.

2 Technical Overview

In this section, we give a high-level overview of our algorithms and discuss several technical challenges that we need to handle for dynamically maintaining k-center on graphs rather than on point sets.

Reduction to k-bounded maximal independent set on threshold graphs. We start by reviewing a known reduction from 2-approximate k-center to k-bounded maximal independent set (MIS) [34]. This reduction is also the basis of some of the fully dynamic k-center algorithms on point sets in arbitrary metric spaces [6, 16]. The idea for obtaining a 2-approximation is to guess the optimal value R^* of the k-center instance via a binary search, and return any maximal distance-$2R^*$ independent set M. It can be shown that M must be of size at most k. Recall that M is a maximal subset of vertices such that no two vertices $u, v \in M$ are within distance $2R^*$ of each other. The vertices of M then correspond to the centers of the k-center instance.

For utilizing this idea in the dynamic setting, rather than guessing the value of R^*, we maintain the MIS on each r-*threshold graph* G_r, for every *distance range* $r \leq (1+\epsilon)^i$, where $i \in [0, \log_{1+\epsilon}(nW)]$ and W is the maximum edge-weight of the graph. Here, by r-threshold graph we mean a graph such that there is an edge between two vertices if and only if they are within distance r. This general framework has been used in the dynamic k-center algorithms on point sets [6, 16]. Actually, they maintain a relaxation of the MIS, called a k-*bounded MIS*. The observation is that it is sufficient to either return an MIS of size at most k on each r-threshold graph G_r, or to simply report that there is an independent set of size at least $k + 1$ as a witness that the distance range r is not the correct guess.

Technical challenges in graphs vs point sets. As discussed before there are several important technical differences between the graph and point sets settings. The first difference is that, unlike the point sets settings, in graphs we do not have direct access to distances. Thus, we also need to maintain the appropriate distances simultaneously while the dynamic MIS is modified on the r-threshold graphs. For this, we would like to combine a dynamic MIS maintenance algorithm with partially dynamic (approximate) shortest paths algorithms. At a high-level, our goal is to maintain a k-bounded MIS M dynamically on each r-threshold graph G_r and at the same time maintain a dynamic (approximate) SSSP algorithm from a super-source which is connected to all the vertices in the dynamic set M. Hence the efficiency of the algorithm will depend on the number of times the set M is modified over all the updates. Equivalently, the efficiency will depend on the number of times the SSSP algorithm is restarted.

In both point sets and graph settings, we need to bound the *recourse*, where by recourse we mean the number of times a new center is introduced by the algorithm. However, we argue that a *stronger recourse guarantee* is needed for graphs. Recall that another important difference between these two settings is that in the point sets setting adding or removing points has a more local impact, whereas in a graph an update may impact the distances between many vertices. In other words, in the graph setting an edge update may *distort* the metric itself. This difference in graphs also requires our algorithm to have an overall recourse guarantee, as opposed to an amortized one that suffices for the point sets settings. Thus an amortized recourse of $\tilde{O}(k)$ *per update* is not enough, and we need the stronger guarantee that the recourse is $\tilde{O}(k)$ *over all updates*. More concretely, this total

bound on recourse will let us argue that in total we need to re-initialize a dynamic $(1 + \epsilon)$-SSSP algorithm $\tilde{O}(k)$ times from each center, and an amortized guarantee would not be enough for getting our desired update bound. We will see that maintaining this stronger recourse guarantee is more challenging in incremental settings than in decremental settings.

Note that this type of recourse guarantee has been studied in metric spaces under the name *consistent* clustering [38, 24, 37]. However, we would like to emphasize that the known sublinear bounds on the total recourse in the point sets setting do not carry over to the graph setting.

Decremental k-center on graphs. We can obtain a decremental $(2+\epsilon)$-approximation algorithm for the k-center problem on graphs, by maintaining a decremental $(1+\epsilon)$-SSSP algorithm from each added center in each of the $O(\log nW)$ r-threshold graphs. Bounding the recourse in the decremental setting is relatively straightforward based on the following observation. Whenever a new center forms a cluster due to a distance increase in a given r-threshold graph, it stays disjoint of other clusters throughout the algorithm and thus it stays a valid center. Furthermore, as soon as we get more than k centers, we move to the next distance range and so, the recourse is upper bounded by k on each r-threshold graph and by $O(k \log nW)$ overall. Hence the $(1 + \epsilon)$-SSSP algorithm is restarted at most $O(k \log nW)$ times in total. This combined with the time needed for maintaining partially dynamic $(1 + \epsilon)$-approximate SSSP leads to our desired $kn^{o(1)}$ amortized update time.

Incremental low recourse ruling sets. Bounding the recourse in the incremental setting is more challenging compared to the decremental setting, for the following reason. After an edge insertion in the input graph, a center c_1 of a cluster may come within distance r of an existing center c_2 of another cluster. In turn, this means that the two vertices c_1 and c_2 become neighbors in the r-threshold graph G_r. We cannot simply merge the corresponding clusters in some way and still maintain a 2-approximation, as some vertices in such a merged cluster would go beyond the desired distance range after each update. Hence we need a new technical idea to keep the recourse low. The idea is to maintain a small (i.e., of size $\tilde{O}(k)$) *dominating set* S on G such that, at a high-level, maintaining a maximal independent set on S will give us an *approximate* maximal independent set on G. More formally, by maintaining a k-bounded MIS M on the dominating set S, we can show that M is also a k-bounded $(2, 2)$-ruling set[1] in G. That is, a subset of vertices of M of size at most k such that: (i) the distance between any pair of vertices in M is at least 2, and (ii) for each vertex $u \in V$ there exists a vertex in M within distance 2. Introducing this small dominating set allows us to maintain a dynamic k-bounded maximal independent set M on the smaller subgraph S more efficiently, at the cost of losing a factor 2 in the approximation due to the fact that M is only a $(2, 2)$-ruling set on G. For maintaining such a dominating set, we use a recursive algorithm that maintains a union of hitting sets on a sequence of sparsified subgraphs of G. The hitting sets are obtained by a standard sampling procedure on the subgraphs corresponding to each recursive call. Informally, the sampling rate of the hitting sets is tuned depending on the densities of these subgraphs and the recursion continues until the remaining set of low degree vertices is sufficiently small. Since we have an incremental graph, the set of *low degree vertices defined based on a specific degree threshold* that are not covered by the hitting sets shrinks over time. This together with an observation regarding the degrees and the sampling, allows us to bound the recursion depth by $O(\log n)$.

Challenges of working with the r-threshold graphs. The high-level idea described above will give us an algorithm for maintaining a k-bounded $(2, 2)$-ruling set on an incremental graph in $\tilde{O}(k)$ amortized update time with an overall recourse of $\tilde{O}(k)$. Similar to the decremental algorithm, our goal is to maintain such a ruling set on all threshold graphs. This, combined with an incremental $(1 + \epsilon)$-SSSP algorithm, will lead to an incremental $(4 + \epsilon)$-approximation algorithm for the k-center problem on graphs. The main remaining challenge is that an edge insertion into the input graph G could lead to many edge insertions in the r-threshold graph. In turn, the density of the r-threshold graphs could be $n^2 \geq m$. To overcome this challenge, we do not explicitly store the r-threshold graphs, but again utilize a small dominating set for each r-threshold graph to ensure that only *relevant* insertions, i.e., those that cause a conflict, are processed and passed to the shortest paths algorithms. Overall, by bounding the number of centers added and the number of edges processed, we ensure that $\tilde{O}(k)$ incremental $(1 + \epsilon)$-SSSP algorithms are re-initialized in total, and this leads to an amortized update time of $kn^{o(1)}$.

---

[1] While our k-center algorithms work for weighted graphs, the ruling set subroutines always perform on unweighted graphs regardless on the input to the k-center problem.

3 Preliminaries

Graphs. Consider a weighted undirected graph $G = (V, E, w)$. We denote by $n = |V|$ the number of vertices, by $m = |E|$ the number of edges, and by W the maximum weight of an edge. Without loss of generality (WLOG), we assume that the minimum edge weight is equal to 1. Moreover, we assume that W is bounded by a polynomial in n (i.e., $W = O(\text{poly}(n))$).

For any two vertices $u, v \in V$, the *distance $d_G(u, v)$ between u and v* is the length of a shortest path from u to v in G. For a fixed subset of vertices $S \subseteq V$ and a vertex $v \in V$, the *distance $d_G(v, S)$ between v and S* is equal to $\min_{u \in S} d_G(v, u)$, namely the distance from v to its closest vertex in S. For a vertex $v \in V$, we denote by $N_G(v)$ the set of neighbors of v in G. A *subgraph H* of a graph G is a graph whose vertex set and edge set are subsets of the vertex set and edge set of G respectively. An *edge-subgraph* of G is a graph whose vertex set is the same as the vertex set of G and whose edge set is a subset of the edge set of G. For a subset of vertices $S \subseteq V$, the *induced subgraph $G[S]$* is the graph with vertex set S, whose edge set consists of all edges in E that have both endpoints in S. We also say that $G[S]$ is the subgraph induced by S. For a graph H, we denote by $V(H)$ the vertex set of H, and by $E(H)$ the edge set of H.

Consider now an unweighted undirected graph $G = (V, E)$. A *distance-α independent set M* is a subset of vertices such that the distance between any two vertices in M is strictly more than α. An *independent set (IS)* is a distance-1 independent set. An *(α, β)-ruling set* is a subset of vertices $M \subseteq V$ such that the distance between any two vertices in M is at least α, and the distance between any vertex in V and its closest vertex in M is at most β. A *maximal independent set (MIS)* is a $(2, 1)$-ruling set.

Dynamic setting. In the dynamic setting, the input graph G is subject to edge updates. Namely, edges can be inserted into G (edge insertions) and/or edges can be removed from G (edge deletions).

A *fully dynamic algorithm* is able to process both types of edge updates (i.e., edge insertions and edge deletions), while a *partially dynamic algorithm* is able to process only one type of edge updates (i.e., either edge insertions or edge deletions). In particular, an *incremental algorithm* can process only edge insertions and a *decremental algorithm* can process only edge deletions.

In our incremental algorithms, we assume that the updates are performed by an *oblivious adversary* who fixes the sequence of updates before the algorithm starts. Namely, the adversary cannot adapt the updates based on the choices of the algorithm during the execution. This is as opposed to an *adaptive adversary*, that instead we consider in the decremental and fully-dynamic settings.

In the incremental setting, let M be an independent set in G. Then for an edge insertion (u, v) in G, we say that the edge (u, v) causes a *conflict* in G when both of its endpoints u and v belong to M before the update.

We say that a dynamic algorithm has *amortized update time $u(n, m)$* if its total time spent for processing any sequence of ℓ updates is bounded by $\ell \cdot u(n, m)$.

k-center on graphs. We formally define the k-center problem as follows:

DEFINITION 3.1. (*k-CENTER ON GRAPH*) *Given a weighted undirected graph $G = (V, E, w)$ and an integer $k \geq 1$, the goal is to output a subset of vertices $S \subseteq V$ of size at most k, such that the value $\max_{v \in V} d_G(v, S)$ is minimized.*

Consider a *k-center instance* $(G = (V, E, w), k)$, which is the pair of the given input graph G and the integer k. For each choice of $S \subseteq V$, we define the radius $r := \max_{v \in V} d_G(v, S)$. The vertices of S are also called *centers*. For a fixed S with radius r, we define a *cluster* for every $c \in S$ containing all vertices within distance r from the center c. We denote by $R^* := \min_{|S| \leq k} \max_{v \in V} d_G(v, S)$ the optimal radius of the given instance, and by S^* any subset with radius R^* (i.e., $R^* = \max_{v \in V} d_G(v, S^*)$). For completeness we also discuss how we may be interested in answering other type of queries in Appendix A. In the dynamic setting, the input graph of the k-center instance is subject to edge updates.

Partially dynamic shortest paths algorithms. Through the paper, we heavily make use of the existing partially dynamic $(1 + \epsilon)$-approximate single-source shortest paths (SSSP) algorithms. In the decremental setting, we can use a deterministic algorithm.

THEOREM 3.1. (*DECREMENTAL $(1 + \epsilon)$-SSSP, [11]*) *Given a graph $G = (V, E)$ subject to edge deletions, a source $s \in V$, and a constant $\epsilon \in (0, 1)$, there is a deterministic algorithm that maintains $(1 + \epsilon)$-approximate shortest paths from s in total update time $m^{1+o(1)}$.*

In the incremental setting, we can use the following randomized partially dynamic algorithm.

THEOREM 3.2. (INCREMENTAL $(1 + \epsilon)$-SSSP, [32, 18, 36]) *Given a graph $G = (V, E)$ subject to edge insertions, a source $s \in V$, and a constant $\epsilon \in (0, 1)$, there is a randomized algorithm (against an oblivious adversary) that maintains $(1 + \epsilon)$-approximate shortest paths from s in total update time $m^{1+o(1)}$.*

The incremental algorithm is not explicitly stated but follows from similar algorithms as the decremental settings such as [32, 18, 36]. In Appendix C, we give a brief sketch of how one can adapt these results to the incremental setting, but the details of this algorithm are beyond the scope of this paper.

4 Reduction from 2-approximate k-center to k-bounded ruling set

It is well-know that the k-center problem can be reduced to the problem of finding an MIS on a graph. This reduction was first given by Hochbaum and Schmoys [34], in order to get a 2-approximation algorithm, and also it has been used by [16, 6] for the fully dynamic k-center problem on point sets in arbitrary metric spaces. In particular, it is sufficient to solve a weaker version of the MIS problem, where we only need to return an MIS of size at most k, or report that there is an independent set of size at least $k + 1$. Formally, we define this problem based on an (α, β)-ruling-set, as follows. A similar definition was also given in [6] for the MIS. Recall that an MIS is a $(2, 1)$-ruling set.

DEFINITION 4.1. (k-BOUNDED (α, β)-RULING SET PROBLEM) *Given an unweighted undirected graph $G = (V, E)$, an integer $k \geq 1$, and parameters α, β such that $\beta \geq \alpha - 1 \geq 0$, the k-bounded (α, β)-ruling set problem asks to either return an (α, β)-ruling set of size at most k, or to report that there is a distance-$(\alpha - 1)$ independent set of size at least $k + 1$.*

The reduction solves the k-bounded (α, β)-ruling set problem on the following type of graphs.

DEFINITION 4.2. (r-THRESHOLD GRAPH) *Given a weighted graph $G = (V, E, w)$ and a parameter $r > 0$, the r-threshold graph $G_r = (V, E_r)$ is defined as the graph with vertex set V and edge set $E_r = \{(u, v) \in V \times V : d_G(u, v) \leq r\}$.*

In other words, the r-threshold graph G_r connects all pairs of vertices that are within distance r in G. Observe that the r-threshold graph G_r is unweighted.

The next lemma is an adjustment of the reduction of Hochbaum and Schmoys [34] to Definition 4.1. The proof is deferred to Appendix B.

LEMMA 4.1. *Consider a k-center instance $(G = (V, E, w), k)$, and a positive constant parameter ϵ. Then by running a k-bounded $(2, \beta)$-ruling set algorithm on the r-threshold graph G_r, for each $r \in \{(1 + \epsilon)^i \mid (1 + \epsilon)^i \leq nW, i \in \mathbb{N}\}$, we can find a $2\beta(1 + \epsilon)$-approximate solution for the k-center instance.*

For the sake of efficiency, in the dynamic setting we do not handle r-threshold graphs, but rather an approximation of them. For this reason, we generalize the previous lemma as follows.

LEMMA 4.2. *Consider a k-center instance $(G = (V, E, w), k)$, constant positive parameters ϵ, ϵ', $r \geq 0$, and $\beta \geq 1$. Let $r' := (1 + \epsilon')r$ and consider the threshold graphs G_r and $G_{r'}$. Suppose that there is an algorithm $\mathcal{A}$ such that, given G, r, ϵ', β,*
- *either reports that there is an independent set in G_r of size at least $k + 1$,*
- *or runs a k-bounded $(2, \beta)$-ruling set algorithm $\mathcal{B}$ on an edge-subgraph H of $G_{r'}$ with the following condition: whenever $\mathcal{B}$ reports that there is an independent set in H of size at least $k + 1$, then there is an independent set in G_r of size at least $k + 1$.*

Then, by running $\mathcal{A}$ with input G, r, ϵ', β, for each $r \in \{(1+\epsilon)^i \mid (1+\epsilon)^i \leq nW, i \in \mathbb{N}\}$, we can find a $2\beta(1+\epsilon)(1+\epsilon')$-approximate solution for the k-center instance.

As already stated, the previous lemma is a generalization of Lemma 4.1. In fact, observe that in the definition of a k-bounded $(2, \beta)$-ruling set problem, we are allowed to report that there is an independent set of size at least $k + 1$. Thus by setting $H = G_r$ and $\epsilon' = 0$ in Lemma 4.2, we get Lemma 4.1 as a corollary.

Before proving Lemma 4.2, we state two auxiliary results that will be useful. Their proofs are deferred to Appendix B.

LEMMA 4.3. *Consider a k-center instance $(G = (V, E, w), k)$ with optimal radius R^*. Then for each $r \geq 2R^*$ and for every $\beta \geq 1$, it holds that every $(2, \beta)$-ruling set in the r-threshold graph G_r is of size at most k.*

OBSERVATION 4.1. *Consider a k-center instance $(G = (V, E, w), k)$ with optimal radius R^*, and let G_r be the r-threshold graph where $r = 2R^*$. Then, there is no independent set in G_r of size at least $k + 1$.*

We proceed now with the proof of Lemma 4.2.

Proof. [Proof of Lemma 4.2] Let $\hat{r}$ be the smallest $r \in \{(1 + \epsilon)^i \mid (1 + \epsilon)^i \leq nW, i \in \mathbb{N}\}$ such that algorithm $\mathcal{A}$ returns a $(2, \beta)$-ruling set M_r of size at most k in an edge-subgraph H of $G_{r'}$, where $r' = (1 + \epsilon')r$. Also, let $\hat{r}' = (1 + \epsilon')\hat{r}$ and let $S := M_{\hat{r}}$ be the solution we return for the k-center instance.

Since H is a subgraph of $G_{\hat{r}'}$, then for every edge $(u, v) \in E(H)$, the distance between u and v in G is at most $\hat{r}'$. Hence, as $M_{\hat{r}}$ is a $(2, \beta)$-ruling set in H, then every vertex is within distance $\beta\hat{r}'$ from its closest center in G. Thus, the returned solution S has radius at most $\beta\hat{r}'$.

We show now that $\hat{r}$ is at most $2(1 + \epsilon)$ times larger than R^*. Based on Observation 4.1, for the fixed choice of $r = 2R^*$, algorithm $\mathcal{A}$ always returns a $(2, \beta)$-ruling set M_r in H of size at most k. By definition of $\hat{r}$, and since the possible values of r are powers of $(1 + \epsilon)$, we have that $\hat{r} \leq 2(1 + \epsilon)R^*$. Therefore, the radius of the returned solution S is at most $2\beta(1 + \epsilon)(1 + \epsilon')R^*$. $\square$

5 Incremental k-center on graphs

We start by recalling the concept of *dominating set*, which we will exploit throughout this section.

DEFINITION 5.1. (DOMINATING SET) *Given a graph $G = (V, E)$, a dominating set $S \subseteq V$ on G is a subset of vertices such that each vertex of G is either in S or has a neighbor in S.*

In particular, we will make use of the following observation.

OBSERVATION 5.1. *Let $G = (V, E)$ be a graph, let S be a dominating set on G, and let M be a (α, β)-ruling set on $G[S]$. Then M is a $(\alpha, \beta + 1)$-ruling set on G.*

In this section, we first develop an incremental algorithm for the k-bounded $(2, 2)$-ruling set problem by finding a small dominating set S and maintaining a k-bounded $(2, 1)$-ruling set in $G[S]$. The idea is to use the reduction of Lemma 4.1 with this algorithm, to solve the incremental k-center problem. In the reduction though, notice that we need to maintain an incremental k-bounded $(2, 2)$-ruling set algorithm on r-threshold graphs, which is more challenging.

Subsequently, we relax our requirements and develop an efficient incremental k-bounded $(2, 2)$-ruling set algorithm that works on approximate versions of r-threshold graphs. Thus, we apply Lemma 4.2 instead of Lemma 4.1, at the cost of an extra $(1 + \epsilon)$ factor in the approximation ratio of the k-center.

5.1 Incremental k-bounded $(2, 2)$-ruling set algorithm
We begin by describing how to detect a small dominating set S on an incremental graph G, and maintain a k-bounded $(2, 1)$-ruling set on the subgraph induced by S.

THEOREM 5.1. *Given a graph $G = (V, E)$ subject to edge insertions, and an integer $k \geq 1$, there is a randomized incremental algorithm which*
- *either reports that there is an independent set in G of size at least $k + 1$, and this is correct w.h.p.,*
- *or finds a dominating set S of size $\tilde{O}(k)$ on G and maintains a k-bounded $(2, 1)$-ruling set in $G[S]$.*

Notice that based on the definitions of dominating set and k-bounded (α, β)-ruling set problem (i.e., Definition 5.1 and Definition 4.1), the algorithm of Theorem 5.1 solves the incremental k-bounded $(2, 2)$-ruling set problem in G. Before describing the algorithm we review two existing algorithms tools. First tool is the following folklore hitting set claim (e.g., see [3], also widely used in decremental settings against an oblivious adversary).

LEMMA 5.1. *Given a graph $G = (V, E)$ and a threshold $\gamma \geq 1$, let S be the set obtained by sampling each vertex independently with probability $\min(c\ln(n)/\gamma, 1)$, for a constant $c > 1$. Then, with probability at least $1 - n^{-(c-1)}$, every vertex of degree more than γ has at least one neighbor in S.*

As noted, e.g., in [41], even though Lemma 5.1 refers to a static graph, it is easy to see that it holds for partially dynamic graphs. Since we are assuming an oblivious adversary, the choice of the random set S is independent of the graph. This and the fact that we have at most $O(n^2)$ versions of the graph in the incremental setting, let us bound the overall probability via a straightforward union bound, and the failure probability is at most $n^{-(c-3)}$.

Second tool, is a fully dynamic k-bounded $(2,1)$-ruling set algorithm with the following guarantees. This algorithm is a trivial extension of any fully dynamic MIS algorithm that returns explicitly the MIS. For that reason, we can either use the MIS algorithm of Behnezhad et al. [7], or the algorithm of Chechik and Zhang [19].

THEOREM 5.2. *Given a graph $G = (V, E)$ subject to edge updates, there is a fully dynamic k-bounded $(2,1)$-ruling set algorithm with $\tilde{O}(1)$ amortized update time.*

Proof. The algorithm of Behnezhad et al. [7] maintains an MIS M under edge updates, in $\tilde{O}(1)$ amortized update time. Recall that an MIS is a $(2,1)$-ruling set. Thus at any moment, if the size of M is at least $k+1$, we report that there is an independent set in G of size at least $k+1$, otherwise we return the set M.   □

Overview of the algorithm. A pseudocode of the algorithm of Theorem 5.1 is provided in Algorithm 1. The algorithm consists of two phases. Roughly speaking, the first phase either detects a dominating set S or reports that there is an independent set in G of size at least $k+1$. The second phase starts when such a dominating set S is detected and is only responsible for maintaining an incremental k-bounded $(2,1)$-ruling set in $G[S]$.

Algorithm 1 k-BOUNDED $(2,2)$-RULING SET

// *In the preprocessing $i = 0$, $L_0 = V$, and k-BOUNDED-RULING-SET() is called with no edge (note that Line 16 where the edge is actually used cannot be reached during the preprocessing)*
// *The index i and the sets L_i, S_i for every i are global*

Procedure k-BOUNDED-RULING-SET(u, v)**:**
  if $|L_i| > 4k$ **then** // *first phase*
    if $i = 0$ *or* $|L_i| \leq \frac{|L_{i-1}|}{2}$ **then** // *recursive sampling*
      $i \leftarrow i + 1$
      $\gamma_i \leftarrow \frac{|L_{i-1}|}{2k} - 1$
      $S_i \leftarrow$ sample vertices of L_{i-1} independently with prob. $\min(10 \ln(n)/\gamma_i, 1)$ // *Lemma 5.1*
      $L_i \leftarrow \{x \in L_{i-1} : N_{G[L_{i-1}]}(x) \cap S_i = \emptyset\}$
      k-BOUNDED-RULING-SET(u, v)
    else
      report there is an independent set in G of size at least $k+1$ // *Lemma 5.2*
  else // *second phase*
    if $\mathcal{B}$ *is not initialized* **then** // $\mathcal{B}$ *is dynamic k-bounded $(2,1)$-ruling set algorithm (Theorem 5.2)*
      $d \leftarrow i$
      $S \leftarrow \bigcup_{j=1}^{d} S_j \cup L_d$ // S *is a dominating set (Lemma 5.4)*
      $\mathcal{B}.initialize(G[S])$
    else if $u \in S$ *and* $v \in S$ **then**
      $\mathcal{B}.update(G[S], u, v)$

Procedure INSERT(u, v)**:**
  $G \leftarrow (V, E \cup \{u, v\})$
  if $u \in L_i$ *and* $\exists j \leq i$ *s.t.* $v \in S_j$ *(resp., $v \in L_i$ and $\exists j \leq i$ s.t. $u \in S_j$)* **then**
    $L_i \leftarrow L_i \setminus \{u\}$ *(resp., $L_i \leftarrow L_i \setminus \{v\}$)*
  k-BOUNDED-RULING-SET(u, v)

In the first phase, the algorithm iteratively adds vertices to the dominating set by recursively sampling a sequence of hitting sets. In each recursive call $i \geq 1$, we set a threshold $\gamma_i = \frac{|L_{i-1}|}{2k} - 1$ and construct two sets S_i and L_i. The set S_i is obtained by sampling each vertex of L_{i-1} independently with probability $\min(c \ln(n)/\gamma_i, 1)$, for a sufficiently large constant c. Roughly speaking the set S_i is the hitting set of the vertices with degree more than γ_i in $G[L_{i-1}]$. Moreover, the set S_i is w.h.p. small in size due to the sampling procedure. The set L_i is

constructed as the subset of vertices of L_{i-1} that do not belong to S_i and do not have a neighbor in S_i. Given the property of the hitting set S_i, the set L_i contains w.h.p. only vertices with degree at most γ_i in $G[L_{i-1}]$. The recursion starts with $L_0 = V$ and it ends when $|L_i| \leq 4k$.

In the i_{th} recursive call, if the size of L_i is at most $|L_{i-1}|/2$ then a new recursive call begins. This implies that the depth of the recursion over all updates is bounded by $O(\log n)$. On the other hand, if the size of the set L_i is greater than $|L_{i-1}|/2$, the recursion pauses and the algorithm reports that there is an independent set in G of size at least $k+1$. In this case i may not be the final recursive call of the algorithm, because on future updates the algorithm can possibly continue the recursion.

Whenever an edge (u,v) is inserted to G during the first phase, we update the set L_i by removing from it one of the endpoints if the other one is contained in S_i. Observe that edge insertions will eventually shrink the size of L_i, forcing the recursion to continue.

The second phase begins when the size of L_i is at most $4k$, and at this moment the recursion ends. We denote by d the index of the last recursive call in the first phase, and let $S := \bigcup_{j=1}^{d} S_j \cup L_d$ be the union of the hitting sets of all recursive calls and of the set L_d. Notice that S can be constructed explicitly, during the first phase of the algorithm. Also, in the updates following the second phase we never re-enter the first phase, and thus the set S is not modified anymore. We show in the analysis, that even though the set S is random, it is always a dominating set on G.

At the beginning of the second phase, the dynamic k-bounded $(2,1)$-ruling set algorithm $\mathcal{B}$ of Theorem 5.2 is initialized on $G[S]$. Whenever an edge (u,v) is inserted to G during the second phase, the algorithm simply forwards the update to $\mathcal{B}$ if $u, v \in S$, and does nothing otherwise.

Proof of Theorem 5.1. The analysis consists of three claims. First, we prove that whenever the algorithm reports that there is an independent set in G of size at least $k+1$, this is correct with high probability. Second, we show that there are $O(\log n)$ recursive calls and that the size of S is $\tilde{O}(k)$. Third, we prove that the set S detected by the algorithm is indeed a dominating set on G.

LEMMA 5.2. *At any stage of the algorithm with $i \geq 1$, if $|L_i| > \frac{|L_{i-1}|}{2}$, then w.h.p. there is an independent set in G of size at least $k+1$.*

Proof. The threshold γ_i is set to $\frac{|L_{i-1}|}{2k} - 1$, and S_i is obtained by sampling each vertex of L_{i-1} independently with probability $\min(c\ln(n)/\gamma_i, 1)$, for a sufficiently large constant c. Then by Lemma 5.1, it holds that w.h.p. every vertex in L_{i-1} of degree more than γ_i in the induced subgraph $G[L_{i-1}]$ has a neighbor in S_i. Hence, w.h.p. every vertex in L_i is of degree at most γ_i in $G[L_{i-1}]$. As $G[L_i]$ is a subgraph of $G[L_{i-1}]$, w.h.p. every vertex of $G[L_i]$ is of degree at most γ_i in $G[L_i]$ as well.

Since w.h.p. the maximum degree in $G[L_i]$ is bounded by γ_i, for any $T \subseteq L_i$ such that $|T| = k$ (note that $|L_i| > 4k$), it holds that w.h.p. the number of vertices which are either in T or have a neighbor in T is at most $k(\gamma_i + 1) \leq \frac{|L_{i-1}|}{2}$. By assumption we have that $|L_i| > \frac{|L_{i-1}|}{2}$, and so T cannot be a maximal independent set. So it holds that w.h.p. there is an independent set in $G[L_i]$ of size at least $k+1$. In turn, as $G[L_i]$ is an induced subgraph of G, it holds that w.h.p. there is an independent set in G of size at least $k+1$ as well. $\square$

LEMMA 5.3. *Over the sequence of updates, there are $d = O(\log n)$ recursive calls. Moreover, the size of S is w.h.p. $O(k \log^2 n)$.*

Proof. Regarding the first claim, at every recursive call $i \geq 1$, it holds that $|L_i| \leq \frac{|L_{i-1}|}{2}$. Initially we have that $|L_0| = n$, and so, the depth of the recursion is $d = O(\log n)$.

Regarding the second claim, at each recursive call $i \geq 1$, we sample each vertex of L_{i-1} independently with probability $\min(c\ln(n)/\gamma_i, 1)$, for a sufficiently large constant c. Recall that $\gamma_i = \frac{|L_{i-1}|}{2k} - 1$ and note that the sampling takes place only if $|L_{i-1}| > 4k$. Then

$$\mathbb{E}[|S_i|] \leq |L_{i-1}| \cdot \frac{c\ln(n)}{\gamma_i} = |L_{i-1}| \cdot \frac{c\ln(n)}{\frac{|L_{i-1}|}{2k} - 1} = |L_{i-1}| \cdot \frac{2k \cdot c\ln(n)}{|L_{i-1}| - 2k} = \frac{2k \cdot c\ln(n)}{1 - 2k/|L_{i-1}|} < 4k \cdot c\ln(n).$$

Moreover note that $|L_d| \leq 4k$. Therefore, by linearity of expectation it holds that $\mathbb{E}[|S|] = |L_d| + \sum_{i=1}^{d} \mathbb{E}[|S_i|] = O(k \log^2 n)$. Finally, since $|S|$ is a sum of independent Poisson trials, a standard application of a Chernoff's bound implies that $|S| = O(k \log^2 n)$ with high probability. $\square$

LEMMA 5.4. *The set S is a dominating set on G.*

Proof. For a fixed vertex $v \in V \setminus S$, let i be the minimum index such that $v \notin L_i$. Note that such an index exists since $v \in V = L_0$ and so $i \geq 1$. If $v \notin L_i$, then by definition of L_i, vertex v must have a neighbor in S_i. Therefore, every vertex $v \in V \setminus S$ has a neighbor in S.

Let $\mathcal{A}$ be a k-bounded (α, β)-ruling set in $G[S]$. If $\mathcal{A}$ reports that there is a distance-$(\alpha - 1)$ independent set in $G[S]$ of size at least $k + 1$, then $\mathcal{A}$ correctly reports that there is a distance-$(\alpha - 1)$ independent set in G as well. This is because $G[S]$ is an induced subgraph of G, and so, any distance-$(\alpha - 1)$ independent set in $G[S]$ is also a distance-$(\alpha - 1)$ independent set in G.

Otherwise, $\mathcal{A}$ returns an (α, β)-ruling set M of size at most k. Then for any vertex $v \in S$, we have that v is of distance at most β from its closest vertex in M. Therefore, since every vertex $v \in V \setminus S$ has at least one neighbor in S, we have that every vertex of G is of distance at most $\beta + 1$ from its closest vertex in M. Thus, the set M is an $(\alpha, \beta + 1)$-ruling set in G of size at most k, and so the claim follows. $\qquad\square$

5.2 Incremental k-bounded $(2, 2)$-ruling set on G_r Our goal here is to extend Theorem 5.1 to r-threshold graphs so that we can apply Lemma 4.1 and maintain an incremental k-center solution. At a high level, our intention is to simulate the two phases of Algorithm 1 on an r-threshold graph G_r. Recall by Definition 4.2 that for any pair of vertices $u, v \in V \times V$, there is an edge in G_r if and only if the distance between u and v in G is at most r. The main challenges in the incremental setting are the following ones.

- We cannot afford to explicitly maintain all the edges of G_r in the incremental setting, because it is very expensive to run an incremental all-pairs shortest paths algorithm on G.
- A single edge insertion in the original graph G could introduce multiple edge insertions in the r-threshold graph G_r.

Note that Algorithm 1 does not need access to all edges of G_r in order to process G_r. Thus, our aim is to describe how to maintain all the necessary information that Algorithm 1 needs, so as to run with implicit input the r-threshold graph G_r.

To extract the relevant information for the r-threshold graph G_r, we make use of the incremental $(1+\epsilon)$-SSSP algorithm of Theorem 3.2. We note that using partially dynamic *exact* SSSP algorithms for this step would be too slow for our purposes, as even in unweighted graphs we would require $\Omega(mr)$ time and r could be very large (i.e., as big as n). Consequently, rather than explicitly maintaining G_r, we maintain an edge-subgraph H of the r'-threshold graph $G_{r'}$, with $r' := (1 + \epsilon)r$. However, whenever the algorithm reports that there is an independent set of size at least $k + 1$ in H, we guarantee that this is also true for the r-threshold graph G_r.

We exploit the fact that Algorithm 1 guarantees that the size of the dominating set is small. Hence, as only the edges in the graph induced by the dominating set are needed, we argue based on Lemma 5.3 that during the whole second phase of the algorithm we maintain $\tilde{O}(k)$ incremental $(1 + \epsilon)$-approximate SSSP instances. Furthermore again by Lemma 5.3, we argue that during the whole first phase of the algorithm, we maintain $\tilde{O}(1)$ incremental $(1 + \epsilon)$-approximate SSSP instances. As a result, in total we maintain only $\tilde{O}(k)$ incremental $(1 + \epsilon)$-approximate SSSP instances over the course of the algorithm, and this is the main ingredient for the efficiency of the algorithm.

THEOREM 5.3. *Consider a graph $G = (V, E, w)$ subject to edge insertions, an integer $k \geq 1$, a positive parameters r and a positive constant $\epsilon < 1$. Let $r' := (1 + \epsilon)r$ and consider the threshold graphs G_r and $G_{r'}$. Then, there is a randomized algorithm which:*

- *either reports that there is an independent set in G_r of size at least $k + 1$, and this is correct w.h.p.,*
- *or finds a dominating set $S \subseteq V$ of size $\tilde{O}(k)$ in an edge-subgraph H of $G_{r'}$ and runs a k-bounded $(2, 1)$-ruling set algorithm $\mathcal{B}$ on $H[S]$ with the following condition: whenever $\mathcal{B}$ reports that there is an independent set in H of size at least $k + 1$, then there is an independent set in G_r of size at least $k + 1$.*

The total update time of the algorithm is w.h.p. $km^{1+o(1)}$.

By definition of a dominating set (see Definition 5.1), the next corollary immediately follows.

COROLLARY 5.1. *Consider the setting of Theorem 5.3. Then, there is a randomized algorithm which:*

- *either reports that there is an independent set in G_r of size at least $k + 1$, and this is correct w.h.p.,*
- *or runs a k-bounded $(2, 2)$-ruling set algorithm $\mathcal{B}$ on an edge-subgraph H of $G_{r'}$ with the following condition: whenever $\mathcal{B}$ reports that there is an independent set in H of size at least $k + 1$, then there is an independent set in G_r of size at least $k + 1$.*

The total update time of the algorithm is w.h.p. $km^{1+o(1)}$.

In the following, we describe the algorithm of Theorem 5.3, which is an adaptation of Algorithm 1 on approximate threshold graphs.

Overview of the algorithm. A pseudocode of the algorithm is provided in Algorithm 2. We adapt Algorithm 1 to process the r-threshold graph G_r implicitly.

Algorithm 2 k-BOUNDED $(2,2)$-RULING SET ON G_r

// *In the preprocessing $i = 0$, $L_0 = V$, and k-BOUNDED-RULING-SET() is called with no edge*

$r' \leftarrow (1 + \epsilon)r$

Procedure k-BOUNDED-RULING-SET(u, v):

  if $|L_i| < 4k$ **then** // *first phase*

    if $i = 0$ *or* $|L_i| \leq \frac{|L_{i-1}|}{2}$ **then**

      $i \leftarrow i + 1$

      $\gamma_i \leftarrow \frac{|L_{i-1}|}{2k} - 1$

      $S_i \leftarrow$ sample vertices of L_{i-1} independently with prob. $\min(10 \ln(n)/\gamma_i, 1)$

      $S^{(i)} \leftarrow \bigcup_{j=1}^{i} S_j$

      $\mathcal{A}_{S^{(i)}}.initialize(G, S^{(i)})$ // $\mathcal{A}_{S^{(i)}}$ *is incremental approx. SSSP algorithm*

                                            // $\mathcal{A}_{S^{(i)}}$ *provides approx. distance* $\delta_{S^{(i)}}(\cdot)$

      $L_i \leftarrow \{x \in V : \delta_{S^{(i)}}(x) > r'\}$

      k-BOUNDED-RULING-SET(u, v)

    else

      report there exists an IS in G_r of size at least $k + 1$

  else // *second phase*

    if $\mathcal{B}$ *not initialized* **then** // $\mathcal{B}$ *is dynamic k-bounded $(2,1)$-ruling set algorithm (Theorem 5.2)*

      $d \leftarrow i$

      $S \leftarrow \bigcup_{j=1}^{d} S_j$

      for $s \in S$ **do**

        $\mathcal{A}_s.initialize(G, s)$ // $\mathcal{A}_s$ *is incremental approx. SSSP algorithm*

      $H \leftarrow (S, E_S)$ s.t. $E_S \leftarrow \{(u, v) \in S \times S : \delta_u(v) \leq r'\}$

      $\mathcal{B}.initialize(H)$

    else

      for $s \in S$ **do**

        $\mathcal{A}_s.insert(G, u, v)$

      while $\exists a, b \in S$ *s.t.* $(a, b) \notin E_S$ *and* $\delta_a(b) \leq r'$ **do**

        $\mathcal{B}.insert(H, a, b)$

Procedure INSERT(G, u, v):

  $\mathcal{A}_{S^{(i)}}.insert(G, u, v)$

  if $\exists x \in L_i$ *s.t.* $\delta_{S^{(i)}}(x) \leq r'$ **then**

    $L_i \leftarrow L_i \setminus x$

  k-BOUNDED-RULING-SET(u, v)

Consider a recursive call $i \geq 1$ of Algorithm 1 during the first phase. The sampling step for obtaining the set S_i does not need access to the edges of the input graph, but only to the vertices of the input graph. Thus, each hitting set S_i can be explicitly constructed. In turn, the union of the sampled sets $S^{(i)} = S_1 \cup \cdots \cup S_i$ is explicitly constructed as well.

The next step of Algorithm 1 is to compute the size of L_i, and decide how to proceed with the recursion depending on the sizes of L_{i-1} and L_i. A simulation of Algorithm 1 on G_r, would construct the set L_i as the set of vertices which are of distance more than r in G from the closest vertex in S_i. Nevertheless, as we use an approximate SSSP algorithm, we construct the set L_i in a slightly different way as follows. In the beginning of the recursive call, we set $S = S^{(i)}$ and $L_i = L_{i-1} \setminus S_i$. At this point, we maintain the incremental $(1 + \epsilon)$-SSSP

algorithm of Theorem [3.2] with super-source S on G, providing distance estimates $\delta_S(\cdot)$.[2] Whenever the distance estimate $\delta_S(v)$ of a vertex $v \in V$ becomes smaller than $(1+\epsilon)r$, we remove v from L_i and add the edge (a, v) to H, where a is the corresponding vertex of S for the distance estimate $\delta_S(v)$. Therefore, we continue with the recursion as in Algorithm [1] by constructing the set L_i in this way, and in turn computing its size.

Moreover, since at every new recursive call the set $S^{(i)}$ is modified (i.e., we sample more vertices), at every new recursive call we set $S = S^{(i)}$ and we restart the incremental $(1+\epsilon)$-SSSP algorithm with super-source S on G.

As in Algorithm [1], the second phase begins when the size of L_i is at most $4k$, and at this moment the recursion ends. We denote by d the index of the last recursive call in the first phase, and S is updated to $S := S^{(d)} \cup L_d$. During the second phase, Algorithm [1] has to maintain an incremental k-bounded $(2, 1)$-ruling set algorithm on $G_r[S]$. Instead, we maintain an incremental k-bounded $(2, 1)$-ruling set algorithm on a subgraph $H[S]$ of $G_{r'}[S]$. The subgraph $H[S]$ is maintained explicitly, as follows. Let E_S be the initially empty set consisting of the edges in $H[S]$. For each vertex $v \in S$, we maintain the incremental $(1+\epsilon)$-SSSP algorithm of Theorem [3.2] with source v on G, providing distance estimates $\delta_v(\cdot)$. Then for any two vertices $u, v \in S$, whenever we have that $\delta_u(v) \leq (1+\epsilon)r$ or $\delta_v(u) \leq (1+\epsilon)r$, the edge (u, v) is added to E_S.

Thus during the second phase, we maintain the incremental k-bounded $(2, 1)$-ruling set algorithm $\mathcal{B}$ of Theorem [5.2] on (S, E_S). Whenever $\mathcal{B}$ reports that there is an independent set in (S, E_S) of size at least $k+1$, we report that there is an independent set in G_r of size at least $k+1$. In the analysis we argue that S is a dominating set in H. Hence, this implies that $\mathcal{B}$ solves the incremental k-bounded $(2, 2)$-ruling set problem in H.

Edge insertions. Consider an edge insertion to G, and let i be the current recursive call of the algorithm before the update arrives. While the algorithm is in the first phase, the update is passed to the incremental $(1+\epsilon)$-SSSP algorithm with super-source $S^{(i)}$. If the algorithm is in the second phase after an insertion, then for every vertex $v \in S$, the inserted edge is passed as an update to the incremental $(1+\epsilon)$-SSSP algorithm with source v. Notice that an edge insertion in the original graph G could introduce multiple updates to E_S. Thus whenever an edge (a, b) is added to E_S, the edge (a, b) is passed as an update to the incremental k-bounded $(2, 1)$-ruling set algorithm $\mathcal{B}$.

Proof of Theorem [5.3]. If we had access to exact distances and we removed a vertex v from L_i whenever its distance estimate is at most r, then the correctness would follow from the arguments of the previous section. However, since for efficiency purposes we are utilizing approximate distances, the analysis has to be adapted. In our case we remove a vertex v from L_i whenever its approximate distance estimate is at most $r' = (1+\epsilon)r$. The next lemma is similar to Lemma [5.2] but now applied to the r-threshold graph G_r.

LEMMA 5.5. *At any stage of the algorithm with $i \geq 1$, if $|L_i| > \frac{|L_{i-1}|}{2}$, then w.h.p. there is an independent in G_r of size at least $k+1$.*

Proof. The threshold γ_i is set to $\frac{|L_{i-1}|}{2k} - 1$, and S_i is obtained by sampling each vertex of L_{i-1} independently with probability $\min(c \ln(n)/\gamma_i, 1)$, for a sufficiently large constant c. Then by Lemma [5.1], it holds that w.h.p. every vertex v in L_{i-1} of degree more than γ_i in the induced subgraph $G_r[L_{i-1}]$ has a neighbor in S_i. This is equivalent of saying that w.h.p. every vertex v of degree more than γ_i in $G_r[L_{i-1}]$ is within distance r from a vertex of S_i in G, that is, $d_G(v, S_i) \leq r$. Then, by Theorem [3.2] we have that $\delta_S(v) \leq (1+\epsilon)r$, which means that v has been removed from L_i. In turn, this implies that w.h.p. every vertex in L_i is of degree at most γ_i in $G_r[L_{i-1}]$. As $G_r[L_i]$ is a subgraph of $G_r[L_{i-1}]$, w.h.p. every vertex of $G_r[L_i]$ is of degree at most γ_i in $G_r[L_i]$ as well.

Since w.h.p. the maximum degree in $G_r[L_i]$ is bounded by γ_i, the claim follows by applying the same process of the second paragraph of Lemma [5.2] on $G_r[L_i]$. $\square$

Notice that after computing the size of L_i, the recursion continues in the same way as in Algorithm [1]. The next lemma says that Lemma [5.3] holds in this algorithm as well. Recall that d is the recursive call after the first phase has ended and just before the second phase begins (i.e., d is the final depth of the recursion).

LEMMA 5.6. *Over the sequence of updates, there are $d = O(\log n)$ recursive calls. Moreover, the size of S is w.h.p. $O(k \log^2 n)$.*

---

[2]Namely, we introduce a fake root x and add an edge (x, v) of zero weight, for every $v \in S$. Then, we run the approximate SSSP algorithm with source x on G.

Remember that we want to use Algorithm 2 as a subroutine in the incremental k-center algorithm. By using the next property of H, we argue that only an extra $(1 + \epsilon)$ factor shows up in the approximation ratio of the k-center algorithm.

LEMMA 5.7. *The graph H is a subgraph of the r'-threshold graph $G_{r'}$, where $r' = (1 + \epsilon)r$.*

Proof. Let $(u, v) \in E(H)$ be an edge of the graph H. Suppose that the edge has been added during the first phase of the algorithm. Then, WLOG it must be the case that $u \in S$ and $\delta_S(v) \le r'$. Based on Theorem 3.2, the distance estimate $\delta_S(\cdot)$ does not underestimate the distances, and so we have that $d_G(u, v) \le \delta_S(v) \le r'$. Thus by definition, the edge (u, v) is part of $G_{r'}$ as well.

Similarly, suppose that the edge has been added during the second phase of the algorithm. Then, WLOG it must be the case that $u, v \in S$ and $\delta_u(v) \le r'$. Using a similar argument as before, we conclude that every edge of H is part of $G_{r'}$. $\quad\square$

During the second phase, whenever the incremental k-bounded $(2, 1)$-ruling set algorithm $\mathcal{B}$ reports that there is an independent set in H of size at least $k + 1$, we report that there is an independent set in G_r of size at least $k + 1$. Since algorithm $\mathcal{B}$ is running on $H[S]$, the following lemma states that in this case, there is definitely an independent set in G_r (and not just with high probability) of size at least $k + 1$.

LEMMA 5.8. *Any independent set in $H[S]$ is also an independent set in $G_r[S]$.*

Proof. Let M be an independent set in $H[S]$, and suppose to the contrary that M is not an independent set in $G_r[S]$. Then there must exist two vertices $u, v \in S \cap M$, such that the edge (u, v) belongs to $G_r[S]$ but not to $H[S]$. Since (u, v) belongs to G_r, the distance between u and v in G is at most r (i.e., $d_G(u, v) \le r$). Also as $u \in S$, in the algorithm we maintain the incremental $(1 + \epsilon)$-SSSP algorithm with source u on G, and by Theorem 3.2 it holds that $\delta_u(v) \le (1 + \epsilon)d_G(u, v) \le (1 + \epsilon)r$. Hence as $v \in S$, the algorithm must have added the edge (u, v) to H, which contradicts the assumption that the edge (u, v) does not belong to $H[S]$. $\quad\square$

LEMMA 5.9. *The set S is a dominating set in H.*

Proof. The claim follows by applying the proof of Lemma 5.4 on H. $\quad\square$

Running Time. During the first phase, the incremental $(1 + \epsilon)$-SSSP algorithm with super-source $S^{(i)}$ on G, is restarted as many times as the number of the recursive calls. By Lemma 5.6, there are at most $O(\log n)$ recursive calls in total, and by Theorem 3.2, the total update time of the incremental $(1 + \epsilon)$-SSSP algorithm is $m^{1+o(1)}$. Thus, the total update time charged for the first phase of the algorithm is $m^{1+o(1)}$.

During the second phase, for every vertex $v \in S$, we maintain an incremental $(1 + \epsilon)$-SSSP algorithm of Theorem 3.2 with source v on G. By Lemma 5.6, the size of S is w.h.p. $\tilde{O}(k)$, and so the total update time for maintaining the edge set E_S is $km^{1+o(1)}$.

OBSERVATION 5.2. *Since G is subject to edge insertion, the edge set E_S of H is non-descreasing.*

The k-bounded $(2, 1)$-ruling set algorithm $\mathcal{B}$ of Theorem 5.2 is running on (S, E_S) (i.e., the induced subgraph $H[S]$). Also, as the edge set E_S contains only edges between vertices in S, the maximum size of E_S is w.h.p. $\tilde{O}(k^2)$. Then based on Theorem 5.2 and Observation 5.2, the total update time charged for $\mathcal{B}$ is $\tilde{O}(k^2)$, which (when amortized over the $\Omega(k^2)$ total edge insertions to E_s) amounts to an amortized update time of $\tilde{O}(1)$. This concludes the running time analysis of Theorem 5.3.

5.3 Incremental k-center on graphs: Putting it together At this point, we have developed all the necessary tools in order to obtain our main theorem for the incremental k-center problem. The idea is to combine the reduction of Lemma 4.2 with Algorithm 2 of Theorem 5.3.

THEOREM 5.4. *Given a weighted undirected graph $G = (V, E, w)$ subject to edge insertions, an integer parameter $k \ge 1$, and a positive constant parameter $\epsilon < 1$, there is a randomized incremental $(4 + \epsilon)$-approximation algorithm for the k-center problem on graphs. The algorithm is correct w.h.p. and has $kn^{o(1)}$ amortized update time over a sequence of $\Theta(m)$ updates.*

Proof. Observe that algorithm $\mathcal{A}$ inside Lemma 4.2 with $\beta = 2$, has the same properties of the algorithm in Corollary 5.1. Hence, let $\mathcal{A}$ be the Algorithm 2 of Corollary 5.1, and $\epsilon_1 = \frac{\epsilon}{12}$. Based on Lemma 4.2, by running $\mathcal{A}$ with input G, r, ϵ_1, for each $r \in \{(1 + \epsilon_1)^i \mid (1 + \epsilon_1)^i \leq nW, i \in \mathbb{N}\}$, we get an incremental $4(1 + \epsilon_1)(1 + \epsilon_1)$-approximation algorithm for the k-center problem. As $\epsilon < 1$ and $\epsilon_1 = \frac{\epsilon}{12}$, the approximation ratio is $(4 + \epsilon)$.

Regarding the running time, by Corollary 5.1 the total update time of $\mathcal{A}$ is $km^{1+o(1)}$. Also, we run $\mathcal{A}$ for at most $O(\log_{1+\epsilon_1}(nW))$ different values of r. Thus, the total update time of the algorithm remains $km^{1+o(1)}$. □

6 Decremental k-center on graphs

In the decremental setting, the input graph of the k-center instance is subject to edge deletions. Based on Lemma 4.1, in order to get a $(2+\epsilon)$-approximation decremental algorithm for the k-center problem, it is sufficient to develop a decremental algorithm for the k-bounded $(2, 1)$-ruling set problem on r-threshold graphs. To maintain the necessary information for r-threshold graphs, we use a decremental SSSP algorithm on G.

6.1 Decremental k-bounded $(2, 1)$-ruling set on G_r

For the sake of efficiency, in order to maintain the necessary information for r-threshold graphs, we make use of the approximate SSSP algorithm of Theorem 3.1. Thus, we obtain instead the following theorem which is a slight relaxation of the decremental k-bounded $(2, 1)$-ruling set problem on r-threshold graphs. This is still sufficient for the k-center problem, as Lemma 4.2 suggests.

THEOREM 6.1. *Consider a graph $G = (V, E, w)$ subject to edge deletions, an integer $k \geq 1$, and positive parameters r, ϵ. Let $r' := (1 + \epsilon)r$ and consider the threshold graphs G_r and $G_{r'}$. Then, there is a deterministic algorithm which:*

- *either reports that there is an independent set in G_r of size at least $k + 1$,*
- *or runs a k-bounded $(2, 1)$-ruling set algorithm $\mathcal{B}$ on an edge-subgraph H of $G_{r'}$ with the following condition: whenever $\mathcal{B}$ reports that there is an independent set in H of size at least $k + 1$, then there is an independent set in G_r of size at least $k + 1$.*

The total update time of the algorithm is $km^{1+o(1)}$.

Recall that in the definition of an (α, β)-ruling set, the first property is that the distance between any two vertices in the (α, β)-ruling set is at least α. The crucial observation here is that under edge deletions, the distance between any two vertices is non-decreasing. Hence the first property is preserved in the decremental setting, and this is the major ingredient for the algorithm.

Algorithm of Theorem 6.1. In the beginning, the algorithm executes a static $(2, 1)$-ruling set algorithm $\mathcal{B}$ on G_r. One simple algorithm for this problem is to run k times the Dijkstra's algorithm on G. In particular, at each iteration we choose a vertex s which has not been covered yet, and we run Dijkstra's algorithm on G with source s. Then, every vertex v of distance at most r from s is set as covered, and the same process is repeated at most k times. The running time of this algorithm is clearly $\tilde{O}(mk)$.

Let M be the $(2, 1)$-ruling set in G_r returned by $\mathcal{B}$. Then, we initialize a decremental approximate SSSP algorithm $\mathcal{A}$ with super-source M on G, providing distance estimates $\delta(\cdot)$.[3] Specifically, we use the $(1 + \epsilon)$-approximate SSSP algorithm of Theorem 3.1. Also let H be a graph whose edge set contains all the edges $(u, v) \in V \times V$ such that $\delta(v) \leq r'$, and $u \in M$ is the corresponding vertex for the distance estimate $\delta(v)$. The graph H can be explicitly constructed during the previous step.

Whenever there is an edge deletion in G, we pass this update to $\mathcal{A}$. In turn, this update can possibly increase the distance estimate $\delta(\cdot)$ of some vertices. In particular, whenever the distance estimate $\delta(v)$ of a vertex $v \in V$ becomes greater than r', we add v to M, and the algorithm $\mathcal{A}$ is restarted with super-source the modified set M. Moreover, the graph H is recomputed from scratch as before.

At any moment, if the size of M has exceeded k, the algorithm reports that there is an independent set in G_r of size at least $k + 1$, and we do not restart the algorithm $\mathcal{A}$ anymore.

Proof of Theorem 6.1. Initially the static algorithm produces a $(2, 1)$-ruling set M in G_r. At any moment, if the size of M becomes at least $k + 1$, the algorithm reports that there is an independent set in G_r of size at least $k + 1$. The next lemma shows the correctness of this step.

---

[3]Namely, we introduce a fake root x and add an edge (x, v) of zero weight, for every $v \in M$. Then, we run a decremental approximate SSSP algorithm with source x on G.

LEMMA 6.1. *If the size of M is at least $k + 1$, then there is an independent set in G_r of size at least $k + 1$.*

Proof. Initially, M is a $(2, 1)$-ruling set in G_r, and by definition M is also an independent set in G_r. Thus, if the size of M is at least $k + 1$ after the execution of the static algorithm, the set M remains an independent set in G_r under edge deletions, and the claim holds.

Hence, we can assume that the size of M became at least $k + 1$ after some edge deletions. We prove the claim by contradiction. Suppose to the contrary that M is not an independent set in G_r after an edge deletion. In this case, the algorithm must have added a vertex v to M which has a neighbor $u \in M$ in G_r (i.e., $d_G(u, v) \le r$). Since $u \in M$, in the algorithm we maintain the decremental $(1 + \epsilon)$-SSSP algorithm with super-source M, and by Theorem 3.1 it holds that $\delta(v) \le (1 + \epsilon) d_G(u, v) \le r'$. But then, the algorithm does not add v to M which yields a contradiction. □

The second part of the algorithm maintains a decremental k-bounded $(2, 1)$-ruling set algorithm on an edge-subgraph H of $G_{r'}$. We prove this fact in the following lemma.

LEMMA 6.2. *The graph H is a subgraph of $G_{r'}$. Moreover, the set M is always a $(2, 1)$-ruling set in H.*

Proof. Let $(u, v) \in E(H)$ be an edge in H. Then, WLOG it must be the case that $u \in M$ and $\delta(v) \le r'$. Based on Theorem 3.1, the distance estimate $\delta(\cdot)$ does not underestimate the distances, and so we have that $d_G(u, v) \le \delta(v) \le r'$. Thus by definition, the edge (u, v) is part of $G_{r'}$ as well.

The algorithm adds a vertex v to M only if the distance estimate $\delta(v)$ becomes greater than r', while an edge is part of H only if the distance of the two endpoints is at most r'. This implies that the set M is an independent set in H. Furthermore, whenever the distance estimate $\delta(v)$ of a vertex $v \in V \setminus M$ becomes greater than r', the set M and the graph H are recomputed. This implies that the distance estimate of any vertex $v \in V \setminus M$ is at most r'. By construction of H, there must exist an edge (u, v), where u is the corresponding vertex of $\delta(v)$. Hence, we can conclude that the set M is a $(2, 1)$-ruling set in H. □

Running time. The running time of the simple static algorithm is $\tilde{O}(mk)$. By Theorem 3.1, the total time of the decremental approximate SSSP algorithm is $m^{1+o(1)}$. As the decremental approximate SSSP algorithm is restarted at most k times, the total update time of the algorithm is $km^{1+o(1)}$. Finally, the time to detect whether a distance estimate is greater than $(1 + \epsilon)r$ is incorporated in the update time of the decremental approximate SSSP algorithm.

6.2 Decremental k-center on graphs: Putting it together We combine Theorem 6.1 with Lemma 4.2 to obtain the next theorem for the decremental k-center problem on graphs. A pseudocode of the algorithm of Theorem 1.3 is provided in Algorithm 3.

THEOREM 6.2. *Given a weighted undirected graph $G = (V, E, w)$ subject to edge deletions, an integer parameter $k \ge 1$, and a positive constant parameter $\epsilon < 1$, there is a deterministic decremental $(2 + \epsilon)$-approximation algorithm for the k-center problem on graphs, with $kn^{o(1)}$ amortized update time over a sequence of $\Theta(m)$ updates.*

Proof. Observe that algorithm $\mathcal{A}$ inside Lemma 4.2 with $\beta = 1$, has the same properties of the algorithm in Theorem 6.1. Hence, let $\mathcal{A}$ be the algorithm of Theorem 6.1, and $\epsilon_1 = \frac{\epsilon}{6}$. Based on Lemma 4.2, by running $\mathcal{A}$ with input G, r, ϵ_1, for each $r \in \{(1 + \epsilon_1)^i \mid (1 + \epsilon_1)^i \le nW, i \in \mathbb{N}\}$, we get a deterministic decremental $2(1 + \epsilon_1)(1 + \epsilon_1)$-approximation algorithm for the k-center problem. As $\epsilon < 1$ and $\epsilon_1 = \frac{\epsilon}{6}$, the approximation ratio is $(2 + \epsilon)$.

Regarding the running time, by Theorem 6.1 the total update time of $\mathcal{A}$ is $km^{1+o(1)}$. Also, we run $\mathcal{A}$ for at most $O(\log_{1+\epsilon_1}(nW))$ different values of r. Thus, the total update time of the algorithm remains $km^{1+o(1)}$. □

7 Fully-dynamic k-center on graphs

In this section we describe how to maintain a $(2 + \epsilon)$-approximate solution to the k-center problem on fully-dynamic graphs. We start by reviewing Gonzalez's algorithm, a classical solution to the problem in the static setting, and then describe how to adapt it to the fully-dynamic setting by using fully-dynamic approximate SSSP algorithms.

Algorithm 3 DECREMENTAL $(2 + \epsilon)$-APPROXIMATION ALGORITHM FOR k-CENTER

Function MAXIMALDISTRIS():

 $i \leftarrow 0$
 while $G \setminus \bigcup_{j=1}^{i} C_j \neq \emptyset$ *and* $i \leq k$ **do**
  $u \leftarrow$ arbitrary vertex from $G \setminus \bigcup_{j=1}^{i} C_j$
  $i \leftarrow i + 1$
  $c_i \leftarrow u$
  $C_i \leftarrow$ cluster with center c_i and radius r
 return i

Procedure FINDRADIUS():

 while MAXIMALDISTRIS() $> k$ **do**
  $r \leftarrow (1 + \epsilon) \cdot r$

Procedure PREPROCESSING():

 $r \leftarrow 1 + \epsilon$
 FINDRADIUS()
 $M \leftarrow \{c_1, \ldots, c_i\}$
 $\mathcal{A}.initialize(G, M)$ // $\mathcal{A}$ *is decremental approx. SSSP algorithm with distance estimates* $\delta(\cdot)$

Procedure UPDATE(u, v):

 $G \leftarrow (V, E \setminus (u, v))$
 $\mathcal{A}.delete(u, v)$
 while $\exists x \in V \ s.t. \ \delta(x) > r$ **do**
  if $i < k$ **then**
   $i \leftarrow i + 1$
   $c_i \leftarrow x$
   $M \leftarrow \{c_1, \ldots, c_i\}$
   $C_i \leftarrow$ cluster with center c_i and radius r
  else
   $r \leftarrow (1 + \epsilon) \cdot r$
   FINDRADIUS()
  $\mathcal{A}.restart(G, M)$

7.1 Gonzalez's algorithm Gonzalez's algorithm [25] is a well-known greedy algorithm for the k-center problem on (possibly weighted and directed) graphs[4]. It works as follows:

1. pick as first center an arbitrary vertex $c_1 \in V$ and set $C = \{c_1\}$;
2. while $|C| < k$, pick the next center $c_i \in \operatorname{argmax}_{v \in V} d_G(C, v)$ and set $C = C \cup \{c_i\}$;
3. return the set of centers C.

THEOREM 7.1. *Gonzalez's algorithm computes a 2-approximation for the k-center problem on graphs and a standard implementation runs in time $O(k(m + n \log n))$.*

DEFINITION 7.1. (α-APPROXIMATE GONZALEZ'S ALGORITHM) *For $\alpha \geq 1$, an α-approximate Gonzalez's algorithm is a relaxation of Gonzalez's algorithm that picks the next center c_i in step 2 above such that $d_G(C, c_i) \geq \alpha^{-1} \cdot \max_{v \in V} d_G(C, v)$.*

THEOREM 7.2. ([1, LEMMA 4.1]) *For $\alpha \geq 1$, an α-approximate Gonzalez's algorithm computes a 2α-approximation for the k-center problem on graphs.*

7.2 Fully-dynamic k-center via fully-dynamic $(1 + \epsilon)$-SSSP Assuming that we have a fully-dynamic $(1+\epsilon)$-SSSP data structure, we show how to use this to get a fully-dynamic k-center data structure in Algorithm 4.

---

[4]The algorithm is also used for the k-center problem in metric spaces.

Algorithm 4 FULLY DYNAMIC $2(1 + \epsilon)$-APPROXIMATION K-CENTER

Function SIMULATEGONZALEZ($\mathcal{D}$, s, k):

    $C = \emptyset$

    for $i = 1, ..., k$ **do**

        $c_i \leftarrow x \in \mathrm{argmax}_{v \in V}\, \delta_{G'}(s, v)$ *// $c_1 \leftarrow$ arbitrary $v \in V$*

                             // s is disconnected at $i = 1$; $\delta_{G'}(s, v) = \infty$, $\forall v \in V$; ties broken arbitrarily

        $\mathcal{D}.insert(s, c_i)$

        $C \leftarrow C \cup \{c_i\}$

    for $i = 1, ..., k$ **do**

        $\mathcal{D}.delete(s, c_i)$

    return C

Function PREPROCESSING(G, k):

    $G' \leftarrow (V \cup \{s\}, E)$ *// augment G with super source s*

    $\mathcal{D} \leftarrow$ INITIALIZE(G', s)   *// $\mathcal{D}$ is fully-dynamic $(1 + \epsilon)$-SSSP, with approx. distance $\delta_{G'}(s, v)$*

    $C \leftarrow$ SIMULATEGONZALEZ($\mathcal{D}$, s, k)

Function UPDATE(u, v):

    $\mathcal{D}.update(u, v)$ *// either insert or delete edge (u, v)*

    $C \leftarrow$ SIMULATEGONZALEZ($\mathcal{D}$, s, k)

THEOREM 7.3. *Given a graph $G = (V, E)$, a positive parameter $\epsilon \leq 1/2$, and a fully-dynamic data structure that maintains $(1 + \epsilon)$-approximate distances from a single source $s \in V$ with worst-case update time $T(n, m, \epsilon)$, Algorithm 4 maintains a $2(1 + 4\epsilon)$-approximate solution to fully-dynamic k-center in time $O(k \cdot (T(n, m, \epsilon) + n))$.*

Proof. We prove that the procedure SIMULATEGONZALEZ in Algorithm 4 is a $(1 + 4\epsilon)$-approximate Gonzalez's algorithm, hence the claim about the approximation follows by Theorem 7.2.

Note that the procedure runs on G', which is a copy of G with an additional super-source vertex s which is initially disconnected. Let us call $\mathcal{D}$ the data structure used to maintain the $(1+\epsilon)$-approximate distances from s in G', e.g., the one given in Theorem 7.4 or in Theorem 7.5. Suppose to be at the i-th iteration of the procedure, i.e., the super-source s is connected to all vertices in $C = \{c_1, ..., c_i\}$ in G'. Note that such additional edges imply that $d_{G'}(s, v) = 1 + d_G(C, v)$, for every $v \in V$. Let $\delta_{G'}(s, v)$ be the approximate distance between s and v maintained by $\mathcal{D}$, which guarantees that $d_{G'}(s, v) \leq \delta_{G'}(s, v) \leq (1 + \epsilon)d_{G'}(s, v)$. Let $v_{\max} \in \mathrm{argmax}_{v \in V}\, d_G(C, v)$ be one among the furthest vertices from C. Let c_{i+1} be the next center selected by the algorithm, i.e., $c_{i+1} \in \mathrm{argmax}_{v \in V}\, \delta_{G'}(s, v)$. Therefore, it holds that

$$1 + d_G(C, v_{\max}) = d_{G'}(s, v_{\max}) \leq \delta_{G'}(s, v_{\max}) \leq \delta_{G'}(s, c_{i+1})$$
$$\leq (1 + \epsilon)d_{G'}(C, c_{i+1}) = (1 + \epsilon)(1 + d_G(C, c_{i+1})).$$

Noting that $d_G(C, v_{\max}) \geq 1$ and since by assumption $\epsilon \in (0, 1/2]$, the previous equation implies

$$d_G(C, c_{i+1}) \geq \frac{1 + d_G(C, v_{\max})}{1 + \epsilon} - 1 = \frac{d_G(C, v_{\max}) - \epsilon}{1 + \epsilon} = \frac{1 - \epsilon/d_G(C, v_{\max})}{1 + \epsilon}d_G(C, v_{\max})$$
$$\geq \frac{1 - \epsilon}{1 + \epsilon}d_G(C, v_{\max}) \geq \frac{1}{1 + 4\epsilon}\max_{v \in V} d_G(C, v),$$

which concludes the approximation proof.

The update procedure requires that $\mathcal{D}$ is updated $2k+1$ times, with a worst-case time of $T(n, m, \epsilon)$ per update, and additionally look for the approximate furthest neighbor k times, each requiring time $O(n)$, i.e., querying the approximate distance $\delta_G(s, v)$, $\forall v \in V$.   ☐

In particular, for the fully-dynamic data structure we use the state-of-the-art algorithm for unweighted graphs by [13].

THEOREM 7.4. ([13]) *Given an unweighted undirected graph $G = (V, E)$ and a single source s, and $0 < \epsilon < 1$, there is a deterministic fully-dynamic data structure for maintaining $(1 + \epsilon)$-distances from s with worst-case update time of $O(n^{1.529}\epsilon^{-2})$ for the current matrix multiplication exponent ω. The algorithm has preprocessing time of $O(n^{\omega}\epsilon^{-2} \log \epsilon^{-1})$, where $\omega \leq 2.373$.*

For weighted graphs the state-of-the-art algorithm is slower and it is given by [14].

THEOREM 7.5. ([14]) *Given a weighted and directed graph $G = (V, E)$, a single source $s \in V$, and a positive parameter $\epsilon < 1$, there is a randomized fully-dynamic algorithm working against an adaptive adversary that maintains $(1+\epsilon)$-distances from s with worst-case update time of $O(n^{1.823}\epsilon^{-2})$ for the current matrix multiplication exponent ω. The algorithm has preprocessing time of $O(n^{\omega}\epsilon^{-2} \log \epsilon^{-1})$, where $\omega \leq 2.373$.*

A combination of Theorem 7.3 with Theorems 7.4 and 7.5 gives the following result.

THEOREM 7.6. *Given a weighted undirected graph $G = (V, E, w)$ subject to edge updates, an integer parameter $k \geq 1$, and a positive constant parameter $\epsilon \leq 1/2$, there are two fully dynamic algorithms for the k-center problem on graphs, that maintain a $(2 + \epsilon)$-approximation with the following guarantees (based on the current value of the matrix multiplication exponent):*
1. *Deterministic algorithm with $O(kn^{1.529}\epsilon^{-2})$ worst-case update time, if G has uniform weights;*
2. *Randomized algorithm, against an adaptive adversary, with $O(kn^{1.823}\epsilon^{-2})$ worst-case update time, if G has general weights.*

Both algorithms have preprocessing time $O(n^{2.373}\epsilon^{-2} \log \epsilon^{-1})$.

Appendix

A Dynamic k-center algorithms queries

The dynamic algorithms for k-center we give in this paper can simply and efficiently answer queries of the following types:
1. Return a set S of at most k centers and the corresponding radius r.
2. Given a vertex $v \in V$, return the center $c \in S$ of the cluster v belongs to.

The first type of queries simply returns the independent sets which have size $\leq k$. The corresponding radius in the first query and the second type of queries can be answered using the dynamic shortest path data structures that we use. When maintaining distances from a super-source, the data structures let us keep the parent nodes along shortest paths which can be used for finding the closest source. The partially dynamic algorithms that we use are based on structures with $n^{o(1)}$ layers, such that we have a parent along the shortest path on each of these levels. Therefore without additional overhead we can keep track of the first parents along the path.

In our fully dynamic algorithm, we can keep track of the cluster center of each vertex by explicitly checking for each vertex whether its distance to the "super-source" changes with each iteration of the simulated Gonzalez's algorithm; the additional $O(kn)$ overhead is already accounted for in our update time.

B Reduction from 2-approximate k-center to k-bounded ruling set: Omitted proofs of Section 4

This section is devoted to the omitted proofs of Section 4.

LEMMA 4.3. *Consider a k-center instance $(G = (V, E, w), k)$ with optimal radius R^*. Then for each $r \geq 2R^*$ and for every $\beta \geq 1$, it holds that every $(2, \beta)$-ruling set in the r-threshold graph G_r is of size at most k.*

Proof. Consider an optimal solution $S^* = \{c_1^*, \ldots, c_{k'}^*\}$ of the k-center instance with $k' \leq k$, and let $C_1^*, \ldots, C_{k'}^*$ be the corresponding clusters, each of radius R^*. Let M be an arbitrary $(2, \beta)$-ruling set in G_r, with $r \geq 2R^*$. We can assume WLOG that the set M is ordered. The proof is by induction on the number of vertices of M. The goal is to prove that for any $i \leq k'$, every vertex $v \in C_i^*$ is a neighbor of the i_{th} vertex of M in G_r. This would imply then that $|M| \leq k' \leq k$, as otherwise there would be two vertices in M which are neighbors in G_r, violating the fact that M is a $(2, \beta)$-ruling set in G_r with $\beta \geq 1$

As a base case, let v_1 be the first vertex of M, and assume WLOG that $v_1 \in C_1^*$. Since all vertices in C_1^* are within distance R^* from c_1^* in G, by triangle inequality it holds that $d_G(v_1, u) \leq 2R^*$, for every vertex $u \in C_1^*$. Hence as $r \geq 2R^*$, we have that every vertex $u \in C_1^*$ is a neighbor of v_1 in G_r.

Let $v_i \in M$ be the i_{th} vertex of M. Since M is a $(2, \beta)$-ruling set in G_r with $\beta \geq 1$, v_i cannot be a neighbor of any other vertex that belongs to M. By inductive hypothesis, every vertex $u \in C_1^* \cup \cdots \cup C_{i-1}^*$ has a neighbor in M, and so v_i cannot be part of $C_1^* \cup \cdots \cup C_{i-1}^*$. As a result, we can assume WLOG that $v_i \in C_i^*$. By following the same approach as in the base case, we have that every vertex $u \in C_i^*$ is a neighbor of v_i in G_r, and so the claim follows. $\square$

OBSERVATION 4.1. *Consider a k-center instance $(G = (V, E, w), k)$ with optimal radius R^*, and let G_r be the r-threshold graph where $r = 2R^*$. Then, there is no independent set in G_r of size at least $k + 1$.*

Proof. Suppose to the contrary that there is an IS M in G_r of size at least $k + 1$, and let M' be an MIS on G_r such that $M \subseteq M'$. Then clearly it holds that $|M'| \geq k + 1$. Also since $\beta \geq 1$, we have that M' is a $(2, \beta)$-ruling set in G_r of size at least $k + 1$. However based on Lemma 4.3, as $r \geq 2R^*$ and $\beta \geq 1$, the size of M' must be at most k, and this yields a contradiction. $\square$

LEMMA 4.1. *Consider a k-center instance $(G = (V, E, w), k)$, and a positive constant parameter ϵ. Then by running a k-bounded $(2, \beta)$-ruling set algorithm on the r-threshold graph G_r, for each $r \in \{(1 + \epsilon)^i \mid (1 + \epsilon)^i \leq nW, i \in \mathbb{N}\}$, we can find a $2\beta(1 + \epsilon)$-approximate solution for the k-center instance.*

Proof. Let r' be the smallest $r \in \{(1+\epsilon)^i \mid (1+\epsilon)^i \leq nW, i \in \mathbb{N}\}$ such that a k-bounded $(2, \beta)$-ruling set algorithm running on G_r returns a $(2, \beta)$-ruling set M_r of size at most k. Let $S = M_{r'}$ be the solution we return for the k-center instance.

Since $M_{r'}$ is a $(2, \beta)$-ruling set in $G_{r'}$, then every vertex is within distance $\beta r'$ from its closest center in G. Thus, the returned solution S has radius at most $\beta r'$. We show now that r' is at most $2(1 + \epsilon)$ times larger than R^*. Based on Observation 4.1, for the fixed choice of $r = 2R^*$, any k-bounded $(2, \beta)$-ruling set algorithm $\mathcal{A}$ running on the r-threshold graph G_r always returns a $(2, \beta)$-ruling set M_r of size at most k. By definition of r', and since the possible values of r are powers of $(1 + \epsilon)$, we have that $r' \leq 2(1 + \epsilon)R^*$. Therefore, the radius of the returned solution S is at most $2\beta(1 + \epsilon)R^*$. $\square$

C Incremental $(1 + \epsilon)$-SSSP

Most of the existing work on partially dynamic $(1 + \epsilon)$-SSSP [10, 32, 18, 36] is presented for the decremental setting, but while not explicitly written, the techniques extend to the incremental setting as well with the same running time. At a high-level these techniques first maintain a hopset (or similar objects like low-hop emulators) of size $\tilde{O}(m^{1+o(1)})$ and hopbound $h = n^{o(1)}$, and maintain an h-hop limited (ES) Even-Schiloach tree [42].

The h-hop limited ES tree algorithm [10] allows us to maintain $(1 + \epsilon)$-approximate single-source shortest path up to h-hops (which finds the approximate shortest path using at most h hops) in $\tilde{O}(mh)$ time.

To use this subroutine several works utilize a hopset [32, 18, 36]. A (β, ϵ)-hopset H' for $G = (V, E)$ is a set of weighted edges such that for all $u, v \in V$, we have that $d_G(u, v) \leq d_{G \cup H'}^{(\beta)}(u, v) \leq (1 + \epsilon)d_G(u, v)$, where $d_{G \cup H'}^{(\beta)}(u, v)$ refers to a shortest path that uses at most β hops.

Much of the technical difficulty in the decremental setting is due to the fact that we have to insert hopset/emulator edges in a decremental data structure. The existing decremental structures use an algorithm called monotone ES tree data structure [31] to handle this, however in the incremental setting a monotone ES tree is not needed. In an incremental setting, an update may require to decrease the weight of an edge or remove it from a hopset/emulator to keep the size small. Handling weight decreases is easy, as we can simply add a new edge with the smaller weight and keep the previous edges in place and this will only impact the number of edges by a logarithmic factor over the sequence of updates. The second issue of removing edges from a hopset/emulator, will also not impact the over all performance of the algorithm for the following reason: In these data structure we would only remove an auxiliary edge $e \in E(H)$ if the weight $w^{(t)}(e)$ (which corresponds to the length of a path in the original input graph G at time t), is reduced by more than a constant factor so that it is within a factor of $(1 - \epsilon')w^{(t-1)}(e)$. It is easy to see that in the incremental setting it will add a logarithmic factor in the size if we keep all of these edges and simply add new parallel edges and thus keep the data structures completely incremental.

References

[1] Amir Abboud et al. "On the Fine-Grained Complexity of Approximating k-Center in Sparse Graphs". In: *2023 Symposium on Simplicity in Algorithms, SOSA 2023, Florence, Italy, January 23-25, 2023*. SIAM, 2023, pp. 145–155. DOI: 10.1137/1.9781611977585.ch14.

[2] Charu C. Aggarwal and Haixun Wang. "A Survey of Clustering Algorithms for Graph Data". In: *Managing and Mining Graph Data*. Ed. by Charu C. Aggarwal and Haixun Wang. Boston, MA: Springer US, 2010, pp. 275–301. ISBN: 978-1-4419-6045-0. DOI: 10.1007/978-1-4419-6045-0_9.

[3] Donald Aingworth et al. "Fast Estimation of Diameter and Shortest Paths (Without Matrix Multiplication)". In: *SIAM J. Comput.* 28.4 (1999). Announced at SODA 1996, pp. 1167–1181. DOI: 10.1137/S0097539796303421.

[4] Maria-Florina Balcan, Nika Haghtalab, and Colin White. "k-center Clustering under Perturbation Resilience". In: *ACM Trans. Algorithms* 16.2 (2020), 22:1–22:39. DOI: 10.1145/3381424.

[5] Sayan Bandyapadhyay, Zachary Friggstad, and Ramin Mousavi. "Parameterized Approximation Algorithms for K-center Clustering and Variants". In: *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*. AAAI Press, 2022, pp. 3895–3903.

[6] MohammadHossein Bateni et al. "Optimal Fully Dynamic k-Center Clustering for Adaptive and Oblivious Adversaries". In: *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*. SIAM, 2023, pp. 2677–2727. DOI: 10.1137/1.9781611977554.ch101.

[7] Soheil Behnezhad et al. "Fully dynamic maximal independent set with polylogarithmic update time". In: *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2019, pp. 382–405.

[8] Suman K. Bera et al. "Fair k-Center Clustering in MapReduce and Streaming Settings". In: *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*. ACM, 2022, pp. 1414–1422. DOI: 10.1145/3485447.3512188.

[9] Mark de Berg, Leyla Biabani, and Morteza Monemizadeh. "k-Center Clustering with Outliers in the MPC and Streaming Model". In: *IEEE International Parallel and Distributed Processing Symposium, IPDPS 2023, St. Petersburg, FL, USA, May 15-19, 2023*. IEEE, 2023, pp. 853–863.

[10] Aaron Bernstein. "Fully dynamic $(2+\varepsilon)$ approximate all-pairs shortest paths with fast query and close to linear update time". In: *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE. 2009, pp. 693–702.

[11] Aaron Bernstein, Maximilian Probst Gutenberg, and Thatchaphol Saranurak. "Deterministic Decremental SSSP and Approximate Min-Cost Flow in Almost-Linear Time". In: *62 Annual IEEE Symposium on Foundatios of Computer Science (FOCS 2022)*. 2021.

[12] Sayan Bhattacharya, Silvio Lattanzi, and Nikos Parotsidis. "Efficient and Stable Fully Dynamic Facility Location". In: *NeurIPS*. 2022. URL: http://papers.nips.cc/paper%5C_files/paper/2022/hash/943d6dca1884955e645d8997ae2fa938-Abstract-Conference.html.

[13] Jan van den Brand, Sebastian Forster, and Yasamin Nazari. "Fast Deterministic Fully Dynamic Distance Approximation". In: *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. 2022, pp. 1011–1022. DOI: 10.1109/FOCS54457.2022.00099.

[14] Jan van den Brand and Danupon Nanongkai. "Dynamic Approximate Shortest Paths and Beyond: Subquadratic and Worst-Case Update Time". In: *FOCS*. IEEE Computer Society, 2019, pp. 436–455.

[15] Matteo Ceccarello, Andrea Pietracaprina, and Geppino Pucci. "Solving k-center Clustering (with Outliers) in MapReduce and Streaming, almost as Accurately as Sequentially". In: *Proc. VLDB Endow.* 12.7 (2019), pp. 766–778. DOI: 10.14778/3317315.3317319.

[16] TH Hubert Chan, Arnaud Guerqin, and Mauro Sozio. "Fully dynamic k-center clustering". In: *Proceedings of the 2018 World Wide Web Conference*. 2018, pp. 579–587.

[17] Moses Charikar et al. "Incremental Clustering and Dynamic Information Retrieval". In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*. Ed. by Frank Thomson Leighton and Peter W. Shor. ACM, 1997, pp. 626–635. DOI: 10.1145/258533.258657.

[18] Shiri Chechik. "Near-optimal approximate decremental all pairs shortest paths". In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2018, pp. 170–181.

[19] Shiri Chechik and Tianyi Zhang. "Fully dynamic maximal independent set in expected poly-log update time". In: *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2019, pp. 370–381.

[20] Vincent Cohen-Addad et al. "Fully Dynamic Consistent Facility Location". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. 2019, pp. 3250–3260. URL: https://proceedings.neurips.cc/paper/2019/hash/fface8385abbf94b4593a0ed53a0c70f-Abstract.html.

[21] David Eppstein, Sariel Har-Peled, and Anastasios Sidiropoulos. "Approximate greedy clustering and distance selection for graph metrics". In: *Journal of Computational Geometry* 11.1 (2020). DOI: 10.20382/jocg.v11i1a25.

[22] Tomás Feder and Daniel H. Greene. "Optimal Algorithms for Approximate Clustering". In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. Ed. by Janos Simon. ACM, 1988, pp. 434–444. DOI: 10.1145/62212.62255.

[23] Andreas Emil Feldmann. "Fixed Parameter Approximations for k-Center Problems in Low Highway Dimension Graphs". In: *42nd International Colloquium on Automata, Languages, and Programming (ICALP)*. Vol. 9135. Lecture Notes in Computer Science. Springer, 2015, pp. 588–600. DOI: 10.1007/978-3-662-47666-6_47.

[24] Hendrik Fichtenberger et al. "Consistent k-Clustering for General Metrics". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, (SODA 2021)*. 2021, pp. 2660–2678. DOI: 10.1137/1.9781611976465.158.

[25] Teofilo F. Gonzalez. "Clustering to Minimize the Maximum Intercluster Distance". In: *Theor. Comput. Sci.* 38 (1985), pp. 293–306. DOI: 10.1016/0304-3975(85)90224-5.

[26] Gramoz Goranci, Monika Henzinger, and Dariusz Leniowski. "A Tree Structure For Dynamic Facility Location". In: *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*. Vol. 112. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 39:1–39:13. DOI: 10.4230/LIPIcs.ESA.2018.39.

[27] Gramoz Goranci et al. "Fully Dynamic k-Center Clustering in Low Dimensional Metrics". In: *Proceedings of the Symposium on Algorithm Engineering and Experiments, ALENEX 2021, Virtual Conference, January 10-11, 2021*. Ed. by Martin Farach-Colton and Sabine Storandt. SIAM, 2021, pp. 143–153.

[28] Xiangyu Guo et al. "Consistent k-Median: Simpler, Better and Robust". In: *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*. Ed. by Arindam Banerjee and Kenji Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, 2021, pp. 1135–1143.

[29] Xiangyu Guo et al. "On the Facility Location Problem in Online and Dynamic Models". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*. Vol. 176. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 42:1–42:23. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2020.42.

[30] Monika Henzinger and Sagar Kale. "Fully-Dynamic Coresets". In: *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*. Ed. by Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders. Vol. 173. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 57:1–57:21. DOI: 10.4230/LIPIcs.ESA.2020.57.

[31] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. "A Subquadratic-Time Algorithm for Decremental Single-Source Shortest Paths". In: *Proc. of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA 2014)*. 2014, pp. 1053–1072. DOI: 10.1137/1.9781611973402.79.

[32] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. "Decremental Single-Source Shortest Paths on Undirected Graphs in Near-Linear Total Update Time". In: *Journal of the ACM* 65.6 (2018). Announced at FOCS 2014, 36:1–36:40. DOI: 10.1145/3218657.

[33] Monika Henzinger, Dariusz Leniowski, and Claire Mathieu. "Dynamic Clustering to Minimize the Sum of Radii". In: *Algorithmica* 82.11 (2020), pp. 3183–3194. DOI: 10.1007/s00453-020-00721-7.

[34] Dorit S. Hochbaum and David B. Shmoys. "A unified approach to approximation algorithms for bottleneck problems". In: *J. ACM* 33.3 (1986), pp. 533–550. DOI: 10.1145/5925.5933.

[35] Wen-Lian Hsu and George L. Nemhauser. "Easy and hard bottleneck location problems". In: *Discret. Appl. Math.* 1.3 (1979), pp. 209–215. DOI: 10.1016/0166-218X(79)90044-1.

[36] Jakub Łącki and Yasamin Nazari. "Near-Optimal Decremental Hopsets with Applications". In: *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

[37] Jakub Łącki et al. "Fully Dynamic Consistent k-Center Clustering". In: *CoRR* abs/2307.13747 (2023). arXiv: 2307.13747.

[38] Silvio Lattanzi and Sergei Vassilvitskii. "Consistent k-Clustering". In: *Proceedings of the 34th International Conference on Machine Learning, (ICML 2017)*. 2017, pp. 1975–1984. URL: http://proceedings.mlr.press/v70/lattanzi17a.html.

[39] Jan Plesnik. "On the computational complexity of centers locating in a graph". In: *Aplikace matematiky* 25.6 (1980), pp. 445–452.

[40] Matthew J. Rattigan, Marc E. Maier, and David D. Jensen. "Graph clustering with network structure indices". In: *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*. Ed. by Zoubin Ghahramani. Vol. 227. ACM International Conference Proceeding Series. ACM, 2007, pp. 783–790. DOI: 10.1145/1273496.1273595.

[41] Liam Roditty and Uri Zwick. "Dynamic approximate all-pairs shortest paths in undirected graphs". In: *SIAM Journal on Computing* 41.3 (2012), pp. 670–683.

[42] Yossi Shiloach and Shimon Even. "An on-line edge-deletion problem". In: *Journal of the ACM (JACM)* 28.1 (1981), pp. 1–4.

[43] Mikkel Thorup. "Quick k-Median, k-Center, and Facility Location for Sparse Graphs". In: *SIAM J. Comput.* 34.2 (2004), pp. 405–432. DOI: 10.1137/S0097539701388884.