Certified Algorithms: Worst-Case Analysis and Beyond

Konstantin Makarychev
Northwestern University, Evanston, IL, USA
https://konstantin.makarychev.net

Yury Makarychev
Toyota Technological Institute at Chicago, Chicago, IL, USA
https://ttic.uchicago.edu/~yury
yury@ttic.edu

Abstract

In this paper, we introduce the notion of a *certified algorithm*. Certified algorithms provide worst-case and beyond-worst-case performance guarantees. First, a γ-certified algorithm is also a γ-approximation algorithm – it finds a γ-approximation no matter what the input is. Second, it exactly solves γ-perturbation-resilient instances (γ-perturbation-resilient instances model real-life instances). Additionally, certified algorithms have a number of other desirable properties: they solve both maximization and minimization versions of a problem (e.g. Max Cut and Min Uncut), solve weakly perturbation-resilient instances, and solve optimization problems with hard constraints.

In the paper, we define certified algorithms, describe their properties, present a framework for designing certified algorithms, provide examples of certified algorithms for Max Cut/Min Uncut, Minimum Multiway Cut, k-medians and k-means. We also present some negative results.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Theory of computation → Approximation algorithms analysis; Mathematics of computing → Approximation algorithms; Theory of computation → Facility location and clustering

Keywords and phrases certified algorithm, perturbation resilience, Bilu–Linial stability, beyond-worst-case analysis, approximation algorithm, integrality

Digital Object Identifier 10.4230/LIPIcs.ITCS.2020.49

Funding *Yury Makarychev*: Supported by NSF CCF-1718820.

1 Introduction

In this paper, we introduce and study certified algorithm, describe their properties, present a framework for designing certified algorithms, provide examples of certified algorithms for Max Cut/Min Uncut, Minimum Multiway Cut, k-medians and k-means. Recall the definition of an instance perturbation, which was given by Bilu and Linial [10].

▶ **Definition 1.** *Consider a combinatorial optimization or clustering problem. Suppose that every instance has a number of parameters $p_1, \dots, p_m > 0$; for example, if the problem is a graph partitioning problem, the parameters are edge weights; if it is a constraint satisfaction problem, the parameters are constraint weights; if it is a clustering problem, the parameters are distances between points.*

Let $\gamma \geq 1$. An instance $\mathcal{I}'$ is a γ-perturbation of $\mathcal{I}$ if it differs from $\mathcal{I}$ only by the values of the parameters, and the parameters $p'_1, \dots, p'_m$ of $\mathcal{I}'$ satisfy the following inequality

$$p_i \leq p'_i \leq \gamma p_i \quad \text{for every } i \tag{1}$$

alternatively, we may require that

$$p_i/\gamma \leq p'_i \leq p_i \quad \text{for every } i. \tag{2}$$

Note that if $\gamma = 1$, then $\mathcal{I}' = \mathcal{I}$. Loosely speaking, the closer γ to 1 is, the closer $\mathcal{I}'$ to $\mathcal{I}$ is. All problems we consider are scale invariant, so it will not matter whether we use formula (1) or (2). It will be convenient to use (1) for combinatorial optimization problems and (2) for clustering problems. The central definition of this paper is that of a certified algorithm.

▶ **Definition 2.** *A γ-certified solution for instance $\mathcal{I}$ is a pair $(\mathcal{I}', s^*)$, where $\mathcal{I}'$ is a γ-perturbation of $\mathcal{I}$ and s^* is an optimal solution for $\mathcal{I}'$. A γ-certified algorithm (or a γ-certified approximation algorithm) is an algorithm that finds a γ-certified solution.*[1]

The definition of a certified algorithm is inspired by the notions of smoothed analysis [18] and perturbation-resilience (also known as Bilu-Linial stability) [10]. Recall that in the smoothed analysis framework, we analyze the performance of an algorithm on a small random perturbation of the input instance. That is, we show that, after we randomly perturb the input, the algorithm can solve it with the desired accuracy in the desired time. A certified approximation algorithm perturbs the input instance *on its own* and then solves the obtained instance exactly. Importantly, the perturbation does not have to be random or small. Now, let us talk about perturbation resilience.

▶ **Definition 3** ([10])**.** *An instance $\mathcal{I}$ is γ-perturbation-resilient*[2] *if every γ-perturbation of $\mathcal{I}$ has the same optimal solution as $\mathcal{I}$ (we require that $\mathcal{I}$ have a unique optimal solution).*

Bilu and Linial [10] initiated the study of perturbation resilience in 2010. Perturbation-resilient instances model practical instances, and the model is particularly well-suited for capturing problems arising in machine learning. As Bilu-Linial (as well as other authors, see [6, 9]) argued most practically relevant instances should be perturbation-resilient. Since the seminal paper by Bilu and Linial, there has been a lot of research on algorithms for perturbation-resilient instances (see e.g., [10, 9, 6, 16, 7, 3, 11, 8], see also [15] for a survey of known results) and by now there are a number of algorithms for perturbation-resilient instances of various graph partitioning, clustering, and other problems. A closely related notion to perturbation resilience is that of weak perturbation resilience.

▶ **Definition 4** ([16])**.** *Consider an instance $\mathcal{I}$. Let s^* be an optimal solution and N be a set of solutions, which includes all optimal solutions. Assume that for every γ-perturbation $\mathcal{I}'$ of $\mathcal{I}$, solution s^* is better than every $s \notin N$ with respect to the $\mathcal{I}'$ objective. Then $\mathcal{I}$ is (γ, N)-weakly perturbation-resilient. We say that an algorithm solves a weakly perturbation-resilient instance $\mathcal{I}$, if given a (γ, N)-weakly perturbation-resilient instance, it finds a solution $s \in N$ (crucially, the algorithm does not know N.)*

Intuitively, N is the set of solutions that are close to s^* in some sense. Say, N might be the set of solutions that are at most ε far from s^* in some metric or have similar structural properties to s^*. Note that $(\gamma, \{s^*\})$-weak perturbation resilience is equivalent to γ-perturbation resilience. In general, the requirement that an instance $\mathcal{I}$ be weakly perturbation-resilient is somewhat less restrictive than the one that $\mathcal{I}$ be perturbation-resilient.

---

[1] We call the solution "certified", because, as we will see later, $\mathcal{I}'$ "certifies" that s^* is a γ-approximation.
[2] Perturbation-resilient instances are also known as Bilu–Linial stable instances.

1.1 Overview: Properties of Certified Algorithms

Consider a γ-certified algorithm $\mathcal{A}$ for a constraint satisfaction or graph partitioning problem (or any problem whose objective is homogeneous of degree 1)[3]. First of all, $\mathcal{A}$ is also a γ-approximation algorithm, it *always* finds a γ-approximate solution. Then it exactly solves γ-perturbation-resilient instances and solves (γ, N)-weakly perturbation-resilient instances. Thus, we have both worst-case and beyond-worst-case guarantees for $\mathcal{A}$. We believe that this property is very desirable in practice. In particular, if our instance is indeed perturbation-resilient, we will find an exact solution; if it is not, we will find a reasonably good approximate solution. Note that other algorithms do not satisfy this property: state-of-the-art approximation algorithms do not solve perturbation-resilient instances and state-of-the-art algorithms for perturbation-resilient instances do not provide a good approximation if the instance is not perturbation-resilient. Additionally, $\mathcal{A}$ also satisfies some other properties; we prove in Theorems 10, 11, and 13 that:

- **Worst-Case Guarantees.** Algorithm $\mathcal{A}$ finds a γ-approximate solution for any instance $\mathcal{I}$; further, it finds a γ-approximate solution for the complimentary objective. For example, if the problem is a constraint satisfaction problem (CSP), then $\mathcal{A}$ finds a solution that is a γ-approximation for the problems of (1) maximizing the weight of the satisfied constraints and (2) minimizing the weight of the unsatisfied constraints. Additionally, if the problem is a CSP, $\mathcal{A}$ is also a γ-approximation algorithm for the variant of the problem with hard constraints.
- **Beyond-Worst-Case Guarantees.** Algorithm $\mathcal{A}$ exactly solves γ-perturbation-resilient instances and solves (γ, N)-weakly perturbation-resilient instances.

▶ Remark 5. The running time of most algorithms that we consider will depend not only on the size of the instance but also on the magnitude of the parameters. In a sense, we will assume that all parameters are given in the unary. More precisely, let ρ be the ratio between the largest and smallest parameters (for constraint satisfaction and graph partitioning problems, ρ is the ratio between the largest and smallest constraint/edge/node weights; for clustering problems, ρ is the aspect ratio of the given metric space, the ratio between the largest and smallest distances). Then the running time will depend polynomially on the input size and ρ. Thus we will call our algorithms *pseudo-polynomial-time algorithms*.

▶ **Definition 6.** *We say that a certified algorithm runs in pseudo-polynomial time if its running time is polynomial in the size of the input n and $\rho = \frac{\max_i p_i}{\min_i p_i}$.*

1.2 Our Results

The main contribution of this paper is conceptual rather than technical. We introduce the notion of a certified algorithm and prove that certified algorithms satisfy the properties listed above. We believe that certified algorithms will prove useful in developing new algorithms for solving worst-case and beyond-worst-case instances. We also believe that even if one is primarily interested in designing an algorithm for solving perturbation-resilient instances, it is often more convenient to design a certified algorithm (as it is guaranteed to solve perturbation-resilient instances).

[3] That is, the value of the objective multiplies by α when we multiply all the parameters (exactly) by α.

We provide a general framework for designing polynomial-time certified algorithms for combinatorial optimization problems and give examples of algorithms for combinatorial optimization and clustering problems. We also present some negative results. In establishing these results, we heavily use techniques from papers on perturbation resilience [16, 3, 11, 8, 12]. Specifically, we give pseudo-polynomial-time γ-certified algorithms for

- Min Uncut and Max Cut with $\gamma = O(\sqrt{\log n}\log\log n)$ (*cf.* the state-of-the-art approximation algorithm for Min Uncut gives an $O(\sqrt{\log n})$ approximation [1]),
- Minimum Multiway Cut with $\gamma = 2 - 2/k + \varepsilon_n$ for every ε_n such that $\varepsilon_n > 1/\operatorname{poly}(n)$ (*cf.* the state-of-the-art approximation algorithm gives a ≈ 1.296 approximation [17])
- k-medians with $\gamma = 3 + \varepsilon$ for every fixed $\varepsilon > 0$ (*cf.* the state-of-the-art algorithm gives a ≈ 2.732 approximation [14]).

We also observe that the algorithm for $(1+\varepsilon)$ -perturbation resilient instances of Euclidean k-means and k-medians by Friggstad, Khodamoradi, and Salavatipour [12] is $(1+\varepsilon)$-certified (see Theorem 24). Additionally, we show that there are no polynomial-time or pseudo-polynomial-time $O(n^{1-\delta})$-certified algorithms for Minimum Vertex Cover, Set Cover, and Min 2-Horn Deletion if $\mathcal{P} \neq \mathcal{NP}$ (for every fixed $\delta > 0$).

▶ Note (Follow-up work). In a follow-up paper [2], Angelidakis, Awasthi, Blum, Chatziafratis, and Dan use our framework to design a number of new certified algorithms for such problems as Euclidean Maximum Independent Set and Node Multiway Cut.

2 Preliminaries

We start with formally defining what an instance of a combinatorial optimization problem is.

▶ **Definition 7.** *An instance of a combinatorial optimization problem is specified by a set of feasible solutions $\mathcal{S}$ (the solution space), a set of constraints $\mathcal{C}$, and constraint weights $w_c > 0$ for $c \in \mathcal{C}$. Typically, the solution space $\mathcal{S}$ is of exponential size and is not provided explicitly. Each constraint is a map from $\mathcal{S}$ to $\{0,1\}$. We say that a feasible solution $s \in \mathcal{S}$ satisfies a constraint c in $\mathcal{C}$ if $c(s) = 1$.*

We consider maximization and minimization objectives.

- *The maximization objective is to maximize the total weight of the satisfied constraints: find $s \in \mathcal{S}$ that maximizes $\operatorname{val}_{\mathcal{I}}(s) \equiv \sum_{c\in\mathcal{C}} w_c c(s)$.*
- *The minimization objective is to minimize the total weight of the unsatisfied constraints: find $s \in \mathcal{S}$ that minimizes $\sum_{c\in\mathcal{C}} w_c(1-c(s)) = w(\mathcal{C}) - \operatorname{val}_{\mathcal{I}}(s)$ (where $w(\mathcal{C}) = \sum_{c\in\mathcal{C}} w(c)$ is the total weight of all the constraints).*

We say that maximization and minimization are complementary objectives. Weights $\{w_c\}_{c\in\mathcal{C}}$ are the parameters of the instance in the sense of Definition 1.

Note that s^* is an optimal solution for $\mathcal{I}$ with the maximization objective if and only if it is an optimal solution for $\mathcal{I}$ with the minimization objective. Accordingly, $(\mathcal{I}', s)$ is a γ-certified solution for $\mathcal{I}$ with the maximization objective if and only if it is a γ-certified solution for $\mathcal{I}$ with the minimization objective.

▶ **Definition 8.** *An optimization problem $\mathcal{P}$ is a family $\mathcal{F}$ of instances. All instances in $\mathcal{F}$ have a fixed objective; either all of them have a maximization or all have a minimization objective. We assume that if instance $(\mathcal{S}, \mathcal{C}, w)$ is in $\mathcal{F}$, then so is $(\mathcal{S}, \mathcal{C}, w')$ for any choice of weights $w_c > 0$.*

This definition captures various constraint satisfaction, graph partitioning, and covering problems. Consider for example Min Uncut. In Min Uncut, the goal is to find a cut $(S, \bar{S})$ in a given graph $G = (V, E, w)$ that minimizes the total weight of the uncut edges (edges that

connect vertices within S or within $\bar{S}$). For Min Uncut, $\mathcal{S}$ is the set of all cuts in G. There is a constraint c_e for every edge $e \in E$; a cut satisfies constraint c_e if and only if it cuts edge e. The objective is to minimize $\sum_{c\in\mathcal{C}} w_c(1 - c((S, \bar{S})))$. The objective for the complementary problem, Max Cut, is $\sum_{c\in\mathcal{C}} w_c c((S, \bar{S}))$.

There are also certified algorithms for clustering problems. In this paper, we describe a $(3 + \varepsilon)$-certified algorithm for k-medians and note that algorithms for Euclidean k-means and k-medians from [12] are $(1 + \varepsilon)$-certified. Recall the definition of k-medians.

▶ **Definition 9.** *In k-medians, we are a given a set of points X, metric d on X, which satisfies triangle inequalities, and a parameter $k \geq 1$. The cost of a cluster $C \subset X$ is*

$$\operatorname{cost} C = \min_{c\in X} \sum_{u\in C} d(u, c).$$

The goal is to partition X into k clusters $C_1, \dots, C_k$ so as to minimize their total cost $\sum_{i=1}^{k} \operatorname{cost} C_i$. The parameters of the problem (in the sense of Definition 1) are the pairwise distances $d(u, v)$.

We say that c is an optimal center for C, if $c \in \arg\min_{c\in X} \sum d(u, c)$. Note that a set of centers $c_1, \dots, c_k$ defines a clustering $C_1, \dots, C_k$ of X; namely, $C_1, \dots, C_k$ is the Voronoi partition for $c_1, \dots, c_k$ (if there are ties, several partitions may correspond to the same set of centers).

A γ-perturbation of an instance (X, d) is an instance (X, d'), such that $\frac{1}{\gamma} d(u, v) \leq d'(u, v) \leq d(u, v)$ for every $u, v \in X$. Note that we do not require that d' satisfy triangle inequalities (if we did, we would get the definition of a metric perturbation; see [3] for details).

3 Properties of Certified Algorithms

In this section, we prove that certified algorithms satisfy the properties we described in Section 1.

▶ **Theorem 10.** *Consider a γ-certified algorithm $\mathcal{A}$ for a combinatorial optimization problem.*

- *$\mathcal{A}$ finds a solution that is a γ-approximate solution w.r.t. both the maximization and minimization objectives.*
- *If the instance is γ-perturbation-resilient, $\mathcal{A}$ finds the optimal solution. If it is (γ, N)-weakly perturbation-resilient, $\mathcal{A}$ finds a solution in N.*

Proof. Consider an instance $\mathcal{I}$. Denote its optimal solution by s^*. Denote the certified solution found by $\mathcal{A}$ by $(\mathcal{I}', s')$. For each $c \in \mathcal{C}$, let w_c and w'_c be its weights in $\mathcal{I}$ and $\mathcal{I}'$, respectively.

I. First, we prove that the algorithm always gives a γ-approximation for both objectives. Consider the maximization objective. We have,

$$\operatorname{val}_{\mathcal{I}}(s') = \sum_{c\in\mathcal{C}} w_c c(s') \geq \sum_{c\in\mathcal{C}} \frac{w'_c}{\gamma} c(s') = \frac{1}{\gamma} \sum_{c\in\mathcal{C}} w'_c c(s')$$
$$\overset{(\star)}{\geq} \frac{1}{\gamma} \sum_{c\in\mathcal{C}} w'_c c(s^*) \geq \frac{1}{\gamma} \sum_{c\in\mathcal{C}} w_c c(s^*) = \frac{\operatorname{val}_{\mathcal{I}}(s^*)}{\gamma}$$

where $(\star)$ holds since s' is an optimal solution for $\mathcal{I}'$. We conclude that s' is a γ-approximate solution for the maximization objective. Similarly, we analyze the minimization objective.

$$\sum_{c\in\mathcal{C}} w_c(1 - c(s')) \leq \sum_{c\in\mathcal{C}} w'_c(1 - c(s')) \leq \sum_{c\in\mathcal{C}} w'_c(1 - c(s^*)) \leq \gamma \sum_{c} w_c(1 - c(s^*)).$$

	problem $\mathcal{P}_{hard}$	problem $\mathcal{P}$
unperturbed instance	$\mathcal{I}$	$\mathcal{J}$
perturbed instance	$\mathcal{I}'$	$\mathcal{J}'$

Figure 1 Instances $\mathcal{I}$, $\mathcal{J}$, $\mathcal{I}'$, and $\mathcal{J}'$. If s is a feasible solution for $\mathcal{I}'$, $\mathrm{val}_{\mathcal{I}'}(s) = \mathrm{val}_{\mathcal{J}'}(s) - W_H$.

II. Now, assume that $\mathcal{I}$ is γ-perturbation-resilient. By the definition of perturbation resilience, $\mathcal{I}$ and $\mathcal{I}'$ have the same optimal solution. Thus, s^* is an optimal solution not only for $\mathcal{I}'$ but also for $\mathcal{I}$. Finally, assume that $\mathcal{I}$ is (γ, N)-weakly perturbation-resilient. Since $\mathcal{I}$ is (γ, N)-weakly perturbation-resilient and $\mathcal{I}'$ is a γ-perturbation of $\mathcal{I}$, we get form Definition 4 that either $s' \in N$ or s^* is better than s' w.r.t. to the $\mathcal{I}'$ objective. The latter is not possible, since s' is an optimal solution for $\mathcal{I}'$. We conclude that $s' \in N$. ◀

Consider an instance of an optimization problem. We may choose a subset of constraints $H \subset \mathcal{C}$ and require that all of them are satisfied. We call them *hard constraints* and the obtained instance an instance with hard constraints. Formally, given an instance $(\mathcal{S}, \mathcal{C}, w)$ and a subset of constraints H, we define the correspondent instance $(\mathcal{S}', \mathcal{C}', w)$ with hard constraints as follows: $\mathcal{S}' = \{a \in \mathcal{S} : c(s) = 1 \text{ for every } c \in H\}$; $\mathcal{C}' = \mathcal{C} \setminus H$; $w'(c) = w(c)$ for $c \in \mathcal{C}'$.

▶ **Theorem 11.** *Assume that $\gamma = \gamma_n$ is at most polynomial in n. If there is a pseudo-polynomial-time γ-certified algorithm for a problem $\mathcal{P}$, then there is also a pseudo-polynomial-time γ-certified algorithm for a variant $\mathcal{P}_{hard}$ of $\mathcal{P}$ with hard constraints.*

Proof. Consider an instance $\mathcal{I}$ of $\mathcal{P}_{hard}$. Let H be the set of hard constraints and S be the set of soft constraints. We transform $\mathcal{I}$ to an instance $\mathcal{J}$ of $\mathcal{P}$ by setting the weight of every hard constraint $c \in H$ to $(\gamma+1)W$ where $W = \sum_{c \in S} w_c$ is the total weight of the soft constraints (see Figure 1). Note that the parameter $\rho_{\mathcal{J}}$ for $\mathcal{J}$ (the ratio between the largest and smallest constraint weights in $\mathcal{J}$; see Definition 6) is polynomial in $\rho_{\mathcal{I}}$ and the input size:

$$\rho_{\mathcal{J}} = \frac{(\gamma+1)W}{\min_{c \in \mathcal{C}} w_c} \leq \frac{|S|(\gamma+1)\max_{c \in \mathcal{C}} w_c}{\min_{c \in \mathcal{C}} w_c} \leq |S|(\gamma+1)\rho_{\mathcal{I}}.$$

We run the algorithm for $\mathcal{P}$ on input $\mathcal{J}$ and get a certified solution $(\mathcal{J}', s^*)$. Let $W_H = \sum_{c \in H} w_c^{\mathcal{J}'}$, where $w_c^{\mathcal{J}'}$ is the weight of c in $w_c^{\mathcal{J}'}$. Every feasible solution s for $\mathcal{I}$ satisfies all the constraints in H and thus $\mathrm{val}_{\mathcal{J}'}(s) \geq W_H$. In particular, if $\mathcal{I}$ has a feasible solution s, then $\mathrm{val}_{\mathcal{J}'}(s^*) \geq \mathrm{val}_{\mathcal{J}'}(s) \geq W_H$. Conversely, every solution s for $\mathcal{J}'$ with $\mathrm{val}_{\mathcal{J}'}(s) \geq W_H$ satisfies all the hard constraints (since the total weight of the soft constraints is less than the weight of any hard constraint in $\mathcal{J}'$) and thus is a feasible solution for $\mathcal{I}$.

If $\mathrm{val}_{\mathcal{J}'}(s^*) < W_H$, we report that $\mathcal{I}$ has no feasible solution. Otherwise, s^* is a feasible solution for $\mathcal{I}$. We let $\mathcal{I}'$ be a perturbation of $\mathcal{I}$, in which all the soft constraints have the same weights as in $\mathcal{J}'$. Observe that for every feasible solution s of $\mathcal{I}'$, $\mathrm{val}_{\mathcal{I}'}(s) = \mathrm{val}_{\mathcal{J}'}(s) - W_H$. It follows that s^* is an optimal solution for $\mathcal{I}'$. We output $(\mathcal{I}', s^*)$. ◀

▶ **Corollary 12.** *Consider a problem $\mathcal{P}$ and its variant with hard constraints $\mathcal{P}_{hard}$. Let γ_n be at most polynomial in n (the instance size). If there is a pseudo-polynomial-time γ_n-certified algorithm for $\mathcal{P}$, then there are pseudo-polynomial-time γ_n-approximation algorithms for maximization and minimization variants of $\mathcal{P}_{hard}$.*

We prove an analog of Theorem 10 for k-medians and k-means in Appendix (its proof is very similar to that of Theorem 10).

▶ **Theorem 13.** *Consider a γ-certified algorithm $\mathcal{A}$ for k-medians or k-means. If $\mathcal{A}$ is for k-medians, then $\mathcal{A}$ is a γ-approximation algorithm; if $\mathcal{A}$ is for k-means, then $\mathcal{A}$ is a γ^2-approximation algorithm. If the instance is γ-perturbation-resilient, $\mathcal{A}$ finds the optimal solution. If it is (γ, N)-weakly perturbation-resilient, $\mathcal{A}$ finds a solution in N.*

4 Designing Certified Algorithms

4.1 Iterative Improvement Algorithms

We use an iterative approach to design certified algorithms: our certified algorithms start with an arbitrary solution and then iteratively improve it. The approach resembles that of local search, except that improvements will not necessarily be local. In this approach, the key component of a certified algorithm is a procedure for improving the current solution; namely, a procedure for solving the following task.

▶ **Task 14.** *Given an instance $\mathcal{I}$ and the current solution s,*
- *either find a new solution s', which is better than s (w.r.t. to the $\mathcal{I}$ objective), or*
- *find a γ certified solution $(\mathcal{I}', s)$.*

▷ Claim 15. Consider a combinatorial optimization or clustering problem. Let $\varepsilon = \varepsilon_n > 1/\operatorname{poly}(n)$. Assume the following.
1. There is a polynomial-time algorithm for Task 14 with $\gamma = \gamma_n$.
2. There is polynomial-time algorithm that finds a feasible solution.

Then there is a pseudo-polynomial-time $(1+\varepsilon_n)\gamma_n$-certified algorithm for the problem.

Proof. Let $p_{min} = \min_i p_i$ be the smallest parameter and $\varepsilon' = \varepsilon/2$. First, we apply a preprocessing step, where we round all parameters p_i to multiples of $q = \varepsilon' p_{min}$ as follows:

$$p'_i = (\lceil p_i/q \rceil + 1)q.$$

It is easy to see that $q \le p'_i - p_i \le 2q$. Thus, instance $\mathcal{I}'$ with parameters p'_i is a $(1+\varepsilon)$ perturbation of $\mathcal{I}$.

If the problem is k-medians (or for that matter another clustering problem), then parameters p_i are distances $d(u,v)$, satisfying triangle inequalities. Then the new distances $d'(u,v)$ also satisfy triangle inequalities:

$$d'(u,v) + d'(v,w) \ge (d(u,v)+q) + (d(v,w)+q) \ge d(u,w) + 2q \ge d'(u,w).$$

Now we proceed as follows. We find a feasible solution for $\mathcal{I}'$ and then iteratively improve it using the procedure for Task 14, until the procedure finds a γ-certified solution $(\mathcal{I}'', s)$ for $\mathcal{I}'$. Clearly, $\mathcal{I}''$ is a $(1+\varepsilon)\gamma$-perturbation of $\mathcal{I}$. Thus, $(\mathcal{I}'', s)$ is a $(1+\varepsilon)\gamma$-certified solution for $\mathcal{I}$.

It remains to prove that the algorithm runs in pseudo-polynomial time. To do so, we need to upper bound the number of iterations. Assume that the problem is a combinatorial optimization problem. Consider the maximization objective. Initially the value of the problem is non-negative. It increases by at least q in every iteration. When the program terminates, it is at most $\sum_{c\in\mathcal{C}} w'_c \le (\rho w_{\min} + 2q)|\mathcal{C}|$ (where ρ as in Definition 6). Thus the algorithm performs at most $\frac{(\rho w_{\min}+2q)|\mathcal{C}|}{q} \le 2(\rho/\varepsilon+1)|\mathcal{C}|$ iterations, which is polynomial in $|\mathcal{C}|$, ρ, and $1/\varepsilon$.

If the problem is k-medians, the cost of the initial clustering is at most $n \max_{u,v\in X} d'(u,v)$. It decreases by at least q in every iteration. The cost of the obtained clustering is non-negative. It is easy to see that the number of iterations is polynomial in n, ρ, and $1/\varepsilon$.

We conclude that the algorithm runs in pseudo-polynomial time. ◁

4.2 Designing Certified Algorithms for Combinatorial Optimization Problems

Consider a combinatorial optimization problem. We show that in order to solve Task 14, it suffices to solve the following task.

▶ **Task 16.** *Assume that we are given an instance $\mathcal{I} = (\mathcal{S}, \mathcal{C}, w)$, a partition of its constraints $\mathcal{C} = C_1 \cup C_2$, and a parameter $\gamma \geq 1$. We need to either*
- *find $s \in \mathcal{S}$ such that $\gamma \sum_{c \in C_1} w_c c(s) > \sum_{c \in C_2} w_c (1 - c(s))$, or*
- *report that for every $s \in \mathcal{S}$: $\sum_{c \in C_1} w_c c(s) \leq \sum_{c \in C_2} w_c (1 - c(s))$.*

(Note that the above options are not mutually exclusive.)

▶ **Theorem 17.** *Assume that (1) there is a polynomial-time procedure for Task 16 with $\gamma = \gamma_n$ and (2) there is a polynomial-time algorithm that finds some $s \in \mathcal{S}$. Then there exists a pseudo-polynomial-time $(\gamma_n + \varepsilon_n)$-certified algorithm for the problem (for every $\varepsilon_n > 1/\operatorname{poly}(n)$).*

Proof. By Claim 15, it suffices to design an algorithm for Task 14. Given a solution s, we will either find a better solution s' or return a certified solution $(\mathcal{I}', s)$. Let $C_1 = \{c \in \mathcal{C} : c(s) = 0\}$ and $C_2 = \{c \in \mathcal{C} : c(s) = 1\}$ be the sets of unsatisfied and satisfied constraints, respectively. Define weights w' as follows:

$$w'_c = \begin{cases} w_c, & \text{if } c \in C_1 \\ \gamma w_c, & \text{if } c \in C_2 \end{cases}$$

We run the procedure for Task 16 on instance $\mathcal{I}' = (\mathcal{S}, \mathcal{C}, w')$. Consider two cases. Assume first that the procedure returns a solution s' such that $\gamma \sum_{c \in C_1} w'_c c(s') > \sum_{c \in C_2} w'_c (1 - c(s'))$. We get that

$$\sum_{c \in C_1} w_c c(s') > \sum_{c \in C_2} w_c (1 - c(s'))$$

and thus

$$\operatorname{val}_{\mathcal{I}}(s') = \sum_{c \in C_1 \cup C_2} w_c c(s') > \sum_{c \in C_2} w_c = \operatorname{val}_{\mathcal{I}}(s).$$

In this case, we return s'. Assume now that the procedure reports that for every solution s':

$$\sum_{c \in C_1} w'_c c(s') \leq \sum_{c \in C_2} w'_c (1 - c(s'))$$

or, equivalently,

$$\operatorname{val}_{\mathcal{I}'}(s') = \sum_{c \in C_1 \cup C_2} w'_c c(s') \leq \sum_{c \in C_2} w'_c = \operatorname{val}_{\mathcal{I}'}(s).$$

Then s is an optimal solution for $\mathcal{I}'$. We return a γ-certified solution $(\mathcal{I}', s)$. ◀

4.3 Certified Algorithm via Convex Relaxations

We describe how to design certified algorithms for combinatorial optimization problems using convex relaxations. Consider a problem and a convex relaxation for it. We refer to problem solutions $s \in \mathcal{S}$ as *combinatorial* solutions and relaxation solutions x as *fractional* solutions;

we say that x is *integral* if it corresponds to a combinatorial solution $s \in \mathcal{S}$. We assume that in the relaxation we have a variable x_c for each constraint c so that $x_c = c(s)$ for every integral solution x and corresponding combinatorial solution s. The relaxations objective is to maximize $\text{fval}(x) = \sum_{c\in\mathcal{C}} w_c x_c$ or (equivalently) minimize $\sum_{c\in\mathcal{C}} w_c(1-x_c)$.

Consider a *randomized* rounding scheme $\mathcal{R}$ that given a fractional solution x outputs a combinatorial solution $\mathcal{R}(x)$. Rounding schemes are widely used for designing approximation algorithms. When designing an algorithm for a *maximization* objective, one typically wants the rounding scheme to satisfy the following approximation condition.

- **Approximation Condition.** The probability that each constraint $c \in \mathcal{C}$ is satisfied by $\mathcal{R}(x)$ is at least x_c/α (the probability is over the random choices made by $\mathcal{R}$).

If $\mathcal{R}$ satisfies this condition, it gives an α approximation for the maximization objective (in expectation). On the other hand, when designing an algorithm for a *minimization* objective, one wants the rounding scheme to satisfy the co-approximation condition.

- **Co-approximation Condition.** The probability that each constraint $c \in \mathcal{C}$ is unsatisfied by $\mathcal{R}(x)$ is at most $\beta(1-x_c)$.

If $\mathcal{R}$ satisfies this condition, it gives a β approximation for the minimization objective (in expectation). Following [16, 3], we consider rounding schemes that simultaneously satisfy the approximation and co-approximation conditions.

▶ **Definition 18.** *We say that a rounding scheme $\mathcal{R}$ is an (α,β)-rounding if it simultaneously satisfies the approximation and co-approximation conditions with parameters α and β.*

We note that (α,β)-rounding schemes have been shown to be very useful for solving perturbation-resilient and weakly perturbation-resilient instances [16, 3]. In particular, if there is an (α,β)-rounding scheme, then the relaxation is integral for $(\alpha\beta)$-perturbation-resilient instances [16]. We now show that (α,β)-rounding schemes can be used for designing certified algorithms.

▶ **Theorem 19.** *Assume that there exists an (α,β)-rounding scheme $\mathcal{R}$. Additionally, assume that the support of $\mathcal{R}$ is of polynomial size and can be found in polynomial time.*

Then there exists a pseudo-polynomial-time certified $(\gamma+\varepsilon_n)$-approximation algorithm for the problem where $\gamma = \alpha\beta$ (for any $\varepsilon > 1/\operatorname{poly}(n)$). Further, the algorithm outputs a certified solution $(\mathcal{I}', s^*)$ such that s^* is an optimal solution not only for $\mathcal{I}'$ but also for the relaxation for $\mathcal{I}'$.

Proof. By Theorem 17, it suffices to design a polynomial-time procedure for solving Task 16. First, we solve the convex relaxation for the problem and obtain an optimal fractional solution $x = x^*$. If $\sum_{c\in C_1} w_c x_c \le \sum_{c\in C_2} w_c(1-x_c)$, then for every $s \in \mathcal{S}$

$$\sum_{c\in C_1} w_c c(s) + \sum_{c\in C_2} w_c c(s) \le \sum_{c\in C_1} w_c x_c + \sum_{c\in C_2} w_c x_c \le \sum_{c\in C_2} w_c(1-x_c) + \sum_{c\in C_2} w_c x_c = \sum_{c\in C_2} w_c. \quad (3)$$

So we report that $\sum_{c\in C_1} w_c c(s) \le \sum_{c\in C_2} w_c(1-c(s))$ for every s (option 2). Note that in this case, the certified algorithm from Theorem 17 returns a certified solution $(\mathcal{I}, s^*)$ of value $\text{val}_{\mathcal{I}'}(s^*) = w(C_2) \equiv \sum_{c\in C_2} w_c$. Eq. (3) shows that the value of every fractional solution (let alone integral) is at most $w(C_2)$. Thus s^* is an optimal solution not only for $\mathcal{I}'$ but also for the relaxation for $\mathcal{I}'$.

Assume now that $\sum_{c\in C_1} w_c x_c > \sum_{c\in C_2} w_c(1-x_c)$. We apply rounding scheme $\mathcal{R}$ and obtain a solution $\mathcal{R}(x)$. From the approximation and co-approximation conditions, we get

$$\mathbf{E}\left[\gamma \sum_{c\in C_1} w_c c(\mathcal{R}(x)) - \sum_{c\in C_2} w_c(1-c(\mathcal{R}(x)))\right] \ge \frac{\gamma}{\alpha}\sum_{c\in C_1} w_c x_c - \beta \sum_{c\in C_2} w_c(1-x_c) > 0.$$

Here, we used that $\gamma = \alpha\beta$. Thus for some solution s in the support of $\mathcal{R}(x)$, we have $\gamma\sum_{c\in C_1} w_c c(s) > \sum_{c\in C_2} w_c(1-c(s))$. We find and return such a solution s. ◀

We note that it is sufficient to design a rounding procedure only for solutions that are close to integral solutions.

▶ **Definition 20.** *Let us say that a fractional solution x is δ-close to an integral if $x = (1-\delta)x^{int} + \delta x^{frac}$ for some integral solution x^{int} and fractional solution x^{frac}. Rounding scheme $\mathcal{R}$ is a δ-local (α, β)-rounding if it is defined and satisfies the approximation and co-approximation conditions for fractional solutions x that are δ-close to an integral solution.*

It turns out that it is sufficient to have a δ-local rounding scheme (for any $\delta > 0$) in Theorem 19. To see that, we slightly change the proof of Theorem 19. We first find an optimal fractional solution x^* and then apply the rest of the argument to solution $x = \delta x^* + (1-\delta)x^s$ (where x^s is the fractional solution corresponding to s).

Finally, we note that if the support of $\mathcal{R}$ is not of a polynomial size, we can get a *randomized* certified algorithm. To do so, instead of trying out all solutions s in the support of $\mathcal{R}(x)$, we apply $\mathcal{R}$ to x sufficiently many times and let s be the best of the obtained solutions (if we use a δ-local rounding scheme, we need that $\delta > 1/\operatorname{poly}(n)$).

5 Examples of Certified Algorithms for Optimization Problems

▶ **Theorem 21.**

I. *There exists a pseudo-polynomial-time $(1+\varepsilon_n)\alpha_n$-certified algorithm for Min Uncut and Max Cut (these problems are complementary), where $\alpha_n = O(\sqrt{\log n}\log\log n)$ is the approximation factor for Sparsest Cut with Non-uniform Demands from [4] and $\varepsilon_n > 1/\operatorname{poly}(n)$.*

II. *There exists a pseudo-polynomial-time $(2 - 2/k + \varepsilon_n)$-certified algorithm for Minimum Multiway Cut.*

Proof.

I. We show how to solve Task 16 in polynomial time. Recall that in our formulation of Min Uncut, $c_e((S,\bar{S})) = 1$ if edge e is cut. Let $E_1 = \{e \in E : c_e \in C_1\}$ and $E_2 = \{e \in E : c_e \in C_2\}$; denote the weight of the edges in E_i cut by $(S,\bar{S})$ by $w(E_i(S,\bar{S}))$. Let $\phi(S) = \frac{w(E_2(S,\bar{S}))}{w(E_1(S,\bar{S}))}$. Then our goal is to either find a cut $(S,\bar{S})$ such that $\phi(S) < \gamma$ or report that $\phi(S) \geq 1$ for every $(S,\bar{S})$. Now the problem of minimizing $\phi(S)$ over all cuts $(S,\bar{S})$ is the same as finding the sparsest cut with non-uniform demands in graph (V, E_2) with edge capacities w, demand pairs E_1, and demand weights w. We run the approximation algorithm for Sparsest Cut and get a cut $(S,\bar{S})$ that approximately – within a factor of γ – minimizes $\phi(S)$. If $\phi(S) < \gamma$, we report cut $(S,\bar{S})$; otherwise, we report that $\phi(S') \geq 1$ for every cut $(S',\bar{S}')$.

II. Consider the LP relaxation for Minimum Multiway Cut by Călinescu, Karloff, and Rabani. It is shown in [3] that there exists a δ-local (α,β)-rounding procedure for it with $\alpha\beta = 2 - 2/k$. It follows from Theorem 19, that there is a $(2 - 2/k + \varepsilon_n)$-certified algorithm. ◀

6 $(3+\varepsilon)$-Certified Local Search Algorithm for k-medians

In this section, we consider k-medians. We show that a local search algorithm is $(3+\varepsilon)$-certified. We note that our analysis is very similar to that in [11, 8].

We first apply the preprocessing step form Theorem 17 (where we round all distances to multiples of some q). Then, we consider an arbitrary set of centers $c_1, \ldots, c_k$ and the corresponding clustering $C_1, \ldots, C_k$. Then, at each iteration, we go over all possible swaps

of size r: we swap r centers among $c_1, \dots, c_k$ with r points outside of $c_1, \dots, c_k$. We choose a swap that decreases the cost of the clustering, if there is one, and recompute $C_1, \dots, C_k$. If there is no such swap, we terminate and output a certified solution $((X, d'), (C_1, \dots, C_k))$, where $d'(u, v) = \frac{1}{\gamma} d(u, v)$ if $u = c_i$ and $v \in C_i$ for some i (or the other way around), and $d'(u, v) = d(u, v)$, otherwise.

▶ **Theorem 22.** *The ρ-local search algorithm for k-medians (described above) is a pseudo-polynomial-time $(3 + O(1/\rho))$-certified algorithm.*

Proof. We use Theorem 14. It guarantees that the algorithm runs in pseudo-polynomial time. We need to show that when the algorithm terminates it indeed outputs a certified solution. Suppose that the algorithm outputs a clustering with centers $L = \{l_1, \dots, l_k\}$.

Consider an arbitrary set of centers $S = \{s_1, \dots, s_k\}$. We need to show that the cost of the k-median clustering with centers in L is at most the cost of the k-median clustering with centers in S with respect to the perturbed distance function d'. Let $l(u)$ and $s(u)$ be the closest centers to point u in L and S respectively with respect to d; and let $l'(u)$ and $s'(u)$ be the closest centers to point u in L and S respectively with respect to d'. Our goal is to prove that

$$\sum_{u \in X} d'(u, l'(u)) \le \sum_{u \in X} d'(u, s'(u)). \tag{4}$$

Observe that for every point $u \in X$, we have $d(u, v) = d'(u, v)$ for all v but $v = l(u)$. Thus, $l'(u) = l(u)$ and $d'(u, l'(u)) = d(u, l(u))/\gamma$. Consequently, the left hand side of (4) equals $\sum_{u \in X} d(u, l(u))/\gamma$. Similarly, $s'(u) = s(u)$ and $d'(u, s'(u)) = d(u, s(u))$ if $l(u) \notin S$. However, if $l(u) \in S$, then $d'(u, s'(u)) = \min\big(d(u, s(u)), d(u, l(u))/\gamma\big)$ as, in this case, the optimal center for u in S w.r.t. d' can be $l(u)$.

Let us split all vertices in X into two groups $A = \{u : l(u) \in S\}$ and $B = \{u : l(u) \notin S\}$. Then, for $u \in A$, we have $d'(u, s'(u)) = \min\big(d(u, s(u)), d(u, l(u))/\gamma\big)$; and for for $u \in B$, we have $d'(u, s'(u)) = d(u, s(u))$. Thus, inequality (4) is equivalent to

$$\sum_{u \in X} \frac{d(u, l(u))}{\gamma} \le \sum_{u \in A} \min\left(d(u, s(u)), \frac{d(u, l(u))}{\gamma}\right) + \sum_{u \in B} d(u, s(u)),$$

which after multiplying both parts by γ can be written as

$$\sum_{u \in X} d(u, l(u)) \le \sum_{u \in A} \min\Big(\gamma d(u, s(u)), d(u, l(u))\Big) + \sum_{u \in B} \gamma d(u, s(u)). \tag{5}$$

For $u \in A$, we have $d(u, s(u)) \le d(u, l(u))$ since both $s(u)$ and $l(u)$ are in S and $s(u) = \arg\min_{v \in S} d(u, v)$. Thus, $\min\big(\gamma d(u, s(u)), d(u, l(u))\big) \ge d(u, s(u))$. Consequently, inequality (5) follows from the following theorem from [11] (see also [5] and [13]).

▶ **Theorem 23** (Local Approximation; [11])**.** *Let L be a r-locally optimal set of centers with respect to a metric d and S be an arbitrary set of k centers. Define sets A and B as above. Then,*

$$\sum_{u \in X} d(u, l(u)) \le \sum_{u \in A} d(u, s(u)) + \gamma \sum_{u \in B} d(u, s(u)),$$

for some $\gamma = 3 + O(1/r)$. ◀

7 Euclidean k-means and k-medians

We note that the algorithm for $(1+\varepsilon)$-perturbation-resilient instances of Euclidean k-means and k-medians (in a fixed dimensional space) by Friggstad, Khodamoradi, and Salavatipour [12] is $(1+\varepsilon)$-certified. The algorithm finds a solution $\mathcal{S}$ and perturbed metric δ' on $X \cup \mathcal{S}$ such that (see Lemma 2.2 in [12])

$$\text{cost}'(\mathcal{S}) \leq \text{cost}'(\mathcal{O}),$$

where cost$'$ is the cost of the clustering w.r.t the perturbed metric δ'. In [12], $\mathcal{O}$ is the optimal solution for the non-perturbed instance; however, the proof does not use that $\mathcal{O}$ is an optimal solution and goes through if $\mathcal{O}$ is an arbitrary solution. Thus, Friggstad, Khodamoradi, and Salavatipour proved that their algorithm finds a solution $\mathcal{S}$ (specified by the list of centers), which is optimal w.r.t. the perturbed distances δ'. We get the following theorem.

▶ **Theorem 24.** *For every fixed $\varepsilon > 0$ and $d \geq 1$, there exist $(1+\varepsilon)$-certified algorithms for Euclidean instances of k-means and k-medians in $\mathbb{R}^d$ (namely, the local search algorithms from [12]). The algorithms run in time polynomial in the bit complexity of the input.*

8 Negative Results

In this section, we present several negative results for certified algorithms. They immediately follow from the properties of certified algorithm we saw in Section 3. We note that similar negative results were shown in [3] for robust algorithms for perturbation-resilient instances.

▶ **Theorem 25.** *There are no pseudo-polynomial-time $O(n^{1-\delta})$-certified algorithms for Minimum Vertex Cover, Set Cover, and Min 2-Horn Deletion if $\mathcal{P} \neq \mathcal{NP}$ (for every fixed $\delta > 0$).*

Proof. Let $\gamma = cn^{1-\delta}$ be the hardness of Maximum Independent Set (MIS) [19].

I. According o Theorem 10, if there were a pseudo-polynomial-time or polynomial-time γ-certified algorithm for Vertex Cover, then there would be a polynomial-time γ-approximation algorithm for Maximum Independent Set (the problem complementary to Minimum Vertex Cover).

II. Since each Vertex Cover instance is also a Set Cover instance, a certified algorithm for Set Cover would also be a certified algorithm for Vertex Cover.

III. Observe that Max 2-Horn SAT with hard constraints is more difficult than MIS. An instance (G, V, E) of MIS is equivalent to the following instance of Max 2-Horn SAT with hard constraints: there is a variable x_u for every $u \in V$, a hard constraint $x_u \vee x_v$ for every $(u, v) \in E$, and a soft constraint x_u. By Corollary 12, since there is no γ-approximation for MIS, there is no polynomial-time algorithm for γ-perturbation-resilient instances of Min 2-Horn Deletion. ◀

References

1 Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. $O(\sqrt{\log n})$ approximation algorithms for Min UnCut, Min 2CNF Deletion, and directed cut problems. In *Proceedings of the Symposium on Theory of Computing*, pages 573–581, 2005.

2 Haris Angelidakis, Pranjal Awasthi, Avrim Blum, Vaggos Chatziafratis, and Chen Dan. Bilu-Linial stability, certified algorithms and the Independent Set problem. In *Proceedings of the European Symposium on Algorithms*, 2019.

3 Haris Angelidakis, Konstantin Makarychev, and Yury Makarychev. Algorithms for stable and perturbation-resilient problems. In *Proceedings of the Symposium on Theory of Computing*, pages 438–451, 2017.

4 Sanjeev Arora, James Lee, and Assaf Naor. Euclidean distortion and the sparsest cut. *Journal of the American Mathematical Society*, 21(1):1–21, 2008.

5 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on computing*, 33(3):544–562, 2004.

6 Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 112(1-2):49–54, 2012.

7 Maria-Florina Balcan, Nika Haghtalab, and Colin White. k-Center Clustering Under Perturbation Resilience. In *International Colloquium on Automata, Languages, and Programming*, 2016.

8 Maria-Florina Balcan and Colin White. Clustering under local stability: Bridging the gap between worst-case and beyond worst-case analysis. *arXiv preprint*, 2017. `arXiv:1705.07157`.

9 Yonatan Bilu, Amit Daniely, Nati Linial, and Michael Saks. On the practically interesting instances of MAXCUT. In *International Symposium on Theoretical Aspects of Computer Science*, 2013.

10 Yonatan Bilu and Nathan Linial. Are Stable Instances Easy? In *Innovations in Computer Science*, pages 332–341, 2010.

11 Vincent Cohen-Addad and Chris Schwiegelshohn. On the local structure of stable clustering instances. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 49–60, 2017.

12 Zachary Friggstad, Kamyar Khodamoradi, and Mohammad R. Salavatipour. Exact Algorithms and Lower Bounds for Stable Instances of Euclidean K-means. In *Proceedings of the Symposium on Discrete Algorithms*, pages 2958–2972, 2019.

13 Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. *arXiv preprint*, 2008. `arXiv:0809.2554`.

14 Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. *SIAM Journal on Computing*, 45(2):530–547, 2016.

15 Konstantin Makarychev and Yury Makarychev. Bilu–Linial Stability. In T. Hazan, G. Papandreou, and D. Tarlow, editors, *Perturbations, Optimization, and Statistics*, chapter 13. MIT Press, 2016.

16 Konstantin Makarychev, Yury Makarychev, and Aravindan Vijayaraghavan. Bilu-Linial stable instances of Max Cut and Minimum Multiway Cut. In *Proceedings of the Symposium on Discrete Algorithms*, pages 890–906, 2014.

17 Ankit Sharma and Jan Vondrák. Multiway Cut, Pairwise Realizable Distributions, and Descending Thresholds. In *Proceedings of the Symposium on Theory of Computing*, 2014.

18 Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004.

19 David Zuckerman. Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. In *Proceedings of the Symposium on Theory of Computing*, 2006.

A Proof of Theorem 13

Proof.

I. Denote the certified solution returned by the algorithm by $((X, d'), (C_1, \dots, C_k))$. Let $C_1^*, \dots, C_k^*$ be an optimal clustering. Let c_i and c_i^* be optimal centers of C_i and C_i^* (respectively). Consider the case of k-medians first. We upper bound the cost of $(C_1, \dots, C_k)$ w.r.t d as follows:

$$\sum_{i=1}^{k}\sum_{u\in C_i} d(u,c_i) \le \gamma\sum_{i=1}^{k}\sum_{u\in C_i} d'(u,c_i) \le \gamma\sum_{i=1}^{k}\sum_{u\in C_i^*} d'(u,c_i^*) \le \gamma\sum_{i=1}^{k}\sum_{u\in C_i^*} d(u,c_i^*).$$

Now, consider the case of k-means.

$$\sum_{i=1}^{k}\sum_{u\in C_i} d(u,c_i)^2 \le \sum_{i=1}^{k}\sum_{u\in C_i} (\gamma d'(u,c_i))^2 \le \gamma^2\sum_{i=1}^{k}\sum_{u\in C_i^*} d'(u,c_i^*) \le \gamma^2\sum_{i=1}^{k}\sum_{u\in C_i^*} d(u,c_i^*).$$

II. The proof is identical to that of Theorem 10. ◀