Complexity of Local Search for Euclidean Clustering Problems

Bodo Manthey
Faculty of Electrical Engineering, Mathematics, and Computer Science, University of Twente, The Netherlands

Nils Morawietz
Institute of Computer Science, Friedrich Schiller University Jena, Germany

Jesse van Rhijn
Faculty of Electrical Engineering, Mathematics, and Computer Science, University of Twente, The Netherlands

Frank Sommer
Institute of Logic and Computation, TU Wien, Austria

Abstract

We show that the simplest local search heuristics for two natural Euclidean clustering problems are PLS-hard. First, we show that the Hartigan–Wong method, which is essentially the Flip heuristic, for k-Means clustering is PLS-hard, even when $k = 2$. Second, we show the same result for the Flip heuristic for Max Cut, even when the edge weights are given by the (squared) Euclidean distances between the points in some set $\mathcal{X} \subseteq \mathbb{R}^d$; a problem which is equivalent to Min Sum 2-Clustering.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Graph algorithms analysis; Theory of computation → Discrete optimization; Theory of computation → Facility location and clustering

Keywords and phrases Local search, PLS-complete, max cut, k-means, partitioning problem, flip-neighborhood

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2024.48

Related Version *Full Version*: `https://arxiv.org/abs/2312.14916`

Funding *Jesse van Rhijn*: Supported by NWO grant OCENW.KLEIN.176.
Frank Sommer: Supported by the Alexander von Humboldt Foundation.

1 Introduction

Clustering problems arise frequently in various fields of application. In these problems, one is given a set of objects, often represented as points in $\mathbb{R}^d$, and is asked to partition the set into *clusters*, such that the objects within a cluster are similar to one another by some measure. For points in $\mathbb{R}^d$, a natural measure is the (squared) Euclidean distance between two objects. In this paper, we consider two Euclidean clustering problems that use this similarity measure: k-Means clustering and Squared Euclidean Max Cut.

k-Means. One well-studied clustering problem is k-Means [10, 23]. In this problem, one is given a set of points $\mathcal{X} \subseteq \mathbb{R}^d$ and an integer k. The goal is to partition $\mathcal{X}$ into exactly k clusters such that the total squared distance of each point to the centroid of its cluster is minimized. Formally, one seeks to minimize the clustering cost

$$\sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mathrm{cm}(C_i)\|^2 \quad \text{where} \quad \mathrm{cm}(C_i) = \frac{1}{|C_i|} \sum_{x \in C_i} x.$$

Being NP-hard even when $k = 2$ [3] or when $\mathcal{X} \subseteq \mathbb{R}^2$ [31], k-Means has been extensively studied from the perspective of approximation algorithms [8, 21, 26, 34]. Nevertheless, local search remains the method of choice for practitioners [10, 23].

The most well-known local search algorithm for k-Means is Lloyd's method [30]. Here, one alternates between two steps in each iteration. In the first step, each point is assigned to its closest cluster center, and in the second step the cluster centers are recalculated from the newly formed clusters.

This algorithm was shown to have worst-case super-polynomial running time by Arthur and Vassilvitskii [7], with the result later improved to exponential running time even in the plane by Vattani [45]. Moreover, Roughgarden and Wang [39] showed it can implicitly solve PSPACE-complete problems. On the other hand, Arthur et al. [6] proved that Lloyd's method has smoothed polynomial running time on Gaussian-perturbed point sets, providing a degree of explanation for its effectiveness in practice.

Recently Telgarsky and Vattani [44] revived interest in another, older local search method for k-Means due to Hartigan and Wong [20]. This algorithm, the Hartigan–Wong method, is much simpler: one searches for a single point that can be reassigned to some other cluster for a strict improvement in the clustering cost. In other words, the Hartigan–Wong method is the Flip heuristic. In the following, we always use Flip instead of Hartigan–Wong to indicate that this is the most simple heuristic for this problem and to keep the name for the used heuristic consistent. Despite this simplicity, Telgarsky and Vattani [44] show that the Flip heuristic is more powerful than Lloyd's method, in the sense that the former can sometimes improve clusterings produced by the latter, while the converse does not hold.

A similar construction to that of Vattani for Lloyd's method shows that there exist instances on which the Flip heuristic can take exponentially many iterations to find a local optimum, even when all points lie on a line [33]. However, this example follows a contrived sequence of iterations. Moreover, k-Means can be solved optimally for instances in which all points lie on a line. Thus, the question remains whether stronger worst-case examples exist, and what the complexity of finding locally optimal clusterings is.

Squared Euclidean Max Cut. Another clustering problem similar to k-Means is Squared Euclidean Max Cut. Recall that Max Cut asks for a subset of vertices S of a weighted graph $G = (V, E)$, such that the total weight of the edges with one endpoint in S and one in $V \setminus S$ is maximized. This problem emerges in numerous applications, from graph clustering to circuit design to statistical physics [9, 12].

In Squared Euclidean Max Cut, one identifies the vertices of G with a set $\mathcal{X} \subseteq \mathbb{R}^d$, and assigns each edge a weight equal to the squared Euclidean distance between its endpoints. This problem is equivalent to Min Sum 2-Clustering (although not in approximation), where one seeks to minimize

$$\sum_{x,y \in X} \|x-y\|^2 + \sum_{x,y \in Y} \|x-y\|^2$$

over all partitions (X, Y) of $\mathcal{X}$. Also this special case of Max Cut is NP-hard [2]. In a clustering context, the problem was studied by Schulman [42] and Hasegawa et al. [21], leading to exact and approximation algorithms.

Given the computational hardness of Max Cut, practitioners often turn to heuristics. Some of the resulting algorithms are very successful, such as the Kernighan-Lin heuristic [27] and the Fiduccia-Mattheyses algorithm [19]. Johnson et al. [24] note that the simple Flip heuristic, where one moves a single vertex from one side of the cut to the other, tends to converge quickly in practice. Schäffer and Yannakakis [40] later showed that it has exponential running time in the worst case.

One may wonder whether Flip performs better for Squared Euclidean Max Cut. Etscheid and Röglin [17, 16] performed a smoothed analysis of Flip in this context, showing a smoothed running time of $2^{O(d)} \cdot \mathrm{poly}(n, 1/\sigma)$ for Gaussian-perturbed instances, where σ denotes the standard deviation of the Gaussian noise. On the other hand, they also exhibited an instance in $\mathbb{R}^2$ on which there exists an exponential-length improving sequence of iterations, with the caveat that not all edges are present in the instance. Like for k-Means, one may ask whether stronger examples exist (e.g. on complete graphs), and what the complexity of finding Flip-optimal solutions is.

Complexity of Local Search. The existence of instances with worst-case exponential running time is common for local search heuristics. To investigate this phenomenon, and local search heuristics in general, Johnson et al. [24] defined a complexity class PLS, for polynomial local search. The class is designed to capture the properties of commonly used local search heuristics and contains pairs consisting of an optimization problem P and a local search heuristic $\mathcal{N}$. In the following we denote such a pair as $\mathrm{P}/\mathcal{N}$. PLS-complete problems have the property that their natural local search algorithms have worst-case exponential running time. Johnson et al. [24] showed that the Kernighan-Lin heuristic for the Max Bisection problem (a variant of Max Cut, where the parts of the partition must be of equal size) is PLS-complete. This stands in contrast to the empirical success of this algorithm [27].

Building on this work, Schäffer and Yannakakis [40] later proved that a host of very simple local search heuristics are PLS-complete, including the Flip heuristic for Max Cut. This refuted a conjecture by Johnson et al., who doubted that such simple heuristics could be PLS-complete. Elsässer and Tscheuschner [15] later showed that this remains true even in the very restricted variant where the input graph has maximum degree five, which we will refer to as Max Cut-5.

Schäffer and Yannakakis defined a new type of PLS-reduction called a *tight* reduction. In addition to showing completeness for PLS, this type of reduction also transfers stronger properties on the running time of local search heuristics between PLS problems.

Since the introduction of PLS, many local search problems have been shown to be PLS-complete, including such successful heuristics as Lin-Kernighan's algorithm for the TSP [38] or the k-Swap-neighborhood heuristic for Weighted Independent Set [28] for $k \geq 3$. For a non-exhaustive list, see Michiels, Korst and Aarts [36, Appendix C].

Our Contribution. Given the existence of k-Means instances where the Flip heuristic has worst-case exponential running time, one may ask whether this heuristic is PLS-hard. In this work, we answer this question in the affirmative.

▶ **Theorem 1.1.** *For each $k \geq 2$, k-Means/Flip is* PLS*-hard.*

Just as with k-Means/Flip, we ask whether Squared Euclidean Max Cut with the Flip heuristic is PLS-hard. Again, we answer this question affirmatively. In addition, we show the same result for Euclidean Max Cut, where the distances between the points are not squared.

▶ **Theorem 1.2.** *Euclidean Max Cut/Flip and Squared Euclidean Max Cut/Flip are* PLS*-hard.*

We note that PLS-hardness results for Euclidean local optimization problems are rather uncommon. We are only aware of one earlier result by Brauer [13], who proved PLS-completeness of a local search heuristic for a discrete variant of k-Means. This variant chooses k cluster centers among the set of input points, after which points are assigned to their closest center. The heuristic they consider removes one point from the set of cluster centers and adds another. In their approach, they first construct a metric instance, and then show that this instance can be embedded into $\mathbb{R}^d$ using a theorem by Schoenberg [41]. In contrast, we directly construct instances in $\mathbb{R}^d$.

In addition to showing PLS-hardness of k-Means/Flip and Squared Euclidean Max Cut/Flip, we also show that there exist specific hard instances of these problems, as well as all other problems considered in this paper.

▶ **Theorem 1.3.** *Each local search problem L considered in this work (see Section 2.2) fulfills the following properties:*

1. *L is* PLS*-hard.*
2. *For each n, one can compute in polynomial time an instance of L of size $n^{\mathcal{O}(1)}$ with an initial solution that is exponentially far away from any local optimum.*
3. *The problem of computing the locally optimal solution obtained from performing a standard local search algorithm based on the neighborhood of L is* PSPACE*-hard for L.*

A formal definition of Property 3 is given in Section 2.1. In particular, this result shows that there exists an instance of k-Means such that there exists an initial solution to this instance that is exponentially many iterations away from *all* local optima [40]. By contrast, the earlier result [33] only exhibits an instance with a starting solution that is exponentially many iterations away from *some* local optimum. Moreover, Theorem 1.3 yields instances where Flip has exponential running time in Squared Euclidean Max Cut on *complete* geometric graphs, unlike the older construction which omitted some edges [16].

2 Preliminaries and Notation

Throughout this paper, we will consider all graphs to be undirected unless explicitly stated otherwise. Let $G = (V, E)$ be a graph. For $v \in V$, we denote by $d(v)$ the *degree* of v in G, and by $N(v)$ the set of *neighbors* of v.

Let $S, T \subseteq V$. We write $E(S, T)$ for the set of edges with one endpoint in S and one endpoint in T. For $S \subseteq V$, we write $\delta(S) = E(S, V \setminus S)$ for the *cut induced by* S. We will also refer to the partition $(S, V \setminus S)$ as a *cut*; which meaning we intend will be clear from the context. If $\big||S| - |V \setminus S|\big| \leq 1$, we will call the cut $(S, V \setminus S)$ a *bisection*. Given $v \in V$ we will also write $\delta(v) = \delta(\{v\})$, which is the set of *edges incident to* v.

Let $F \subseteq E$. Given a function $f : E \to \mathbb{R}$, we denote by $f(F) = \sum_{e \in F} f(e)$ the *total value* of f on the set of edges F. If $F = E(X, Y)$ for some sets $X, Y \subseteq V$, we will abuse notation to write $f(X, Y) = f(F)$.

2.1 The Class PLS

For convenience, we summarize the formal definitions of local search problems and the associated complexity class PLS, as devised by Johnson et al. [24].

A *local search problem* P is defined by a set of instances I, a set of feasible solutions $F_I(x)$ for each instance $x \in I$, a *cost function* c that maps pairs of a solution of $F_I(x)$ and an instance x to $\mathbb{Z}$, and a *neighborhood function* $\mathcal{N}$ that maps a solution of $F_I(x)$ and an instance x to a subset of $F_I(x)$. Typically, the neighborhood function is constructed so that it is easy to compute some $s' \in \mathcal{N}(s, x)$ for any given $s \in F_I(x)$.

This characterization of local search problems gives rise to the *transition graph* defined by an instance of such a problem.

▶ **Definition 2.1.** *Given an instance $x \in I$ of a local search problem P, we define the* transition graph *$T(x)$ as the directed graph with vertex set $F_I(x)$, with an edge from s to s' if and only if $s' \in \mathcal{N}(s, x)$ and $c(s, x) < c(s', x)$ (assuming P is a maximization problem; otherwise, we reverse the inequality). The height of a vertex s in $T(x)$ is the length of the shortest path from s to a sink of $T(x)$.*

The class PLS is defined to capture the properties of local search problems that typically arise in practical applications. Formally, P is contained in PLS if the following are all true:

1. There exists a deterministic polynomial-time algorithm A that, given an instance $x \in I$, computes some solution $s \in F_I(x)$.
2. There exists a deterministic polynomial-time algorithm B that, given $x \in I$ and $s \in F_I(x)$, computes the value of $c(s, x)$.
3. There exists a deterministic polynomial-time algorithm C that, given $x \in I$ and $s \in F_I(x)$, either computes a solution $s' \in \mathcal{N}(s, x)$ with $c(s', x) > c(s, x)$ (in the case of a maximization problem), or outputs that such a solution does not exist.

Intuitively, algorithm A gives us some initial solution from which to start an optimization process, algorithm B ensures that we can evaluate the quality of solutions efficiently, and algorithm C drives the local optimization process by either determining that a solution is locally optimal or otherwise giving us an improving neighbor. Based on the algorithms A and C, one can define the "standard algorithm problem" for P as follows.

▶ **Definition 2.2.** *Let P be a local search problem and let I be an instance of P. Moreover, let $s^*(I)$ be the unique local optimum obtained by starting with the solution outputted by algorithm A and replacing the current solution by the better solution outputted by C, until reaching a local optimum. The* standard algorithm problem *for P asks for a given instance I of P and a locally optimal solution s' for I with respect to $\mathcal{N}$, whether s' is exactly the solution $s^*(I)$.*

It was shown that for many local search problems the standard algorithm problem is PSPACE-complete [13, 36, 37, 40].

Given problems $P, Q \in$ PLS, we say that P is PLS-*reducible* to Q (written $P \leq_{\mathsf{PLS}} Q$) if the following is true.

1. There exist polynomial-time computable functions f, g, such that f maps instances x of P to instances $f(x)$ of Q, and g maps pairs (solution s of $f(x), x$) to feasible solutions of P.
2. If a solution s of $f(x)$ is locally optimal for $f(x)$, then $g(s, x)$ is locally optimal for x.

The idea is that, if $Q \in$ PLS is efficiently solvable, then P is also efficiently solvable: simply convert an instance of P to Q using f, solve Q, and convert the resulting solution back to a solution of P using g. As usual in complexity theory, if P is *complete* for PLS and $P \leq_{\mathsf{PLS}} Q$, then Q is also complete for PLS.

In addition to this standard notion of a PLS-reduction, Schäffer and Yannakakis [40] defined so-called *tight* reductions. Given PLS problems P and Q and a PLS-reduction (f, g) from P to Q, the reduction is called *tight* if for any instance x of P we can choose a subset $\mathcal{R}$ of the feasible solutions of $f(x)$ of Q such that:

1. $\mathcal{R}$ contains all local optima of $f(x)$.
2. For every feasible solution s of x, we can construct a feasible solution $q \in \mathcal{R}$ of $f(x)$ such that $g(q, x) = s$.
3. Suppose the transition graph $T(f(x))$ of $f(x)$ contains a directed path from s to s' such that $s, s' \in \mathcal{R}$, but all internal vertices lie outside of $\mathcal{R}$, and let $q = g(s, x)$ and $q' = g(s', x)$. Then either $q = q'$, or the transition graph $T(x)$ of x contains an edge from q to q'.

The set $\mathcal{R}$ is typically called the set of *reasonable solutions to* $f(x)$. Here, the intuition is that tight reductions make sure that the height of a vertex s of $T(f(x))$ is not smaller than that of $g(s, x)$ in $T(x)$. Note that a reduction (f, g) is trivially tight if $T(f(x))$ is isomorphic to $T(x)$.

Tight PLS-reductions have two desired properties [1, Chapter 2]. Suppose P reduces to Q via a tight reduction. First, if the standard algorithm problem for P is PSPACE-complete, then the standard algorithm problem for Q is PSPACE-complete as well. Second, if there exists an instance x of P such that there exists a solution of x that is exponentially far away from any local optimum, then such an instance exists for Q as well. Note that this first property holds irrespective of the choices made by the algorithm C for Q [40].

2.2 Definitions of Local Search Problems

We will be concerned with various local search problems. In the following we provide a summary of the problems that appear in this paper, and provide definitions for each. Some of the problems considered in this paper are not the most natural ones, but we need them as intermediate problems for our reductions. Moreover, these problems might be useful to show PLS-hardness of other problems having cardinality constraints.

Before introducing the problems themselves, we first provide a more abstract view of the problems, since they have many important aspects in common. Each problem in the list below is a type of partitioning problem, where we are given a finite set S and are asked to find the "best" partition of S into k sets (indeed, for all but one problem, we have $k = 2$). What determines whether some partition is better than another varies; this is determined by the cost function of the problem in question.

▶ **Definition 2.3.** *Given a partition $\mathcal{P} = \{S_1, \dots, S_k\}$ of S, a partition $\mathcal{P}'$ is a neighbor of $\mathcal{P}$ in the* Flip neighborhood *if $\mathcal{P}'$ can be obtained by moving exactly one element from some $S_i \in \mathcal{P}$ to some other $S_j \in \mathcal{P}$. In other words, if $\mathcal{P}' = \{S_1, \dots, S_i', \dots, S_j', \dots, S_k\}$ where for some $v \in S_i$ we have $S_i' = S_i \setminus \{v\}$ and $S_j' = S_j \cup \{v\}$.*

The Flip neighborhood as defined above is perhaps the simplest neighborhood structure for a partitioning problem. For each problem in the list below, we consider only the Flip neighborhood in this paper. Recall that the Flip heuristic for k-Means is also referred to as the Hartigan–Wong method [20].

Max Cut

Input: A graph $G = (V, E)$ with non-negative edge weights $w : E \to \mathbb{Z}_{\geq 0}$.

Output: A partition (X, Y) of V such that $w(X, Y)$ is maximal.

We will mainly be concerned with several variants of Max Cut. For some fixed integer d, by Max Cut-d we denote the restriction of the problem to graphs with maximum degree d. In Densest Cut, one aims to maximize $\frac{w(X,Y)}{|X| \cdot |Y|}$ rather than just $w(X, Y)$. The minimization version of this problem is called Sparsest Cut. The problem Odd Max Bisection is identical to Max Cut, with the added restrictions that the number of vertices must be odd and that the two halves of the cut differ in size by exactly one. The minimization version of the problem is called Odd Min Bisection.

The definitions of Odd Max/Min Bisection are somewhat unconventional, as one usually considers these problem with an even number of vertices and with the Swap neighborhood, where two solutions are neighbors if one can be obtained from the other by swapping a pair of vertices between parts of the partition. Hardness of Max Bisection/Swap was shown by Schäffer and Yannakakis [40] in a short reduction from Max Cut/Flip, providing as a corollary also a simple hardness proof for the Kernighan-Lin heuristic [27]. The reason we require the Flip neighborhood is that we aim to reduce this problem to Squared Euclidean Max Cut/Flip, where we run into trouble when we use the Swap neighborhood (see Section 3 for details).

Squared Euclidean Max Cut

Input: A set of n points $\mathcal{X} \subseteq \mathbb{R}^d$.

Output: A partition (X, Y) of $\mathcal{X}$ such that $\sum_{x \in X} \sum_{y \in Y} \|x - y\|^2$ is maximal.

Euclidean Max Cut is defined similarly; the only difference is that the actual distances between points enter the objective function, rather than the squared distances.

k-Means

Input: A set of n points $\mathcal{X} \subseteq \mathbb{R}^d$ and an integer $k \geq 2$.

Output: A partition $(C_1, \dots, C_k)$ of $\mathcal{X}$ such that $\sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mathrm{cm}(C_i)\|^2$ is minimal.

The sets $\{C_1, \dots, C_k\}$ are called *clusters*. Note that in this formulation, both k-Means/Flip and Squared Euclidean Max Cut/Flip are not contained in PLS, as their cost functions can take non-integer values. However, we still obtain PLS-hardness for each of these problems, and the existence of specific hard instances (cf. Theorem 1.3). Moreover, this hardness still holds for restricted versions of the problems which do belong to PLS. More technical details are given in the full version.

Positive Not-All-Equal k-Satisfiability (Pos NAE k-SAT)

Input: A boolean formula with clauses of the form $\mathrm{NAE}(x_1, \dots, x_\ell)$ with $\ell \leq k$, where each clause is satisfied if its constituents, all of which are positive, are neither all true nor all false. Each clause C has a weight $w(C) \in \mathbb{Z}$.

Output: A truth assignment of the variables such that the sum of the weights of the satisfied clauses is maximized.

In Odd Half Pos NAE k-SAT, additionally the number of variables is odd and it is required that the number of `true` variables and the number of `false` variables in any solution differ by exactly one. This is analogous to the relationship between Max Cut and Odd Max Bisection.

2.3 Strategy

Both Squared Euclidean Max Cut and k-Means are NP-hard [2, 3]. The reductions used to prove this are quite similar, and can be straightforwardly adapted into PLS-reductions: In the case of Squared Euclidean Max Cut/Flip, we obtain a reduction from Odd Min Bisection/Flip, while for k-Means/Flip we obtain a reduction from Densest Cut/Flip. The latter reduction even works for $k = 2$. These results are given in Lemma 4.3 (k-Means), and Lemmas 4.5 and 4.6 ((Squared) Euclidean Max Cut).

What remains then is to show that the problems we reduce from are also PLS-complete, which takes up the bulk of the work. Figure 1 shows the reduction paths we use.

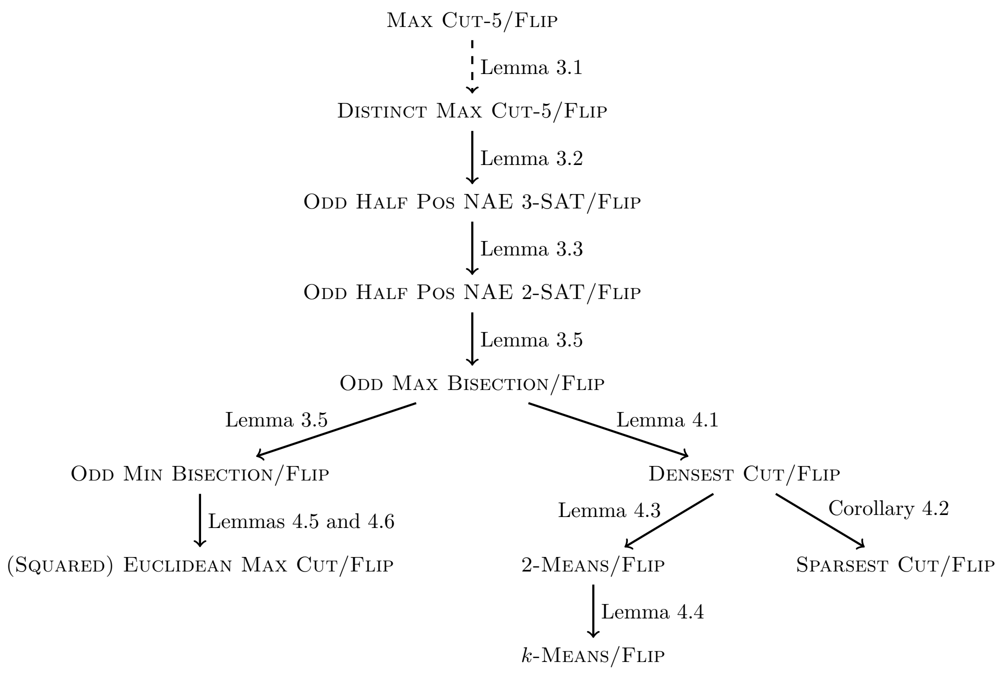


Figure 1 Graph of the PLS-reductions used in this paper. Reductions represented by solid lines are tight, reductions represented by dashed lines are not.

The starting point will be the PLS-completeness of Max Cut-5/Flip, which was shown by Elsässer and Tscheuschner [15]. An obvious next step might then be to reduce from this problem to Max Bisection/Swap, and then further reduce to Odd Max Bisection/Flip. Unfortunately, this turns out to be rather difficult, as the extra power afforded by the Swap neighborhood is not so easily reduced back to the Flip neighborhood. Using this strategy, we can only obtain PLS-hardness of the 2-Flip neighborhood for Squared Euclidean Max Cut, where two points may flip in a single iteration.

We thus take a detour through Odd Half Pos NAE 3-SAT/Flip in Lemma 3.2, which then reduces down to Odd Half Pos NAE 2-SAT/Flip in Lemma 3.3 and finally to Odd Max Bisection/Flip in Lemma 3.5, using a reduction by Schäffer and Yannakakis [40].

From this point, hardness of Squared Euclidean Max Cut/Flip (and with a little extra work, Euclidean Max Cut/Flip) is easily obtained. For k-Means/Flip, we need some more effort, as we still need to show hardness of Densest Cut/Flip. Luckily, this can be done by reducing from Odd Max Bisection/Flip as well, as proved in Lemma 4.1.

Due to space constraints, our proofs are deferred to the full version.

3 Reduction to Odd Min/Max Bisection

The goal of this section is to obtain PLS-completeness of Odd Min/Max Bisection/Flip, from which we can reduce further to our target problems; see Figure 1. We will first construct a PLS-reduction from Max Cut-5/Flip to Odd Half Pos NAE 3-SAT/Flip in Lemma 3.2.

A subtlety is that the reduction only works when we assume that the Max Cut-5 instance we reduce from has distinct costs for any two neighboring solutions. The following lemma ensures that we can make this assumption.

▶ **Lemma 3.1.** *Distinct Max Cut-5/Flip is* PLS*-complete. More precisely, there exists a* PLS*-reduction from Max Cut-5/Flip to Distinct Max Cut-5/Flip.*

Unfortunately, this reduction is not tight. Hence, to prove the last two items of Theorem 1.3, simply applying the reductions from Figure 1 is not sufficient, as these properties (viz. PSPACE-completeness of the standard algorithm problem and the existence of certain hard instances) do not necessarily hold for Distinct Max Cut-5/Flip. We must instead separately prove that they hold for this problem. To accomplish this, we recall a construction by Monien and Tscheuschner [37] that shows these properties for Max Cut-4/Flip. It can be verified straightforwardly that the construction they use is already an instance of Distinct Max Cut-4/Flip.

In the remainder of this work, we present a sequence of tight reductions starting from Distinct Max Cut-5/Flip to all of our other considered problems. First, we reduce from Distinct Max Cut-5/Flip to Odd Half Pos NAE 3-SAT/Flip.

▶ **Lemma 3.2.** *There exists a tight* PLS*-reduction from Distinct Max Cut-5/Flip to Odd Half Pos NAE 3-SAT/Flip.*

As mentioned in Section 2.3, it may seem more straightforward to reduce from Min Bisection/Swap to Odd Min Bisection/Flip. The problem with this approach is that a solution to Odd Min Bisection/Flip can be locally optimal for two reasons: either no vertex can flip to obtain a cut of larger weight, or the vertices that could are in the smaller part of the partition. This makes the Flip neighborhood much less powerful than Swap in this problem variant; we were thus not able to find a direct reduction from Min Bisection/Swap. Instead, we apply a new technique that allows us to prove PLS-hardness for this very restricted problem.

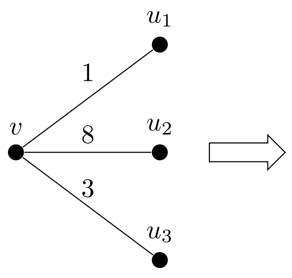
We first prove PLS-hardness of Odd Half Pos NAE 3-SAT/Flip, and subsequently use existing reductions to obtain hardness of Odd Min Bisection/Flip. With the expressiveness of this SAT variant we gain a great deal of freedom to handle the problem restrictions. The main challenge is in encoding the restriction that the number of true and false variables must differ by exactly one without weakening the neighborhood.

Next, we briefly sketch and motivate some of the ideas in the reduction in more detail. See also Figure 2.

Sketch of Proof for Lemma 3.2. We first embed the Distinct Max Cut-5 instance, given by a weighted graph $G = (V, E)$, in Pos NAE SAT. This can be done rather straightforwardly, by a reduction used by Schäffer and Yannakakis [40]: each vertex becomes a variable, and an edge uv becomes a clause NAE(u, v). This instance is directly equivalent to the original Distinct Max Cut-5 instance. We call these variables the *level 1 variables*, and the clauses the *level 1 clauses*. A level 1 clause NAE(u, v) gets a weight $M \cdot w(uv)$ for some large integer M.

A solution to the original Distinct Max Cut-5 instance is obtained by placing the `true` level 1 variables in a feasible truth assignment on one side of the cut, and the `false` level 1 variables on the other side.

Given the reduction so far, suppose we have some locally optimal feasible truth assignment s. We partition the variables into the sets T and F of `true` and `false` variables; thus, (T, F) is the cut induced by s. Suppose $|T| = |F| + 1$. If no level 1 variable can flip from T to F, then also no vertex can flip from T to F in the induced cut. However, we run into trouble when there exists some $v \in F$ that can flip in the cut. Since $|F| < |T|$, we are not allowed to flip the level 1 variable v, and so the truth assignment may be locally optimal even though the induced cut is not.



NAE(v, u_1)		weight: M	Level 1
NAE(v, u_2)		weight: $8M$	
NAE(v, u_3)		weight: $3M$	
NAE(q_v, u_1)		weight: $-L$	Level 2
NAE(q_v, u_2)		weight: $-8L$	
NAE(q_v, u_3)		weight: $-3L$	
NAE(v, q_v, a_i)	$\{u_1, u_2, u_3\}$	weight: -1	Level 3
NAE(v, q_v, a_i)	$\{u_1, u_2\}$	weight: -1	
NAE(v, q_v, a_i)	$\{u_1, u_3\}$	weight: 0	
NAE(v, q_v, a_i)	$\{u_2, u_3\}$	weight: -1	
NAE(v, q_v, a_i)	$\{u_1\}$	weight: 0	
NAE(v, q_v, a_i)	$\{u_2\}$	weight: -1	
NAE(v, q_v, a_i)	$\{u_3\}$	weight: 0	
NAE(v, q_v, a_i)	$\emptyset$	weight: 0	

Figure 2 Schematic overview of the reduction used in the proof of Lemma 3.2. On the left we have a vertex $v \in V$ and its neighbors $\{u_1, u_2, u_3\}$ in a Max Cut instance, with weights on the edges between v and its neighbors. The NAE clauses on the right are the clauses constructed from v. In the actual reduction, these clauses are added for all level 3 variables. The right-most column shows the weights assigned to the clauses. The middle column shows for the level 3 clauses which subset of $N(v)$ corresponds to which clause. The constants L and M are chosen so that $1 \ll L \ll M$.

To deal with this situation, we will introduce two more levels of variables and clauses. The weights of the clauses at level i will be scaled so that they are much larger than those at level $i+1$. In this way, changes at level i dominate changes at level $i+1$, so that the Distinct Max Cut-5 instance can exist independently at level 1.

For each vertex $v \in V$, we add a variable q_v to the instance, and for each $u \in N(v)$, we add a clause NAE(q_v, u) with weight proportional to $-w(uv)$. We call these variables the *level 2 variables*, and these clauses the *level 2 clauses.*

Finally, we add $N = 2n+1$ more variables $\{a_i\}_{i=1}^{N}$, which we call the *level 3 variables.* The number N is chosen so that for any truth assignment such that the number of `true` and `false` variables differ by one, there must exist a level 3 variable in the larger of the two sets. We then add more clauses as follows: for each level 3 variable a_i, for each $v \in V$, for each $Q \subseteq N(v)$, we add a clause $C_i(v, Q) = \text{NAE}(v, q_v, a_i)$. We give this clause a weight of -1 if and only if v can flip when each of the vertices in Q are present in the same half of the cut as v. We call these the *level 3 clauses*

Now consider the aforementioned situation, where a truth assignment s is locally optimal, but there exists some $v \in V \cap F$ that can flip in the induced cut. Carefully investigating the structure of such a locally optimal truth assignment shows that some level 2 or level 3 variable can flip for a strict improvement in the cost. This contradicts local optimality, and so we must conclude that locally optimal truth assignments induce locally optimal cuts, satisfying the essential property of PLS-reductions. ◀

As far as we are aware, this technique for overcoming size constraints in local search problems is novel. We believe that it may be useful to prove PLS-hardness results for simple heuristics for other size-constrained problems, such as balanced clustering problems.

A reduction from Pos NAE 3-SAT/Flip to Max Cut/Flip was provided by Schäffer and Yannakakis [40]. Since Max Cut is equivalent to Pos NAE 2-Sat, we can use the same reduction to reduce from Odd Half Pos NAE 3-SAT/Flip to Odd Half Pos NAE 2-SAT/Flip.

▶ **Lemma 3.3** (Schäffer and Yannakakis [40]). *There exists a tight* PLS*-reduction from Odd Half Pos NAE 3-SAT/Flip to Odd Half Pos NAE 2-SAT/Flip.*

While our reductions so far have used negative-weight clauses in Odd Half Pos NAE k-SAT/Flip, it may be of interest to have a PLS-completeness result also when all clauses have non-negative weight.

▶ **Corollary 3.4.** *Odd Half Pos NAE 2-SAT/Flip is* PLS*-complete even when all clauses have non-negative weight. More precisely, there exists a tight* PLS*-reduction from Odd Half Pos NAE 2-SAT/Flip to Odd Half Pos NAE 2-SAT/Flip where all clauses have non-negative weight.*

Finally, we reduce from Odd Half Pos NAE 2-SAT/Flip to Odd Min Bisection/Flip.

▶ **Lemma 3.5.** *There exists a tight* PLS*-reduction from Odd Half Pos NAE 2-SAT/Flip to both Odd Max Bisection/Flip and Odd Min Bisection/Flip.*

A reduction from Pos NAE 2-SAT/Flip to Max Cut/Flip is given by Schäffer and Yannakakis [40]. It is easy to see that this reduction also works with our constraint on the number of `true` and `false` variables, which yields a reduction to Odd Max Bisection/Flip.

4 Reduction to Clustering Problems

Armed with the PLS-completeness of Odd Min Bisection/Flip (see Lemma 3.5), we now proceed to prove hardness of the Euclidean clustering problems of interest.

k-Means. We provide a tight PLS-reduction from Odd Min Bisection/Flip to k-Means/Flip. This is done in three steps (see Figure 1). First, we show PLS-completeness of Densest Cut/Flip. The construction of the proof of our PLS-completeness of Densest Cut/Flip is rather simple (we only add a large set of isolated edges), but the analysis of the correctness is quite technical. Second, we show PLS-hardness of 2-Means/Flip by slightly adapting an NP-hardness reduction of 2-Means [3]. Finally, we extend this result to k-Means/Flip.

Now, we show PLS-completeness of Densest Cut/Flip. We impose the additional constraint that there are no isolated vertices in the reduced instance. This is a technical condition which is utilized in Lemma 4.3 for the PLS-hardness of k-Means/Flip.

For an illustration of the reduction of Lemma 4.1 we refer to Figure 3.

▶ **Lemma 4.1.** *There exists a tight* PLS*-reduction from Odd Max Bisection/Flip to Densest Cut/Flip without isolated vertices.*

Next, we show that also the closely related Sparsest Cut is PLS-complete under the Flip neighborhood. Note that Densest Cut and Sparsest Cut are both NP-hard [35]. Sparsest Cut is studied intensively in terms of approximation algorithms [5] and integrality gaps [25], and is used to reveal the hierarchical community structure of social networks [32] and in image segmentation [43].

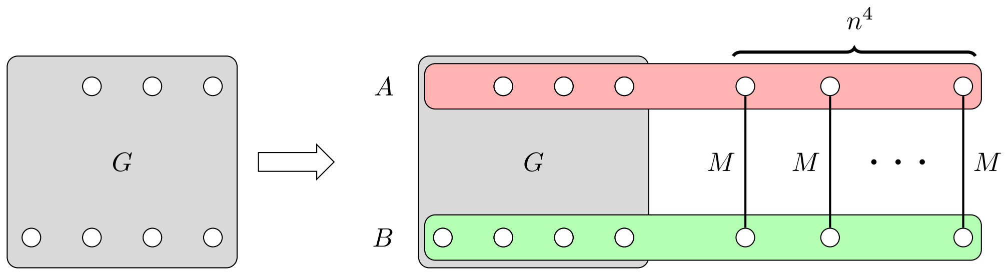


Figure 3 Schematic overview of the reduction used in the proof of Lemma 4.1. On the left side we have an instance of Odd Max Bisection/Flip and on the right side we have the corresponding instance of Densest Cut/Flip. The edges inside of G together with their weights are not depicted but are identical in both instances. Let (A, B) be the partition corresponding to some locally optimal solution of the Densest Cut/Flip instance. Then, A contains exactly one endpoint of each of the n^4 isolated edges and $\big|\,|A \cap V(G)| - |B \cap V(G)|\,\big| = 1$.

▶ **Corollary 4.2.** *There exists a tight* PLS*-reduction from Densest Cut/Flip to Sparsest Cut/Flip.*

The penultimate step is to show that 2-Means/Flip is PLS-hard. We achieve this by modifying a proof of NP-hardness of 2-Means by Alois et al. [3].

▶ **Lemma 4.3.** *There exists a tight* PLS*-reduction from Densest Cut/Flip without isolated vertices to 2-Means/Flip.*

Finally, we provide a generic reduction to show PLS-hardness for general k.

▶ **Lemma 4.4.** *For each $k \geq 2$, there exists a tight* PLS*-reduction from k-Means/Flip to $(k+1)$-Means/Flip.*

Now, Theorem 1.1 follows by applying the tight PLS-reductions according to Figure 1.

Squared Euclidean Max Cut. We construct a PLS-reduction from Odd Min Bisection/Flip to Squared Euclidean Max Cut/Flip. The reduction is largely based on the NP-hardness proof of Euclidean Max Cut of Ageev et al. [2]. The main difference is that we must incorporate the weights of the edges of the Odd Min Bisection/Flip instance into the reduction.

▶ **Lemma 4.5.** *There exists a tight* PLS*-reduction from Odd Min Bisection/Flip to Squared Euclidean Max Cut/Flip.*

With a few modifications, the proof can be adapted to a reduction to Euclidean Max Cut/Flip. The main challenge in adapting the proof is that the objective function is now of the form $\sum \|x - y\|$, rather than $\sum \|x - y\|^2$. However, by suitably modifying the coordinates of the points, the distances $\|x - y\|$ in the Euclidean Max Cut instance can take the same value as $\|x - y\|^2$ in the Squared Euclidean Max Cut instance.

▶ **Lemma 4.6.** *There exists a tight* PLS*-reduction from Odd Min Bisection/Flip to Euclidean Max Cut/Flip.*

5 Discussion

Theorems 1.1 and 1.3 show that no local improvement algorithm using the FLIP heuristic can find locally optimal clusterings efficiently, even when $k = 2$. This result augments an earlier worst-case construction [33]. Theorem 1.2 demonstrates that finding local optima in SQUARED EUCLIDEAN MAX CUT is no easier than for general MAX CUT under the FLIP neighborhood. Thus, the Euclidean structure of the problem yields no benefits with respect to the computational complexity of local optimization.

Smoothed Analysis. Other PLS-hard problems have yielded under smoothed analysis. Chiefly, MAX CUT/FLIP has polynomial smoothed complexity in complete graphs [4, 11] and quasi-polynomial smoothed complexity in general graphs [14, 18]. We hope that our results here serve to motivate research into the smoothed complexity of k-MEANS/FLIP and SQUARED EUCLIDEAN MAX CUT/FLIP, with the goal of adding them to the list of hard local search problems that become easy under perturbations.

Reducing the Dimensionality. Our reductions yield instances of k-MEANS and (SQUARED) EUCLIDEAN MAX CUT in $\Omega(n)$ dimensions. Seeing as our reductions cannot be obviously adapted for $d = o(n)$, we raise the question of whether the hardness of SQUARED EUCLIDEAN MAX CUT/FLIP and k-MEANS/FLIP is preserved for $d = o(n)$. This seems unlikely for SQUARED EUCLIDEAN MAX CUT/FLIP for $d = O(1)$, since there exists an $O(n^{d+1})$-time exact algorithm due to Schulman [42]. A direct consequence of PLS-hardness for $d = f(n)$ would thus be an $O\big(n^{f(n)}\big)$-time general-purpose local optimization algorithm. Concretely, PLS-hardness for $d = \operatorname{polylog} n$ would yield a quasi-polynomial time algorithm for all problems in PLS.

For k-MEANS/FLIP, the situation is similar: For $d = 1$, k-MEANS is polynomial-time solvable for any k. However, already for $d = 2$, the problem is NP-hard [31] when k is arbitrary. When both k and d are constants, the problem is again polynomial-time solvable, as an algorithm exists that finds an optimal clustering in time $n^{O(kd)}$ [21]. Thus, PLS-hardness for $kd \in O(f(n))$ would yield an $n^{O(f(n))}$-time algorithm for all PLS problems in this case.

Euclidean Local Search. There appear to be very few PLS-hardness results for Euclidean local optimization problems, barring the result of Brauer [13] and now Theorem 1.1 and Theorem 1.2. A major challenge in obtaining such results is that Euclidean space is very restrictive; edge weights cannot be independently set, so the intricacy often required for PLS-reductions is hard to achieve. Even in the present work, most of the work is done in a purely combinatorial setting. It is then useful to get rid of the Euclidean structure of the problem as quickly as possible, which we achieved by modifying the reductions of Ageev et al. [2] and Alois et al. [3].

With this insight, we pose the question of what other local search problems remain PLS-hard for Euclidean instances. Specifically, is TSP with the k-opt neighborhood still PLS-hard in Euclidean (or squared Euclidean) instances, for sufficiently large k? This is known to be the case for general metric instances for some large constant k [29] (recently improved to all $k \geq 17$ [22]), but Euclidean instances are still a good deal more restricted.

References

1 Emile Aarts and Jan Karel Lenstra, editors. *Local Search in Combinatorial Optimization*. Princeton University Press, 2003. doi:10.2307/j.ctv346t9c.

2 A. A. Ageev, A. V. Kel'manov, and A. V. Pyatkin. Complexity of the weighted max-cut in Euclidean space. *Journal of Applied and Industrial Mathematics*, 8(4):453–457, October 2014. doi:10.1134/S1990478914040012.

3 Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, May 2009. doi:10.1007/s10994-009-5103-0.

4 Omer Angel, Sébastien Bubeck, Yuval Peres, and Fan Wei. Local max-cut in smoothed polynomial time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 429–437, New York, NY, USA, June 2017. Association for Computing Machinery. doi:10.1145/3055399.3055402.

5 Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM*, 56(2):5:1–5:37, April 2009. doi:10.1145/1502793.1502794.

6 David Arthur, Bodo Manthey, and Heiko Röglin. Smoothed Analysis of the k-Means Method. *Journal of the ACM*, 58(5):19:1–19:31, October 2011. doi:10.1145/2027216.2027217.

7 David Arthur and Sergei Vassilvitskii. How slow is the k-means method? In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, SoCG '06, pages 144–153, New York, NY, USA, June 2006. Association for Computing Machinery. doi:10.1145/1137856.1137880.

8 David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, USA, January 2007. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1283383.1283494.

9 Francisco Barahona, Martin Grötschel, Michael Jünger, and Gerhard Reinelt. An Application of Combinatorial Optimization to Statistical Physics and Circuit Layout Design. *Operations Research*, 36(3):493–513, June 1988. doi:10.1287/opre.36.3.493.

10 P. Berkhin. A Survey of Clustering Data Mining Techniques. In *Grouping Multidimensional Data: Recent Advances in Clustering*, pages 25–71. Springer, Berlin, Heidelberg, 2006. doi:10.1007/3-540-28349-8_2.

11 Ali Bibak, Charles Carlson, and Karthekeyan Chandrasekaran. Improving the Smoothed Complexity of FLIP for Max Cut Problems. *ACM Transactions on Algorithms*, 17(3):19:1–19:38, July 2021. doi:10.1145/3454125.

12 Y.Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 105–112 vol.1, July 2001. doi:10.1109/ICCV.2001.937505.

13 Sascha Brauer. Complexity of Single-Swap Heuristics for Metric Facility Location and Related Problems. In *Algorithms and Complexity*, Lecture Notes in Computer Science, pages 116–127, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-57586-5_11.

14 Xi Chen, Chenghao Guo, Emmanouil V. Vlatakis-Gkaragkounis, Mihalis Yannakakis, and Xinzhi Zhang. Smoothed complexity of local max-cut and binary max-CSP. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 1052–1065, New York, NY, USA, June 2020. Association for Computing Machinery. doi:10.1145/3357713.3384325.

15 Robert Elsässer and Tobias Tscheuschner. Settling the Complexity of Local Max-Cut (Almost) Completely. In *International Colloquium on Automata, Languages, and Programming*, Lecture Notes in Computer Science, pages 171–182, Berlin, Heidelberg, 2011. Springer. doi:10.1007/978-3-642-22006-7_15.

16 Michael Etscheid. *Beyond Worst-Case Analysis of Max-Cut and Local Search*. PhD thesis, Universitäts- und Landesbibliothek Bonn, August 2018. URL: `https://bonndoc.ulb.uni-bonn.de/xmlui/handle/20.500.11811/7613`.

17 Michael Etscheid and Heiko Röglin. Smoothed Analysis of the Squared Euclidean Maximum-Cut Problem. In *Algorithms - ESA 2015*, Lecture Notes in Computer Science, pages 509–520, Berlin, Heidelberg, 2015. Springer. `doi:10.1007/978-3-662-48350-3_43`.

18 Michael Etscheid and Heiko Röglin. Smoothed Analysis of Local Search for the Maximum-Cut Problem. *ACM Transactions on Algorithms*, 13(2):25:1–25:12, March 2017. `doi:10.1145/3011870`.

19 C.M. Fiduccia and R.M. Mattheyses. A Linear-Time Heuristic for Improving Network Partitions. In *19th Design Automation Conference*, pages 175–181, June 1982. `doi:10.1109/DAC.1982.1585498`.

20 J. A. Hartigan and M. A. Wong. Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979. `doi:10.2307/2346830`.

21 Susumu Hasegawa, Hiroshi Imai, Mary Inaba, and Naoki Katoh. Efficient Algorithms for Variance-Based k-Clustering. *Proceedings of Pacific Graphics 1993*, February 2000.

22 Sophia Heimann, Hung P. Hoang, and Stefan Hougardy. The k-Opt algorithm for the Traveling Salesman Problem has exponential running time for $k \geq 5$. In *International Colloquium on Automata, Languages, and Programming*, volume 297 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 84:1–84:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. `doi:10.4230/LIPICS.ICALP.2024.84`.

23 Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, June 2010. `doi:10.1016/j.patrec.2009.09.011`.

24 David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, August 1988. `doi:10.1016/0022-0000(88)90046-3`.

25 Daniel M. Kane and Raghu Meka. A PRG for lipschitz functions of polynomials with applications to sparsest cut. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 1–10, New York, NY, USA, June 2013. Association for Computing Machinery. `doi:10.1145/2488608.2488610`.

26 T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, July 2002. `doi:10.1109/TPAMI.2002.1017616`.

27 B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, February 1970. `doi:10.1002/j.1538-7305.1970.tb01770.x`.

28 Christian Komusiewicz and Nils Morawietz. Finding 3-Swap-Optimal Independent Sets and Dominating Sets Is Hard. In *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, volume 241 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 66:1–66:14, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.MFCS.2022.66`.

29 M.W. Krentel. Structure in locally optimal solutions. In *30th Annual Symposium on Foundations of Computer Science*, pages 216–221, October 1989. `doi:10.1109/SFCS.1989.63481`.

30 S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982. `doi:10.1109/TIT.1982.1056489`.

31 Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is NP-hard. *Theoretical Computer Science*, 442:13–21, July 2012. `doi:10.1016/j.tcs.2010.05.034`.

32 Charles F. Mann, David W. Matula, and Eli V. Olinick. The use of sparsest cuts to reveal the hierarchical community structure of social networks. *Social Networks*, 30(3):223–234, July 2008. doi:10.1016/j.socnet.2008.03.004.

33 Bodo Manthey and Jesse van Rhijn. Worst-Case and Smoothed Analysis of the Hartigan-Wong Method for k-Means Clustering. In *41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.STACS.2024.52.

34 J. Matoušek. On Approximate Geometric k -Clustering. *Discrete & Computational Geometry*, 24(1):61–84, January 2000. doi:10.1007/s004540010019.

35 David W. Matula and Farhad Shahrokhi. Sparsest cuts and bottlenecks in graphs. *Discrete Applied Mathematics*, 27(1):113–123, May 1990. doi:10.1016/0166-218X(90)90133-W.

36 Wil Michiels, Jan Korst, and Emile Aarts. *Theoretical Aspects of Local Search*. Monographs in Theoretical Computer Science, An EATCS Series. Springer, Berlin, Heidelberg, 2007. doi:10.1007/978-3-540-35854-1.

37 Burkhard Monien and Tobias Tscheuschner. On the Power of Nodes of Degree Four in the Local Max-Cut Problem. In *Algorithms and Complexity*, Lecture Notes in Computer Science, pages 264–275, Berlin, Heidelberg, 2010. Springer. doi:10.1007/978-3-642-13073-1_24.

38 Christos H. Papadimitriou. The Complexity of the Lin–Kernighan Heuristic for the Traveling Salesman Problem. *SIAM Journal on Computing*, 21(3):450–465, June 1992. doi:10.1137/0221030.

39 Tim Roughgarden and Joshua R. Wang. The Complexity of the k-means Method. In *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 78:1–78:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2016.78.

40 Alejandro A. Schäffer and Mihalis Yannakakis. Simple Local Search Problems that are Hard to Solve. *SIAM Journal on Computing*, 20(1):56–87, February 1991. doi:10.1137/0220004.

41 I. J. Schoenberg. Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, 44(3):522–536, 1938. doi:10.1090/S0002-9947-1938-1501980-0.

42 Leonard J. Schulman. Clustering for edge-cost minimization (extended abstract). In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, STOC '00, pages 547–555, New York, NY, USA, May 2000. Association for Computing Machinery. doi:10.1145/335305.335373.

43 Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000. doi:10.1109/34.868688.

44 Matus Telgarsky and Andrea Vattani. Hartigan's Method: K-means Clustering without Voronoi. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 820–827. JMLR Workshop and Conference Proceedings, March 2010. URL: https://proceedings.mlr.press/v9/telgarsky10a.html.

45 Andrea Vattani. K-means Requires Exponentially Many Iterations Even in the Plane. *Discrete & Computational Geometry*, 45(4):596–616, June 2011. doi:10.1007/s00454-011-9340-1.