Superpolynomial smoothed complexity of 3-FLIP in Local Max-Cut

Lukas Michel[†] Alex Scott[†]

28 March 2024

Abstract

Local search algorithms for NP-hard problems such as Max-Cut frequently perform much better in practice than worst-case analysis suggests. Smoothed analysis has proved an effective approach to understanding this: a substantial literature shows that when a small amount of random noise is added to input data, local search algorithms typically run in polynomial or quasi-polynomial time. In this paper, we provide the first example where a local search algorithm for the Max-Cut problem fails to be efficient in the framework of smoothed analysis. Specifically, we construct a graph with n vertices where the smoothed runtime of the 3-FLIP algorithm can be as large as $2^{\Omega(\sqrt{n})}$.

Additionally, for the setting without random noise, we give a new construction of graphs where the runtime of the FLIP algorithm is $2^{\Omega(n)}$ for any pivot rule. These graphs are much smaller and have a simpler structure than previous constructions.

1 Introduction

In the Max-Cut problem, the vertices of a weighted graph have to be partitioned into two sets such that the sum of the weights of all edges crossing between the sets is maximal. Since this problem is computationally hard to solve [Kar72], local search is often used to compute reasonably good solutions in an acceptable time. A standard form of local search for the Max-Cut problem is the FLIP algorithm which repeatedly moves individual vertices across the cut until it reaches a local optimum; similarly, the k-FLIP algorithm moves up to k vertices in each step. The FLIP algorithm performs well in practice [GNWZ95, DHL99], despite the fact that there exist graphs where the algorithm takes exponential time to terminate [SY91, ET11, MT10].

An important approach for closing the gap between the worst-case analysis and the empirical performance of algorithms is through *smoothed analysis* [ST04]. This method has been particularly effective for local search algorithms [Man15]. In this framework, the runtime of the FLIP algorithm is analysed after a small amount of random noise is added to the edge weights of the graph. Recent work has shown that the smoothed runtime of the FLIP algorithm is polynomial or quasi-polynomial in various different

[†]Mathematical Institute, University of Oxford, United Kingdom ({michel,scott}@maths.ox.ac.uk). Research of Alex Scott supported by EPSRC grant EP/V007327/1.

settings [ER17, CGVG+20, ET11, GGM22, ABPW17, BCC21]. Similarly, the 2-FLIP algorithm has been shown to have a smoothed quasi-polynomial runtime in complete graphs [CGVGY23].

In this paper, we consider the 3-FLIP algorithm. In light of the previous work, it might be expected that smoothed analysis should give the same picture, with a polynomial or near-polynomial smoothed runtime. We show that this is not the case: the smoothed runtime of the 3-FLIP algorithm in an n-vertex graph can be exponentially large in $\sqrt{n}$.

Theorem 1.1. *For all $n \in \mathbb{N}$ there exist graphs G with $\mathcal{O}(n)$ vertices such that the following holds. Let $a < b$ be real numbers. Suppose that the edge weights of G are chosen uniformly at random in $[a, b]$ and consider a uniformly random initial cut of G. Then, with high probability there are executions of the 3-FLIP algorithm from this initial cut that take $2^{\Omega(\sqrt{n})}$ steps.*

This is the first example of a local search algorithm for the Max-Cut problem whose smoothed runtime can be inefficient. We note that Theorem 1.1 is also quite robust: the weights can be smoothed across any interval, and the proofs could be adjusted to handle other smoothing distributions.

Returning to the non-smoothed problem, we also give a new construction of graphs with maximum degree four where the FLIP algorithm takes exponential time to terminate. While graphs with these properties were previously constructed by Monien and Tscheuschner [MT10], our construction has a simpler structure and uses fewer vertices. Additionally, there is exactly one execution of the FLIP algorithm in our graph that ends in a local optimum, while in previous constructions there were multiple.

Theorem 1.2. *For all $n \in \mathbb{N}$ there exist graphs with $\mathcal{O}(n)$ vertices and maximum degree four which have an initial cut from which the unique execution of the FLIP algorithm that ends in a local optimum takes $\Omega(2^n)$ steps.*

This paper is organized as follows. In Sections 1.1 and 1.2 we give some background on local search and smoothed analysis for the Max-Cut problem. We then provide the new construction of graphs with the properties from Theorem 1.2 in Section 2. Theorem 1.1 is proved in Section 3. We conclude with some discussion in Section 4.

1.1 Local Search

The Max-Cut problem has found many practical applications, including circuit layout design [BGJR88, CKC83, Pin84, CD87], clustering [PZ06], the analysis of newsgroups [ARSX03], and more [PT95, BH91, CCG04, Bra07]. However, Max-Cut is NP-hard [Kar72], and so there likely exists no efficient algorithm solving this problem. Instead, in applications of Max-Cut, local search is often used to compute large cuts in a short amount of time.

Local search algorithms are successful in a variety of optimization problems, such as linear programming [ST04] or the travelling salesman problem [JM97]. For Max-Cut, a simple but successful form of local search is the FLIP algorithm, which starts with some initial cut and then repeatedly moves individual vertices across the cut until it reaches a local optimum. A pivot rule determines which vertex to move if there are multiple vertices that could be moved to yield a local improvement. *Local Max-Cut* is the problem of finding a local optimum, that is a cut which cannot be improved by flip-

ping the position of a single vertex. The Local Max-Cut problem is related to Hopfield neural networks [Hop82] and Nash equilibria in party affiliation games [FPT04].

While the FLIP algorithm performs well in practice [GNWZ95, DHL99], worst-case analysis cannot explain its good performance. Schäffer and Yannakakis [SY91] have shown that the Local Max-Cut problem is PLS-complete where PLS is the complexity class of all polynomial local search problems. This is already the case even for Local Max-Cut on graphs whose maximum degree is five [ET11]. Not only does this prove that the Local Max-Cut problem cannot be solved efficiently unless PLS $\subseteq$ P, it also implies that there exist graphs where the FLIP algorithm takes exponential time to terminate. In fact, Monien and Tscheuschner [MT10] have shown that there are graphs of maximum degree four with this property. We provide a new construction of such graphs, proving Theorem 1.2. Note that the condition of having maximum degree four cannot be reduced since the FLIP algorithm runs in polynomial time on graphs whose maximum degree is three [Pol95].

1.2 Smoothed Analysis

Besides the FLIP algorithm, there are also many other algorithms whose worst case analysis incorrectly predicts their real-world performance. To explain this difference for the simplex algorithm, Spielman and Teng [ST04] introduced smoothed analysis. This framework adds a small amount of random noise to the input data to model slight imprecisions occurring in the real world. If the expected runtime of an algorithm on these perturbed inputs is efficient, this may explain why the algorithm has a good practical performance. Smoothed analysis has been applied to many areas in computer science [MR11, ST09], in particular to local search algorithms [ERV14, AMR11, Man15].

To perform a smoothed analysis of the FLIP algorithm, we add a small amount of random noise to each edge weight of the graph. In this setting, Etscheid and Röglin [ER17] have shown that the FLIP algorithm runs in smoothed quasi-polynomial time. This means that the expected runtime is bounded by $\phi n^{\mathcal{O}(\log n)}$ where ϕ is a parameter controlling magnitude of the perturbations applied to the inputs. This bound was later improved to $\phi n^{\mathcal{O}(\sqrt{\log n})}$ by [CGVG+20]. Moreover, in graphs with logarithmic maximum degree and in complete graphs, the FLIP algorithm runs in smoothed polynomial time [ET11, GGM22, ABPW17]. The best known smoothed runtime bound for the FLIP algorithm in complete graphs is $\mathcal{O}(\phi n^{7.83})$ [BCC21].

There are also local search algorithms for the Max-Cut problem which move more than one vertex across the cut in each step [KL70, GNWZ95]. One such algorithm is the k-FLIP algorithm which moves up to k vertices in each step. k-Opt Local Max-Cut is the corresponding problem of finding a cut which cannot be improved even if we are allowed to move up to k vertices across the cut at once. As already noted, the 2-FLIP algorithm has a smoothed quasi-polynomial runtime in complete graphs [CGVGY23]. This result assumes that the pivot rule never moves two vertices if moving just one of them provides a higher improvement.

The proofs of these results show that with a small amount of random noise, the expected number of steps of every possible execution of the local search from any initial cut is at most quasi-polynomial. In contrast, we show with Theorem 1.1 that this is not true for the 3-FLIP algorithm. Even if a large amount of random noise is added

to the graph and if we consider a random initial cut, the runtime of the 3-FLIP algorithm can nevertheless be as large as $2^{\Omega(\sqrt{n})}$. This provides the first example where a local search algorithm for the Max-Cut problem has a superpolynomial smoothed runtime. However, we also note that the long local search sequences in our example contain moves with three vertices where moving just one of them would yield a higher improvement. It therefore remains open whether the 3-FLIP algorithm could have an efficient smoothed runtime for pivot rules that do not make such moves, such as the greedy pivot rule, or for restricted graph classes.

1.3 Preliminaries

For any integer $k \in \mathbb{N}$, we write $[k] := \{1, \dots, k\}$. If $G = (V, E)$ is a graph, we denote by $N(v)$ the neighbourhood of a vertex v. A *cut* of G is a function $\sigma\colon V \to \{-1, 1\}$. This cut partitions the vertices of G into $\{v \in V : \sigma(v) = 1\}$ and $\{v \in V : \sigma(v) = -1\}$. We say that an edge $uv \in E$ *crosses the cut* if $\sigma(u) \neq \sigma(v)$. If G is weighted, we denote its *edge weights* by $X \in \mathbb{R}^E$. The *value of a cut* σ is

$$v(\sigma) := \frac{1}{2} \sum_{uv \in E} (1 - \sigma(u)\sigma(v)) X_{uv}.$$

Thus, the Max-Cut problem is the problem of finding a cut σ which maximizes $v(\sigma)$.

A *k-flip sequence* is a sequence $L = V_1, \dots, V_\ell$ of subsets $V_i \subseteq V$ such that $|V_i| \leq k$ for all $i \in [\ell]$. L can be interpreted as a sequence of ℓ moves whose i-th step moves all vertices from V_i across the cut. For some initial cut σ, we define σ^L to be the cut obtained from σ by performing all moves from L. This means that σ^L is defined by $\sigma^L(v) := (-1)^{n(v)}\sigma(v)$ where $n(v)$ denotes the number of occurrences of v in L. The *change in the cut value* caused by performing all moves from L is $\Delta^L(\sigma) := v(\sigma^L) - v(\sigma)$. For example, for a single vertex v, it can be checked that

$$\Delta^v(\sigma) = \sum_{u \in N(v)} \sigma(u)\sigma(v)X_{uv}.$$

Finally, L is *improving* from σ_0 if $\Delta^{V_i}(\sigma_{i-1}) > 0$ for all $i \in [\ell]$ where σ_i is defined inductively by $\sigma_i := \sigma_{i-1}^{V_i}$. With these definitions, an execution of the k-FLIP algorithm corresponds to an improving k-flip sequence ending in a local optimum.

2 Runtime of the FLIP algorithm

First, we present a small graph where the FLIP algorithm takes exponential time to terminate. Our example is given by the graph G_n and its initial cut depicted in Figure 1. Using $\mathcal{O}(n)$ vertices, this graph generates exponential improving sequences of length $\Omega(3^n)$. The graph is constructed inductively as follows:

- The base case is a graph F_0 consisting of a single edge $v_{0,1}v_{0,8}$ with weight 7 whose endpoints are on opposite sides of the cut.
- We construct a graph F_n from F_{n-1} by adding a path with eight new vertices $v_{n,1}, \dots, v_{n,8}$ to the graph, with consecutive vertices positioned on alternating

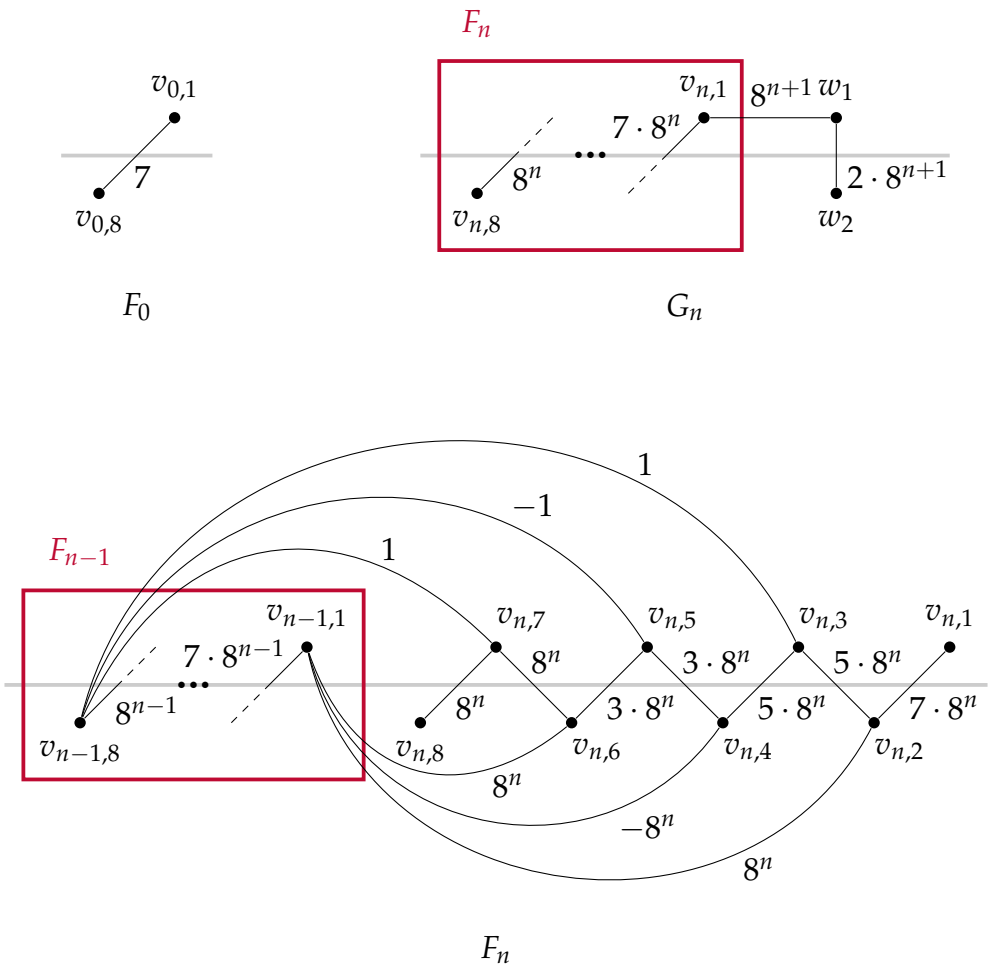

Figure 1: A graph G_n with $\mathcal{O}(n)$ vertices where the FLIP algorithm runs in time $\Omega(3^n)$.

sides of the cut and $v_{n,1}$ on the same side of the cut as $v_{n-1,1}$. For all $k \in [7]$, the edge $v_{n,k}v_{n,k+1}$ is assigned a weight of $(7 - 2\lfloor k/2 \rfloor) \cdot 8^n$.

The new vertices are joined to those of F_{n-1} as follows: $v_{n,2}$, $v_{n,4}$, and $v_{n,6}$ are joined to $v_{n-1,1}$. These edges are assigned a weight of 8^n except for the edge $v_{n,4}v_{n-1,1}$ which receives the negative weight -8^n. The vertices $v_{n,3}$, $v_{n,5}$, and $v_{n,7}$ are connected to $v_{n-1,8}$ with edges of weight of 1, expect that the weight of the edge $v_{n,5}v_{n-1,8}$ is -1.

- To obtain G_n, two new vertices w_1 and w_2 are added to F_n with w_1 on the same side of the cut as $v_{n,1}$ and w_2 on the opposite side. Then, w_1 is joined to $v_{n,1}$ with an edge of weight 8^{n+1} and w_2 is joined to w_1 with an edge of weight $2 \cdot 8^{n+1}$.

In total, G_n has $8n + 4 \in \mathcal{O}(n)$ vertices and $13n + 3 \in \mathcal{O}(n)$ edges whose weights range (in modulus) from 1 to $2 \cdot 8^{n+1}$. The maximum degree of G_n is four.

The idea behind this construction is that the initial configuration of F_n is a local max-cut, but moving the *start vertex* $v_{n,1}$ across this cut will trigger an exponential improving sequence. This sequence will then move the vertices $v_{n,2}$ to $v_{n,8}$ one-by-one across the cut. Moreover, while doing so, it forces the improving sequence of F_{n-1} to be performed three times, once each after the moves of $v_{n,2}$, $v_{n,4}$, and $v_{n,6}$. This results in the exponential growth of the length of that sequence. Because the additional vertices

of G_n ensure that moving $v_{n,1}$ across the cut in the first step is improving, G_n will perform this exponential improving sequence. Moreover, we can prove that this is in fact the unique improving sequence of G_n. This gives the following result.

Theorem 2.1. *For the graph G_n and its initial cut depicted in Figure 1, the unique improving sequence that ends in a local optimum has length $\Omega(3^n)$.*

Proof. Let $L_0 := v_{0,1}, v_{0,8}$ and

$$L_n := v_{n,1}, v_{n,2}, L_{n-1}, v_{n,3}, v_{n,4}, L_{n-1}, v_{n,5}, v_{n,6}, L_{n-1}, v_{n,7}, v_{n,8}.$$

The length of L_n is $\Omega(3^n)$, and a simple induction shows that L_n moves every vertex of F_n an odd number of times across the cut (w_1 and w_2 remain fixed). We claim that L_n is improving from the initial cut of G_n. For this purpose, we will show by induction on n that every step of L_n, except for potentially the first step, is improving when starting from the initial cut of F_n. This is clearly satisfied for the base case L_0. For any $n > 0$, consider the following cases:

- Let $k \in \{2, 4, 6, 8\}$. Then, since the weight of the edge $v_{n,k}v_{n,k-1}$ is higher than the absolute weights of all other edges incident to $v_{n,k}$ combined, moving $v_{n,k}$ across the cut is improving if and only if $v_{n,k}$ and $v_{n,k-1}$ are on the same side of the cut. This condition is satisfied during the move of $v_{n,k}$ in L_n since $v_{n,k}$ and $v_{n,k-1}$ start on opposite sides of the cut but $v_{n,k-1}$ moves across the cut immediately before $v_{n,k}$. Hence, the move of $v_{n,k}$ in L_n is improving.
- Let $k \in \{3, 7\}$. In this case, the edges $v_{n,k}v_{n,k-1}$ and $v_{n,k}v_{n,k+1}$ have the same weight while the third edge $v_{n,k}v_{n-1,8}$ has a weight of 1. So, there are two situations in which moving $v_{n,k}$ across the cut is improving: Either $v_{n,k-1}$ and $v_{n,k+1}$ are both on the same side of the cut as $v_{n,k}$, or this is satisfied by one of these two vertices and additionally the vertex $v_{n-1,8}$.

  Because $v_{n,k}$ and $v_{n-1,8}$ start on opposite sides of the cut and L_{n-1} moves $v_{n-1,8}$ an odd number of times across the cut, $v_{n-1,8}$ will be on the same side of the cut as $v_{n,k}$ during the move of $v_{n,k}$ in L_n. This is also true for $v_{n,k-1}$ since $v_{n,k-1}$ starts on the opposite side of the cut from $v_{n,k}$ but moves exactly once across the cut before the move of $v_{n,k}$. Hence, the second condition from above is satisfied, and so the move of $v_{n,k}$ in L_n is also improving.
- For $v_{n,5}$, the behaviour is almost exactly the same as for $v_{n,3}$ and $v_{n,7}$, except that $v_{n-1,8}$ must be on the opposite side of the cut from $v_{n,5}$ to make the move of $v_{n,5}$ improving. Since this is satisfied in L_n, the move of $v_{n,5}$ is also improving.
- Whenever $v_{n-1,1}$ moves in L_n as the first step of L_{n-1}, it can be checked that either $v_{n,2}$ and $v_{n,6}$ will both be on the same side of the cut as $v_{n-1,1}$, or this is satisfied by one of these two vertices and additionally $v_{n,4}$ is on the opposite side of the cut. In both of these cases, the contributions of the mentioned vertices outweigh all other edges incident to $v_{n-1,1}$ combined, ensuring that the moves of $v_{n-1,1}$ in L_n are improving.
- We also have to reverify that the moves of $v_{n-1,8}$ in L_n as part of L_{n-1} are improving because we have connected new edges to that vertex. However, this is no problem: The weight of the edge $v_{n-1,8}v_{n-1,7}$ is still higher than the absolute weights of all other edges incident to $v_{n-1,8}$ combined. As for $v_{n,8}$, we therefore get that the moves of $v_{n-1,8}$ in L_n are improving.

- Finally, all other moves of L_{n-1} are improving in L_n by induction since we connected no new edges to these vertices and so the improvements of these moves in L_n when starting from F_n are the same as the improvements of these moves in L_{n-1} when starting from F_{n-1}.

Hence, every move of L_n, except for potentially the first move, is improving when starting from the initial cut of F_n. Due to the additional vertices and edges from G_n, the first move of L_n in G_n will also be improving. Therefore, L_n is an improving sequence of length $\Omega(3^n)$ in G_n.

Lastly, we will show that L_n is the only improving sequence of G_n ending in a local optimum, which concludes the proof of the theorem. For this, we show that each move of L_n is the unique improving move at that time step. Then, every improving sequence must perform exactly the moves from L_n, as required. Again, we divide into cases:

- For $k \in \{2, 4, 6, 8\}$, recall that moving $v_{n,k}$ across the cut is improving if and only if $v_{n,k}$ and $v_{n,k-1}$ are on the same side of the cut. However, these two vertices start on opposite sides of cut, and this remains true until $v_{n,k-1}$ moves. Moreover, because $v_{n,k}$ moves immediately after $v_{n,k-1}$, these vertices will again be on opposite sides of the cut after the move of $v_{n,k}$. Hence, $v_{n,k}$ and $v_{n,k-1}$ are only on the same side of the cut during the move of $v_{n,k}$ in L_n, and so $v_{n,k}$ cannot move at any other time step. Similar arguments apply to $v_{n-1,8}$.
- For $k \in \{3, 7\}$, recall that moving $v_{n,k}$ across the cut is improving if and only if either $v_{n,k-1}$ and $v_{n,k+1}$ are both on the same side of the cut as $v_{n,k}$, or if this is satisfied by one of these two vertices and additionally the vertex $v_{n-1,8}$. It can be checked that the first case never occurs while the second case is only satisfied when $v_{n,k}$ moves in L_n. Hence, $v_{n,k}$ cannot move at any other time step than during its move in L_n. Similar arguments apply to $v_{n,5}$.
- For $v_{n-1,1}$, note that a move of that vertex can only become improving if one of its adjacent vertices has moved. Since we already know that $v_{n-1,1}$ moves immediately after every move of $v_{n,2}$, $v_{n,4}$, or $v_{n,6}$ in L_n, this could only be problematic if a move of $v_{n-1,1}$ becomes improving after one of the moves of $v_{n-1,2}$ in L_n. However, this is never the case because $v_{n-1,2}$ will always move to the opposite side of the cut from $v_{n-1,1}$ and will therefore only reinforce the current position of $v_{n-1,1}$ on its side of the cut.
- Finally, for all other vertices of F_{n-1}, we can show by induction that they cannot move at any other time step than during their moves in L_n.

Hence, a move of a vertex of F_n is only improving when this vertex moves in L_n. Moreover, a move of w_1 or w_2 is never improving in G_n. This is again due to similar reasons as for $v_{n,2}$, $v_{n,4}$, $v_{n,6}$, and $v_{n,8}$: Because the edge between w_1 and w_2 outweighs all other edges incident to these vertices, moving one of these two vertices is only improving if they are both on the same side of the cut, but this is never the case. Hence, as required, the moves of L_n are the unique improving moves at their time steps. □

In particular, because the FLIP algorithm performs an improving sequence until it reaches a local optimum, it needs an exponential number of steps to terminate for the graph G_n, and this holds regardless of the chosen pivot rule. This proves Theorem 1.2.

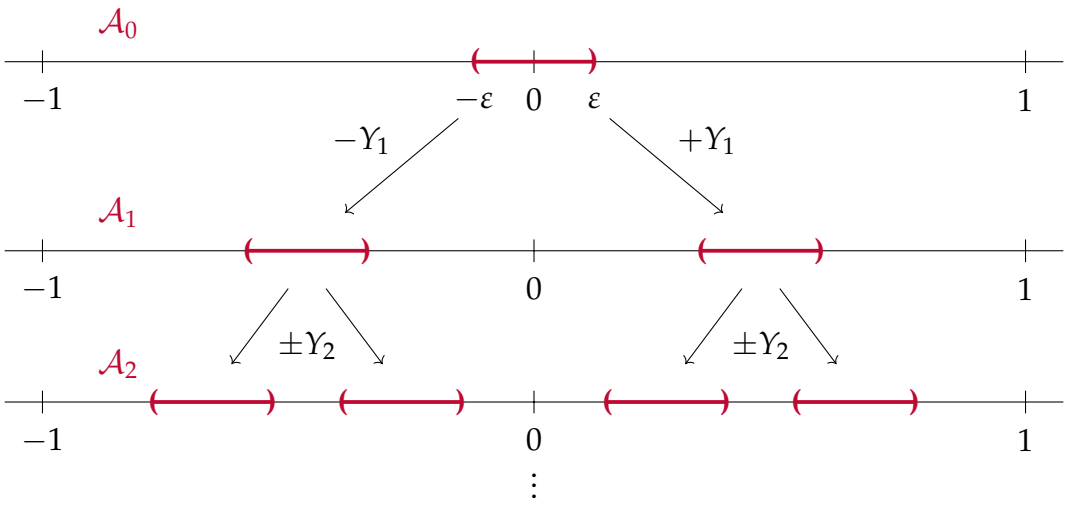

Figure 2: Proof strategy for Theorem 3.1.

3 Smoothed runtime of the 3-FLIP algorithm

Next, we show that the 3-FLIP algorithm can have a superpolynomial smoothed runtime even if we choose all edge weights of the graph uniformly at random.

3.1 Linear combinations of uniform random variables

We will need a result about linear combinations of uniform random variables with coefficients -1 and 1. We will prove that with high probability these linear combinations can approximate any value in a large interval up to some exponentially small error. Since the local improvements during the execution of the 3-FLIP algorithm correspond to such linear combinations, this will allow us to control these local improvements in graphs whose edge weights are chosen uniformly at random.

Theorem 3.1. *There exists $c > 0$ such that the following holds. Let $n \in \mathbb{N}$ be a natural number, $b \in \mathbb{R}$ be a real number, and $X_1, \dots, X_{2n}$ be independent random variables chosen uniformly at random in the interval $[b, b+1]$. Then, it holds that*

$$\mathbb{P}\left(\forall x \in \left[-\frac{n}{5}, \frac{n}{5}\right] : \min_{a_1,\dots,a_{2n} \in \{-1,1\}} \left| x - \sum_{i=1}^{2n} a_i X_i \right| < 2^{-cn}\right) \geq 1 - 2^{-\Omega(n)}.$$

The idea of the proof of this theorem is as follows. For $Y_i := X_i - X_{n+i}$, we want to look at the set $\mathcal{A}_k$ of all values in $[-1, 1]$ that we can approximate up to an error of $\varepsilon := 2^{-cn}$ using linear combinations of $Y_1, \dots, Y_k$, that is

$$\mathcal{A}_k := \left(\left\{\sum_{i=1}^{k} a_i Y_i : a_i \in \{-1, 1\}\right\} + (-\varepsilon, \varepsilon)\right) \cap [-1, 1]$$

where $A + B := \{x + y : x \in A, y \in B\}$ (see Figure 2). If some $\mathcal{A}_k$ covers the entire interval $[-1, 1]$ and if the sum of the remaining random variables is sufficiently large, it will follow that $\mathcal{A}_n$ will cover $[-n/5, n/5]$ as required by Theorem 3.1. To prove that some $\mathcal{A}_k$ covers $[-1, 1]$, we would like to argue that the Lebesgue measure of $\mathcal{A}_k$ grows

by a constant factor $C > 1$ from $\mathcal{A}_k$ to $\mathcal{A}_{k+1}$, so that $\mathcal{A}_n$ is C^n times larger than the set $\mathcal{A}_0 = (-\varepsilon, \varepsilon)$ and must therefore, for a suitable ε, cover $[-1, 1]$ as wanted.

Unfortunately, it is sometimes unlikely that the measure of $\mathcal{A}_k$ grows, for example if $\mathcal{A}_k$ covers almost all of $[-1, 1]$ or if $\mathcal{A}_k$ is close to the boundary of that interval. So instead, we consider linear combinations with coefficients 0 and 1. This ensures that $\mathcal{A}_k$ is always a subset of $\mathcal{A}_{k+1}$. As a consequence, we will show that $\lambda(\mathcal{A}_{k+1}) \geq C \cdot \lambda(\mathcal{A}_k)$ holds with constant positive probability as long as the measure of $\mathcal{A}_k$ is not yet too large. Moreover, once $\lambda(A_k)$ is large, we will show that $\mathcal{A}_{2k}$ is likely to cover the entire interval. We do this by splitting $\mathcal{A}_{2k}$ into two independent copies of $\mathcal{A}_k$ and combining them appropriately. By transforming the coefficients back to -1 and 1, this yields the following proof of Theorem 3.1.

Proof. Let $m := \lfloor n/40 \rfloor$ and $\varepsilon := (200/201)^{m/400}$. Define $Y_i := 2(X_i - X_{n+i})$. For $k \in [n]$ and $\delta > 0$, let

$$\mathcal{A}_k^\delta := \left\{ \sum_{i=1}^{k} a_i Y_i : a_i \in \{0, 1\} \right\} + (-\delta, \delta) \quad \text{and} \quad \mathcal{A}_k := \mathcal{A}_k^\varepsilon \cap [-1, 1].$$

Thus, $\mathcal{A}_k$ is the set of all values in the interval $[-1, 1]$ that can be approximated up to an error of ε using linear combinations of $Y_1, \ldots, Y_k$ with coefficients 0 or 1. Note that $\mathcal{A}_k \subseteq \mathcal{A}_{k+1}$ and so $\lambda(\mathcal{A}_{k+1}) \geq \lambda(\mathcal{A}_k)$. If $\lambda(\mathcal{A}_k) \leq 8/5$, we will show that there is a constant positive probability that $\lambda(\mathcal{A}_{k+1})$ is significantly larger than $\lambda(\mathcal{A}_k)$.

Claim 3.2. *If* $\lambda(\mathcal{A}_k) \leq 8/5$, *then* $\mathbb{P}(\lambda(\mathcal{A}_{k+1}) \geq (201/200) \cdot \lambda(\mathcal{A}_k)) \geq 1/200$.

Proof. Define $\mathcal{B}_k := [-1, 1] \setminus \mathcal{A}_k$. Then, $\mathcal{A}_k \cup ((\mathcal{A}_k + Y_{k+1}) \cap \mathcal{B}_k)$ is a disjoint union contained in $\mathcal{A}_{k+1}$ and so $\lambda(\mathcal{A}_{k+1}) \geq \lambda(\mathcal{A}_k) + \lambda((\mathcal{A}_k + Y_{k+1}) \cap \mathcal{B}_k)$. Since Y_{k+1} takes values in the interval $[-2, 2]$ with probability density function $(2 - |y|)/4$,

$$\begin{aligned}
\mathbb{E}(\lambda((\mathcal{A}_k + Y_{k+1}) \cap \mathcal{B}_k)) &= \int_{-2}^{2} \frac{2 - |y|}{4} \lambda((\mathcal{A}_k + y) \cap \mathcal{B}_k) \, \mathrm{d}y \\
&\geq \frac{1}{20} \int_{-\frac{9}{5}}^{\frac{9}{5}} \lambda((\mathcal{A}_k + y) \cap \mathcal{B}_k) \, \mathrm{d}y \\
&= \frac{1}{20} \int_{-\frac{9}{5}}^{\frac{9}{5}} \int_{\mathcal{A}_k} \mathbb{1}_{x+y \in \mathcal{B}_k} \, \mathrm{d}x \, \mathrm{d}y \\
&= \frac{1}{20} \int_{\mathcal{A}_k} \int_{-\frac{9}{5}}^{\frac{9}{5}} \mathbb{1}_{y+x \in \mathcal{B}_k} \, \mathrm{d}y \, \mathrm{d}x \\
&= \frac{1}{20} \int_{\mathcal{A}_k} \lambda\left(\left(\left[-\frac{9}{5}, \frac{9}{5} \right] + x \right) \cap \mathcal{B}_k \right) \mathrm{d}x.
\end{aligned}$$

Note that $x \in \mathcal{A}_k \subseteq [-1, 1]$ and $\mathcal{B}_k \subseteq [-1, 1]$. Therefore, $\lambda(([-9/5, 9/5] + x) \cap \mathcal{B}_k) \geq \lambda(\mathcal{B}_k) - 1/5 = 9/5 - \lambda(\mathcal{A}_k) \geq 1/5$ and thus

$$\mathbb{E}(\lambda((\mathcal{A}_k + Y_{k+1}) \cap \mathcal{B}_k)) \geq \frac{1}{20} \int_{\mathcal{A}_k} \frac{1}{5} \, \mathrm{d}x = \frac{\lambda(\mathcal{A}_k)}{100}.$$

On the other hand, we know that $\lambda((\mathcal{A}_k + Y_{k+1}) \cap \mathcal{B}_k) \leq \lambda(\mathcal{A}_k + Y_{k+1}) = \lambda(\mathcal{A}_k)$. If $\mathbb{P}(\lambda((\mathcal{A}_k + Y_{k+1}) \cap \mathcal{B}_k) \geq \lambda(\mathcal{A}_k)/200) \leq 1/200$, then

$$\mathbb{E}(\lambda((\mathcal{A}_k + Y_{k+1}) \cap \mathcal{B}_k)) \leq \frac{199}{200}\frac{\lambda(\mathcal{A}_k)}{200} + \frac{1}{200}\lambda(\mathcal{A}_k) < \frac{\lambda(\mathcal{A}_k)}{100},$$

which gives a contradiction. Thus, $\mathbb{P}(\lambda((\mathcal{A}_k + Y_{k+1}) \cap \mathcal{B}_k) \geq \lambda(\mathcal{A}_k)/200) \geq 1/200$. Because the event $\lambda((\mathcal{A}_k + Y_{k+1}) \cap \mathcal{B}_k) \geq \lambda(\mathcal{A}_k)/200$ implies that $\lambda(\mathcal{A}_{k+1}) \geq \lambda(\mathcal{A}_k) + \lambda((\mathcal{A}_k + Y_{k+1}) \cap \mathcal{B}_k) \geq (201/200) \cdot \lambda(\mathcal{A}_k)$, we conclude that

$$\mathbb{P}\left(\lambda(\mathcal{A}_{k+1}) \geq \frac{201}{200}\lambda(\mathcal{A}_k)\right) \geq \mathbb{P}\left(\lambda((\mathcal{A}_k + Y_{k+1}) \cap \mathcal{B}_k) \geq \frac{\lambda(\mathcal{A}_k)}{200}\right) \geq \frac{1}{200}. \qquad \blacksquare$$

By a Chernoff bound, it follows that with high probability approximately $m/200$ of the first m sets $\mathcal{A}_1, \ldots, \mathcal{A}_m$ will either have a measure of at least $8/5$ or otherwise be larger than their predecessor by a factor of $201/200$. Since the original set $\mathcal{A}_0$ has a measure of 2ε, our choice of ε then implies that the measure of $\mathcal{A}_m$ must be at least $8/5$.

Claim 3.3. $\mathbb{P}(\lambda(\mathcal{A}_m) \geq 8/5) \geq 1 - 2^{-\Omega(n)}$.

Proof. Denote by E_k the event $E_k := \{\lambda(\mathcal{A}_k) \geq 8/5 \text{ or } \lambda(\mathcal{A}_k) \geq (201/200) \cdot \lambda(\mathcal{A}_{k-1})\}$. From Claim 3.2 we know that $\mathbb{P}(E_k) \geq 1/200$ holds regardless of the occurrence of $E_1, \ldots, E_{k-1}$. Hence, the sequence of indicator variables $\mathbb{1}_{E_k}$ of these events stochastically dominates a collection of independent Bernoulli random variables $Z_1, \ldots, Z_m$ which take the value 1 with probability $1/200$. With $Z := \sum_{i=1}^{m} Z_i$, we get from a Chernoff bound that

$$\mathbb{P}\left(\sum_{i=1}^{m} \mathbb{1}_{E_i} \geq \frac{m}{400}\right) \geq \mathbb{P}\left(Z \geq \frac{m}{400}\right) \geq 1 - 2^{-\Omega(n)}.$$

If at least one of the events E_k occurs because $\lambda(\mathcal{A}_k) \geq 8/5$ is satisfied, then by monotonicity we will have $\lambda(\mathcal{A}_m) \geq 8/5$. On the other hand, if all of these events occur only because $\lambda(\mathcal{A}_k) \geq (201/200) \cdot \lambda(\mathcal{A}_{k-1})$, then

$$\lambda(\mathcal{A}_m) \geq \left(\frac{201}{200}\right)^{\frac{m}{400}} \lambda(\mathcal{A}_0) = \left(\frac{201}{200}\right)^{\frac{m}{400}} 2\varepsilon = 2 \geq \frac{8}{5}.$$

Hence, in all cases $\sum_{i=1}^{m} \mathbb{1}_{E_i} \geq m/400$ implies that $\lambda(\mathcal{A}_m) \geq 8/5$, and so

$$\mathbb{P}\left(\lambda(\mathcal{A}_m) \geq \frac{8}{5}\right) \geq \mathbb{P}\left(\sum_{i=1}^{m} \mathbb{1}_{E_i} \geq \frac{m}{400}\right) \geq 1 - 2^{-\Omega(n)}. \qquad \blacksquare$$

We have shown that with high probability, $\mathcal{A}_m$ covers at least $4/5$ of the interval $[-1, 1]$ using only the random variables $Y_1, \ldots, Y_m$; the same arguments also apply to $Y_{m+1}, \ldots, Y_{2m}$. We now show that this implies that $\mathcal{A}_{2m}^{2\varepsilon}$ covers the entire interval $[-1, 1]$ with high probability.

Claim 3.4. $\mathbb{P}([-1, 1] \subseteq \mathcal{A}_{2m}^{2\varepsilon}) \geq 1 - 2^{-\Omega(n)}$.

Proof. Define $\mathcal{B}_m := (\{\sum_{i=m+1}^{2m} -a_i Y_i : a_i \in \{0,1\}\} + (-\varepsilon,\varepsilon)) \cap [-1,1]$. Note that $-Y_i$ has the same distribution as Y_i. Thus, as for the set $\mathcal{A}_m$, we get from Claim 3.3 that $\mathbb{P}(\lambda(\mathcal{B}_m) \geq 8/5) \geq 1 - 2^{-\Omega(n)}$. Applying the union bound yields

$$\mathbb{P}\left(\min\{\lambda(\mathcal{A}_m), \lambda(\mathcal{B}_m)\} \geq \frac{8}{5}\right) \geq 1 - 2^{-\Omega(n)}.$$

Assume now that $\min\{\lambda(\mathcal{A}_m), \lambda(\mathcal{B}_m)\} \geq 8/5$ is satisfied, and let $x \in [-1,1]$ be arbitrary. Since $|x| \leq 1$, the set $\mathcal{C} := (\mathcal{B}_m + x) \cap [-1,1]$ satisfies $\lambda(\mathcal{C}) \geq \lambda(\mathcal{B}_m) - 1 \geq 3/5$. If $\mathcal{A}_m$ and $\mathcal{C}$ were disjoint, this would imply $11/5 = 8/5 + 3/5 \leq \lambda(\mathcal{A}_m) + \lambda(\mathcal{C}) = \lambda(\mathcal{A}_m \cup \mathcal{C}) \leq \lambda([-1,1]) = 2$, giving a contradiction. Hence, $\mathcal{A}_m \cap \mathcal{C} \neq \varnothing$ and so there exists some $y \in \mathcal{A}_m \cap \mathcal{C}$.

By definition of $\mathcal{A}_m$ and $\mathcal{C}$, this means that there exist $a_1, \ldots, a_{2m} \in \{0,1\}$ as well as $\delta_1, \delta_2 \in (-\varepsilon,\varepsilon)$ such that $\sum_{i=1}^{m} a_i Y_i + \delta_1 = y = \sum_{i=m+1}^{2m} -a_i Y_i + \delta_2 + x$. Reordering this equation yields $x = \sum_{i=1}^{2m} a_i Y_i + \delta_1 - \delta_2$ with $\delta_1 - \delta_2 \in (-2\varepsilon, 2\varepsilon)$ and thus $x \in \mathcal{A}_{2m}^{2\varepsilon}$. Since $x \in [-1,1]$ was arbitrary, we get $[-1,1] \subseteq \mathcal{A}_{2m}^{2\varepsilon}$, and so we conclude that

$$\mathbb{P}\left([-1,1] \subseteq \mathcal{A}_{2m}^{2\varepsilon}\right) \geq \mathbb{P}\left(\min\{\lambda(\mathcal{A}_m), \lambda(\mathcal{B}_m)\} \geq \frac{8}{5}\right) \geq 1 - 2^{-\Omega(n)}. \qquad \blacksquare$$

We have now shown that every value in the interval $[-1,1]$ can be approximated using linear combinations of $Y_1, \ldots, Y_{2m}$ with coefficients $a_i \in \{0,1\}$ up to an error of 2ε. Next, we transfer this result to linear combinations of $X_1, \ldots, X_n$ with $a_i \in \{-1,1\}$. This effectively shifts the set $\mathcal{A}_{2m}^{2\varepsilon}$ away from the origin, and we use the remaining variables to recenter the set to the origin. This only works if the sum of the absolute values of those variables is sufficiently large. To show that this is the case, recall the following version of Hoeffding's Inequality [Hoe63].

Lemma 3.5 (Hoeffding's Inequality)**.** *Let $Z_1, \ldots, Z_n$ be independent bounded random variables with $Z_i \in [a,b]$ and define $Z := \sum_{i=1}^{n} Z_i$. Then,*

$$\mathbb{P}(|Z - \mathbb{E}(Z)| \geq tn) \leq 2\exp\left(-\frac{2nt^2}{(b-a)^2}\right).$$

We apply this inequality in our setting to the absolute value of $Z_i := X_i - X_{n+i}$.

Claim 3.6. $\mathbb{P}(n/4 \leq \sum_{i=2m+1}^{n} |Z_i|) \geq 1 - 2^{-\Omega(n)}$.

Proof. Let $Z := \sum_{i=2m+1}^{n} |Z_i|$. As $\mathbb{E}(|Z_i|) = 1/3$, we have $\mathbb{E}(Z) = (n-2m)/3 \geq 19n/60$ and so by Hoeffding's Inequality

$$\mathbb{P}(Z \leq n/4) \leq \mathbb{P}\left(|Z - \mathbb{E}(Z)| \geq \frac{n}{15}\right) \leq 2^{-\Omega(n)}. \qquad \blacksquare$$

Finally, we show that the event from the previous claim allow us to recenter the shifted copy of $A_{2m}^{2\varepsilon}$ to the origin. We need the following observation.

Claim 3.7. *Let $x \in [-n/4, n/4]$ be arbitrary and suppose that $n/4 \leq \sum_{i=2m+1}^{n} |Z_i|$. Then, there exist $a_{2m+1}, \ldots, a_n \in \{-1,1\}$ such that $x + \sum_{i=2m+1}^{n} a_i Z_i \in [-1,1]$.*

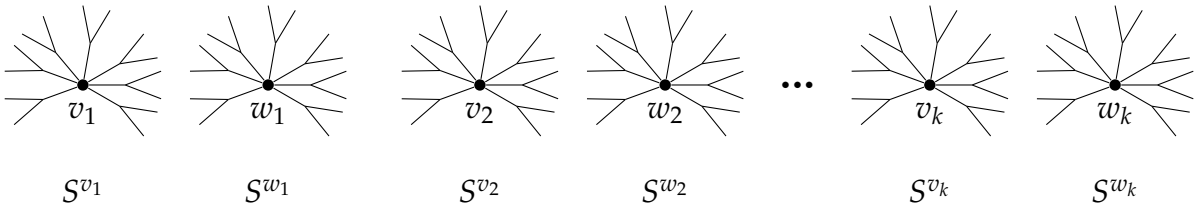

Figure 3: A graph H_k with $n \in \mathcal{O}(k^2)$ vertices where the 3-FLIP algorithm can have a smoothed runtime of up to $2^{\Omega(\sqrt{n})}$.

Proof. We may assume that all Z_i are positive. Start with $a_{2m+1} = \cdots = a_n = 1$ and change these coefficients one-by-one from 1 to -1. Note that $x + \sum_{i=2m+1}^{n} Z_i \geq 0$ and $x + \sum_{i=2m+1}^{n} -Z_i \leq 0$, so this process will change the sum $x + \sum_{i=2m+1}^{n} a_i Z_i$ from a positive to a negative value. Moreover, changing a_i from 1 to -1 changes the value of the sum by at most 2. Thus, at some point it holds that $x + \sum_{i=2m+1}^{n} a_i Z_i \in [-1, 1]$. ■

Finally, assume that the two events from Claims 3.4 and 3.6 occur simultaneously. Let $x \in [-n/5, n/5]$ be arbitrary. Note that $x + \sum_{i=1}^{2m} Z_i \in [-n/4, n/4]$. So, by Claim 3.7 there exist $a_{2m+1}, \ldots, a_n \in \{-1, 1\}$ with $y := x + \sum_{i=1}^{2m} Z_i + \sum_{i=2m+1}^{n} a_i Z_i \in [-1, 1]$. Hence, $y \in \mathcal{A}_{2m}^{2\varepsilon}$ and so there exist $a_1, \ldots, a_{2m} \in \{0, 1\}$ and some $\delta \in (-2\varepsilon, 2\varepsilon)$ such that $y = \sum_{i=1}^{2m} a_i Y_i + \delta$. By the definition of Y_i and Z_i, this yields

$$x + \sum_{i=1}^{2m} (X_i - X_{n+i}) + \sum_{i=2m+1}^{n} a_i (X_i - X_{n+i}) = y = \sum_{i=1}^{2m} 2a_i (X_i - X_{n+i}) + \delta$$

and so

$$x - \sum_{i=1}^{2m} \underbrace{(2a_i - 1)}_{\in\{-1,1\}} (X_i - X_{n+i}) - \sum_{i=2m+1}^{n} \underbrace{-a_i}_{\in\{-1,1\}} (X_i - X_{n+i}) = \delta \in (-2\varepsilon, 2\varepsilon).$$

Note that there exists $c > 0$ such that $2\varepsilon \leq 2^{-cn}$. Since $x \in [-n/5, n/5]$ was arbitrary, we can now apply the union bound to Claims 3.4 and 3.6 to conclude that

$$\mathbb{P}\left(\forall x \in \left[-\frac{n}{5}, \frac{n}{5}\right] : \min_{a_1,\ldots,a_n \in \{-1,1\}} \left| x - \sum_{i=1}^{n} a_i X_i \right| < 2^{-cn} \right) \geq 1 - 2^{-\Omega(n)}. \qquad \square$$

3.2 Construction of a superpolynomial smoothed example

Using Theorem 3.1, we now construct a graph H_k with $n \in \mathcal{O}(k^2)$ vertices where the 3-FLIP algorithm can have a smoothed runtime of $\Omega(2^k) = 2^{\Omega(\sqrt{n})}$. For any $k \in \mathbb{N}$, let $n_k \in \mathcal{O}(k)$ be such that $2^{-cn_k/128} < 3^{-k}$ for the c given by Theorem 3.1. The graph H_k, depicted in Figure 3, is a disjoint union of the following graphs:

- For every $i \in [k]$, the graph contains two trees S^{v_i} and S^{w_i}. These two trees are obtained by taking a copy of $K_{1,2n_k}$ and attaching two new vertices to each leaf. We denote the centers of S^{v_i} and S^{w_i} by v_i and w_i respectively.

Because S^{v_i} and S^{w_i} have $6n_k + 1 \in \mathcal{O}(k)$ vertices each, the graph H_k has a total of $2k(6n_k + 1) \in \mathcal{O}(k^2)$ vertices as claimed. We also remark that we could make H_k

connected by arbitrarily adding edges between the leaves of the trees of H_k. Such edges are irrelevant to the improving 3-flip sequence that we construct.

To study the smoothed complexity of the 3-FLIP algorithm in H_k, suppose that the edge weights of H_k are chosen uniformly at random and consider a uniformly random initial cut σ of H_k. To show that the 3-FLIP algorithm can have a superpolynomial runtime in this setting, we start by moving a subset of the neighbours of v_i and w_i across the cut. This allows us to get a very precise control over the local improvements of a move of v_i or w_i later in the execution of the algorithm. This is given by the following result.

Lemma 3.8. *Let $b \in \mathbb{R}$ be a real number. Suppose that the edge weights of the graph H_k are chosen uniformly at random in the interval $[b, b+1]$ and that σ is a uniformly random cut of H_k. Then, with high probability there exists a 3-flip sequence L that is improving from σ such that $|\Delta^u(\sigma^L) - 3^{-(i-1)}| < 3^{-k}$ for all $i \in [k]$ and $u \in \{v_i, w_i\}$.*

Proof. We may assume that $b \geq -1/2$ since the case $b \leq -1/2$ is analogous. Let $i \in [k]$ and $u \in \{v_i, w_i\}$. For each $s \in N(u)$, denote by $s_1, s_2 \in N(s) \setminus \{u\}$ the two neighbours of s apart from u. Let E_s be the event that $\sigma(s) = \sigma(s_1) = \sigma(s_2)$ and $X_{ss_1} + X_{ss_2} > b+1$, and let $S_u \subseteq N(u)$ be the subset of those neighbours s of u where the event E_s occurs. These events are independent with $\mathbb{P}(E_s) \geq 1/32$. So, a Chernoff bound implies that

$$\mathbb{P}\left(|S_u| \geq \frac{n_k}{32}\right) = \mathbb{P}\left(|S_u| \geq \frac{|N(u)|}{64}\right) \geq 1 - 2^{-\Omega(n_k)}.$$

Define

$$D_u := \sum_{s \in N(u) \setminus S_u} \sigma(u)\sigma(s).$$

Note that $\sigma(u)\sigma(s)$ is chosen uniformly at random in $\{-1, 1\}$, independent of the event E_s. Since D_u is a sum of at most $2n_k$ such independent random variables, a Chernoff bound implies that

$$\mathbb{P}\left(|D_u| \leq \frac{n_k}{64}\right) \geq 1 - 2^{-\Omega(n_k)}.$$

If E_u is the event that $|S_u| \geq n_k/32$ and $|D_u| \leq n_k/64$ are both satisfied, this implies that $\mathbb{P}(E_u) = 1 - 2^{-\Omega(n_k)}$.

Let $m := \lfloor n_k/128 \rfloor$ and let $T_u \subseteq S_u$ be a subset of size $2m$. If E_u occurs, then we have that $|S_u \setminus T_u| \geq n_k/64 \geq |D_u|$. So, the fact that $|N(u) \setminus S_u| + |S_u \setminus T_u| = 2(n_k - m)$ is even implies that there exist coefficients $a_{us} \in \{-1, 1\}$ for $s \in S_u \setminus T_u$ such that $D_u + \sum_{s \in S_u \setminus T_u} a_{us} = 0$. Define

$$\Delta_u := \sum_{s \in N(u) \setminus S_u} \sigma(u)\sigma(s)X_{us} + \sum_{s \in S_u \setminus T_u} a_{us}X_{us}.$$

Note that X_{us} is distributed uniformly at random in $[b, b+1]$, independent of the events E_u and E_s. By pairing up variables with positive and negative coefficients, Δ_u can be written as a sum of $n_k - m$ independent random variables taking values in $[-1, 1]$, each with expectation 0. So $\mathbb{E}(\Delta_u) = 0$ and Hoeffding's Inequality implies that

$$\mathbb{P}\left(|\Delta_u| \leq \frac{n_k}{1024}\right) \geq 1 - 2^{-\Omega(n_k)}.$$

Let F_u be the event that there exist coefficients $a_{us} \in \{-1, 1\}$ for $s \in T_u$ such that $|3^{-(i-1)} - \Delta_u - \sum_{s \in T_u} a_{us} X_{us}| < 3^{-k}$. If E_u and $|\Delta_u| \leq n_k/1024$ are both satisfied, then $|3^{-(i-1)} - \Delta_u| \leq 1 + n_k/1024 \leq m/5$. Moreover, the random variables X_{us} for all $s \in T_u$ are independent of these two conditions. So, Theorem 3.1 shows that the probability of F_u conditioned on these two events will be at least $1 - 2^{-\Omega(n_k)}$, and thus

$$\mathbb{P}(F_u) = \mathbb{P}\left(F_u \;\middle|\; E_u \text{ and } |\Delta_u| \leq \frac{n_k}{1024}\right) \mathbb{P}\left(E_u \text{ and } |\Delta_u| \leq \frac{n_k}{1024}\right) \geq 1 - 2^{-\Omega(n_k)}.$$

Applying the union bound shows that with high probability all the events F_u will occur. Consider the case where this happens.

Let L be the 3-flip sequence moving every vertex $s \in S_u$ with $\sigma(u)\sigma(s) \neq a_{us}$ individually across the cut. Because the event E_s occurs for every $s \in S_u$, these moves are improving. Let $\tau := \sigma^L$ be the resulting cut, so $\tau(s) = -\sigma(s)$ for exactly those $s \in S_u$ with $\sigma(u)\sigma(s) \neq a_{us}$. In particular, this implies that $\tau(u)\tau(s) = a_{us}$ for all $s \in S_u$ while $\tau(u)\tau(s) = \sigma(u)\sigma(s)$ for all $s \in N(u) \setminus S_u$. Therefore,

$$\Delta^u(\tau) = \sum_{s \in N(u)} \tau(u)\tau(s) X_{us} = \Delta_u + \sum_{s \in T_u} a_{us} X_{us},$$

and so we conclude that $|\Delta^u(\tau) - 3^{-(i-1)}| < 3^{-k}$ because the event F_u occurs. □

Given these controlled local improvements, we now construct an improving 3-flip sequence of length $\Omega(2^k)$ in H_k. This 3-flip sequence acts like a binary counter on the vertices $v_1, \ldots, v_k$. Whenever v_i is on the side $\sigma^L(v_i)$ of the cut, this corresponds to a 0 at position i of the binary counter, while otherwise it corresponds to a 1. For any i, the sequence will first perform a binary counter sequence on $v_{i+1}, \ldots, v_k$, will then move v_i across the cut while resetting the positions of $v_{i+1}, \ldots, v_k$, and will afterwards again perform a binary counter sequence on $v_{i+1}, \ldots, v_k$. Since this construction requires a move of v_i to reset the positions of $v_{i+1}, \ldots, v_k$ and these vertices cannot all move in the same step, the vertices $w_{i+1}, \ldots, w_k$ are used to pass on the task of resetting the positions of $v_{i+1}, \ldots, v_k$. This yields the following result.

Lemma 3.9. *Let σ be a cut of the graph H_k with $|\Delta^u(\sigma) - 3^{-(i-1)}| < 3^{-k}$ for all $i \in [k]$ and $u \in \{v_i, w_i\}$. Then, there exists at least one 3-flip sequence that is improving from σ and has length $\Omega(2^k)$.*

Proof. Let $V_i := \{v_i, \ldots, v_k\}$, $U_i := V_i \cup \{w_i, \ldots, w_k\}$, and $U := U_1$. If τ is a cut of H_k satisfying $\tau|_{V \setminus U} = \sigma|_{V \setminus U}$, we have for $u \in U$ that

$$\Delta^u(\tau) = \sum_{s \in N(u)} \tau(u) \underbrace{\tau(s)}_{=\sigma(s) \text{ since } s \notin U} X_{us} = \tau(u)\sigma(u) \sum_{s \in N(u)} \sigma(u)\sigma(s) X_{us} = \tau(u)\sigma(u)\Delta^u(\sigma).$$

Hence, as long as we move only vertices from U, moving u from the side $\sigma(u)$ to the side $-\sigma(u)$ of the cut changes the cut value by $\Delta^u(\sigma)$ while moving u from $-\sigma(u)$ to $\sigma(u)$ yields a change of $-\Delta^u(\sigma)$. In particular, if $i < k$, moving $u \in \{v_i, w_i\}$ from the side $\sigma(u)$ to the side $-\sigma(u)$ of the cut while simultaneously moving v_{i+1} and w_{i+1} from $-\sigma(v_{i+1})$ to $\sigma(v_{i+1})$ and $-\sigma(w_{i+1})$ to $\sigma(w_{i+1})$ respectively increases the cut value by

$$\Delta^u(\sigma) - \Delta^{v_{i+1}}(\sigma) - \Delta^{w_{i+1}}(\sigma) > (3^{-(i-1)} - 3^{-k}) + 2 \cdot (-3^{-i} - 3^{-k}) \geq 0, \tag{1}$$

implying that such a move is improving.

We use this to provide an improving 3-flip sequence of length $\Omega(2^k)$ in H_k. Consider any cut τ with $\tau|_{V\setminus U} = \sigma|_{V\setminus U}$ and $\tau|_{V_i} = \sigma|_{V_i}$. We will show by induction on $k-i$ that there exists a 3-flip sequence L of length $\geq 2^{k-i}$ which is improving from τ and uses only vertices from U_i. The base case $i = k$ is trivially satisfied by the 3-flip sequence $L := (v_k)$ which moves only the vertex v_k across the cut since this move increases the cut value by $\Delta^{v_k}(\sigma) > 3^{-(k-1)} - 3^{-k} > 0$ and is therefore improving.

Now, consider an arbitrary $i \in [k-1]$ and assume that the induction hypothesis holds for $i+1$. Suppose that $\tau|_{V\setminus U} = \sigma|_{V\setminus U}$ and $\tau|_{V_i} = \sigma|_{V_i}$. Since $\tau|_{V_{i+1}} = \sigma|_{V_{i+1}}$, we can immediately apply the induction hypothesis and get an improving 3-flip sequence L_1 of length $\geq 2^{k-(i+1)}$ starting from τ and using only vertices from U_{i+1}. Let $\tau_1 := \tau^{L_1}$ be the resulting cut after the moves from L_1.

Next, let L_2 be the 3-flip sequence moving every vertex $u \in U_{i+1}$ with $\tau_1(u) = \sigma(u)$ individually to the side $-\sigma(u)$ of the cut. Again, these moves increase the cut value by $\Delta^u(\sigma) > 0$. Moreover, the resulting cut $\tau_2 := \tau_1^{L_2}$ will satisfy $\tau_2|_{U_{i+1}} = -\sigma|_{U_{i+1}}$ while $\tau_2(v_i) = \sigma(v_i)$ since v_i was not yet moved at all. That is, v_i is on the side $\sigma(v_i)$ of the cut while v_j and w_j for $j > i$ are on the sides $-\sigma(v_j)$ and $-\sigma(w_j)$ respectively. Then, we perform the following sequence of moves:

$$L_3 := (v_i, v_{i+1}, w_{i+1}),\ (w_{i+1}, v_{i+2}, w_{i+2}),\ (w_{i+2}, v_{i+3}, w_{i+3}),\ \dots,\ (w_{k-1}, v_k, w_k).$$

Since each step of this sequence moves a vertex $u \in \{v_j, w_j\}$ from the side $\sigma(u)$ to the side $-\sigma(u)$ of the cut while moving v_{j+1} and w_{j+1} from $-\sigma(v_{j+1})$ to $\sigma(v_{j+1})$ and $-\sigma(w_{j+1})$ to $\sigma(w_{j+1})$ respectively, we know from (1) that all of these moves are improving. Moreover, because L_3 moves every vertex from V_{i+1} exactly once across the cut, the resulting cut $\tau_3 := \tau_2^{L_3}$ satisfies $\tau_3|_{V_{i+1}} = -\tau_2|_{V_{i+1}} = \sigma|_{V_{i+1}}$. Hence, we can again apply the induction hypothesis and perform an improving 3-flip sequence L_4 of length $\geq 2^{k-(i+1)}$ starting from τ_3 and using only the vertices from U_{i+1}.

Let L be the concatenation of L_1, L_2, L_3, and L_4. Since each of these four sequences is improving, L is improving. Moreover, L uses only vertices from U_i since this holds for L_1, L_2, L_3, and L_4. Lastly, because the lengths of L_1 and L_4 are $\geq 2^{k-(i+1)}$, the length of L is $\geq 2 \cdot 2^{k-(i+1)} = 2^{k-i}$. Therefore, the induction hypothesis holds for i and so by induction for all $i \in [k]$. With $i = 1$ and $\tau = \sigma$, this yields a 3-flip sequence of length $\geq 2^{k-1} \in \Omega(2^k)$ which is improving from σ. □

Combining these two lemmas and rescaling the edge weights proves Theorem 1.1.

We note that the 3-FLIP algorithm can be easily adapted in this example to avoid the improving 3-flip sequence of superpolynomial length. For instance, this can be achieved by using the greedy pivot rule which always chooses the move maximizing the local improvement in each step. We also note that our example is quite sparse. So, for specific pivot rules and dense graphs, the smoothed complexity of the 3-FLIP algorithm remains open.

It is also interesting to consider the 2-FLIP algorithm. However, the example from above cannot be adapted to show that the 2-FLIP algorithm would have a superpolynomial smoothed runtime since it would only yield improving 2-flip sequences of length $\Omega(k^2)$. In fact, we can prove the following.

Theorem 3.10. *Let G be a graph and $I \subseteq V$ be an independent set. Then, every improving 2-flip sequence of G which uses only vertices from I has a length of $\mathcal{O}(n^2)$.*

Proof. Let $I = \{v_1, \dots, v_k\}$. Since I is an independent set, moving a vertex v_i across the cut will always result in the same change to the cut value. Denote this change by $\Delta_i > 0$ and let s_i be such that moving v_i from the side s_i to $-s_i$ of the cut changes the cut value by Δ_i while moving v_i from $-s_i$ to s_i changes the cut value by $-\Delta_i$. By sorting the vertices $v_1, \dots, v_k$, we may assume that $\Delta_1 \geq \Delta_2 \geq \cdots \geq \Delta_k$.

Let L be any improving 2-flip sequence of G. For any cut σ of G, assign a token to every vertex v_i with $\sigma(v_i) = s_i$ and consider how these tokens change during the execution of L. For this purpose, assume that a single step of L moves two vertices v_i and v_j across the cut where $i < j$. Since this move is improving and $\Delta_i \geq \Delta_j$, this step must move v_i from the side s_i to $-s_i$ of the cut. In particular, v_i loses a token during this step. If v_j gains a token during this move, we imagine the token to be moved from v_i to v_j.

Hence, during every step of L, either at least one token gets removed from G or a token moves from some vertex v_i to a vertex v_j with $j > i$. Since there are at most k tokens at the beginning and each of these tokens can move at most $k - 1$ times, it follows that the length of L is at most $k^2 \in \mathcal{O}(n^2)$. □

As the 3-flip sequence from Lemma 3.9 moved only vertices from an independent set, no small change to this example can yield superquadratic improving 2-flip sequences.

4 Discussion and open problems

For NP-hard problems, smoothed analysis is an extremely powerful framework for proving that algorithms typically run in polynomial or near-polynomial time. However, the results in this paper show that the effectiveness of this framework can depend on the details of the analysed algorithms. It would be of great interest to find more general conditions under which polynomial runtime can be achieved in this framework.

Many interesting questions remain open for further investigation.

- Is the smoothed runtime of the FLIP algorithm on arbitrary graphs polynomial? Only quasi-polynomial smoothed runtime bounds are known [ER17, CGVG+20].
- Is the smoothed runtime of the 2-FLIP algorithm on complete graphs polynomial? Again, only quasi-polynomial runtime bounds are known [CGVGY23].
- What is the behaviour of the 2-FLIP algorithm on arbitrary graphs? Boodaghians, Kulkarni, and Mehta [BKM18] showed that an efficient smoothed runtime of the 2-FLIP algorithm would also prove that the sequential-better-response algorithm for k-strategy network coordination games has an efficient smoothed runtime.
- We have shown that the smoothed runtime of the 3-FLIP algorithm can be as large as $2^{\Omega(\sqrt{n})}$. Is it possible to avoid this by choosing the right pivot rule? It would be very interesting to determine whether the 3-FLIP algorithm has an efficient smoothed runtime for specific pivot rules (for example the greedy pivot rule, or the uniform random pivot rule) or in certain graph classes (like complete graphs). It seems difficult to adapt the example from this paper to these settings. The same questions arise for the k-FLIP algorithm with $k > 3$.

References

[ABPW17] Omer Angel, Sébastien Bubeck, Yuval Peres, and Fan Wei (2017). Local Max-Cut in smoothed polynomial time. *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 429–437. ↑2, 3

[AMR11] David Arthur, Bodo Manthey, and Heiko Röglin (2011). Smoothed analysis of the k-means method. *Journal of the ACM (JACM)* **58**(5), 1–31. ↑3

[ARSX03] Rakesh Agrawal, Sridhar Rajagopalan, Ramakrishnan Srikant, and Yirong Xu (2003). Mining newsgroups using networks arising from social behavior. *Proceedings of the 12th International Conference on World Wide Web*, 529–535. ↑2

[BCC21] Ali Bibak, Charles Carlson, and Karthekeyan Chandrasekaran (2021). Improving the smoothed complexity of FLIP for max cut problems. *ACM Transactions on Algorithms (TALG)* **17**(3), 1–38. ↑2, 3

[BGJR88] Francisco Barahona, Martin Grötschel, Michael Jünger, and Gerhard Reinelt (1988). An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research* **36**(3), 493–513. ↑2

[BH91] Endre Boros and Peter L. Hammer (1991). The Max-Cut problem and quadratic 0–1 optimization; polyhedral aspects, relaxations and bounds. *Annals of Operations Research* **33**(3), 151–180. ↑2

[BKM18] Shant Boodaghians, Rucha Kulkarni, and Ruta Mehta (2018). Smoothed efficient algorithms and reductions for network coordination games arXiv:1809.02280. ↑16

[Bra07] Yann Bramoullé (2007). Anti-coordination and social interactions. *Games and Economic Behavior* **58**(1), 30–49. ↑2

[CCG04] Yangchi Chen, Melba M. Crawford, and Joydeep Ghosh (2004). Integrating support vector machines in a hierarchical output space decomposition framework. *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium*, vol. 2, 949–952 (IEEE). ↑2

[CD87] Kevin C. Chang and David H. Du (1987). Efficient algorithms for layer assignment problem. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **6**(1), 67–78. ↑2

[CGVG+20] Xi Chen, Chenghao Guo, Emmanouil V. Vlatakis-Gkaragkounis, Mihalis Yannakakis, and Xinzhi Zhang (2020). Smoothed complexity of local Max-Cut and binary Max-CSP. *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, 1052–1065. ↑2, 3, 16

[CGVGY23] Xi Chen, Chenghao Guo, Emmanouil-Vasileios Vlatakis-Gkaragkounis, and Mihalis Yannakakis (2023). Smoothed complexity of SWAP in local graph partitioning arXiv:2305.15804. ↑2, 3, 16

[CKC83] Ruen-Wu Chen, Yoji Kajitani, and Shu-Park Chan (1983). A graph-theoretic via minimization algorithm for two-layer printed circuit boards. *IEEE Transactions on Circuits and Systems* **30**(5), 284–299. ↑2

[DHL99] Oliver Dolezal, Thomas Hofmeister, and Hanno Lefmann (1999). A comparison of approximation algorithms for the MaxCut-problem. *Manuscript, Universität Dortmund, Lehrstuhl Informatik* . ↑1, 3

[ER17] Michael Etscheid and Heiko Röglin (2017). Smoothed analysis of local search for the maximum-cut problem. *ACM Transactions on Algorithms (TALG)* **13**(2), 1–12. ↑2, 3, 16

[ERV14] Matthias Englert, Heiko Röglin, and Berthold Vöcking (2014). Worst case and probabilistic analysis of the 2-Opt algorithm for the TSP. *Algorithmica* **68**(1), 190–264. ↑3

[ET11] Robert Elsässer and Tobias Tscheuschner (2011). Settling the complexity of local Max-Cut (almost) completely. *International Colloquium on Automata, Languages, and Programming*, 171–182 (Springer). ↑1, 2, 3

[FPT04] Alex Fabrikant, Christos Papadimitriou, and Kunal Talwar (2004). The complexity of pure Nash equilibria. *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, 604–612. ↑3

[GGM22] Yiannis Giannakopoulos, Alexander Grosz, and Themistoklis Melissourgos (2022). On the smoothed complexity of combinatorial local search arXiv:2211.07547. ↑2, 3

[GNWZ95] Wilsin Gosti, Giao Nguyen, Marlene Wan, and Min Zhou (1995). Approximation algorithm for the Max-Cut problem. *Technical Report, University of California, Department of Electrical Engineering and Computer Science* . ↑1, 3

[Hoe63] Wassily Hoeffding (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association* **58**, 13–30. ↑11

[Hop82] John J. Hopfield (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences* **79**(8), 2554–2558. ↑3

[JM97] David S. Johnson and Lyle A. McGeoch (1997). The traveling salesman problem: A case study, 215–310 (John Wiley and Sons). ↑2

[Kar72] Richard M. Karp (1972). Reducibility among combinatorial problems. *Complexity of Computer Computations*, 85–103. ↑1, 2

[KL70] Brian W. Kernighan and Shen Lin (1970). An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal* **49**(2), 291–307. ↑3

[Man15] Bodo Manthey (2015). Smoothed analysis of local search algorithms. *Algorithms and data structures, Lecture Notes in Computer Science*, vol. 9214, 518–527. ↑1, 3

[MR11] Bodo Manthey and Heiko Röglin (2011). Smoothed analysis: Analysis of algorithms beyond worst case. *IT-Information Technology* **53**(6), 280–286. ↑3

[MT10] Burkhard Monien and Tobias Tscheuschner (2010). On the power of nodes of degree four in the local Max-Cut problem. *International Conference on Algorithms and Complexity*, 264–275 (Springer). ↑1, 2, 3

[Pin84] Ron Y. Pinter (1984). Optimal layer assignment for interconnect. *Advances in VLSI and Computer Systems* **1**(2), 123–137. ↑2

[Pol95] Svatopluk Poljak (1995). Integer linear programs and local search for Max-Cut. *SIAM Journal on Computing* **24**(4), 822–839. ↑3

[PT95] Svatopluk Poljak and Zsolt Tuza (1995). Maximum cuts and large bipartite subgraphs. *DIMACS Series* **20**, 181–244. ↑2

[PZ06] Jan Poland and Thomas Zeugmann (2006). Clustering pairwise distances with missing data: Maximum cuts versus normalized cuts. *International Conference on Discovery Science*, 197–208 (Springer). ↑2

[ST04] Daniel A. Spielman and Shang-Hua Teng (2004). Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)* **51**(3), 385–463. ↑1, 2, 3

[ST09] DANIEL A. SPIELMAN and SHANG-HUA TENG (2009). Smoothed analysis: An attempt to explain the behavior of algorithms in practice. *Communications of the ACM* **52**(10), 76–84. ↑3

[SY91] ALEJANDRO A. SCHÄFFER and MIHALIS YANNAKAKIS (1991). Simple local search problems that are hard to solve. *SIAM Journal on Computing* **20**(1), 56–87. ↑1, 3