Online Discrepancy with Recourse for Vectors and Graphs

Anupam Gupta[*] Vijaykrishna Gurunathan [†] Ravishankar Krishnaswamy [‡]
Amit Kumar [§] Sahil Singla [¶]

Abstract

The vector-balancing problem is a fundamental problem in discrepancy theory: given T vectors in $[-1,1]^n$, find a signing $\sigma(a) \in \{\pm 1\}$ of each vector a to minimize the discrepancy $\| \sum_a \sigma(a) \cdot a\|_\infty$. This problem has been extensively studied in the static/offline setting. In this paper we initiate its study in the fully-dynamic setting with *recourse*: the algorithm sees a stream of T insertions and deletions of vectors, and at each time must maintain a low-discrepancy signing, while also minimizing the amortized recourse (the number of times any vector changes its sign) per update.

For general vectors, we show algorithms which almost match Spencer's $O(\sqrt{n})$ offline discrepancy bound, with $O(n \operatorname{polylog} T)$ amortized recourse per update. The crucial idea behind our algorithm is to compute a basic feasible solution to the linear relaxation in a distributed and recursive manner, which helps find a low-discrepancy signing. We bound the recourse using the distributed computation of the basic solution, and argue that only a small part of the instance needs to be re-computed at each update.

Since vector balancing has also been greatly studied for sparse vectors, we then give algorithms for *low-discrepancy edge orientation*, where we dynamically maintain signings for 2-sparse vectors in an n-dimensional space. Alternatively, this can be seen as orienting a dynamic set of edges of an n-vertex graph to minimize the discrepancy, i.e., the absolute difference between in- and out-degrees at any vertex. We present a deterministic algorithm with $O(\operatorname{polylog} n)$ discrepancy and $O(\operatorname{polylog} n)$ amortized recourse. The core ideas are to dynamically maintain an expander-decomposition with low recourse (using a very simple approach), and then to show that, as the expanders change over time, a natural local-search algorithm converges quickly (i.e., with low recourse) to a low-discrepancy solution. We also give strong lower bounds (with some matching upper bounds) for local-search discrepancy minimization algorithms for vector balancing and edge orientation.

1 Introduction

In the Online Vector Balancing problem introduced by Spencer [Spe77], vectors $a_1, a_2, \ldots, a_T \in [-1,1]^n$ arrive online, and the algorithm irrevocably assigns a sign $\sigma(a_t)$ immediately upon seeing a_t, with the goal of minimizing the *discrepancy of the signed sum*, i.e., $\| \sum_t \sigma(a_t) \cdot a_t\|_\infty$. Following a sequence of works [BS20, BJSS20, BJM+21], the state-of-the-art bounds for this problem is an elegant randomized algorithm that maintains a discrepancy of $O(\sqrt{n}\log(nT))$ [ALS21]. Their result assumes an *oblivious* adversary, so that the choice of arriving vectors does not depend on the internal state of the algorithm. Indeed, if we allow *adaptive* adversaries then every online algorithm incurs $\Omega(\sqrt{T})$ discrepancy [Spe77]. We think of $T \gg n$, so $\Omega(\sqrt{T})$ is much larger than $O(\sqrt{n}\log(nT))$.

We initiate the study of Fully-Dynamic Vector Balancing, where vectors can both arrive or depart at each time step, and the algorithm must always maintain a low-discrepancy signing of the vectors present in the

[*](anupamg@cs.cmu.edu) Computer Science Department, Carnegie Mellon University. Supported in part by NSF awards CCF-1907820, CCF1955785, and CCF-2006953.

[†](krishnavijay1999@gmail.com) Computer Science Department, Stanford University

[‡](rakri@microsoft.com) Microsoft Research.

[§](amitk@cse.iitd.ac.in) Department of Computer Science and Engineering, Indian Institute of Technology Delhi.

[¶](ssingla@gatech.edu) School of Computer Science, Georgia Tech.

system at all times. Since it is easy to construct examples where no algorithm can guarantee non-trivial discrepancy bounds if it is forced to commit to the sign of a vector upon arrival, we study the problem where the algorithm can *re-sign the vectors from time to time.* Indeed, many real-world applications that motivate such discrepancy-based methods (such as in fair allocations, sparsification routines, etc.) have a fully-dynamic flavor to them, with the corresponding inputs being dynamic in nature due to both insertions and deletions.

Problem (Fully-Dynamic Vector Balancing). *We start with an empty collection of* active *vectors $A(0)$. At each time/update $t \in [T]$, an adaptive adversary either inserts a new vector $a_t \in [-1,1]^n$, i.e., $A(t) = A(t-1) \cup \{a_t\}$, or removes an existing vector $a \in A(t-1)$, i.e., $A(t) = A(t-1) \setminus \{a\}$. The goal is to maintain* signings *$\sigma_t : A(t) \to \{\pm 1\}$ to minimize the norm $\| \sum_{a \in A(t)} \sigma_t(a) \cdot a \|_\infty$. The algorithm can reassign the sign of a vector a (i.e., set $\sigma_t(a) \neq \sigma_{t-1}(a)$), and the total* recourse *is the sum total of the reassignments.*

Two trivial solutions exist: (a) recomputing low-discrepancy signings on the active set of vectors after every update operation incurs optimal offline discrepancy guarantees with a recourse of $\Theta(T)$ per update, and (b) an independent and uniformly random signing of every new vector maintains at any time t a signing of discrepancy $\Theta(\sqrt{T \log n})$ w.h.p., while performing no recourse whatsoever. Since $T \gg n$, this is much larger than the optimal offline discrepancy bounds of $O(\sqrt{n})$ for any collection of T vectors in $[-1,1]^n$ [Spe85, Ban10, LM15]. We ask: *can we get near-optimal*[1] *discrepancy bounds with a small amount of recourse?*

1.1 Our Results and Techniques

Fully-Dynamic Vector Balancing. Our first main contribution is the design of an algorithm which maintains low-discrepancy signings for the fully-dynamic problem that nearly matches the offline discrepancy bounds while giving an amortized recourse that is only logarithmic in the sequence length T.

Theorem 1.1 (Fully-Dynamic: General Vector Balancing). *There is an efficient algorithm for Fully-Dynamic Vector Balancing with update vectors in $[-1,1]^n$ which maintains signings $\sigma_t(\cdot)$ with discrepancy $O(\sqrt{n})$ and an amortized recourse of $O(n \log T)$ per update, even against adaptive adversaries. For Komlos' setting, i.e., if all the updates vectors have ℓ_2 length at most 1 (instead of ℓ_∞ length), the algorithm achieves discrepancy $O(\sqrt{\log(n)})$ with an amortized recourse of $O(n \log T)$ per update.*

Since in this theorem we are competitive against *adaptive adversaries*, it illustrates the power of recourse: in the absence of recourse, we get $\Omega(\sqrt{T})$ lower bounds on the discrepancy even for arrival-only sequences of 2-dimensional vectors. This is because the adversary can always make the next vector to be orthogonal to the current signed sum.

At a very high level, our algorithm divides the instance into many parts of size $O(n)$, obtains a good partial signing for each part (such that all but n vectors are signed), and recurses on the residual instance. The algorithm imposes a tree-like hierarchy on these parts, so that it can easily adapt to inserts or deletes with bounded recourse by only re-running the computations on the part suffering the insertion/deletion, and on any internal node on the corresponding root-leaf path from that part to the root. If we are not careful, the discrepancy of the overall vector can be proportional to the number of parts, since we could accrue error in each part. However, we use linear algebraic ideas inspired by [BG81] to *couple all the parts*, thereby always ensuring that the sum of the partial signings across all nodes of the tree (except the root) is zero.

Fully-Dynamic Edge Orientation/Carpooling and Local Search. Given the general result above, next we focus on the special case of orienting edges of a graph to minimize the maximum imbalance between the in- and out-degrees. Fagin and Williams [FW83] posed the *carpooling problem*, which corresponds to vector balancing with vectors of the form $(0,\ldots,0,1,0,\ldots,0,-1,0,\ldots) \in \mathbb{R}^n$, and the graph discrepancy objective is precisely the $\|\cdot\|_\infty$ of the signed sum of vectors. [FW83, AAN+98] use this problem to model fairness in scheduling, where edges represent shared commitments (such as carpooling), orientations give primary

[1]In this paper, we use "near-optimal" to mean optimal up to poly-logarithmic factors.

and secondary partners of the commitment (e.g., driver and co-driver), and hence the discrepancy measures fairness for individuals, in terms of how many commitments he/she is the primary partner for, relative to the total number of commitments he/she is a part of.

Somewhat surprisingly, [AAN+98] showed that any algorithm must suffer $\Omega(n)$ discrepancy on some worse-case adaptive sequence of edge arrivals. On the other hand, for an oblivious sequence of edge arrivals, it is easy to maintain orientations with $O(\sqrt{n \log n})$ discrepancy by simply orienting edges randomly (while always orienting repeated parallel edges (u, v) oppositely). To mitigate such strong lower bounds, [AAN+98] and recently Gupta et al. [GKKS20] study a stochastic version of the problem where the arriving edges are sampled from a known distribution: they design algorithms to maintain $\operatorname{polylog}(n, T)$-discrepancy. The recent algorithm of Alweiss et al. [ALS21] also extends to this special case giving $O(\log(nT))$ discrepancy bounds for any oblivious sequence of edge arrivals, not just stochastic ones. None of these prior algorithms extend to a fully-dynamic input consisting of both insertions and deletions. Moreover, Theorem 1.1 guarantees near-optimal discrepancy only with $O(n \log T)$ amortized recourse. In this paper, we give *deterministic* near-optimal discrepancy algorithms with near-optimal amortized recourse.

Theorem 1.2 (Fully-Dynamic Edge Orientation). *There is an efficient deterministic algorithm that maintains an orientation of* $\operatorname{polylog} n$ *discrepancy while performing an amortized recourse of* $\operatorname{polylog} n$ *per update.*

Since this algorithm is deterministic, the guarantees also hold against adaptive adversaries: there are $\Omega(n)$ discrepancy bounds for no-recourse algorithms against such adversaries, even for the setting of only arrivals.

At a high level, our algorithm can be seen as a composition of two modules. Firstly, we consider a simple local-search procedure, which flips an edge from $u \to v$ to $v \to u$ if the current discrepancy of v exceeds that of u by more than 2. Clearly, this reduces the discrepancy of the maximum of these two vertices. Our crucial observation is that this process always maintains low-discrepancy signings when the graph is an *expander*. We find this interesting, since we can show that there are bad local optima with $\operatorname{poly}(n)$ discrepancy for general graphs. Secondly, we show how to dynamically maintain a partitioning of the edge set of an arbitrary graph G into a disjoint collection of expanders $G_1, G_2, \ldots, G_\ell$ with each vertex appearing in at most $\operatorname{polylog} n$ many expanders, such that the *amortized number of changes* to G_1, G_2, $\ldots, G_\ell$, per update to G is bounded. (This expander decomposition can be viewed as a "preconditioning" step.) We build on ideas recently developed for dynamic graph algorithms [SW19, BvdBG+20]: our challenge is to show that dynamic expander decomposition can be done along with local search on the individual expanders, and specifically to control the potential functions that guide our proofs.

Indeed, using the above two modules to obtain Theorem 1.2 requires new ideas. When an update (insertion or deletion) occurs to G, we first modify our expander decomposition, and re-run local search *starting from the prior local optima* in each expander. While this ensures good discrepancy bounds, it could lead to many local search moves. In order to bound the latter quantity, our idea is to use a potential function in each expander to bound the recourse, such that each step of local search always decreases the potential by at least a constant. This is somewhat delicate: a single update in G can change any particular expander G_i by a lot (even though the amortized recourse is bounded), and hence the single-step potential change can be huge. We show how to maintain some Lipschitzness properties for our potential function under inserts and deletes, which gives us the final bounds of $\operatorname{polylog} n$ on both the discrepancy and the recourse.

Along the way, we also develop a better understanding of the strengths and limitations of local search as a technique for discrepancy minimization problems, both for graphs and for general vectors.

Theorem 1.3 (Informal: Discrepancy of Local Optima). *For edge orientation in expanders, any locally optimal solution for local search using the simple potential $\Phi = \sum_{v \in V} disc(v)^2$ has discrepancy $O(\log n)$. For arbitrary graphs, however, the discrepancy can be as bad as $\Omega(n^{1/3})$. For general vectors in $\{\pm 1\}^n$ (and in $[-1, 1]^n$), the local search bound using the simple potential $\Phi = \sum_{i \in [n]} \left(\sum_t \sigma(a_t) \cdot a_t(i) \right)^2$ deteriorates to $\Omega(2^n)$ (and to $\Omega(\sqrt{T})$).*

Signing s-Sparse Vectors for Online Arrivals. Finally, we consider the problem with s-sparse vectors, which interpolates between the graphical case of $s = 2$ and the general case. In the offline setting, the classical linear-algebraic algorithm of Beck and Fiala [BF81] constructs a signing with disrepancy $2s - 1$ (independent of n and T). Subsequent works by Banaszczyk [Ban98] and Bansal, Dadush, and Garg [BDG16] develop techniques to get discrepancy $O(\sqrt{s \log n})$, and a long-standing question in discrepancy theory is to improve this bound to $O(\sqrt{s})$. Here we study the Online Vector Balancing problem only with arrivals. In this setting, the algorithm of [ALS21] maintains signings of discrepancy $O(\sqrt{s}\log(nT))$ without recourse, but against an *oblivious* adversary. In Section 6 we give a generic reduction that can maintain near-optimal discrepancy for Online Vector Balancing against adaptive adversaries, with small recourse.

Theorem 1.4 (Arrivals Only: Online Vector Balancing with Recourse)**.** *There is an efficient algorithm for Online Vector Balancing with s-sparse vectors that achieves $O(\sqrt{s \log n} \log T)$ discrepancy and $O(\log T)$ amortized recourse per update against an adaptive adversary.*

1.2 Further Related Work

Discrepancy theory is a rich and vibrant area of research [Cha01, Mat09]. While some initial works [Spe77, Bá79] focused on the online discrepancy problem, the majority of research dealt with the offline setting, where the T vectors are given upfront. Near-optimal results are known for settings such as discrete set systems [Spe85, Ban10, LM15] (i.e., vectors in $\{0, 1\}^n$), *sparse* set systems [BF81] (s-sparse binary vectors), and general vectors in the unit ball [Ban98, Bec81, Gia97, Rot14, BDGL19].

There has been a renewed interest in the online discrepancy setting, where many of the techniques developed for the offline setting no longer extend. Most of the results for online vector discrepancy deal with *stochastic settings* of the problem where the arriving vectors satisfy some distributional assumptions [AAN+98, BS20, BJM+21]. A recent breakthrough work [ALS21] gives a very elegant randomized algorithm with near-optimal discrepancy for general vectors arriving online. However, to the best of our knowledge, none of these ideas easily extend to the fully-dynamic setting where vectors can also depart—which is the focus of this paper. In fact, we do not know how to adapt existing ideas to establish non-trivial results for even the simple *deletions-only* setting: starting with T vectors, a uniformly random subset of $T/2$ of these vectors are deleted one-by-one. Can we always maintain a low-discrepancy signing of the remaining vectors with small recourse?

The study of dynamic algorithms also has a rich history, both in the *recourse* model, which measures the number of updates made the algorithm per update, and the *update-time* model, which measures the running time of the algorithm per update. Apart from graph problems, these models have been studied in a variety of settings such network design [IW91, GK14, GGK16, ŁOP+15], clustering [GKLX20, CAHP+19], matching [GKKV95, CDKL09, BLSZ14], and scheduling [PW93, Wes00, AGZ99, SSS09, SV10, EL14, GKS14], and set cover [BHI18, BCH17, BK19, AAG+19, BHN19, BHNW20, GKKP17].

A different version of edge-orientation, commonly known as graph balancing, involves minimizing just the maximum in-degree (see, e.g., [BF99, Kow07, KKPS14]): the techniques used for that version seem quite different from those needed here.

Paper Outline. We present the results for Fully-Dynamic Vector Balancing and specifically Theorem 1.1 in §2. The results for graph balancing appear in §3. Other results for local-search algorithms appear in §4 and §5. We close with an insertion-only algorithm for sparse vectors, and conclusions and open problems in §6.

2 Fully-Dynamic Vector Balancing

In this section, we prove Theorem 1.1. Given a set of vectors $a_1, a_2, \ldots, a_T \in [-1, 1]^n$, the Bárány-Grinberg algorithm signs them such that the discrepancy of the signed sum is at most $2n$. However, this signing is highly sensitive to insert or delete operations. We address this issue by recursively dividing the input sequence such that we lose only $O(n)$ discrepancy at each level of this recursion tree—we call this the distributed Bárány-Grinberg algorithm. We then show how it can easily handle insert and delete operations with low recourse.

The main idea underlying the Bárány-Grinberg algorithm [BG81] is the following linear algebraic lemma.

Lemma 2.1 (Rounding Lemma [BG81])**.** *Let $a_1, a_2, \dots, a_T \in [-1,1]^n$ be the columns of matrix $A \in [-1,1]^{n\times T}$. For any initial fractional signing $x \in [-1,1]^T$, there exists a (near-integral) signing y with all but n variables being ± 1 such that $Ay = Ax$.*

The signing y is obtained by moving to a basic feasible solution (BFS) of the following set of linear constraints $\{Ay = Ax,\ y \in [-1,1]^T\}$, where x is treated as being fixed. Based on Lemma 2.1, Bárány and Grinberg [BG81] gave the following offline algorithm: starting with the all-zeros vector as the fractional signing (i.e., $x = \mathbf{0}$), let y be the almost-integral vector satisfying $Ay = 0$. Now randomly rounding the fractional variables (with bias given by the y_i values) and using concentration bounds shows a discrepancy of $O(\sqrt{n \log n})$, or using sophisticated rounding schemes can give the tight $O(\sqrt{n})$ discrepancy [Spe85, Ban10, LM15].

2.1 An Equivalent, Recursive Viewpoint

A natural question is: *can we extend the above Bárány-Grinberg algorithm to the dynamic case*? Naively using the rounding lemma does not work, since the rounded solutions y and y' for matrices A and A' differing in one column could be very different. Our idea is to simulate the Bárány-Grinberg algorithm in a distributed and recursive manner. We divide the sequence $\{1, \dots, T\}$ into sub-sequences of length $2n$ each, which gives us a set of $m := T/2n$ sub-sequences (assume w.l.o.g., e.g., by padding, that $T/2n$ is a power of 2). Let $P_1, \dots, P_m$ denote these sub-sequences ordered from left to right. We build a binary tree $\mathcal{T}$ of height $\log_2 m$ on m leaves, where leaf j corresponds to the sub-sequence P_j. Similarly, for an internal node v, define P_v as the sub-sequence formed by taking the union of P_j over all leaves j below v.

The signing algorithm DBG(v), where v is a node of $\mathcal{T}$ is shown in Algorithm 1. It assigns values $y_i^v \in [-1,1]$ to the vectors a_i for $i \in P_v$ such that the following two conditions are satisfied:

(I1) $\sum_{i \in P_v} y_i^v a_i = 0$, and

(I2) all but at most n variables $y_i^v, i \in P_v$ are either $+1$ or -1.

Applying this property to the root node yields Lemma 2.1. While the end result is identical to the one-shot Bárány-Grinberg algorithm, this yields some crucial advantages in the dynamic setting. Indeed, when a vector is inserted/deleted, only a single leaf's sub-sequence changes. We will show that this leads to making changes in the signing assigned by the ancestors of just this leaf, giving a total recourse of $\widetilde{O}(n)$ per update!

For a subset I of indices, let A_I denote the submatrix of A given by the columns corresponding to I. Similarly, for a vector z indexed by P_v and a subset F of P_v, define $z|_F$ as the restriction of z to F. The algorithm DBG(v) begins by recursively assigning values to the two sub-sequences corresponding to its two children. Since these assignments, satisfy the two invariant conditions above, combining the two solutions into a new solution x (in line 4) leads to at most $2n$ fractional variables. Using Lemma 2.1, we reduce the number of fractional variables to n. Finally, we can maintain a (integral) signing σ by randomly assigning signs to the fractional variables F_r at the root r and retaining the values y^r for rest of the vectors. We now show by induction that the two invariant properties are satisfied, the proof is deferred to Appendix A.

Lemma 2.2. *The variables $y_i^v, i \in P_v$ satisfy the invariant properties (I1) and (I2) at the end of DBG(v).*

2.2 Dealing with Update Operations

Before describing the insert/delete operations, we describe a useful subroutine UPDATEVECTOR, which given an assignment y^{old} to $a_1, \dots, a_T$, updates it to a new assignment y^{new} when one of the vectors a_j in the

Algorithm 1 Distributed-Bárány-Grinberg: DBG(v)

Input: A node v of $\mathcal{T}$.

Output: (y^v, F_v): an assignment $y_i^v \in [-1,1]$ for each $i \in P_v$, and $F_v \subseteq P_v$ is the index set of "fractionally" signed vectors, i.e., indices i such that $-1 < y_i^v < 1$.

```
1: if $v$ is not a leaf then
2:     Let $v_L$ and $v_R$ be the left and the right children of $v$ respectively.
3:     $(y^{v_L}, F_{v_L}) \leftarrow$ DBG($v_L$), $(y^{v_R}, F_{v_R}) \leftarrow$ DBG($v_R$).
4:     Define $F := F_{v_L} \cup F_{v_R}$, $x_i := y_i^{v_L}$ for all $i \in P_{v_L}$, $x_i := y_i^{v_R}$ for all $i \in P_{v_R}$.
5: else
6:     Define $F := P_v$, $x_i = 0$ for all $i \in P_v$.
7: Using Lemma 2.1 find a vector $y' \in [-1,1]^{|F|}$ such that (i) $A_F \cdot y' = A_F \cdot x|_F$, and (ii) there are at most $n$
   indices (denoted by the set $F_v \subseteq F$) having $-1 < y'_i < 1$.
8: Define $y_i^v = x_i$ for $i \in P_v \setminus F$ and $y_i^v = y'_i$ for $i \in F$.
9: Return $(y^v, F_v)$.
```

sequence changes. The algorithm is very similar to Algorithm 1, but it needs to recurse on only one child of v. the one containing the index j. As a result, the vectors y^{old} and y^{new} differ in at most $O(n \log T)$ coordinates. Details are deferred to Appendix A.

Dynamic Insert and Delete. We now discuss the algorithm when an insert or delete operation happens. The algorithm works in phases: a new phase starts when the number of vectors becomes 2^ℓ for some ℓ, and ends when this quantity reaches $2^{\ell-1}$ or $2^{\ell+1}$. Whenever a new phase starts, we run DBG algorithm to find an assignment y. During a phase, we always maintain exactly $2^{\ell+1}$ vectors – this can be ensured at the beginning of this phase by padding with 2^ℓ zero vectors. This ensures that the tree $\mathcal{T}$ does not change during a phase.

When a delete operation happens, we call DBGUpdate, where the deleted vector gets updated to the zero vector. Similarly, when an insert operation happens, we update one of the zero vectors to the inserted vector. Thus, we get the following result:

Lemma 2.3. *The amortized recourse of this fully-dynamic algorithm is $O(n \log N)$ per update operation, where N is the maximum number of active vectors at any point in time.*

Proof. The work done at the beginning of a phase can be charged to the length of the input sequence at this time. This results in $O(1)$ amortized recourse. We show in Corollary A.2 that the amortized recourse after each update during a phase is $O(n \log N)$. This proves the overall amortized recourse bound. □

Finally, since there are only n fractional variables at the root, we can use any state-of-the-art offline discrepancy minimization algorithm to sign these vectors, e.g., to get $O(\sqrt{n})$ discrepancy for vectors with unit ℓ_∞-norm [Spe85, Ban10, LM15], or to get $O(\sqrt{\log n})$ discrepancy for vectors with unit ℓ_2-norm [BDG16, BDGL19]. This proves Theorem 1.1.

3 Fully-Dynamic Edge Orientation

We next consider the case of dynamically orienting edges in a graph to maintain bounded discrepancy. In this problem, at each time/update an adaptive adversary either inserts a new edge $e_t = (u,v)$ or removes an existing edge e from a graph $G(t)$. Assigning an *orientation* to each edge (u,v) as $u \longrightarrow v$ or $v \longrightarrow u$, the discrepancy of a vertex v is $\mathrm{disc}(v) = \big|\, |\delta_{\text{in}}(v)| - |\delta_{\text{out}}(v)| \,\big|$, where $\delta_{\text{in}}(v)$ and $\delta_{\text{out}}(v)$ are the sets of in- and out-edges incident at v. Our goal is to minimize $\max_{v \in V} \mathrm{disc}(v)$. The algorithm is allowed to re-orient any edge e, and the *amortized recourse* is the average number of re-orientations per edge insertion/deletion. We now present the first fully-dynamic algorithms with polylog(n) discrepancy and recourse.

Useful Notation For an undirected graph G and any set $S \subseteq V$, define $E(S)$ as the set of edges whose endpoints are both in S; for sets S, T, define $E(S,T) = \{e \in E \mid |e \cap S| = |e \cap T| = 1\}$. Define the *volume* of a set S to be $\mathrm{vol}(S) = \sum_{v \in S} \deg(v)$.

Definition 3.1 (ϕ-Expander)**.** A graph G is a *ϕ-expander* if for all subsets $S \subseteq V$,

$$|E(S, V-S)| \geq \phi \cdot \min(\mathrm{vol}(S), \mathrm{vol}(V-S)) \ .$$

In this case, we also say the graph G has *conductance* at least ϕ.

Definition 3.2 (γ-Weak-Regularity)**.** For $\gamma \in [0,1]$, an undirected graph G is *γ-weakly-regular* if the minimum degree of any vertex is at least γ times the average degree $2m/n$.

3.1 High Level Overview

We now provide a detailed overview of our algorithm, and then delve into the individual components. A natural algorithm for the edge orientation problem is a *local search* procedure: while there exists an edge (u,v) currently oriented $u \to v$ such that $\mathrm{disc}(v) > \mathrm{disc}(u) + 2$, flip its orientation to $v \to u$. Although locally optimal orientations could have discrepancy $\Omega(n^{1/3})$ for general graphs (see an example in Section 4.3), our first crucial result is that they always have low discrepancy on *expanders*.

Theorem 3.3. *Let $G(V,E)$ be a γ-weakly-regular ϕ-expander. Then the discrepancy of any solution produced by* Local-Search *is $O\big(\frac{\log m}{\phi\gamma}\big)$.*

The proof of this theorem appears in Section 3.2. In order to apply our local search algorithm to arbitrary graphs, our plan is to use the powerful idea of expander decompositions (see, e.g., [ST04, SW19, BvdBG+20]). At a high level, such schemes decompose any graph G into a disjoint union of expanders with each vertex appearing in a small number of them. For concreteness, we use the following result from [GKKS20, Theorem 19][2].

Theorem 3.4 (Decomposition into Weakly-Regular Expanders)**.** *Any graph $G = (V,E)$ can be decomposed into an edge-disjoint union of smaller graphs $G_1, G_2, \ldots, G_k$ such that: (a) each vertex appears in at most $O(\log^2 n)$ many smaller graphs, and (b) each of the smaller subgraphs G_i is a $\phi/4$-weakly-regular ϕ-expander, where $\phi = \Theta(1/\log n)$.*

In order to make this into a dynamic decomposition, our algorithm follows a natural idea of maintaining $\log m$ *levels/scales*, and placing each edge of the current graph G at one of these levels. We use G_i to denote the subgraph formed by the level-i edges; crucially, we ensure that G_i has at most 2^i edges. For each level i, we maintain the expander decomposition of G_i into $\bigcup_j G_{i,j}$ where $G_{i,j}$ represents the j^{th} expander in this decomposition. Since each vertex appears in at most $\log^2 n$ expanders at every level, overall any vertex will appear in $O(\log^3 n)$ expanders. Hence, our goal is to maintain a low-discrepancy signing for each expander, with bounded number of re-orientations as the expander changes due to updates. Next we discuss how insertions are easier to handle, but deletions require several new ideas.

Insertions. When edges are inserted into G, we insert it into G_1 (the lowest scale) and orient it arbitrarily. Whenever a level i becomes full, i.e., $|G_i| > 2^i$, we remove all edges and add them to the higher level $i+1$, and *recompute the expander decomposition using Theorem 3.4 from scratch for the graph consisting of all edges in this level.* We also recompute an optimal offline low-discrepancy discrepancy orientation for each expander[3].

[2]For ease of exposition, we use a result that runs in exponential time; using approximate low-conductance cuts gives a polynomial runtime with additional logarithmic factors.

[3]It is easy to optimally orient any graph in the offline setting: we consistently orient the edges of all cycles, to be left with a forest. We can then again orient all the maximal paths between pairs of leaves in a consistent manner, to end up with an orientation where every vertex has discrepancy in $\{-1, 0, 1\}$.

Of course, we may need to cascade to higher levels if the next level also overflows. However, the total cost of all these edge reorientations can be easily charged to the recent arrivals that caused the overflow.

Deletions. Our insertion procedure guarantees that an expander $G_{i,j}$ only observes deletions in its lifetime (before the expander decomposition at its level is recomputed). So when the adversary deletes an edge from G (called a *primary deletion*), we can remove it from the corresponding expander $G_{i,j}$ it belongs to, and simply re-run local search from the current orientation if it continues to have expansion at least, say $\phi/6$. We can then bound the recourse by tracking the changes to the associated ℓ_2 potential Φ for this graph. However, *what do we do when $G_{i,j}$ ceases to be an expander?* Our idea is to simply identify a cut of sparsity $< \phi/6$ and remove the smaller side ΔP from the graph $G_{i,j}$, and repeat if necessary. This is called the Prune procedure and we formally describe it in Section 3.3. The edges which are incident to ΔP are re-inserted into the system using the insertion algorithm. In Theorem 3.5, we bound the number of pruned edges (also called *secondary deletions*) in terms of the number of actual adversarial edge deletions which caused the drop in expansion, and so we are able to amortize the recourse of re-inserting these pruned edges back into our algorithm.

Theorem 3.5. *Let $G_0 = (V_0, E_0)$ be a ϕ-expander with m edges, n vertices, and minimum degree δ. For a subset $S \subseteq V_0$, let $\mathrm{vol}_0(S)$ denote its initial volume in G_0. There is an algorithm called Prune (described in Section 3.3), which for every adversarial deletion of any edge in G_0, outputs a (possibly empty) set of vertices ΔP to be pruned/removed which satisfies the following properties.*

Let P_t denote the aggregate set of vertices pruned over a sequence of t adversarial deletions inside G_0, i.e., $V_t := V_0 \setminus P_t$ and G_t is the graph with the undeleted edges of E_0 that are induced on V_t. Then, for each $1 \le t \le \phi^2 m/20$:

(i) $P_t \subseteq P_{t+1}$.

(ii) G_t *is a $\phi/6$-"strong expander", i.e., for any subset $A \subseteq V_t$,*

$$|E_t(A, V_t \setminus A)| \ge (\phi/6) \cdot \min\left(\mathrm{vol}_0(A), \mathrm{vol}_0(V_t \setminus A)\right).$$

Hence the minimum degree of a vertex in V_t is at least $\phi\delta/6$.

(iii) $\mathrm{vol}_0(P_t) \le 6t/(5\phi)$.

Similar ideas have recently been used for dynamic graph algorithms, e.g. in [SW19, BvdBG+20], but our algorithms and analyses are more direct since we are concerned only with the amortized recourse rather than the update time. However, new challenges appear due to our discrepancy minimization setting.

A 'Potential' Problem. While the above procedure identifies the set of edges to prune, so that the residual graph remains an expander, we still need to maintain a low-discrepancy orientation on the expander as it undergoes deletions and prunings. Indeed, the above ideas essentially allow us to cleanly reduce the fully dynamic problem to the follow special case of only handling deletions on expanders: let $G = (V, E)$ be a $\phi/6$-expander, currently oriented according to local search. Then, suppose e is an adversarial deletion, and suppose ΔP is the set of vertices to be removed as computed by the Prune procedure. Then, how many flips would we need to end-up at a locally-optimal orientation on $G[V \setminus \Delta P]$, which we know has bounded discrepancy since $G[V \setminus \Delta P]$ is an expander? If we can bound this in terms of the number of edges incident to ΔP, then we would be done, since these are precisely the number of secondary deletions, which are in turn bounded in terms of adversarial deletions.

A natural attempt is to simply re-run local search on $G[V \setminus \Delta P]$ starting from the current orientation. While this will converge to a low-discrepancy solution because $G[V \setminus \Delta P]$ is an expander, our recourse analysis proceeds by tracking a quadratic potential function, and this could increase a lot if we suddenly remove all edges incident to ΔP en masse. Removing the edges one by one is also also an issue as the intermediate graphs won't satisfy the desired expansion to argue both discrepancy as well as recourse (which indirectly depends on having good discrepancy bounds to control the potential). To resolve this issue, we craft a collection of "fake" intermediate graphs that interpolate between the graphs G and $G[V \setminus \Delta P]$ which ensure that (i) all of them have good expansion properties, and (ii) the potential change in moving from one to another is bounded. Our overall algorithm is to then repeatedly re-run local search after moving to each intermediate graph, until we end up with the final orientation on $G[V \setminus \Delta P]$.

We now formalize this in the following theorem, which bounds the recourse needed to move from a locally optimal orientation in $G[V]$ to one in $G[V \setminus \Delta P]$. Let H denote any graph with a current orientation represented by $\vec{H}$. We then define the following potential

$$\Phi(\vec{H}) := \sum_{v \in V(H)} \mathrm{disc}(v)^2 .$$

Theorem 3.6. *Let $G_{t-1} = (V_{t-1}, E_{t-1})$ be a $\phi/6$-expander as maintained by our algorithm, and suppose the adversary deletes an edge $e_t \in E_{t-1}$. Moreover, suppose an associated set of vertices $\Delta P \subseteq V_{t-1}$ are pruned by* Prune *to obtain the graph $G_t = (V_t, E_t)$ which is a $\phi/6$-expander, where $V_t = V_{t-1} \setminus \Delta P$ and E_t is the subset of $E_{t-1} \setminus \{e_t\}$ induced on V_t.*

Then, starting from a locally optimal orientation $\vec{G}_{t-1}$ we can compute a locally optimal orientation $\vec{G}_t$ by performing at most L_t flips satisfying

$$L_t \leq \left(\frac{\log m}{\phi^2 \gamma} + \frac{\mathrm{vol}_0(\Delta P) \log m}{\phi^2 \gamma} \right) + \Phi(\vec{G}_{t-1}) - \Phi(\vec{G}_t) .$$

With this our algorithm description is complete. For the discrepancy analysis, note that our algorithm at all times maintains a locally optimal orientation in each expander at each level, and every vertex appears in at most polylog(n) expanders from Theorem 3.4, giving us an overall discrepancy of polylog(n) by combining with Theorem 3.3. For the recourse analysis, any time the insertion algorithm overflows and a rebuild happens in the higher level, we can charge the recourse to the adversarial insertions as well as re-insertions of the edges removed by Prune. The latter is in turn bounded in terms of the adversarial deletions by Theorem 3.5. Finally, we bound the total recourse within an expander, as parts of it are pruned out, for which we appeal to Theorem 3.6. Since Theorem 3.5 (iii) ensures that the total volume of all the sets which are pruned can be bounded in terms of $O(1/\phi)$ times the number of adversarial deletions, we get that the total number of flips done over a sequence of adversarial deletions in any expander is at most $\frac{\log m}{\phi^3 \gamma}$ times the number of adversarial deletions plus the potential $\Phi(\vec{G}_0)$ of the initial expander, which is small since we start with an optimal orientation where each vertex has discrepancy at most 1 when the expander is formed.

3.2 Local-Search for Weakly-Regular Expanders

In this section we prove Theorem 3.3 that local search ensures low discrepancy on any weakly-regular expander. Recall that the local search flips an edge (u, v) oriented from u to v whenever $\mathrm{disc}(v) > \mathrm{disc}(u) + 2$.

Algorithm 2 Local-Search

Input: Graph $G = (V, E)$ and an initial partial orientation.
Output: Revised orientation which is a local optimum.

```
1: Arbitrarily direct any undirected edges in $G$.
2: While there exists a directed edge $(u, v)$ such that flipping it decreases $\Phi := \sum_u \mathrm{disc}(u)^2$, flip it.
```

Proof of Theorem 3.3. Let $\vec{G} = (V, \vec{E})$ be the directed graph corresponding to a local optimum. Consider the node v with largest discrepancy k; without loss of generality, assume $k \geq 0$. We perform a breadth-first-search (BFS) in $\vec{G}$ starting from v, but only following the *incoming* edges at each step. Let L_i be the vertices at level i during this BFS, i.e., L_i is the set of vertices w for which the shortest path in $\vec{G}$ to v contains i edges. Let S_i denote the set of vertices up to level i, i.e., $S_i := \bigcup_{i'=0}^{i} L_{i'}$. The fact that $\vec{G}$ is a local optimum means there are no improving flips, and hence the discrepancy of any vertex in L_i is at least $k - 2i$. In turn, this implies that there are at least $k/2$ layers, and the discrepancy of any vertex in $S_{k/4}$ is at least $k/2$. We now show that the volume of $S_{k/4}$'s complement is large.

Claim 3.7. $\mathsf{vol}(V \setminus S_{k/4}) \geq 2\gamma m/3$.

Proof. Each node in $S_{k/4}$ has discrepancy at least $k/2$, and each node in $V \setminus S_{k/4}$ has discrepancy at least $-k$. Since the total discrepancy of all the vertices in V is 0, it follows that

$$0 \;=\; \sum_{v \in V} \mathsf{disc}(v) \;\geq\; |S_{k/4}| \cdot k/2 - |V \setminus S_{k/4}| \cdot k.$$

This implies that $|V \setminus S_{k/4}| \geq n/3$. Now γ-weak-regularity implies each vertex in G has degree at least $\gamma \cdot \frac{2m}{n}$, and hence the sum of the degrees of the vertices in $V \setminus S_{k/4}$ is at least $2m\gamma/3$. □

We now show that the size of the edge set $E(S_i)$ increases geometrically.

Claim 3.8. *For any $i \leq k/4$, $|E(S_{i+1})| \geq (1 + \phi\gamma/3)|E(S_i)|$.*

Proof. Given a directed graph $\vec{G}$ and a subset X of vertices, let $\delta^-(X)$ and $\delta^+(X)$ denote the set of incoming edges into X (from $V \setminus X$), and the set of outgoing edges from X (to $V \setminus X$) respectively. Since the discrepancy of each vertex in S_i is positive,

$$0 \;\leq\; \sum_{w \in S_i} \mathsf{disc}(w) \;=\; \sum_{w \in S_i} \big(|\delta^-(w)| - |\delta^+(w)|\big) \;=\; |\delta^-(S_i)| - |\delta^+(S_i)|.$$

The expansion property now implies that

$$|\delta^-(S_i)| \;\geq\; \tfrac{1}{2}|\delta(S_i)| \;\geq\; \tfrac{\phi}{2} \cdot \min\big(\mathsf{vol}(S_i), \mathsf{vol}(V - S_i)\big). \tag{1}$$

Since all edges in $\delta^-(S_i)$ are directed from L_{i+1} to L_i,

$$\begin{aligned} |E(S_{i+1})| \;\geq\; |E(S_i)| + |\delta^-(S_i)| \;&\overset{(1)}{\geq}\; |E(S_i)| + \frac{\phi}{2} \min\big(\mathsf{vol}(S_i), \mathsf{vol}(V - S_i)\big) \\ &\geq\; |E(S_i)| + \frac{\phi}{2} \min\big(|E(S_i)|, 2\gamma m/3\big), \end{aligned}$$

where we used $\mathsf{vol}(S_i) \geq |E(S_i)|$ and Claim 3.7 for the two terms of the last inequality. Since $m \geq |E(S_i)|$ and $\gamma \leq 1$, the RHS above is at least $|E(S_i)| \cdot (1 + \phi\gamma/3)$. □

Finally the fact that $S_0 = \{v\}$ implies that $|E(S_1)| \geq |\delta^-(S_0)| \geq k$, and using Claim 3.8 we get

$$|E(S_{k/4})| \geq k \, \left(1 + {}^{\phi\gamma}\!/_3\right)^{k/4-1}.$$

Since $|E(S_{k/4})| \leq m$, we get $k = O\big(\frac{\log m}{\phi\gamma}\big)$. □

The weak-regularity property was used only in Claim 3.7 above; it is easy to alter the proof to show that $\mathsf{vol}(V \setminus S_{k/4}) \geq {}^{\gamma m}\!/_4$ even if all but ${}^{n}\!/_7$ vertices in G satisfy the weak-regularity property. This proves Theorem 3.3.

Corollary 3.9. *Let G be a ϕ-expander, such that degree of every vertex, except perhaps a subset of at most ${}^{n}\!/_7$ vertices, is at least $2\gamma m/n$. Then the discrepancy of* Local-Search *is $O\big(\frac{\log m}{\phi\gamma}\big)$.*

The expansion plays a crucial role here: in Section 4 we show that locally-optimum solutions of Local-Search can have large discrepancy for general graphs.

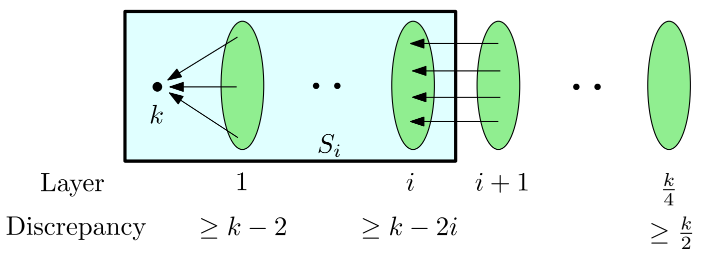

Figure 1: local-search for expanders.

3.3 Dynamic Expander Pruning

In this section we prove Theorem 3.5. We first recall the setup. Suppose we start with a ϕ-expander $G_0 = (V_0, E_0)$, and at each step t some edge e_t is deleted by the adversary (call these *primary deletions*). The goal is to remove a small portion of the graph so that the remaining portion continues to be, say, a $\phi/6$-expander. Since the graph may may violate the expansion requirement due to deletions, we perform additional *secondary deletions* at each step to maintain a slightly smaller subgraph $G_t = (V_t, E_t)$ which is $\phi/6$-expanding, such that $\{e_1, \ldots, e_t\} \cap E_t = \varnothing$. Crucially, the number of secondary deletions is only a factor $1/\phi$ more than the number of primary deletions until that point. The idea of this greedy *expander pruning algorithm* is simple: whenever the edge deletion e_t creates a sparse cut in G_{t-1}, we iteratively remove the smaller side of such a sparse cut from the current graph until we regain expansion. (See Algorithm 3 for the formal definition. Again we assume we can find low-conductance cuts; using an approximation algorithm would lose logarithmic factors.)

Algorithm 3 PRUNE(G_{t-1}, e_t)

Input: Graph $G_{t-1} = (V_{t-1}, E_{t-1})$ and an edge $e_t \in E_{t-1}$ which gets deleted at this step.
Output: A set of vertices ΔP that have to be pruned from G_{t-1} to get G_t.

```
1: Define $G_t$ with edges $E_t \leftarrow E_{t-1} \setminus e_t$ and vertices $V_t \leftarrow V_{t-1}$.
2: $\Delta P \leftarrow \varnothing$.
3: while there is a subset $A \subseteq V_t$ with $|E_t(A, V_t \setminus A)| < (\phi/6) \cdot \mathrm{vol}_0(A)$ do
4:     Assume that $A$ is the smaller side of the cut
5:     Remove $A$ from $G_t$, i.e., $V_t \leftarrow V_t \setminus A$, $E_t \leftarrow G_t[V_t \setminus A]$, and $\Delta P \leftarrow \Delta P \cup A$.
6: Return $\Delta P$
```

For a subset $S \subseteq V_0$, let $\mathrm{vol}_0(S)$ denote its initial volume in G_0. Observe that the expansion in line 3 above is measured with respect to $\mathrm{vol}_0(A)$, the volume of the set A in G_0, which is a stronger condition than comparing to the volume in the current graph. Define P_t to be the set of vertices pruned in the first t iterations, i.e., $P_t = V_0 \setminus V_t$, so that $P_0 = \varnothing$. We now show that this algorithm maintains a "strong expansion" property at all times (i.e., the expansion property holds with respect to the initial volume vol_0).

Proof of Theorem 3.5. The first property uses that P_t is the set of vertices removed in the first t steps. The second property uses the stopping condition of the algorithm, and the fact that $\mathrm{vol}_0(\{v\}) \geq \delta$ for all vertices.

To prove the final property, let the subsets pruned by the algorithm be $A^1, A^2, \ldots, A^s$ in the order they are pruned. For an index ℓ, let $B^\ell := \cup_{\ell'=1}^{\ell} A^{\ell'}$. Let $E'_t := E_0 \setminus \{e_1, \ldots, e_t\}$. The following claim bounds the number of edges leaving B^ℓ.

Claim 3.10. *If set A^ℓ is pruned in iteration t, then $|E'_t(B^\ell, V_0 - B^\ell)| \leq \frac{\phi}{6}\mathrm{vol}_0(B^\ell)$. Also, $\mathrm{vol}_0(B^\ell) \leq \mathrm{vol}_0(V_0 \setminus B^\ell)$.*

Before we prove this claim, we use it to prove the third property: note that $|E_0(B^\ell, V_0 \setminus B^\ell)| \geq \phi \cdot \mathrm{vol}_0(B^\ell)$ because G_0 is a ϕ-expander, and moreover $\mathrm{vol}_0(B^\ell) \leq \mathrm{vol}_0(V_0 \setminus B^\ell)$ by the second part of Claim 3.10. Moreover,

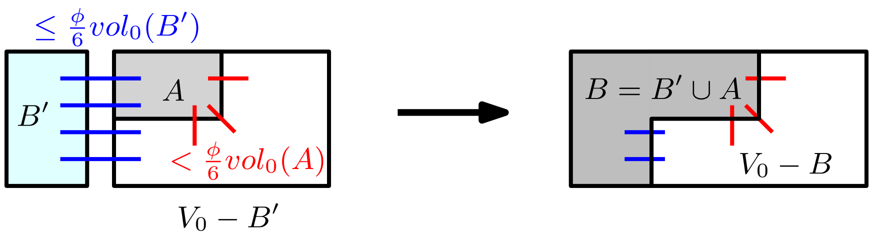

Figure 2: Dynamic expander pruning.

the first part of the claim implies that $|E'_t(B^\ell, V_0 \setminus B^\ell)| \leq \frac{\phi}{6}\mathrm{vol}_0(B^\ell)$. Since E'_t changes by at most one edge per deletion, it follows that

$$t \;\geq\; |E_0(B^\ell, V_0 \setminus B^\ell)| - |E'_t(B^\ell, V_0 \setminus B^\ell)| \geq \phi \cdot \mathrm{vol}_0(B^\ell) - \frac{\phi}{6} \cdot \mathrm{vol}_0(B^\ell) \;=\; \frac{5\phi}{6}\mathrm{vol}_0(B^\ell). \tag{2}$$

Since P_t is the same as B^ℓ for some ℓ, this completes the proof. □

Proof of Claim 3.10. We proceed by induction on ℓ. When $\ell = 0$ this is trivial since $B^\ell = \varnothing$. Suppose the claim holds for $B^{\ell-1}$, and we need to prove it for $B^\ell = B^{\ell-1} \cup A^\ell$. For sake of brevity, we denote $B' = B^{\ell-1}, B = B^\ell$ and $A = A^\ell$ (see Figure 2). Any edge in $E'_t(B, V_0 \setminus B)$ either lies in $E'_t(B', V_0 \setminus B) \subseteq E'_t(B', V_0 \setminus B')$ or in $E'_t(A, V_0 \setminus B)$. By the induction hypothesis, the former is at most $\frac{\phi}{6}\mathrm{vol}_0(B')$. The latter, by the condition in line 3, is at most $(\phi/6) \cdot \mathrm{vol}_0(A)$. Summing the two, we get $|E'_t(B, V_0 \setminus B)| \leq (\phi/6)\mathrm{vol}_0(B') + (\phi/6)\mathrm{vol}_0(A)$, which proves the first part of the claim.

To prove second part of the claim, it suffices to show $\mathrm{vol}_0(B) \leq m$. By the induction hypothesis, $\mathrm{vol}_0(B') \leq m$. Further, by (2) we have $\mathrm{vol}_0(B') \leq \frac{6t}{5\phi}$. Now two cases arise:

1. $\mathrm{vol}_0(B') \geq \frac{\phi}{6}\mathrm{vol}_0(A)$: In this case,

$$t \overset{(2)}{\geq} \frac{5\phi}{6}\mathrm{vol}_0(B') \;\geq\; \frac{5\phi}{12}\mathrm{vol}_0(B') + \frac{5\phi^2}{72}\mathrm{vol}_0(A) \;\geq\; \frac{5\phi^2}{72}\mathrm{vol}_0(B' + A) \;=\; \frac{5\phi^2}{72}\mathrm{vol}_0(B) \;.$$

   Since $t \leq \phi^2 m/20$, it follows that $\mathrm{vol}_0(B) \leq m$.

2. $\mathrm{vol}_0(B') < \frac{\phi}{6}\mathrm{vol}_0(A)$: Consider the cut $(A, V_0 \setminus B)$. Now,

$$|E_0(A, V_0 \setminus B)| \;=\; |E_0(A, V_0 \setminus A)| - |E_0(A, B')| \;\geq\; \phi \cdot \mathrm{vol}_0(A) - \mathrm{vol}_0(B') \;\geq\; 5/6 \cdot \mathrm{vol}_0(A).$$

   On the other hand, A is pruned by our algorithm because it is a sparse cut, i.e., $|E'_t(A, V_0 \setminus B)| < \frac{\phi}{6} \cdot \mathrm{vol}_0(A)$. Therefore, the total number of deletions t is at least $\frac{5\phi}{6}\mathrm{vol}_0(A) - \frac{\phi}{6}\mathrm{vol}_0(A) = \frac{4\phi}{6}\mathrm{vol}_0(A)$. Now we argue as in the first case. We get

$$t \;>\; \frac{3\phi}{6}\mathrm{vol}_0(A) + \frac{\phi}{6}\mathrm{vol}_0(A) \;>\; \frac{3\phi}{6}\mathrm{vol}_0(A) + \mathrm{vol}_0(B') \;\geq\; \frac{3\phi}{6}\mathrm{vol}_0(B' + A) \;=\; \frac{\phi}{2}\mathrm{vol}_0(B) \;.$$

   Since $t \leq \phi^2 m/20$, we again get $\mathrm{vol}_0(B) \leq m$. □

3.4 Dynamic Local-Search on Expanders with Deletions

In this section we show how to dynamically maintain a locally-optimal orientation of an expander, as parts of it are pruned out over time, thereby proving Theorem 3.6. The algorithm appears as Algorithm 4. We assume that the expander G_{t-1} is maintained by dynamic pruning procedure Prune and satisfies the expansion properties of Theorem 3.5. We also assume that we have a locally optimal orientation $\vec{G}_{t-1}$ inductively maintained by Algorithm 4. Then, when the adversary deletes an edge e_t and Prune computes a set ΔP of vertices to remove from G_{t-1} to obtain a graph G_t, we show how to compute a locally optimal orientation $\vec{G}_t$ with a bounded number of flips.

Recall that we do this via a potential function argument. For any graph H and current orientation $\vec{H}$, the potential of this orientation is $\Phi(\vec{H}) := \sum_{v \in V(H)} \mathsf{disc}(v)^2$. Indeed, the main issue is that there could be some vertices in $V_{t-1} \setminus \Delta P$ which are incident to many edges from ΔP. Hence, if we remove ΔP in one shot, the potential of the residual graph could increase a lot. To resolve this, we replace ΔP by a set F of an equal number $|\Delta P|$ of *fake vertices*, and replace all the edges between $V_t := V_{t-1} \setminus \Delta P$ and ΔP with edges between V_t and F in a balanced round-robin manner to preserve the discrepancy of every vertex of V_t w.r.t. its discrepancy in $\vec{G}_{t-1}$. Due to this balanced way of distributing the edges, we can show that the potential of the fake graph over $V_t \cup F$ is no more than that of $\vec{G}_{t-1}$, and moreover, even after deleting a subset $F' \subseteq F$ of fake vertices, $G_t \cup (F \setminus F')$ is an expander. These properties motivate running the following algorithm: transition from $G_t \cup F$ to G_t by removing the fake vertices (and its incident edges) one-by-one, and re-running local search *after each deletion.*

Algorithm 4 PRUNE-AND-REORIENT$(\vec{G}_{t-1}, e_t, \Delta P)$

Input: Graph $G_{t-1} = (V_{t-1}, E_{t-1})$ with orientation $\vec{G}_{t-1}$, deleted edge e_t, and pruned set $\Delta P \subseteq V_{t-1}$.
Output: A low-discrepancy orientation $\vec{G}_t$ for $G_t = (V_t, E_t)$ where $V_t := V_{t-1} \setminus \Delta P$ and E_t is the subset of $E_{t-1} \setminus \{e_t\}$ induced on V_t.

```
1: Create fake vertices $F := \{f_1, \dots, f_N\}$, where $N = |\Delta P|$, and define $H_t := (V_t \cup F, E_t)$.
2: Let $E^+$ be edges in $E_{t-1}(V_t, \Delta P) \setminus \{e_t\}$ oriented from $V_t$ to $\Delta P$. Denote $E^+ = \{e_1, \dots, e_r\}$, such that all edges
   incident to a vertex in $V_t$ appear consecutively.
3: for $i = 1, \dots, r$ do
4:     For edge $e_i = (v_i, p_i) \in E^+$, add edge $(v_i, f_{(i \bmod N)+1})$ into $H_t$ oriented from $v_i$ to $f_{(i \bmod N)+1}$.
5: Repeat above loop for edges $E^-$ in $E_{t-1}(V_t, \Delta P) \setminus \{e_t\}$ oriented into $V_t$; adds more edges to $H_t$.
6: Run LOCAL-SEARCH on $H_t$.
7: for each $1 \le j \le N$ do
8:     Remove vertex $f_j$ and incident edges from $H_t$.
9:     Run LOCAL-SEARCH on (the current graph) $H_t$.
10: Define the final orientation $\vec{G}_t$ to be the final orientation $\vec{H}_t$ of $H_t$ (there are no fake vertices).
```

Proof of Theorem 3.6. Let H_t^j denote the graph H_t after the removal of the fake vertices $f_1, \dots, f_j$, so that $H_t^0 = H_t$. Since t is fixed, we suppress the subscript t for the rest of this discussion.

We begin with some useful claims towards bounding the total number of flips L_t. Firstly, we show that each of the intermediate graphs is a reasonable expander. The idea is that the pruned set ΔP is small compared to V_t, because whenever it becomes sufficiently large, the dynamic expander decomposition algorithm rebuilds the expander and we can charge the recourse to the adversarial deletions. As a result, both the number of fake vertices as well as their volume is substantially smaller than that of the "real" vertices V_t, and so the expansion properties of V_t are approximately retained in each of the intermediate graphs H^j.

Lemma 3.11. *For each $j \in \{0, \dots, N\}$, the graph H^j is a $\phi/36$-expander.*

Proof. Let $F^j := \{f_{j+1}, \dots, f_N\}$ be a suffix of the fake vertices. The vertex set of H^j is $W^j := V_t \cup F^j$. Let E^j denote the set of edges in H^j—these are the union of edges between V_t, i.e., $E_{t-1}(V_t)$, and those going between F^j and V_t. Let S be a subset of vertices in H^j; we need to show that

$$|E^j(S, W^j \setminus S)| \ \geq\ \frac{\phi}{36} \min(\mathrm{vol}^j(S), \mathrm{vol}^j(W^j \setminus S)), \tag{3}$$

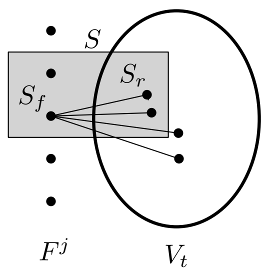

Figure 3: Expansion property of intermediate graphs W^j.

where vol^j denotes the volume with respect to E^j. Let vol_t denote the volume with respect to the edges $E_t = E_{t-1}(V_t)$. Recall that vol_0 denotes the volume with respect to G_0 (i.e., the expander graph before any deletions were performed).

Without loss of generality, assume that $\mathrm{vol}^j(S) \le \mathrm{vol}^j(W^j \setminus S)$. Let S_r and S_f ("real" and "fake") denote $S \cap A_t$ and $S \cap F^j$ respectively (see Figure 3).

Claim 3.12. $\mathrm{vol}_t(S_r) \le \frac{2}{2-\phi} \cdot \mathrm{vol}_t(V_t \setminus S_r)$.

Proof. We know that $V_0 = P_t \cup V_t$. Since $t \le \phi^2 m/20$, Theorem 3.5(iii) shows that $\mathrm{vol}_0(P_t) \le \frac{6}{5\phi} \cdot \frac{\phi^2 m}{20} \le \phi m/4$. Therefore, $\mathrm{vol}^j(V_t)$ is at least $2\left(m - \frac{\phi m}{4}\right) = 2m - \frac{\phi m}{2}$. Now,

$$\mathrm{vol}^j(S_r) \;\le\; \mathrm{vol}^j(S) \;\le\; \mathrm{vol}^j(W^j \setminus S) \;\le\; \mathrm{vol}^j(V_t \setminus S_r) + \mathrm{vol}_0(P_t).$$

Using $\mathrm{vol}_0(P_t) \le \phi m/2$, this gives $\mathrm{vol}^j(S_r) - \mathrm{vol}^j(V_t \setminus S_r) \le \phi m/2$. We also know that $\mathrm{vol}^j(S_r) + \mathrm{vol}^j(V_t \setminus S_r) = \mathrm{vol}^j(V_t) \ge 2m - \frac{\phi m}{2}$. Eliminating m from these two inequalities gives the claim. □

Theorem 3.5 shows that

$$|E^j(S_r, V_t \setminus S_r)| \;\ge\; \frac{\phi}{6}\min(\mathrm{vol}_0(S_r), \mathrm{vol}_0(V_t \setminus S_r)) \;\ge\; \frac{\phi}{6} \cdot \frac{2-\phi}{2} \cdot \mathrm{vol}^j(S_r) \;\ge\; \frac{\phi}{12}\mathrm{vol}^j(S_r),$$

where the second-last inequality follows from Claim 3.12, and the last inequality uses $\phi \le 1$.

We now consider two cases:

1. $|E^j(S_f, V_t \setminus S_r)| \ge \mathrm{vol}^j(S_f)/2$: In this case,

$$|E^j(S, W^j \setminus S)| \;\ge\; |E^j(S_r, V_t \setminus S_r)| + |E^j(S_f, V_t \setminus S_r)| \;\ge\; \frac{\phi}{12}\mathrm{vol}^j(S_r) + \frac{1}{2}\mathrm{vol}^j(S_f) \;\ge\; \frac{\phi}{12}\mathrm{vol}^j(S).$$

2. $|E^j(S_f, V_t \setminus S_r)| \le \mathrm{vol}^j(S_f)/2$: This implies $\mathrm{vol}^j(S_r) \ge \mathrm{vol}^j(S_f)/2$. Therefore,

$$|E^j(S, W^j \setminus S)| \;\ge\; |E^j(S_r, V_t \setminus S_r)| \;\ge\; \frac{\phi}{12}\mathrm{vol}^j(S_r) \;=\; \frac{\phi}{36}(\mathrm{vol}^j(S_r) + 2\mathrm{vol}^j(S_r)) \;\ge\; \frac{\phi}{36}\mathrm{vol}^j(S).$$

Hence the proof of Lemma 3.11 follows. □

Next we show that Local-Search gives low discrepancy on the graphs H^j, *even though they may not be weakly-regular.*

Lemma 3.13. *For every $j \in \{0, \dots, N\}$, the discrepancy of H^j at a local optimum is $O\big(\frac{\log m}{\phi^2 \gamma}\big)$.*

Proof. We apply Corollary 3.9 to H^j. Lemma 3.11 implies that H^j is $\phi/36$-expander, so it suffices to show that a large fraction of the vertices of H^j have large degree.

First of all, since we allow at most $D = \phi^2 m/20$ deletions, $\mathrm{vol}_0(P_t) \le \frac{6}{5\phi} \cdot \frac{\phi^2 m}{20} = \frac{3\phi m}{50}$ by Theorem 3.5(iii). Using γ-weak-regularity of G_0 and $\gamma = \phi/4$ (Theorem 3.4), this implies $|P_t| \le \frac{n}{2m\gamma} \cdot \frac{3\phi m}{50} = \frac{3n}{25}$. So H^j has at least $22n/25$ vertices. It follows that $|P_t| \le 3n'/22 \le n'/7$, where n' denotes the number of vertices in H^j. Now, Theorem 3.5(ii) implies that the degree of any vertex belonging to set V_t in the graph G_t is at least $\phi\delta/6$, where $\delta \ge 2\gamma m/n$. (This uses that G_0 is γ-weakly-regular). Thus, the degree of any vertex in V_t in the intermediate graph H^j is also at least

$$\frac{\phi\delta}{6} = \frac{2\gamma\phi m}{6n} \ge \frac{22\gamma\phi m'}{75n'} = \left(\frac{11\gamma\phi}{75}\right) \cdot \frac{2m'}{n'},$$

where m', n' denote the number of edges and vertices in H^j respectively (since $m' \le m$ and $n' \ge 22n/25$ as shown above). The desired result now follows from Corollary 3.9. □

We are now ready to conduct the potential-based analysis for bounding the number of flips. We bound the recourse by studying the ℓ_2-potential $\Phi(\vec{H}^j) := \sum_{w\in V(H^j)} \mathrm{disc}(w)^2$ as we transition from G_{t-1} to G_t. Indeed, note that a flip made by local search decreases the potential by at least 1, so the recourse is at most the *total increase* in the potential. This increase happens during Algorithm 4 when we replace ΔP by F to get the graph H_t (with its resulting orientation), and when we remove the fake vertex f_j from H_t^{j-1} to get H_t^j (in line 8). We bound the potential increase during each of these steps.

We give some notation first. Let $\vec{G}_{t-1}, \vec{G}_t, \vec{H}_t^j$ be the subgraphs G_{t-1}, G_t, H_t^j oriented after running Local-Search respectively. Let $\vec{G}'_{t-1}$ denote $\vec{G}_{t-1} \setminus \{e_t\}$. Recall that E^+, E^- (and therefore H_t) are defined using edges of $\vec{G}'_{t-1}$. Let $\widetilde{H}_t$ be the orientation of H_t just after we replace ΔP by F in $\vec{G}'_{t-1}$ (i.e., before line 6). Similarly let $\widetilde{H}^j$ (we again suppress the subscript t for ease of notation) be the orientation of H^j just after we remove f_j but before we run Local-Search on it (in line 9). Since edges are added in a round-robin manner between V_t and ΔP in H_t, there are no parallel edges.

Claim 3.14. *$\Phi(\vec{G}'_{t-1}) - \Phi(\vec{G}_{t-1})$ is at most $O(\frac{\log m}{\phi^2\gamma})$.*

Proof. Recall that $\vec{G}'_{t-1}$ is obtained by removing e_t from $\vec{G}_{t-1}$. Let d be the maximum discrepancy of a vertex in $\vec{G}_{t-1}$. Theorem 3.5 implies that G_{t-1} is an $\Omega(\phi\gamma)$-weakly-regular $\Omega(\phi)$-expander, so Theorem 3.3 implies that the discrepancy d is $O(\frac{\log m}{\phi^2\gamma})$. Hence the removal of e_t from $\vec{G}_{t-1}$ can increase the potential by at most $2((d+1)^2 - d^2) = 2(2d+1) = O(\frac{\log m}{\phi^2\gamma})$, thus proving the claim. □

Next, we show that the potential cannot increase while going from $\vec{G}'_{t-1}$ to $\widetilde{H}_t$. This uses the fact that we essentially re-distributed all the edges in E^+ and E^- in a balanced round-robin manner.

Claim 3.15. $\Phi(\widetilde{H}_t) - \Phi(\vec{G}'_{t-1}) \le 0$.

Proof. For a given sum s and variables satisfying $\sum_{i=1}^N x_i = s$, the optimal (w.r.t. ℓ_2 norm) integer assignment of variables has $x_i \in \{\lfloor\frac{a_+-a_-}{N}\rfloor, \lfloor\frac{a_+-a_-}{N}\rfloor + 1\}$ for all i and is unique up to permutations. For our problem, $s = a_+ - a_-$ and the x_i's denote the discrepancies of the fake vertices. So it suffices to prove the following:

Claim 3.16. *For each addition of an edge $(v, f) \in (E^+ \cup E^-)$ in Algorithm 4, $\exists\, d'$ such that just after the addition, $\{\mathrm{disc}(f') \mid f' \in F\} \subseteq \{d', d'+1\}$. In particular, after the addition of all edges, $\{\mathrm{disc}(f') \mid f' \in F\} \subseteq \{\lfloor\frac{a_+-a_-}{N}\rfloor, \lfloor\frac{a_+-a_-}{N}\rfloor + 1\}$.*

Proof. Recall that we first add the edges in E^+. Since they are added in round robin fashion, the claim is trivially true up to this point. At this point, there will be some prefix of vertices $F' \subseteq F$ with discrepancy $d'+1$ and the rest have discrepancy d'. Now consider the addition of edges in E^-. If $|E^-| \le |F'|$, then nodes in $F \setminus F'$ remain unchanged and the discrepancy of some nodes in F' will become d', thus still satisfying the desired property. If $|E^-| > |F'|$, then after $|F'|$ insertions, all nodes will have discrepancy d', and after this point, discrepancies decrease by 1 in a round-robin fashion, thus maintaining the desired property. In particular, after the insertion of all edges, we have $d' = \lfloor \frac{a_+ - a_-}{N} \rfloor$. This is because if $\frac{a_+ - a_-}{N}$ is integral, then all vertices in F will have discrepancy $\frac{a_+ - a_-}{N} = \lfloor \frac{a_+ - a_-}{N} \rfloor$ and if $\frac{a_+ - a_-}{N}$ is non-integral, since it is the average discrepancy, it is a convex combination of d' and $d' + 1$, implying $d' = \lfloor \frac{a^- - a^+}{N} \rfloor$. $\square$

As explained in the beginning of the proof, the claim immediately implies that $\Phi(\widetilde{H}_t) \leq \Phi(\vec{G}'_{t-1})$. $\square$

Claim 3.17. *For any $j \in \{1, \ldots, N\}$, if $\delta(f_j)$ is the degree of f_j in H_t, the potential change is*

$$\Phi(\widetilde{H}^j) - \Phi(\vec{H}^{j-1}) \leq O\left(\frac{\delta(f_j)\ \log m}{\phi^2 \gamma}\right).$$

Proof. Let d be the maximum discrepancy of a vertex in $\vec{H}^{j-1}$. When we remove the fake vertex f_j from $\vec{H}^{j-1}$, the discrepancy of the neighbors of f_j changes by 1, and so the potential increases by at most $2d\delta(f_j)$. Lemma 3.13 shows that d is $O\left(\frac{\log m}{\phi^2 \gamma}\right)$. $\square$

Claims 3.14, 3.15 and 3.17 show that the total increase in the potential due to deletion of e_t, creation of H_t and deletion of a fake vertices is at most $\frac{\log m}{\phi^2 \gamma} + \frac{\mathrm{vol}_0(\Delta P) \log m}{\phi^2 \gamma}$. If L_t denotes the number of flips performed by Local-Search during Prune-and-Recolor(G_{t-1}, e_t), then

$$\Phi(\vec{G}_t) - \Phi(\vec{G}_{t-1}) \leq \left(\frac{\log m}{\phi^2 \gamma} + \frac{\mathrm{vol}_0(\Delta P) \log m}{\phi^2 \gamma}\right) - L_t.$$

This completes the proof of Theorem 3.6. $\square$

We end this section by using Theorem 3.6 in an aggregate sense, over a sequence of t adversarial deletions.

Theorem 3.18. *Let $G_0 = (V_0, E_0)$ be a γ-weakly-regular ϕ-expander with m edges and n vertices. Suppose at most $D = \phi^2 m/20$ edges are deleted adversarially. Then for any $t \leq D$, the total number of edge flips performed by Algorithm 4 during the first t deletions is at most $O\left(\frac{\log m}{\phi^3 \gamma} \cdot t + m\right)$.*

Proof. The proof is to simply combine Theorems 3.5 and 3.6 over the sequence of adversarial deletions. Indeed, we can use the facts that the total volume of the pruned set is at most $\mathrm{vol}_0(P_t) \leq 6t/(5\phi)$ along with $\Phi(\vec{G}_0) \leq m$ (optimal offline orientation of G_0 has discrepancy at most $n \leq m$), and $\Phi(\vec{G}_t) \geq 0$ to complete the proof. $\square$

3.5 Putting Everything Together

We now formally describe our overall algorithms and analyses. To keep track of the internal states of the algorithms, we maintain an internal clock which is initialized at *moment* $\tau = 0$ (but eventually τ will exceed time t). At any moment τ, we maintain a decomposition of the current graph $G_{\mathrm{curr}}(\tau)$ into several subgraphs $\{G_i(\tau)\}_{i \geq 0}$, where $G_i(\tau)$ is the *level-i* subgraph of $G_{\mathrm{curr}}(\tau)$. These subgraphs maintain the following invariants:

(I1) For each moment τ and level i, the graph $G_i(\tau)$ has at most 2^i edges.

(I2) For every τ and i, subgraph $G_i(\tau)$ has a *creation moment* τ_0 which is at most τ. Graph $G_i(\tau)$ is a subgraph of $G_i(\tau_0)$, i.e., we only delete edges from this level between τ_0 and τ.

(I3) For each τ and i, we maintain a decomposition of $G_i(\tau)$ into subgraphs $G_{i,j}(\tau)$ for $j \geq 1$, such that any vertex appears in at most $\log^2 n$ of these subgraphs. Moreover, if τ_0 is the creation moment of $G_i(\tau)$, then $G_{i,j}(\tau_0)$ is γ-weakly-regular ϕ-expander for all j, and $G_{i,j}(\tau)$ is a subgraph of $G_{i,j}(\tau_0)$ for all j.

Although not mentioned explicitly in the invariants, the subgraph $G_{i,j}(\tau)$ also has expansion and the weak-regularity properties given by Theorem 3.5: in the notation of this theorem, $G_{i,j}(\tau) = G_t$, where G_0 is the corresponding subgraph at the creation moment τ_0 of $G_i(\tau)$.

Edge Insertions. We first consider the (easier) case of adversarial edge insertions. The algorithm appears in Algorithm 5. We first insert the edge e into level-1. Whenever a level-j subgraph overflows (i.e., has more than 2^j edges), we empty this level and move all the edges to the subsequent level. If this process stops at level j, we build a new expander decomposition of the graph at this level using Theorem 3.4, and also recompute an optimal offline low-discrepancy discrepancy orientation for each expander. As mentioned before, it is easy to optimally orient any graph in the offline setting: we consistently orient the edges of all cycles, to be left with a forest. We can then again orient all the maximal paths between pairs of leaves in a consistent manner, to end up with an orientation where every vertex has discrepancy in $\{-1, 0, 1\}$. Note that since it is the optimal discrepancy solution, it is also a locally optimal orientation.

Algorithm 5 INSERT(e, τ)

Input: Edge e to be inserted in $G_{\text{curr}}(\tau)$.
Output: Graph $G_{\text{curr}}(\tau + 1)$ with decomposition into levels.

```
1: Find the smallest $i$ such that $\{e\} \cup G_1(\tau) \cup \cdots \cup G_i(\tau)$ has at most $2^i$ edges.
2: Set $G_{i'}(\tau + 1) = \emptyset$ for $i' = 1, \ldots, i - 1$.
3: Set $G_{i'}(\tau + 1) \leftarrow G_{i'}(\tau)$, for all $i' > i$ and $G_i(\tau + 1) \leftarrow \{e\} \cup G_1(\tau) \cup \cdots \cup G_i(\tau)$.
4: Let the expander decomposition of $G_i(\tau + 1)$ (using Theorem 3.4) return subgraphs $G_{i,j}(\tau + 1), j \geq 1$;
   define their *creation moment* to be $\tau + 1$.
5: Find a discrepancy-at-most-1 orientation for each $G_{i,j}(\tau + 1), j \geq 1$.
6: $\tau \leftarrow \tau + 1$
```

Edge Deletions. For the case of adversarial edge deletions, when an edge e is deleted from subgraph $G_{i,j}(\tau)$, we first check if $\frac{\phi^2 m_{i,j}}{20}$ edges have been deleted from $G_{i,j}(\tau_0)$, where τ_0 is the creation moment of $G_i(\tau)$ and $m_{i,j}$ was the number of edges in $G_{i,j}(\tau_0)$. If so, we remove the subgraph $G_{i,j}(\tau)$ and re-insert these edges (these are called *internal inserts*). Otherwise, we run Algorithm 3 on $G_{i,j}(\tau)$ and edge e to get subset ΔP, and then call Algorithm 4 which removes ΔP (via *secondary deletes*) and reorients edges of $G_{i,j}(\tau)$. Finally, the edges of ΔP are re-inserted (causing more *internal inserts*). The algorithm is shown formally in Algorithm 6.

Algorithm 6 DELETE(e, τ)

Input: Edge e to be deleted from $G_{\text{curr}}(\tau)$.
Output: Graph after deletion of edge e.

```
1: Find the level $i$ and index $j$ such that $e$ belongs to $G_{i,j}(\tau)$.
2: Let $\tau_0 \leftarrow$ creation moment of $G_i(\tau)$, and $m_{i,j} \leftarrow$ number of edges in $G_{i,j}(\tau_0)$.
3: Let $T_{i,j}$ be the set of $\tau' \in [\tau_0, \tau]$ at which an adversarial edge deletion happened in $G_{i,j}(\tau')$.
4: if $|T_{i,j}| \geq \phi^2 m_{i,j}/20$ then
5:     Remove all edges in $G_{i,j}(\tau)$, re-insert all except $e$ one-by-one using Algorithm 5 while incrementing
   $\tau$.
6: else
7:     $\Delta P \leftarrow$ PRUNE$(G_{i,j}(\tau), e)$ (see Algorithm 3, where $t = |T_{i,j}|$, and $G_{t-1} = G_{i,j}(\tau)$).
8:     $G_{i,j}(\tau + 1) \leftarrow$ PRUNE-AND-REORIENT$(G_{i,j}(\tau), e, \Delta P)$ (see Algorithm 4) and $\tau \leftarrow \tau + 1$.
9:     Reinsert edges of $\Delta P$ except $e$ one-by-one while incrementing $\tau$ using Algorithm 5.
```

We are now ready to analyze the discrepancy of $G_{\text{curr}}(\tau)$ for all τ, as well as the amortized recourse. We will prove the following quantitative version of Theorem 1.2.

Theorem 3.19 (Main Theorem: Graph Orientation)**.** *Suppose we start with the empty graph on n vertices and it undergoes adversarial edge insertions and deletions. There is an algorithm that maintains discrepancy of $O(\log^7 n)$ with an amortized recourse of $O(\log^5 n)$ per update.*

Proof. Firstly, we can assume w.l.o.g. that we never have parallel edges, as we can handle repetitions in the following black-box manner. Let E denote the set of active edges and let E' denote the set of active edges in the no-repetitions black-box. For the copies of an edge e in E (call it T_e), we will maintain the following invariants: (a) if $|T_e|$ is even, then $e \notin E'$ and half of them are signed +1 and, (b) if $|T_e|$ is odd, then $e \in E'$ and the 1's and −1's in T_e differ by at most one such that overall the signs add up to $\sigma'(e)$. These invariants ensure that the discrepancy for E is equal to that for E'. To maintain these invariants:

1. If $|T_e|$ is even and e is added/deleted in E, then call insert procedure with e into E' and for each edge e' whose orientation changes in E', you have to flip exactly one copy in $T_{e'}$ to satisfy the invariant.
2. If $|T_e|$ is odd and e is added/deleted in E, then you have to re-orient at most one of edge in T_e to ensure that +1's and −1's are equal in T_e. Then call delete procedure on e from E'. Again, for each edge e' flipped in E', you have to flip exactly one copy in $T_{e'}$ to satisfy the invariant.

For the rest of the proof we assume that there are no parallel edges. We first bound the recourse of the algorithm. There are two sources of recourse:

(a) While performing a discrepancy-at-most-1 orientation in Line 5 of Insert (Algorithm 5): this could happen due to adversarial insert or *internal* inserts, i.e., due to lines 5 or 9 in Algorithm 6.

(b) During Local-Search performed inside procedure Prune-and-Reorient, called in line 8 of Algorithm 6.

We first bound the number of calls to the Insert procedure. Let T denote the total number of adversarial inserts and deletes.

Claim 3.20. *The total number of calls to Insert procedure is at most $\frac{2T}{\phi^2\gamma}$.*

Proof. Clearly, the number of adversarial inserts is at most T. First consider the internal inserts caused by line 5 of Algorithm 6. These can be charged to the $\phi^2\gamma m_{i,j}$ adversarial deletes in the set $T_{i,j}$. Therefore, the number of such calls to Insert procedure is at most $\frac{T}{\phi^2\gamma}$. Similarly, the number of inserts in line 9 of Algorithm 6 is at most $\mathrm{vol}_0(\Delta P)$, the total number of such internal inserts corresponding to a fixed expander graph which undergoes $D' \leq D$ adversarial edge deletions is at most $\frac{6D'}{5\phi}$ (by Theorem 3.5). Summing over all expanders, this quantity is at most $\frac{6T}{5\phi}$. Thus, the overall number of calls to Insert is at most $\frac{2T}{\phi^2\gamma}$. □

We now bound the recourse caused by rebuilding of levels due to insertions.

Claim 3.21. *The total re-orientations due to Line 5 of Insert is at most $\frac{4T\log m}{\phi^2\gamma}$.*

Proof. For a fixed level $i^\star$, let T' be the set of τ when we call Insert and the index i selected in line 1 in Algorithm 5 happens to be $i^\star$. For any such moment τ, the total number of edges added to $G_i(\tau+1)$ is at most $2^{i^\star}$ and at least $2^{i^\star-1}$. It follows that the number of re-orientations due to Line 5 of Insert at this moment is also at most $2^{i^\star}$. Note that we empty the levels $1, \ldots, i^\star - 1$ at this moment. Therefore, between any two consecutive moments in T', we must have inserted at least $2^{i^\star-1}$ edges (these could be either adversarial or internal inserts). Thus, the total number of re-orientations due to Line 5 of Insert for all moments in T' is at most twice the total number of calls to Insert, which is at most $\frac{2T}{\phi^2\gamma}$ (by Claim 3.20).

Since there are at most $\log m$ levels, the desired result follows. □

We will next bound the recourse due to local-search steps in the Prune-and-Reorient procedure.

Claim 3.22. *The total number of re-orientations due to Local-Search called in line 6 of Algorithm 4 is $O\left(\frac{T\log n}{\gamma\phi^3}\right)$.*

Proof. Consider any particular expander graph $G_{i,j}(\tau_0)$ created at moment τ_0. Consider the calls to the Prune-and-Reorient procedure where the deleted edge belongs to $G_{i,j}(\tau')$ for some $\tau' \geq \tau_0$. We know that $D' \leq D$, this inequality could be strict because we may remove all the level i edges at some moment because of line 2 in Algorithm 5. Theorem 3.18 shows that the total number of re-orientations due to Local-Search called in line 6 of Prune-and-Reorient procedure during these D' moments is at most (a constant factor of) $\frac{D' \log m}{\gamma\phi^3} + m_{i,j}$, where $m_{i,j}$ is the number of edges in $G_{i,j}(\tau)$. When we add the above for all expanders, the first term is at most $\frac{T \log m}{\gamma\phi^3}$. The second term is the sum over all expanders that get created during the algorithm of the number of edges in the expander. Expanders are created in line 4 of Algorithm 5, and so their total size can be bounded in the same manner as the argument used in the proof of Claim 3.21. Hence this quantity is at most $\frac{4T \log m}{\phi^2\gamma}$. □

Since we use $\phi, \gamma = \Theta(1/\log n)$, the above results show that the amortized recourse is at most $O\left(\gamma^{-1}\phi^{-3}\log n\right) = O\left(\log n \cdot \log^3 n \cdot \log n\right) = O(\log^5 n)$. We now bound the discrepancy of any vertex.

Claim 3.23. *The discrepancy of any vertex is bounded by $O(\log^7 n)$ at all times.*

Proof. From Lemma 3.13, discrepancy of a vertex in any expander is $O(\phi^{-2}\gamma^{-1}\log m)$, which is $O(\log^4 n)$ since we use $\gamma, \phi = \Theta(1/\log n)$. Since each vertex appears in at most $O(\log^3 n)$ expanders, we get that the discrepancy is bounded by $O(\log^7 n)$ at all times. □

This completes the proof of Theorem 3.19. □

4 Lower Bounds for Local Search

In this section, we will show that for general vectors and general graphs, typical ℓ_2-potential local search procedures do not guarantee low discrepancy.

4.1 The ℓ_2-Potential for General Vectors

The ℓ_2-potential for a signing $\{\varepsilon_i\}$ for vectors a_i is $\sum_j(\sum_i \varepsilon_i a_{ij})^2 = \|S\|^2$, where $S := \sum_i \varepsilon_i a_i$. Hence we are at a local optimum if for each i, the potential change $\|S - 2\varepsilon_i a_i\|_2 - \|S\|_2$ due to flipping a_i is non-negative. We will show that there exist locally optimal solutions on T vectors with discrepancy $\Omega(\sqrt{T})$.

Consider the following set of vectors in 2 dimensions: $T/2$ vectors of the form $(1, 1/\sqrt{T})$ and $T/2$ vectors of the form $(-1, 1/\sqrt{T})$ with signing $\varepsilon = \mathbf{1}$. Adding these vectors, we get $S = (0, \sqrt{T})$. Now, for any vector a (w.l.o.g. $a = (1, 1/\sqrt{T})$) in the collection,

$$\|S - 2a\|_2^2 - \|S\|_2^2 \;=\; \|(-2, \sqrt{T} - \frac{2}{\sqrt{T}})\|_2^2 - \|(0, \sqrt{T})\|_2^2 \;=\; 4 + \left(\sqrt{T} - \frac{2}{\sqrt{T}}\right)^2 - T \;=\; \frac{4}{T} \;>\; 0.$$

Hence, this is indeed a local optimum and has discrepancy $\sqrt{T}$.

4.2 The ℓ_2-Potential for $\{\pm 1\}$-Vectors

The ℓ_2-potential for a signing $\{\varepsilon_i\}$ for ± 1-vectors a_i is $\sum_j(\sum_i \varepsilon_i a_{ij})^2 = \|S\|^2$, where $S := \sum_i \varepsilon_i a_i$. Hence we are at a local optimum if for each i, the potential change $\|S - 2\varepsilon_i a_i\|_2 - \|S\|_2$ due to flipping a_i is non-negative. Since $\|a_i\|_2^2 = n$, the above condition is equivalent to showing

$$S^\intercal(\varepsilon_i a_i) \leq n \tag{4}$$

for all i. We can show an $\Omega(2^{n/2})$ locality gap in this case.

Lemma 4.1. *There is a family of instances of $\{\pm 1\}$ vectors, one for each n that is a multiple of 8, having local optima with discrepancy $\Omega(2^{n/2})$ but global optima having zero discrepancy.*

Proof. We construct a $\{\pm 1\}$ matrix M with n columns and $2 \cdot \sum_{i=1}^{n/2} r_i$ rows, where r_i is set later. We prove that giving signs $\varepsilon = 1$ to the rows of this matrix is a local optimum with large discrepancy. Let $S = \boldsymbol{\varepsilon} M$ denote the sum of the rows of M. Our construction consists of *repeating units*, where the i^{th} repeating unit (for $i = 1, \ldots, n/2$) is the following $2 \times n$ sub-matrix:

$$\left[\begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix} \text{ repeated } i-1 \text{ times}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \text{ repeated } \tfrac{n}{2} - i \text{ times}\right]_{2\times n}.$$

This unit is repeated r_i times. We will later set r_i to an even number, implying that any vector appears an even number of times. Therefore, by signing these even number of copies in an alternating fashion, we get that the global optimum has discrepancy 0. By construction, $S = (s_1, s_1, s_2, s_2, ..., s_{n/2}, s_{n/2})$ for some integers s_j. Define $\vec{s} := (s_1, s_2, \ldots, s_{n/2})$.

Claim 4.2. *Let B be the $\frac{n}{2} \times \frac{n}{2}$ lower-triangular matrix with 1s on the diagonal and -1s in the lower triangle. Then using $\vec{r} := (r_1, \ldots, r_{n/2})^\top = \frac{n}{4}(B^{-1})^\top B^{-1}\mathbf{1}$ results in M whose row-sum $S = (s_1, s_1, s_2, s_2, \ldots, s_{n/2}, s_{n/2})$ satisfies $\vec{s} := (s_1, s_2, \ldots, s_{n/2})^\top = 2B^\top \vec{r}$. Moreover, $\epsilon_i = 1$ for all i is a local optimum.*

Proof. The row sum $(s_1, s_1, s_2, s_2, \ldots, s_{n/2}, s_{n/2})$ satisfies:

$$\begin{aligned} 2r_1 - \ldots - 2r_{n/2-1} - 2r_{n/2} &= s_1 \\ &\vdots \\ 2r_{n/2-1} - 2r_{n/2} &= s_{n/2-1} \\ 2r_{n/2} &= s_{n/2} \end{aligned}$$

which implies $\vec{s} = 2B^\top \vec{r}$. Next, we check the condition (4) for local optimality: for any vector a in the i^{th} repeating unit,

$$\langle S, a\rangle = -2s_1 - 2s_2 - \ldots - 2s_{i-1} + 2s_i = 2(B\vec{s})_i = 2(B \cdot 2B^\top \vec{r})_i = n. \qquad \square$$

Putting the facts from Claim 4.2 together, the discrepancy vector $\vec{s} = 2B^\top \vec{r} = \frac{n}{2}B^{-1}\mathbf{1}$. We explicitly write down the inverse of the lower-triangular matrix as follows:

$$\begin{bmatrix} 1 & & & & \\ -1 & \ddots & & & \\ \vdots & \ddots & \ddots & & \\ \vdots & & \ddots & \ddots & \\ -1 & \cdots & \cdots & -1 & 1 \end{bmatrix}_{k\times k}^{-1} = \begin{bmatrix} 1 & & & & \\ 1 & \ddots & & & \\ 2 & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \\ 2^{k-2} & \cdots & 2 & 1 & 1 \end{bmatrix},$$

Using this, we get that $\vec{s}$ has entries of value $\Omega(2^{n/2})$, which proves Lemma 4.1. $\square$

Remark 4.3. Since our example has repetitions, we could try to use this structure to assign opposite signs to these multiple copies, and thereby get low discrepancy. However, it is easy to extend this to avoid repetitions. Since $(r_1, \ldots, r_{n/2}) = \frac{n}{4}(B^{-1})^\top B^{-1}\mathbf{1}$, the total number of rows $R = \sum_i r_i$ in our original example is even and at most 2^{4n}. Take the original matrix M, append $2^{4n} - R$ rows of $(1, -1, 1, -1, \ldots)$ and $(-1, 1, -1, 1, \ldots)$ in alternation to obtain a $2^{4n} \times n$ matrix M'. Moreover, append a $2^{4n} \times 4n$ matrix of all possible $\{\pm 1\}^{4n}$ vectors to the right of M' to get the final $2^{4n} \times 5n$ matrix M''. The row sum is now $S'' = (S, \mathbf{0}^{4n})$, and any vector in the first R rows continues to satisfy $\langle S'', a\rangle = n \le 5n$ by construction. For any vector after that, $\langle S'', a\rangle = 0 \le 5n$, because $\langle (s_1, s_1, \ldots, s_{n/2}, s_{n/2}), (1, -1, 1, -1, \ldots)\rangle = 0$. So M'' has rows in $\{\pm 1\}^{5n}$, and it is a local optimum with discrepancy $\Omega(2^{n/2})$.

4.3 The ℓ_2-Potential for General Graphs

We saw that the local search with the ℓ_2 potential was effective on expanders: however, it fails for general graphs. We now show instances with n vertices and local optima having discrepancy $\Omega(n^{1/3})$.

Lemma 4.4. *There is an infinite family of graph instances with local optima for the ℓ_2-potential having discrepancy $\Omega(n^{1/3})$.*

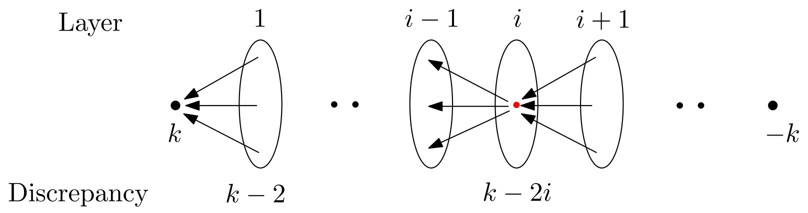

Figure 4: Lower bound of $\Omega(n^{1/3})$ discrepancy for local search on graphs.

Proof. For even integer $k = \Omega(n^{1/3})$, construct a layered digraph with k layers (see Figure 4). Denote the vertices in layer i by L_i. For every $u \in L_i$ and $v \in L_{i+1}$, add a directed edge from v to u. The number of vertices n_i in layer L_i is chosen so that the root has discrepancy k and each node in L_i has discrepancy $k - 2i$. Since a node $u \in L_i$ has incoming edges from every $v \in L_{i+1}$ and outgoing edges to every $v' \in L_{i-1}$, it suffices to have $n_{i+1} - n_{i-1} = \text{disc}(u) = k - 2i$. The base cases are $n_0 = 1$, $n_1 = k$. Since k is even, this recurrence results in a symmetric instance with a zero-discrepancy layer at the center. There are k layers, and increase in size from n_{i-1} to n_{i+1} is at most k. So the total number of nodes (up to constant factors) is at most $k + 2k + \ldots + (k)k = (1 + 2 + \ldots + k)k = O(k^3)$.

Finally, each node in L_i has discrepancy equal to $k - 2i$ by construction. Since all edges (in the directed graph) are of the form $u \longrightarrow v$ with $u \in L_j, v \in L_{j-1}$ for some j, we have that $\text{disc}(v) - \text{disc}(u) = (k-2(j-1)) - (k-2j) = 2$, and hence the orientation is indeed a local optimum. □

5 Upper-Bounds for Local Search on Unstructured Graphs and Vectors

In Section 3 we saw how local search with the ℓ_2-potential on expander graphs results in small discrepancy; and Section 4 gave strong lower bounds for general graphs, and general collections of vectors. In this section we give an upper bound for general graphs that matches the lower bounds; for vectors, we get an exponential-in-n (but quantitatively weaker) upper bound.

5.1 Local Search Upper Bound of $n^{O(n)}$ for $\{\pm 1\}$ vectors

Let us consider a locally optimal configuration. Without loss of generality, $\varepsilon = 1$. For each participating vector a, using the fact that $\|a\|_2^2 = n$, we can rewrite the condition $\|S - 2a\|_2 \geq \|S\|_2$ as $S^T a \leq n$. So for the worst case we are interested in the following program.

$$\max_{a_1,\ldots,a_T \in \{\pm 1\}^n,\, S = \sum_i a_i} \|S\|_\infty$$
$$\text{s.t. } \langle S, a_i \rangle \leq n \;\forall i \in [T].$$

We can show that for $\{\pm 1\}$ vectors, discrepancy at a local optimum is bounded by a function of n, independent of T. Without loss of generality, S has all positive coordinates. For any vector $u \geq 1$,

$$\|S\|_\infty \leq \langle S, u \rangle$$

Suppose there is such a u of the form $\sum x_i a_i$ with $x_i \geq 0$, then we will get $\|S\|_\infty \leq n \cdot (\sum x_i)$ since $\langle S, a_i \rangle \leq n$. Clearly $x_i = 1$ is a feasible solution, but instead let us optimize it as follows.

$$\begin{aligned} &\min \textstyle\sum_{i=1}^{T} x_i \\ &x_i \geq 0 && \text{for } i = 1, \ldots, T \ (T \text{ constraints}) \\ &\textstyle\sum_{i=1}^{T} x_i a_i \geq 1 && (n \text{ constraints}) \end{aligned}$$

Let x^* be a corner in the feasible region. By definition of a corner, we will have T linearly independent tight constraints at x^*. Let $n' \leq n$ of them be of the second kind and $T - n'$ of the first kind. That is, there are n' non-zero x_i^*'s. Then we will have $\sum_{i:x_i^* \neq 0} x_i^* a_i' = 1$, where a_i' is obtained from a_i by retaining only the n' coordinates corresponding to tight constraints. Arranging these n'-dimensional vectors as columns of a matrix, we get a $n' \times n'$ matrix V with $V\beta = 1$, where β is a n' dimensional vector containing the nonzero x_i^*'s. V is full rank since if the rows had a non-trivial linear combination giving zero, then using those coefficients for the n' type-2 tight constraints will give a vector that is non-zero only at positions i where $x_i^* = 0$. So it is a linear combination of the $T - n'$ type 1 tight constraints of the form $x_i = 0$. This contradicts the linear independence of the tight constraints at a vertex. Hence V is full rank, which implies $\beta = V^{-1}\mathbf{1}$.

Claim 5.1. *The entries of β are bounded by $n^{O(n)}$.*

Proof. For a $k \times k$ matrix M, $\det(A) = \sum_{\pi \in S_k} sign(\pi) m_{1,\pi(1)} m_{2,\pi(2)} \ldots m_{k,\pi(k)}$. Since there are at most k^k permutations, any $k \times k$ matrix with ± 1 entries has determinant at most k^k. We know that $V^{-1} = \frac{adj(V)}{\det(V)}$ and each entry of $adj(V)$ is the determinant of a $(n'-1) \times (n'-1)$ submatrix of V (i.e., by removing a row and column). Also since V is an invertible ± 1 matrix, $|\det(V)| \geq 1$. Hence we have that entries of V^{-1} are bounded by $(n'-1)^{n'-1} = O(n^n)$, and since $\beta = V^{-1}\mathbf{1}$, the entries of β are also bounded by $n^{O(n)}$. □

Now recall that $\|S\|_\infty \leq n \cdot (\sum_{i=1}^{T} \alpha_i) = n \cdot (\sum_{j=1}^{n'} \beta_j)$ since β is a n' dimensional vector containing the non-zero x_i's. Claim 5.1 implies that this quantity is at most $n \cdot (\sum_{j=1}^{n'} n^{O(n)}) \leq n \cdot (n' \cdot n^{O(n)}) = n^{O(n)}$.

5.2 Local Search for General Graphs: Upper Bounds

In this section, we will show upper bounds for a simple variant of Local-Search involving flips along directed paths instead of single edges. We will refer to it by Path-Local-Search with parameter L (Algorithm 7). This is simpler than the method involving expander decompositions (Theorem 3.19), but does not guarantee logarithmic bounds. (In the next section, we show our analysis is tight.)

Algorithm 7 Path-Local-Search

Input: Graph $G = (V, E)$ and an initial partial coloring, Parameter L.

Output: Revised orientation which is a local optimum.

```
1: Arbitrarily direct any undirected edges in $G$.
2: While there exists a directed path $(u_0, \ldots, u_l)$ with $l \leq L$ such that $\text{disc}(u_l) > \text{disc}(u_0) + 2$, flip all the
   edges in the directed path.
```

Theorem 5.2. *Suppose we start with the empty graph on n vertices and it undergoes adversarial edge insertions and deletions such that at any point, the graph does not have multiple edges. Then there is a deterministic algorithm that achieves $O(D)$ discrepancy with $O(\sqrt{n/D})$ amortized recourse for any $\Omega(1) \leq D \leq O(n)$.*

Proof. Recall that Path-Local-Search uses the following rule: if there is a directed path $a \to b$ of length $\leq L$ such that $\text{disc}(b) > \text{disc}(a) + 2$, then flip all the edges in the path. Let us bound the discrepancy at a local optimum. Let $\vec{G} = (V, \vec{E})$ be the directed graph corresponding to a local optimum. Consider the node v with largest discrepancy k; without loss of generality, assume $k \geq 0$. We perform BFS in $\vec{G}$ starting from

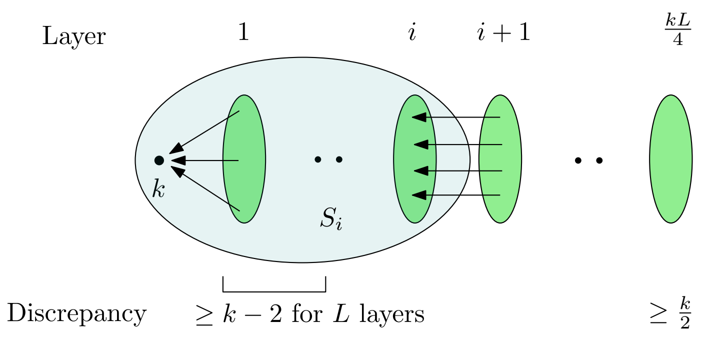

Figure 5: Discrepancy of a local optimum of PATH-LOCAL-SEARCH

v, following *incoming* edges at each step. Let L_i be the vertices at level i during this BFS, i.e., L_i is the set of vertices w for which the shortest path in $\vec{G}$ to v contains i edges. Let S_i denote the set of vertices in the first i layers, i.e., $S_i := \bigcup_{i'=0}^{i} L_{i'}$. The fact that $\vec{G}$ is a local optimum means there are no improving flips, and hence the discrepancy of any vertex in L_i is at least $k - 2\left(\lfloor \frac{i-1}{L} \rfloor + 1\right)$ (see Figure 5). In turn, this implies that there are at least $kL/2$ layers, and the discrepancy of any vertex in $S_{kL/4}$ is at least $k/2$.

We now prove the following claim about the rate at which the number of nodes grows.

Claim 5.3. *Let n_i be the number of nodes in layer i for $0 \leq i \leq \frac{kL}{4}$, then $|S_i| = \sum_{j=0}^{i} n_j \geq \frac{i^2k}{8}$.*

Proof. By induction. Base case: Trivial since $|S_0| = 1$. Induction step: Given a directed graph $\vec{G}$ and a subset X of vertices, let $\delta^-(X)$ denote the set of incoming edges into X (from $V \setminus X$). Since there are no repeated edges and all edges in $\delta^-(S_i)$ are directed from L_{i+1} to L_i, we have $|\delta^-(S_i)| \leq n_i n_{i+1}$. Since all vertices in S_i have discrepancy at least $k/2$, the number of incoming edges $|\delta^-(S_i)|$ must be at least $\frac{k}{2} \cdot |S_i|$. Combining the two inequalities, we have $n_i n_{i+1} \geq \frac{k}{2} \cdot |S_i|$. Now using this this with the definition of S_{i+1},

$$\begin{aligned}
|S_{i+1}| \;=\; |S_{i-1}| + n_i + n_{i+1} &\geq \frac{(i-1)^2k}{8} + 2\sqrt{n_i n_{i+1}} \quad \text{(induction hypothesis and AM-GM inequality)} \\
&\geq \frac{(i-1)^2k}{8} + 2\sqrt{\frac{k}{2} \cdot |S_i|} \\
&\geq \frac{(i-1)^2k}{8} + 2\sqrt{\frac{k}{2} \cdot \frac{i^2k}{8}} \quad \text{(induction hypothesis)} \\
&\geq \frac{(i-1)^2k}{8} + \frac{ik}{2} = \frac{(i+1)^2k}{8} \quad ,
\end{aligned}$$

which completes the proof of Claim 5.3. □

Now, Claim 5.3 implies $|S_{kL/4}| \geq (kL/4)^2k/8 = k^3L^2/128$. Since we also have $|S_{kL/4}| \leq n$, we get $k \leq O(n^{1/3}/L^{2/3})$. To get the amortized recourse, let us track the l_2 potential $\Phi = \sum_u \text{disc}(u)^2$, which is initially zero. Before each insertion/deletion, we are at a local optimum, which has a discrepancy of at most $n^{1/3}/L^{2/3}$. So Φ can change by at most $O(n^{1/3}/L^{2/3})$ when you insert/delete. In each step of local search, Φ decreases by at least 1. Since $\Phi \geq 0$, the total number of local search steps is at most $T \cdot \frac{n^{1/3}}{L^{2/3}}$. Since a local search step flips at most L edges, the total recourse is at most L times the number of local search steps. This implies that the amortized recourse is at most $\frac{n^{1/3}}{L^{2/3}} \cdot L = n^{1/3}L^{1/3}$. This proves Theorem 5.2 in the range $\Omega(1) \leq D \leq O(n^{1/3})$.

For the range $\Omega(n^{1/3}) \leq D \leq n$, we use the following lemma.

Lemma 5.4. *Perform local search with $L = 1$ but perform a step when $\text{disc}(v) - \text{disc}(u) > \delta$. Then a local optimum has discrepancy at most $n^{1/3}\delta^{2/3}$.*

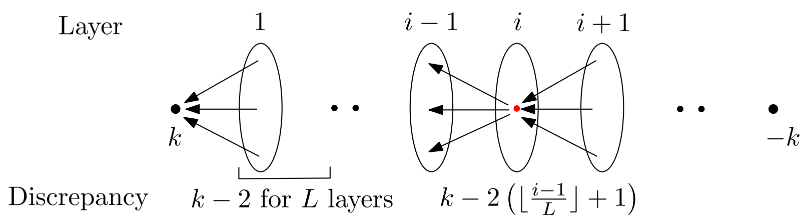

Figure 6: Example showing discrepancy bound at a local optimum of Path-Local-Search *in Section 5.2 is tight.*

Proof. Very similar to the previous analysis. There will now be $k/2\delta$ layers, which will imply that total number of vertices is at least $(k/2\delta)^2 k/8 = \Omega(k^3/\delta^2)$. Since this must be at most n, this implies that the discrepancy $k = O(n^{1/3}\delta^{2/3})$. The potential drops by at least δ in each local search step, so amortized recourse will be at most $O(n^{1/3}\delta^{2/3}/\delta) = O(n^{1/3}/\delta^{1/3})$. □

This completes the proof of Theorem 5.2. □

As an aside, setting $L = 1$ and $\delta = 2$ in Lemma 5.4 shows the local search in Section 3 achieves $O(n^{1/3})$ discrepancy when performed on general graphs instead of on expanders; this matches the lower bound in Lemma 4.4.

5.3 Tightness of discrepancy bound of Path-Local-Search

We now provide a family of instances to show that the discrepancy bound for Path-Local-Search given in Section 5.2 is tight. Note that Path-Local-Search with $L = 1$ is the same as Local-Search, and so using $L = 1$ in the lemma below reproduces the $\Omega(n^{1/3})$ lower bound of Lemma 4.4.

Lemma 5.5. *For* Path-Local-Search *with parameter L, there is a family of instances of increasing size that are each a local optima and have discrepancy k and number of nodes $O(k^3L^2)$.*

Proof. We will construct a layered directed graph with $O(kL)$ layers (see Figure 6). The instance will be symmetric and we will use even k and odd L. Let us denote layer i by L_i. For every $u \in L_i$, $v \in L_{i+1}$, there is a directed edge from v to u. We will set up the number of vertices n_i in each layer so that the root has discrepancy k and each node in L_i has discrepancy $k - 2\left(\lfloor\frac{i-1}{L}\rfloor + 1\right)$. Let us look at a node $u \in L_i$. It has incoming edges from every $v \in L_{i+1}$ and has outgoing edges to every $v' \in L_{i-1}$. So it suffices to have $n_{i+1} - n_{i-1} = \mathrm{disc}(u) = k - 2\left(\lfloor\frac{i-1}{L}\rfloor + 1\right)$. The base cases are $n_0 = 1$, $n_1 = k$. It is not hard to see that since k is even and L is odd, this recurrence will result in a symmetric instance with an odd number of zero discrepancy layers in the center. There are $O(kL)$ layers and from n_{i-1} to n_{i+1}, the increase is at most k. So the total number of nodes (up to constant factors) is at most $k + 2k + \ldots + (kL)k = (1 + 2 + \ldots + kL)k = O(k^3L^2)$. □

One can get tightness for the other side of the tradeoff (Lemma 5.4) by the same idea.

5.4 Local Search for Forests

If we are promised that at each time t the underlying graph is a forest (in the undirected sense), then we can obtain constant discrepancy with logarithmic recourse using the variant of local search that flips along directed paths.

Lemma 5.6. *For forests, performing* Path-Local-Search *with $L = \log n$ gives $O(1)$ discrepancy and $O(\log n)$ amortized recourse.*

Proof. First, we bound the discrepancy at a local optimum by 3. For this, suppose for contradiction, the discrepancy (wlog it is positive) is at least 4. Consider the node with discrepancy at least 4 (see Figure 7). It will have a at least 4 incoming edges. Due to local optimum condition with $L = \log n$, each of these children will have discrepancy at least 2, and therefore will have at least 2 children. Since $L = \log n$ and forests do

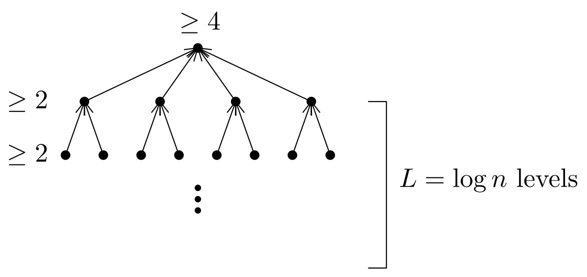

Figure 7: Discrepancy of a local optimum in the forest case

not have cycles, we can continue this binary-tree like growth argument till $\log n$ layers. Now the number of nodes will be at least $2^{\log n+1} - 1 = 2n - 1$, which is a contradiction. So we get $O(1)$ discrepancy and by the potential based argument as in previous proofs (e.g., Theorem 5.2), we get $O(\log n)$ amortized recourse. □

6 Conclusions and Insert-Only Algorithms

In this paper we initiate the study of fully dynamic discrepancy problems, where vectors/edges can both arrive or depart at each time step, and the algorithm must always maintain a low-discrepancy signing. Prior algorithms for online discrepancy could only handle arrivals, and that too only for an oblivious adversary. We obtain near-optimal discrepancy bounds for both the edge-orientation and the vector balancing cases. We achieve the former in near-optimal $\tilde{O}(1)$ amortized recourse, and the latter in $O(n)$ amortized recourse (which is exponentially better in T than the naive algorithm which recolors after each update).

The main open question left by our work is whether we can achieve near-optimal discrepancy for vector balancing in $\tilde{O}(1)$ amortized recourse per update. If true, this would imply our edge-orientation results as a special case.

Open Problem 6.1. For fully-dynamic vector balancing with vectors of ℓ_2 length at most 1 arriving or departing, can we achieve polylog(nT) discrepancy in polylog(nT) amortized recourse per update?

Currently, we don't know how to achieve near-optimal discrepancy even using $o(n)$ amortized recourse. However, in the case of insertions only (i.e., no departures), we can answer the above question affirmatively using the following Dyadic Resigning algorithm. Note that prior works, e.g. [ALS21, LSS21], could only achieve this against an oblivious adversary.

The Dyadic Resigning algorithm is simple: when the t^{th} vector arrives, let ℓ be the largest power of 2 that divides t. Use any offline algorithm $\mathcal{A}$ to construct a fresh signing of the most recently arrived 2^ℓ vectors.

Theorem 6.2. *For any sequence of T adaptive inserts, suppose that an offline algorithm $\mathcal{A}$ can produce a signing of discrepancy at most D when given any subset of these vectors. Then the Dyadic Resigning algorithm achieves at each timestep $t \leq T$ discrepancy at most $D\lceil \log_2 t \rceil$. Moreover, each vector is assigned a new sign at most $\lceil \log_2 T \rceil$ times.*

Proof. Let $t = 2^{a_1} + 2^{a_2} + \ldots + 2^{a_s}$, where $a_1 > a_2 > \ldots > a_s \geq 0$. Let $\tau_i = \sum_{j \leq i} 2^{a_j}$. To prove the discrepancy bound, the main observation is that at each timestep t, the current signing consists of the output of $\mathcal{A}$ on $\lceil \log_2 t \rceil$ different subintervals of the input sequence: $\{v_1, \ldots, v_{\tau_1}\}$, $\{v_{\tau_1+1}, \ldots, v_{\tau_2}\}$, all the way down to $\{v_{\tau_{s-1}+1}, \ldots, v_{\tau_s} = v_t\}$. (This can be proved using an inductive argument.) The discrepancy for each of these logarithmically-many is at most D, by our assumption on the algorithm $\mathcal{A}$, which proves the first claim. The second claim uses that each time a vector is given a new sign, it belongs to an subinterval of twice the length; this can happen only $O(\log T)$ times. □

Using the algorithm of [Ban98, BDG16] gives us the following result.

Corollary 6.3. *There is an algorithm for the insert-only setting that ensures a discrepancy of $O(\sqrt{\log n}\log T)$ for any sequence of vectors of ℓ_2 length at most 1 (and hence discrepancy of $O(\sqrt{s\log n}\log T)$ for any sequence of s-sparse vectors with entries in $[-1,1]$) in $O(\log T)$ amortized recourse per update.*

Another interesting future direction is to get near-optimal discrepancy for small *worst-case recourse* per update (instead of amortized recourse). E.g., in the setting of Open Problem 6.1, can we achieve $\tilde{O}(1)$ discrepancy in $\tilde{O}(1)$ worst-case recourse per update? It will also be interesting to improve Corollary 6.3 to get $O(1)$ amortized recourse per update, or to even get $O(1)$ worst-case recourse per update.

A Missing Details of Section 2

We first prove Lemma 2.2, which is restated below.

Lemma 2.2. *The variables $y_i^v, i \in P_v$ satisfy the invariant properties (I1) and (I2) at the end of DBG(v).*

Proof. The proof is by induction on the height of v: we also add to the induction hypothesis the statement that all the indices $i \in P_v$ such that $-1 < y_i^v < 1$, belong to F_v. For a leaf node, the set $F = P_j$, and so using line 7, we see that $\sum_{i\in P_j} y'_i a_i = 0$. Since $y_i^v = y'_i$ for all $i \in P_j$, the invariant (I1) follows. Invariant (I2) holds because of Lemma 2.1.

Now suppose v is an internal node and assume that the induction hypothesis holds for its children v_L and v_R. Since the assignment x just combines y^{v_L} and y^{v_R} (line 5), it follows from induction hypothesis that $\sum_{i\in P_v} x_i a_i = 0$. We ensure in line 5 that $\sum_{i\in F} y'_i a_i = \sum_{i\in F} a_i x_i$. Therefore,

$$\sum_{i\in P_v} y_i^v a_i = \sum_{i\in F} y'_i a_i + \sum_{i\in P_v\setminus F} x_i a_i = \sum_{i\in P_v} x_i a_i = 0.$$

This proves that (I1) is satisfied for y^v. For property (I2), first observe that if $i \notin F$, then $x_i \in \{-1,+1\}$ by induction hypothesis, and so $y_i^v = x_i \in \{-1,+1\}$ as well. For the indices $i \in F$, at most n of these satisfy $y'_i \in (-1,+1)$ (by Lemma 2.1) and so all the variables $y_i^v, i \notin F_v$ are either $+1$ or -1. □

We now give details of the procedure DBGUpdate in Algorithm 8. When a vector a_h changes to a_h^{new}, we only run the algorithm in Lemma 2.1 for the ancestors of the leaf j in $\mathcal{T}$ for which $h \in P_j$. We now show that this procedure has the desired properties:

Claim A.1. *Suppose the assignment y^{old} satisfies the following properties for every node v: (i) $\sum_{i\in P_v} y_i^{old} a_i = 0$, and (ii) there are at most n indices $i \in P_v$ for which $-1 < y_i^{old} < +1$. Then the assignment y^r, where r is the root node, returned by DBGUpdate$(r, y^{old}, h, a_h^{new})$ also satisfies these properties for every node v (with a_h replaced by a_h^{new}). Further y^r and y^{old} differ in at most $O(n\log T)$ coordinates.*

Proof. Let the index h belong to P_j, where j is a leaf node in $\mathcal{T}$. Let $w_0 = j, w_1, w_2, \ldots, w_H = r$, be the path from j to the root r of $\mathcal{T}$. We prove the following by induction on ℓ. The assignment (y^{w_ℓ}, F_{v_ℓ}) returned by DBGUpdate$(w_\ell, y^{old}, h, a_h^{new})$ has the following properties: (i) $\sum_{i\in P_{w_\ell}} y_i^{w_\ell} a_i = 0$, (ii) If $-1 < y_i < 1$ for some $i \in P_{w_\ell}$, then $i \in F_{w_\ell}$, (iii) $y^{old}|_{P_{w_\ell}}$ and y^{w_ℓ} differ in at most $2n(\ell+1)$ coordinates.

The base case when $\ell = 0$ follows easily because of line 8 and Lemma 2.1. Now suppose the induction hypothesis is true for $\ell - 1$. Assume wlog that $w_{\ell-1}$ is the left child of w_ℓ and w' be the right child of w_ℓ. By induction hypothesis and property of y^{old}, we see that (here x is the assignment defined during DBGUpdate for w_ℓ):

$$\sum_{i\in P_{w_\ell}} x_i a_i = \sum_{i\in P_{w_{\ell-1}}} y_i^{w_{\ell-1}} a_i + \sum_{i\in P_{w'}} y_i^{old} a_i = 0.$$

This proves property (i). Property (ii) can be shown similarly. Again, it follows from induction hypothesis and the property of y^{old} that $|F| \le 2n$, and so (i) x and y^{w_ℓ} differ in at most $2n$ coordinates, and (ii) $y^{old}|_{P_{w_\ell}}$ and x differ in at most $2n\ell$ coordinates. This implies property (iii). □

Algorithm 8 Distributed-Bárány-Grinberg Update: DBGUpdate$(v, y^{old}, h, a_h^{new})$

Input: A node v of $\mathcal{T}$, assignment y^{old} satisfying (I1) and (I2), an index $j \in P_v$ where the corresponding vector a_j changes to a_j^{new}.
Output: (y^v, F_v): an assignment $y_i^v \in [-1, 1]$ for each $i \in P_v$, and $F_v \subseteq P_v$ is the index set of "fractionally" signed vectors, i.e., indices i such that $-1 < y_i^v < 1$.

```
1: if $v$ is not a leaf then
2:     Let $v_L$ and $v_R$ be the left and the right children of $v$ respectively.
3:     Let $\ell \in \{L, R\}$ be such that $j \in P_{v_\ell}$ and $\ell'$ denote $\{L, R\} \setminus \{\ell\}$.
4:     $(y^{v_\ell}, F_{v_\ell}) \leftarrow \text{DBGUpdate}(a_\ell, y^{old}, j, a_j^{new})$, and $F_{v_{\ell'}} := \{i \in P_{v_{\ell'}} \mid -1 < y_i^{old} < 1\}$.
5:     Define $F := F_{v_\ell} \cup F_{v_{\ell'}}, x_i := y_i^{v_\ell}$ for all $i \in P_{v_\ell}, x_i := y_i^{old}$ for all $i \in P_{v_{\ell'}}$.
6: else
7:     Define $F := P_v, x_i = 0$ for all $i \in P_v$.
8: Using Lemma 2.1 find a vector $y' \in [-1, 1]^{|F|}$ such that (i) $A_F \cdot y' = A_F \cdot x|_F$, (ii) there are at most $n$ indices,
   call it $F_v \subseteq F$, such that $-1 < y_i' < 1$ (note that if $h \in F$, then column $h$ of $A_F$ is $a_h^{new}$).
9: Define $y_i^v = x_i$ for $i \in P_v \setminus F$ and $y_i^v = y_i'$ for $i \in F$.
10: Return $(y^v, F_v)$.
```

Corollary A.2. *The amortized recourse during a phase of the DBGUpdate algorithm is $O(n \log N)$.*

Proof. When a phase begins, we run Algorithm 1 to ensure that the assignment y satisfies the conditions stated in Claim A.1 (for the assignment y^{old}). Using this result, we see that after each update operation, these conditions continue to be satisfied. Therefore, Claim A.1 shows that the recourse encountered after each update operation is $O(n \log N)$. ◻

References

[AAG+19] Amir Abboud, Raghavendra Addanki, Fabrizio Grandoni, Debmalya Panigrahi, and Barna Saha. Dynamic set cover: improved algorithms and lower bounds. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 114–125, 2019. 4

[AAN+98] Miklos Ajtai, James Aspnes, Moni Naor, Yuval Rabani, Leonard J Schulman, and Orli Waarts. Fairness in scheduling. *Journal of Algorithms*, 29(2):306–357, 1998. 2, 3, 4

[AGZ99] Matthew Andrews, Michel X Goemans, and Lisa Zhang. Improved bounds for on-line load balancing. *Algorithmica*, 23(4):278–301, 1999. 4

[ALS21] Ryan Alweiss, Yang P. Liu, and Mehtaab Sawhney. Discrepancy minimization via a self-balancing walk. In *Proceedings of STOC*, 2021. 1, 3, 4, 25

[Ban98] Wojciech Banaszczyk. Balancing vectors and Gaussian measures of n-dimensional convex bodies. *Random Struct. Algorithms*, 12(4):351–360, 1998. 4, 25

[Ban10] Nikhil Bansal. Constructive Algorithms for Discrepancy Minimization. In *Proceedings of FOCS 2010*, pages 3–10, 2010. 2, 4, 5, 6

[BCH17] Sayan Bhattacharya, Deeparnab Chakrabarty, and Monika Henzinger. Deterministic fully dynamic approximate vertex cover and fractional matching in $O(1)$ amortized update time. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 86–98. Springer, 2017. 4

[BDG16] Nikhil Bansal, Daniel Dadush, and Shashwat Garg. An algorithm for komlós conjecture matching banaszczyk's bound. In *Proceedings of FOCS 2016*, pages 788–799, 2016. 4, 6, 25

[BDGL19] Nikhil Bansal, Daniel Dadush, Shashwat Garg, and Shachar Lovett. The Gram-Schmidt walk: A cure for the Banaszczyk blues. *Theory Comput.*, 15:1–27, 2019. 4, 6

[Bec81] József Beck. Balanced two-colorings of finite sets in the square I. *Combinatorica*, 1(4):327–335, 1981. 4

[BF81] József Beck and Tibor Fiala. “Integer-making” theorems. *Discrete Appl. Math.*, 3(1):1–8, 1981. 4

[BF99] Gerth Stø lting Brodal and Rolf Fagerberg. Dynamic representations of sparse graphs. In *Algorithms and data structures (Vancouver, BC, 1999)*, volume 1663 of *Lecture Notes in Comput. Sci.*, pages 342–351. Springer, Berlin, 1999. 4

[BG81] Imre Bárány and Victor S Grinberg. On some combinatorial questions in finite-dimensional spaces. *Linear Algebra and its Applications*, 41:1–9, 1981. 2, 5

[BHI18] Sayan Bhattacharya, Monika Henzinger, and Giuseppe F. Italiano. Deterministic fully dynamic data structures for vertex cover and matching. *SIAM Journal on Computing*, 47(3):859–887, 2018. 4

[BHN19] Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. A New Deterministic Algorithm for Dynamic Set Cover. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 406–423. IEEE, 2019. 4

[BHNW20] Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Xiaowei Wu. An Improved Algorithm for Dynamic Set Cover. *arXiv preprint arXiv:2002.11171*, 2020. 4

[BJM$^+$21] Nikhil Bansal, Haotian Jiang, Raghu Meka, Sahil Singla, and Makrand Sinha. Online discrepancy minimization for stochastic arrivals. In *Proceedings of SODA*, pages 2842–2861, 2021. 1, 4

[BJSS20] Nikhil Bansal, Haotian Jiang, Sahil Singla, and Makrand Sinha. Online vector balancing and geometric discrepancy. In *Proceedings of STOC*, pages 1139–1152, 2020. 1

[BK19] Sayan Bhattacharya and Janardhan Kulkarni. Deterministically Maintaining a $(2+\epsilon)$-Approximate Minimum Vertex Cover in $O(1/\epsilon^2)$ Amortized Update Time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1872–1885. SIAM, 2019. 4

[BLSZ14] Bartlomiej Bosek, Dariusz Leniowski, Piotr Sankowski, and Anna Zych. Online bipartite matching in offline time. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 384–393. IEEE, 2014. 4

[BS20] Nikhil Bansal and Joel H. Spencer. On-line balancing of random inputs. *Random Struct. Algorithms*, 57(4):879–891, 2020. 1, 4

[BvdBG$^+$20] Aaron Bernstein, Jan van den Brand, Maximilian Probst Gutenberg, Danupon Nanongkai, Thatchaphol Saranurak, Aaron Sidford, and He Sun. Fully-dynamic graph sparsifiers against an adaptive adversary. *CoRR*, abs/2004.08432, 2020. 3, 7, 8

[Bá79] I Bárány. On a class of balancing games. *Journal of Combinatorial Theory, Series A*, 26(2):115–126, 1979. 4

[CAHP$^+$19] Vincent Cohen-Addad, Niklas Oskar D Hjuler, Nikos Parotsidis, David Saulpic, and Chris Schwiegelshohn. Fully Dynamic Consistent Facility Location. In *Advances in Neural Information Processing Systems*, pages 3250–3260, 2019. 4

[CDKL09] Kamalika Chaudhuri, Constantinos Daskalakis, Robert D. Kleinberg, and Henry Lin. Online bipartite perfect matching with augmentations. In *IEEE INFOCOM 2009*, pages 1044–1052. IEEE, 2009. 4

[Cha01] Bernard Chazelle. *The discrepancy method: randomness and complexity*. Cambridge University Press, 2001. 4

[EL14] Leah Epstein and Asaf Levin. Robust algorithms for preemptive scheduling. *Algorithmica*, 69(1):26–57, 2014. 4

[FW83] Ronald Fagin and John H. Williams. A fair carpool scheduling algorithm. *IBM J. Res. Dev.*, 27(2):133–139, March 1983. 2

[GGK16] Albert Gu, Anupam Gupta, and Amit Kumar. The power of deferral: maintaining a constant-competitive Steiner tree online. *SIAM Journal on Computing*, 45(1):1–28, 2016. 4

[Gia97] Apostolos A Giannopoulos. On some vector balancing problems. *Studia Mathematica*, 122(3):225–234, 1997. 4

[GK14] Anupam Gupta and Amit Kumar. Online Steiner tree with deletions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 455–467. SIAM, 2014. 4

[GKKP17] Anupam Gupta, Ravishankar Krishnaswamy, Amit Kumar, and Debmalya Panigrahi. Online and Dynamic Algorithms for Set Cover. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 537–550, New York, NY, USA, 2017. ACM. 4

[GKKS20] Anupam Gupta, Ravishankar Krishnaswamy, Amit Kumar, and Sahil Singla. Online carpooling using expander decompositions. In *Proceedings of FSTTCS*, pages 23:1–23:14, 2020. 3, 7

[GKKV95] Edward F Grove, Ming-Yang Kao, P. Krishnan, and Jeffrey Scott Vitter. Online perfect matching and mobile computing. In *Workshop on Algorithms and Data Structures*, pages 194–205. Springer, 1995. 4

[GKLX20] Xiangyu Guo, Janardhan Kulkarni, Shi Li, and Jiayi Xian. The Power of Recourse: Better Algorithms for Facility Location in Online and Dynamic Models. *arXiv preprint arXiv:2002.10658*, 2020. 4

[GKS14] Anupam Gupta, Amit Kumar, and Cliff Stein. Maintaining assignments online: Matching, scheduling, and flows. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 468–479. SIAM, 2014. 4

[IW91] Makoto Imase and Bernard M Waxman. Dynamic Steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, 1991. 4

[KKPS14] Tsvi Kopelowitz, Robert Krauthgamer, Ely Porat, and Shay Solomon. Orienting fully dynamic graphs with worst-case time bounds. In *Automata, languages, and programming. Part II*, volume 8573 of *Lecture Notes in Comput. Sci.*, pages 532–543. Springer, Heidelberg, 2014. 4

[Kow07] Ł ukasz Kowalik. Adjacency queries in dynamic sparse graphs. *Inform. Process. Lett.*, 102(5):191–195, 2007. 4

[LM15] Shachar Lovett and Raghu Meka. Constructive Discrepancy Minimization by Walking on the Edges. *SIAM J. Comput.*, 44(5):1573–1582, 2015. 2, 4, 5, 6

[ŁOP$^+$15] Jakub Łacki, Jakub Oćwieja, Marcin Pilipczuk, Piotr Sankowski, and Anna Zych. The power of dynamic distance oracles: Efficient dynamic algorithms for the Steiner tree. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 11–20, 2015. 4

[LSS21] Yang P. Liu, Ashwin Sah, and Mehtaab Sawhney. A gaussian fixed point random walk. *CoRR*, abs/2104.07009, 2021. 25

[Mat09] Jiří Matousek. *Geometric discrepancy: An illustrated guide*, volume 18. Springer Science & Business Media, 2009. 4

[PW93] Steven Phillips and Jeffery Westbrook. Online load balancing and network flow. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 402–411, 1993. 4

[Rot14] Thomas Rothvoß. Constructive Discrepancy Minimization for Convex Sets. In *Proceedings of FOCS 2014*, pages 140–145, 2014. 4

[Spe77] Joel Spencer. Balancing games. *Journal of Combinatorial Theory, Series B*, 23(1):68–74, 1977. 1, 4

[Spe85] Joel Spencer. Six standard deviations suffice. *Trans. Am. Math. Soc.*, 289(2):679–706, 1985. 2, 4, 5, 6

[SSS09] Peter Sanders, Naveen Sivadasan, and Martin Skutella. Online scheduling with bounded migration. *Mathematics of Operations Research*, 34(2):481–498, 2009. 4

[ST04] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 81–90, 2004. 7

[SV10] Martin Skutella and José Verschae. A robust PTAS for machine covering and packing. In *European Symposium on Algorithms*, pages 36–47. Springer, 2010. 4

[SW19] Thatchaphol Saranurak and Di Wang. Expander decomposition and pruning: Faster, stronger, and simpler. In *Proceedings of SODA*, pages 2616–2635, 2019. 3, 7, 8

[Wes00] Jeffery Westbrook. Load balancing for response time. *Journal of Algorithms*, 35(1):1–16, 2000. 4