Random Walks that Find Perfect Objects and the Lovász Local Lemma

Dimitris Achlioptas
Deparment of Computer Science
University of California, Santa Cruz

Fotis Iliopoulos
Deparment of Electrical Engineering
National Technical University of Athens

***Abstract*—We give an algorithmic local lemma by establishing a sufficient condition for the uniform random walk on a directed graph to reach a sink quickly. Our work is inspired by Moser's entropic method proof of the Lovász Local Lemma (LLL) for satisfiability and completely bypasses the Probabilistic Method formulation of the LLL. In particular, our method works when the underlying state space is entirely unstructured. Similarly to Moser's argument, the key point is that algorithmic progress is measured in terms of entropy rather than energy (number of violated constraints) so that termination can be established even under the proliferation of states in which *every* step of the algorithm (random walk) increases the total number of violated constraints.**

***Index Terms*—Lovász Local Lemma, Entropic Method, Random Walks**

I. Introduction

Let Ω be a (large) set of objects and let F be a collection of subsets of Ω, each subset comprising objects sharing some negative feature. For example, in a CNF formula on n variables with clauses $c_1, c_2, \ldots, c_m$, each clause c_i corresponds to the subset of $\Omega = \{0,1\}^n$, i.e., the subcube of truth assignments, violating c_i. We will refer to each subset $f \in F$ as a *flaw* and, following linguistic rather than mathematical convention, say that f is present in σ if $f \ni \sigma$. We will say that an object $\sigma \in \Omega$ is *flawless* (perfect) if no flaw is present in σ.

Given Ω and F we can often prove the *existence* of flawless objects using the Probabilistic Method and in many interesting cases this is the only way we know how to do so. In order to employ the Probabilistic Method we introduce a probability measure μ on Ω and consider the collection of ("bad") events $\mathcal{A}$ corresponding to the flaws (one event per flaw). The existence of flawless objects is then equivalent to μ assigning strictly positive measure to the intersection of the complements of the bad events. Clearly, such positivity always holds if the events in $\mathcal{A}$ are independent and none of them has measure 1. One of the most powerful tools of the Probabilistic Method is the Lovász Local Lemma (LLL) asserting that such positivity also holds under a condition of limited dependence among the events in $\mathcal{A}$. Below we state a highly compact corollary of the LLL which is nonetheless sufficient for most applications (we discuss the LLL extensively in Section III).

Research supported by ERC Starting Grant 210743 and an Alfred P. Sloan Research Fellowship. The research of DA was partially performed at CTI and the Department of Informatics and Telecommunications, University of Athens.

Asymmetric LLL. *Let $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$ be a set of events, and let $D_i \subseteq \mathcal{A}$ denote the dependency set of event A_i, i.e., A_i is mutually independent of all events not in D_i. If for every event A_i,*

$$\sum_{A_j \in D_i} \Pr[A_j] \leq \frac{1}{4} \ , \tag{1}$$

then with positive probability none of the events in $\mathcal{A}$ occurs.

Remark 1. *If $\Pr[A_i] < \epsilon$ for all i, then $1/4$ in (1) can be replaced by $1/\mathrm{e} - \phi(\epsilon)$, where $\phi \to 0$ as $\epsilon \to 0$.*

As one can imagine, the amount of time between proving that flawless objects exist and asking whether they can be found efficiently is short. In contrast, making the LLL constructive has been a long quest, starting with the work of Beck [3], with subsequent works of Alon [2], Molloy and Reed [16], Czumaj and Scheideler [5], Srinivasan [22] and others, each such work establishing a method for finding flawless objects efficiently, but in all cases under significant additional conditions beyond those required for existence by the LLL. The breakthrough was made by Moser [17] who showed that a shockingly simple algorithm nearly matches the LLL condition for k-CNF formulas. Very shortly afterwards, Moser and Tardos [18] established a general framework within which the LLL is efficiently constructive. We discuss the framework of [18] and its relationship to our work in Sections III and IV.

Inspired by the work of Moser [17] we take a more direct approach to finding flawless objects, bypassing the probabilistic formulation of the existence question. Without introducing a measure on Ω we seek flawless objects as follows: start with an arbitrary object in Ω; keep transforming it until it (hopefully) becomes flawless by taking a random walk on Ω. With this in mind, we will refer to the elements of Ω as states. Each state transformation (step of the walk) $\sigma \to \tau$ will be taken to *address* a flaw present at σ. A step may eradicate other flaws beyond the one addressed but may also introduce new flaws (and, in fact, may fail to eradicate the addressed flaw).

Concretely, for each $\sigma \in \Omega$, let $U(\sigma) = \{f \in F : \sigma \in f\}$, i.e., $U(\sigma)$ is the set of flaws present in σ. For each $\sigma \in \Omega$ and $f \in U(\sigma)$, we require a *non-empty* $A(f, \sigma) \subseteq \Omega$ which we will refer to as the set of possible *actions* for addressing flaw f in state σ. To address flaw f in state σ we select uniformly at random an element $\tau \in A(f, \sigma)$ and walk to state τ, noting that possibly $\tau = \sigma \in A(f, \sigma)$. We note that the set

of actions, $A(f, \sigma)$, for addressing a flaw f in each state σ can depend *arbitrarily* on σ. As we discuss in Section IV, this fact moves the Local Lemma from the Probabilistic Method squarely within the purview of Algorithm Design.

We represent the set of all possible state transformations as a multi-digraph D on Ω formed as follows: for each state σ, for each flaw $f \in U(\sigma)$, for each state $\tau \in A(f, \sigma)$ place an arc $\sigma \xrightarrow{f} \tau$ in D, i.e., an arc labeled by the flaw being addressed. Thus, D may contain pairs of states σ, τ with multiple $\sigma \to \tau$ arcs, each such arc labeled by a different flaw, each such flaw f having the property that moving to τ is one of the actions for addressing f at σ, i.e., $\tau \in A(f, \sigma)$. Since we require that the set $A(f, \sigma)$ is non-empty for every flaw in $U(\sigma)$ we see that a vertex of D is a sink iff it is flawless.

We focus on digraphs satisfying the following condition.

Atomicity. *D is* atomic *if for every flaw f and state τ there is* at most *one arc incoming to τ labeled by f.*

The purpose of atomicity is to capture "locality of action". In particular, note that if D is atomic, then every walk on D can be reconstructed from its final state and the sequence of labels on the arcs traversed, as atomicity allows one to trace the walk backwards unambiguously. We want to emphasize that atomicity does not represent an algorithmic limitation beyond what is inherent in locality of action. A fruitful way to think about this point is to consider the case where Ω has product structure over a set of variables, e.g., a Constraint Satisfaction Problem. In that case atomicity is equivalent to the following:

1) Each constraint (flaw) forbids exactly *one* joint value assignment to its underlying variables.
2) Each state transition modifies *only* the variables of the violated constraint (flaw) that it addresses.

Condition 1 expresses a syntactic requirement of breaking down compound constraints to constituent parts akin to satisfiability constraints. So, for example, to encode graph q-colorability we must write q constraints (flaws) per edge, one for each color. This flattening enables a uniform treatment and we do not pay any price for it. In fact, in many cases it is strictly advantageous as it affords a more refined accounting of conflict between constraints. Condition 2, a genuine restriction, expresses the idea of "focusing" introduced by Papadimitriou for 2-SAT [19], i.e., that every state transformation is the result of attempting to eradicate some specific flaw.

Conditions 1, 2 are sufficient (and necessary) for atomicity. To see sufficiency, imagine that there exist two arcs $\sigma_1 \xrightarrow{f} \tau$ and $\sigma_2 \xrightarrow{f} \tau$, i.e., two state transformations involving only variables of f, leading to state τ. Since f must be present in both σ_1 and σ_2, Condition 1 implies that if $\sigma_1 \neq \sigma_2$, then there exists at least one variable v not bound by f which takes different values in σ_1, σ_2. In that case, though, Condition 2 implies that v will have the same value before and after each of the two transformations, leading to a contradiction.

Having defined the multi-digraph D on Ω we will now define a digraph C on the set of flaws F, reflecting some of the structure of D.

Causality Digraph. *Given a multi-digraph D on Ω with arcs labeled by elements of F, let $C' = C'(\Omega, F, D)$ be the multi-digraph on F formed by mapping each arc $\sigma \xrightarrow{f} \tau$ of D as follows:*

- *If $f \in U(\tau)$, then add a self-loop $f \to f$ in C'.*
- *For each $g \in U(\tau) \setminus U(\sigma)$ add an arc $f \to g$ in C'.*

The causality digraph $C = C(\Omega, F, D)$ *is the simple digraph that results by retaining one occurrence of each arc in C'. The* neighborhood *of a flaw f is $\Gamma(f) = \{g : f \to g$ exists in $C\}$.*

In other words, C captures "potential causality", where f causes g in $\sigma \xrightarrow{f} \tau$ if g is present in τ but was not present in σ, or if $g = f$ is (still) present in τ, i.e., the action failed to eradicate the flaw it addressed. It is important to note that C contains an arc $f \to g$ if there exists *even one* state transition aimed at addressing f that causes g to appear in the new state. In that sense, C is a "pessimistic" estimator of causality (or, alternatively, a lossy compression of D). This pessimism is both the strength and the weakness of our approach (and of the entropic method in general). On one hand, it makes it possible to extract results about algorithmic progress without tracking state evolution. On the other hand, it only gives good results when C can remain sparse even in the presence of such stringent arc inclusion. We feel that this tension is meaningful: maintaining sparsity requires that the actions for addressing each flaw across different states are *coherent* with respect to the flaws they cause.

Without loss of generality (and to avoid certain trivialities), we can assume that C is strongly connected. To see this, let $C_1, \ldots, C_k$ be the strongly connected components of C and consider the DAG with vertices $c_1, \ldots, c_k$, where for $i \neq j$, c_i points to c_j iff there exist $f \in C_i$ and $g \in C_j$ such that $f \to g$ exists in C. If c_i is a source vertex in the DAG, we can first only address flaws in C_i until we reach a state $\sigma_i \in \Omega$ in which no such flaws are present. If σ_i is flawed, we remove c_i from the DAG, select a new source vertex c_j, and repeat the same idea continuing from σ_i. The actions that will be taken to address flaws in C_j will never introduce flaws in C_i etc.

So far we have avoided discussing *which* flaw to address in each flawed state, demanding instead a non-empty set $A(f, \sigma)$ of actions for each flaw f present in a state σ. We will discuss the motivation for this in Section II. For now, suffice it to say that we consider algorithms which employ an *arbitrary* ordering π of F and in each flawed state address the greatest flaw present according to π. Formally, we define the following.

Definition 1. *If π is any ordering of F, let $I_\pi : 2^F \to F$ be the function mapping each subset of F to its greatest element according to π, with $I_\pi(\emptyset) = \emptyset$. We will sometimes abuse notation and for a state $\sigma \in \Omega$, write $I_\pi(\sigma)$ for $I_\pi(U(\sigma))$ and also write I for I_π when π is clear from context.*

$D_\pi \subseteq D$ is the result of retaining for each state σ only the outgoing arcs with label $I_\pi(\sigma)$.

The next definition reflects that since actions are selected uniformly, the *number* of actions available to address a flaw, i.e., the breadth of the "repertoire", is important.

Definition 2. *For each flaw $f \in F$, let*

$$A_f = \min_{\sigma \in f} |A(f, \sigma)| \ . \tag{2}$$

Specifically, A_f will be used to bound from below the amount of randomness consumed every time f is addressed. (The minimum in (2) is often inoperative with $|A(f, \sigma)|$ being the same for all $\sigma \in F$.)

Last is our key definition, strongly reminiscent of (1) in the asymmetric LLL.

Amenability. *D is* amenable *if for every $f \in F$,*

$$\sum_{g \in \Gamma(f)} \frac{1}{A_g} < \frac{1}{\mathrm{e}} \ . \tag{3}$$

We are now ready to state our main result.

Theorem 1 (Main result)**.** *Fix any ordering π of F and any $\sigma_1 \in \Omega$. If D is amenable, then the uniform random walk on D_π started from σ_1 reaches a sink within $O(|F| + \log |\Omega|)$ steps with high probability. Specifically, the probability that a sink is not reached within $t = (T_0 + s)/\delta$ steps is 2^{-s}, where*

$$\begin{aligned} T_0 &= \log_2 |\Omega| + 2|U(\sigma_1)| \ , \\ \delta &= 1 - \max_{f \in F} \sum_{g \in \Gamma(f)} \frac{\mathrm{e}}{A_g} \ . \end{aligned}$$

If A_f is the same for all $f \in F$, then $T_0 = \log_2 |\Omega| + |U(\sigma_1)|$.

Theorem 1 has a few features worth noting. The first is that since the initial state of the walk can be arbitrary, any foothold on Ω suffices to apply the theorem, in contrast to the requirement of being able to sample from Ω according to some distribution. While being able to sample from Ω is generally a non-issue in existing applications of the LLL, this is only true precisely because Ω has been a highly structured set and μ a very simple measure on Ω, as we discuss in Section IV.

The second feature is that the running time depends only on the number of flaws present in the initial state, rather than on $|F|$. This has an implication conceptually analogous to the work of Hauepler, Saha, and Srinivasan [11] on core events: it is possible to get an efficient algorithm even when $|F|$ is very large, e.g., super-polynomial in the problem's encoding, if we can show that $|U(\sigma_1)|$ is small, e.g., by proving that in every state only polynomially many flaws may be present. This feature provides great flexibility in the design of flaws, which we promptly explore in one of our first applications [1].

Lastly, the bound on the running-time is sharper than a typical high probability bound, being instead akin to cutoff bounds for Markov chains [6] (wherein the distance to the stationary distribution drops from near 1 to near 0 in a very small number of steps past a critical point). Here the walk first makes T_0/δ steps without any guarantee of progress but from that point on every single step has constant probability of being the last step. While, pragmatically, a high probability bound would be just as useful, the fact that our bound naturally takes this form suggests a potential deeper connection between amenability and Markov chains.

II. Flaw Selection

The astute reader will notice that the definition of amenability does not entail any element of flaw choice. As a result, Theorem 1 asserts that if D is amenable, a uniform random walk on D_π reaches a sink quickly for *every* permutation π of the flaws. As this is an unnecessarily luxurious conclusion, it is natural to try to sharpen our result by selecting the flaw order π *first*, so that the causality digraph defining amenability is the image of the (much) sparser D_π instead of D. However, since an arc $f \to g$ exists in the causality digraph C as long as there is even one transition addressing f that causes g in D, it is not at all clear that sparsifying D using a generic π helps significantly. Moreover, when a "special" π does help significantly, coming up with π is non-trivial.

For example, in the setting of satisfiability, if f, g are clauses that share variable v with opposite signs, then not having the arc $f \to g$ in C requires either that addressing f should never involve flipping v, cutting A_f by half, or finding a permutation of the clauses such that in *every* state in which f is the greatest violated clause, g is satisfied by some variable other than v. The only non-trivial case we know where the latter can be done is when F is satisfiable by the pure literal heuristic.

As far as we know, the method by which a bad event (flaw) is selected in each step does not affect the performance of any of the algorithmic extensions of the LLL even though in the setting of [18] this choice can be arbitrary. The only use we know of this freedom lies in enabling parallelization when Ω is a structured set [18], [13], [14], [4]. Since we allow Ω to be completely amorphous, it is not readily clear how to approach parallelization in our setting.

At the same time it is clear that in practice the choice of which flaw to address in each step matters. Inspired by the so-called LeftHanded version of the Local Lemma introduced by Pedgen [20], in Theorem 2 (see Section VI) we give a condition under which the flaw order π can be chosen in a *provably* beneficial way, by organizing the flaws in an order akin to an elimination sequence. Specifically, we show that when one or more flaws fail the amenability condition (3), we can seek a permutation π and a "responsibility digraph" R derived from C to model a "shift of responsibility" from flaws failing the condition to flaws that have slack. While R need not be a sparsification of C, if all flaws satisfy (3) with $\Gamma_R(f)$ replacing $\Gamma(f) = \Gamma_C(f)$, our result asserts that a flawless object can be found quickly. Interestingly, to reap this benefit we must employ a recursive, i.e., non-Markovian, process to select which flaw to address in each step. In particular, different flaws present in a state may be addressed in different steps. This is why we required a non-empty set of actions for every flaw present in a state, and why the definition of the causality digraph ignores the effect of clause choice.

Finally, we note that flaw choice in our framework is not really restricted to using a single permutation. For example, in the non-recursive setting, before beginning the walk we can select an arbitrary infinite sequence of permutations $\pi_1, \pi_2, \ldots$ of F and in the i-th step of the walk address the greatest flaw

present according to π_i. If $\pi_1 = \pi_2 = \cdots$ we are back to the single-permutation setting, while if, for example, each π_i is an independent uniformly random permutation, the algorithm addresses a uniformly random flaw present in each step. At the same time, we must make clear that our framework does *not* accomodate arbitrary flaw selection functions and, in fact, we do not see how to extend it beyond permutation-based choice. To keep the presentation of our results uniform (and compact) we have stated both Theorems 1 and 2 in terms of a single permutation. We do point out the one place in our proofs that changes (trivially) to handle multiple permutations.

III. The Lovász Local Lemma

The idea of the Local Lemma was first circulated by Lovász in the early 1970s in an unpublished note. It was published by Erdős and Lovász in [9]. The general form, also due in unpublished form to Lovász, was given by Spencer in [21].

General LLL ([9], [21]). *Let $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$ be a set of m events, and let $D_i \subseteq \mathcal{A}$ denote the dependency set of A_i, i.e., A_i is mutually independent of all events not in D_i. If there exist $x_1, \ldots, x_m \in [0,1)$ such that for all $i \in [m]$, $\Pr(A_i) \leq x_i \prod_{A_j \in \mathcal{D}_i}(1 - x_j)$, then the probability that none of the events in $\mathcal{A}$ occur is at least $\prod_{i=1}^{m}(1 - x_i) > 0$.*

The main value of the LLL lies in that it can deliver strong results even when μ is chosen without *any* consideration of the flaws/events, rendering it indistinguishable from magic. For example, when Ω has product structure over a set V, such as when coloring the vertices of a graph, μ is typically taken to be the product measure in which for each $v \in V$, the measure $\mu(v)$ is uniform over the the domain of v. Such choice also greatly simplifies the search for suitable $\{x_i\}$, to the point that corollaries offering "pre-packaged" $\{x_i\}$, such as the Asymmetric LLL in (1), suffice for most applications.

In [10], Erdős and Spencer noted that one can replace the LLL's requirement that each bad event is dependent with few other bad events with the weaker requirement that each bad event is *negatively correlated* with few other bad events. That is, the non-occurrence of a bad event may boost the probability of a few bad events, but all other bad events should be either unaffected, or become less likely. One can represent the boosting relationships as a so-called *lopsided dependence* digraph, wherein each event points to the events it may boost. For v a vertex in a digraph L, we let $\Gamma(v) = \Gamma_L(v)$ denote the out-neighbourhood of v in L.

Lopsided LLL ([10]). *Let $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$ be a set of m events and let L be a digraph on $[m]$. If there exist $x_1, \ldots, x_m \in [0,1)$ such that for all $i \in [m]$ and all $S \subseteq [m] \setminus \Gamma(i)$,*

$$\Pr\left(A_i \,\middle|\, \bigcap_{j \in S} \overline{A_j}\right) \leq x_i \prod_{j \in \Gamma(i)} (1 - x_j),$$

then the probability that none of the events in $\mathcal{A}$ occur is at least $\prod_{i=1}^{m}(1 - x_i) > 0$.

A natural setting for the lopsided LLL arises when one seeks a collection of permutations satisfying a set of constraints and considers the uniform measure on them. While the bad events (constraint violations) are now typically heavily dependent (since all elements of each permutation affect one another), one can often establish sufficient negative correlation among the bad events to apply the lopsided LLL.

A. Constructive Versions

In a landmark paper [18], Moser and Tardos made the General LLL efficiently constructive for the case of *product* measures over explicitly presented variables. In this so-called *variable setting*, each event A_i is associated with the set of variables $\mathrm{vbl}(A_i)$ that determine it, and A_i, A_j are dependent whenever $\mathrm{vbl}(A_i) \cap \mathrm{vbl}(A_j) \neq \emptyset$. They proved that if the condition of the general LLL holds, then repeatedly selecting *any* occurring event in $\mathcal{A}$ (flaw present) and (addressing it by) resampling every variable in it independently of all others, leads to an elementary event where no event in $\mathcal{A}$ holds (flawless object) after a polynomial number of resamplings. Since the dependence relationship in the variable setting is inherently symmetric, the results of [18] do not apply for the lopsided LLL (with the notable exception of CNF-SAT).

More recently, Harris and Srinivasan [13] made the lopsided LLL constructive for the uniform measure on [Cartesian products of] permutations, by far the most common use case. Pointing out that the lopsided LLL has been gainfully applied to other measures as well, they asked if their results can be extended beyond permutations, leaving as a canonical open problem whether the results of Dudek, Frieze and Ruciński [8] regarding Hamilton Cycles in edge colored hypergraphs can be made constructive.

In Section VII we show how our framework yields efficient algorithms for the results of [8] with minimal effort.

IV. Comparison to Our Work

Besides dispensing with the need for Ω to have product structure (variables) or symmetry (permutations), our setting has two additional benefits.

A. State-dependent Transformations

The LLL, framed as a result in probability, *begins* with a measure μ. Its algorithmic versions amount to moving within Ω by applying probabilistic transformations *consistent with* μ. Specifically, in the variable setting of [18], the only transformation allowed is resampling all variables of a bad event according to the measure's projection on them. As μ must be a product measure, this means that each variable is resampled independently of all others and using the same distribution *every* time it is resampled, i.e., obliviously to the current state. In the partial resampling framework of [12] one can resample a subset of a bad event's variables, but again only in the fashion of [18]. Similarly, when μ is the uniform measure on permutations [13], the elements whose images form a violated constraint must be reshuffled in a very specific way, again without regard for the violation of other constraints.

In contrast, our framework allows the set of transformations for addressing each flaw present in a state σ to depend arbitrarily on σ. As a fundamental example, we can recover the (elementary) fact that a graph G with maximum degree Δ can be colored with $q = \Delta + 1$ colors, overcoming a well-known shortcoming of the LLL (see the survey of Szegedy [23]). Specifically, imagine that to recolor a monochromatic edge e we select an endpoint v of e arbitrarily and assign v a new color c. When the choice of c must be uniform among *all* colors, as mandated when using the uniform measure in the Moser-Tardos variable setting [18], the obliviousness of the choice necessitates the use of many colors relative to Δ for collisions to become rare. Specifically, the LLL can only work when $q > \mathrm{e}\Delta$. On the other hand, in our setting, the color c can be selected uniformly among the colors *available* for v, i.e., the colors not appearing in v's neighborhood, by taking the set of actions $A(f, \sigma)$ to be precisely the set of states that result by assigning available colors to v in σ. Thus, as soon as $q \geq \Delta + 1$, every $A(f, \sigma) \neq \emptyset$ and $\Gamma(f) = \emptyset$ for all $f \in F$, trivializing the amenability condition (3).

B. Dependencies vs. Actions

Unlike the general LLL and the variable setting of Moser and Tardos [18] where the (dependency) relation between events is symmetric, our (causality) relation, similarly to the lopsided LLL, is not. We consider asymmetry a significant structural feature of our work as it can yield strictly sparser relations, as demonstrated in [23]. Asymmetry is also essential in our development of structured clause choice in Section VI.

To enable asymmetry our setting replaces the limited negative dependence condition of the lopsided LLL, which can be highly non-trivial to establish [15], with limited causality under atomicity, a condition that is not only far less restrictive, but also much easier to check. Moreover, to our pleasant surprise, in all applications we have considered so far [1], we have found atomicity to be a very valuable *aid* in the design of flaws and actions, i.e., "a feature not a bug".

V. Proof of Theorem 1

A. Versions of Flaws

For any ordering π of the flaws atomicity implies that the digraph D_π is simple, since for every state σ we only retain the outgoing arcs in D labeled by $I_\pi(\sigma) = I(\sigma)$. For the purposes of the proof it will be convenient to turn D_π to a multidigraph D^*_π as described below. We emphasize that this transformation is trivial from an algorithmic point of view, but helps with the associated eventual counting, leading to a more compact proof. Let $Z \geq 1$ be the least common multiple of the integers $\{A_f : f \in F\}$.

We replace each arc $\sigma \xrightarrow{f} \tau$ in D_π with Z/A_f new arcs from σ to τ, carrying labels $f_1, f_2, \ldots, f_{Z/A_f}$. We refer to each such label as a *version* of flaw f. Observe that since all arcs leaving each $\sigma \in D_\pi$ have the same label, all arcs leaving a state σ are replaced by the same number $Z/A_{I(\sigma)}$ of versioned arcs. Therefore, the uniform random walk on the versioned graph started at σ_1 induces exactly the same probability distribution on sequences of vertices as the uniform walk on D_π (indeed the two walks can be coupled so that the sequence of vertices visited is the same). For the purposes of the analysis it will be convenient to also connect each sink vertex of D^*_π to σ_1 using Z distinct, parallel arcs (not labeled by elements of F), so that the walk continues even after it reaches a flawless state.

Definition 3. *A walk $\Sigma = \sigma_1 \xrightarrow{w_1} \sigma_2 \xrightarrow{w_2} \sigma_3 \cdots \sigma_t \xrightarrow{w_t} \sigma_{t+1}$ is called a t-trajectory. A t-trajectory is* bad *if it only goes through flawed states.*

Intuitively, in order to move away from a flawed state σ the uniform random walk on D^*_π first choses $\tau \in A(I(\sigma), \sigma)$ uniformly at random, i.e., the next state, and then consumes an additional amount of randomness to "choose a version" of $I(\sigma)$, i.e., to chose one of the $Z/A_{I(\sigma)}$ arcs from σ to τ. Based on this view, we will typically speak of the flaw f addressed by the algorithm at each step, rather than of the version of f. Note that to move from any flawed state σ to the next state, the walk must select among

$$|A\left(I(\sigma), \sigma\right)| \cdot \frac{Z}{A_{I(\sigma)}} \geq Z$$

possibilities. Also, any step from a flawless state to σ_1 requires choosing amongst exactly Z possibilities. Therefore, the probability of any t-trajectory is at most $1/Z^t$. Having a uniform upper bound on the probability of each trajectory as a function of its length was precisely why we introduced versioned flaws.

To prove Theorem 1 we will give $T_0 = T_0(|\Omega|, |U(\sigma_1)|)$ such that the probability that a $(T_0 + s)$-trajectory on D^*_π is bad is exponentially small in s. To do this, let $\mathrm{Bad}(t)$ be the set of bad t-trajectories starting from σ_1 and notice that, per our discussion above, each such trajectory has probability at most $1/Z^t$. Therefore, it suffices to bound $|\mathrm{Bad}(t)|/Z^t$.

Our first step is the same as Moser's [17] for SAT, generalized to the notion of atomicity, and amounts to defining an almost-1-to-1 map from bad t-trajectories to sequences of t flaws. A crucial point is that while the map is not 1-1, it becomes 1-1 with the addition of a piece of information whose size is *independent* of t. Specifically, Claim 1 below implies that $|\mathrm{Bad}(t)|$ is bounded by the number of possible witnesses of bad t-trajectories multiplied by $|\Omega|$.

Definition 4. *If $\Sigma = \sigma_1 \xrightarrow{w_1} \sigma_2 \xrightarrow{w_2} \sigma_3 \cdots \sigma_t \xrightarrow{w_t} \sigma_{t+1}$ is a bad t-trajectory, the sequence $W(\Sigma) = w_1, w_2, \ldots, w_t$, i.e., the sequence of versioned flaws labeling the arcs Σ, is the* witness *of Σ.*

Claim 1. *If D is atomic then the map from bad t-trajectories $\Sigma \to \langle W(\Sigma), \sigma_{t+1} \rangle$ is one-to-one.*

Proof. Let f_t be the flaw of which w_t is a version. The atomicity of D implies that σ_t is the unique state in D with an arc $\sigma_t \xrightarrow{f_t} \sigma_{t+1}$. Etc. □

The next step, which is novel, amounts to representing the witness (stripped of version information) as a sequence of sets in a way that reflects causality.

B. Break Sequences

To strip a bad t-trajectory Σ of its version information we let $q_i = I(\sigma_i)$, i.e., q_i is the (unversioned) flaw of which w_i is a version. We define the Version Sequence, $V(\Sigma)$, of Σ to be the sequence of t integers in $[\max_{f\in F} Z/A_f]$ whose i-th element is the version of $I(\sigma_i)$. So, if $w_i = f_j$, then $q_i = f$ and the i-th element of V is j. Clearly, given the sequence $q_1, q_2, \ldots, q_t$ and $V(\Sigma)$ it is trivial to reconstruct $W(\Sigma)$.

Definition 5. *Let $B_0 = U(\sigma_1)$. For all $1 \le i \le t-1$, let $B_i = U(\sigma_{i+1}) \setminus (U(\sigma_i) \setminus I(\sigma_i))$.*

Thus, B_i is the set of flaws "introduced" by the i-th step of the algorithm, where a flaw f "introduces itself" if it remains present after an action from $A(f, \sigma_i)$ is taken (per our earlier discussion note that we are referring to flaws not versions thereof). Let $B_i^* \subseteq B_i$ comprise the flaws in B_i addressed by the walk, i.e., $B_i \setminus B_i^*$ consists of the flaws in B_i eradicated "collaterally" by an action taken to address some other flaw (comprising set O_i below), and the flaws in B_i which remained present in every subsequent state without being addressed (comprising set N_i below).

Definition 6. *For $0 \le i \le t-1$, let*

$$\begin{aligned} O_i &= \{f \in B_i \mid \exists j \in [i+1, t] : \\ &\qquad f \notin U(\sigma_{j+1}) \wedge \forall \ell \in [i+1, j] : f \neq q_\ell\} \\ N_i &= \{f \in B_i \mid \forall j \in [i+1, t] : \\ &\qquad f \in U(\sigma_{j+1}) \wedge \forall \ell \in [i+1, t] : f \neq q_\ell\} \\ B_i^* &= B_i \setminus \{O_i \cup N_i\} \ . \end{aligned}$$

Given $B_0^*, B_1^*, \ldots, B_{i-1}^*$ we can determine $q_1, q_2, \ldots, q_i$ inductively, as follows. Let $E_1 = B_0^*$, while for $i \ge 1$ let

$$E_{i+1} = (E_i - q_i) \cup B_i^* \ . \tag{4}$$

By construction, the set $E_i \subseteq U(\sigma_i)$ is guaranteed to contain $q_i = I(\sigma_i) = I(U(\sigma_i))$. Since $I = I_\pi$ returns[1] the greatest flaw in its input according to π, it must be that $I_\pi(E_i) = q_i$. We note that this is the only place we ever use that the function I is derived by an ordering of the flaws, thus guaranteeing that for every $f \in F$ and $S \subseteq F$, if $I(S) \neq f$ then $I(S \backslash f) = I(S)$. We refer to the sequence $B^* = B_0^*, B_1^*, \ldots, B_{t-1}^*$ as the *Break Sequence* of Σ and note that, as described above, it suffices to reconstruct the sequence $q_1, q_2, \ldots, q_t$.

C. Break Forests

Next we give another 1-to-1 map, showing how to represent the Break Sequence of a bad t-trajectory as a labelled forest with t nodes, called the *Break Forest* of Σ.

The Break Forest of a bad t-trajectory Σ has precisely one vertex per step of Σ, labelled by the flaw addressed by the step. The set B_0^* will provide the labels for the roots of the trees of the forest, while for each $i \ge 1$, the children of the vertex corresponding to the i-th step are labelled with the flaws in B_i^*. So, for example, if $B_i^* = \emptyset$, then the vertex corresponding to the i-th step will be a leaf in the forest. Thus, the Break Forest $\phi = (\Phi, l_\phi)$ of a bad t-trajectory is a finite, rooted, forest Φ with exactly t vertices, organized into no more than $|U(\sigma_1)|$ trees, together with a function (labeling) $l_\phi : V(\Phi) \to F$. Formally, if $B_0^*, B_1^*, \ldots, B_{t-1}^*$ is the Break Sequence of Σ, then $\phi = \phi(\Sigma)$ consists of:

- $|B_0^*|$ trees, the roots of which are labelled by distinct elements of $|B_0^*|$.
- If v is the vertex of ϕ corresponding to the i-th step, then v has $|B_i^*|$ children labelled by the elements of B_i^*.

Observe that while neither the trees, nor the nodes inside each tree of the Break Forest are ordered, we can still reconstruct the sequence $q_1, q_2, \ldots, q_t$ from $\phi(\Sigma)$, by the following simple procedure:

1: $j \leftarrow 1$
2: $E \leftarrow$ The set of labels of the roots of ϕ
3: **while** $E \neq \emptyset$ **do**
4: $\quad q'_j = I(E)$
5: $\quad$ Let B be the set of labels of the children of q'_j in ϕ.
6: $\quad$ Let $E \leftarrow (E \setminus \{q'_j\}) \cup B$
7: $\quad j \leftarrow j + 1$

By induction, whenever $j = i$ in the above procedure, the set E equals the set E_i in (4) and, as a result, $q'_i = q_i$.

D. Counting

We are now ready to conclude the proof by bounding the number of possible ⟨Break Forest, Version Sequence⟩ pairs. Recall that neither the trees, nor the nodes inside each tree in a Break Forests are ordered. To proceed with the count we fix an arbitrary ordering of F, not necessarily π, and first convert the Break Forest into an ordered forest by ordering the trees according to their roots and similarly ordering the children of each vertex. Having induced this ordering, for the purpose of the counting it will be helpful to incorporate the Version Sequence by relabeling the vertices of the Break Forest so as to carry not only the flaw flaw addressed, but also the integer denoting its version. We refer to the resulting object as the (ordered) versioned Break Forest.

We will map each tree T of the versioned Break Forest to a sequence of $|T|$ arrays, each array representing the progeny of exactly one vertex of T, the first array representing the progeny of the root, the second array (if it exists) representing the progeny of the root's leftmost child, etc. in breadth first order. To define the arrays we recall that each node in the Break Forest that is labelled by a flaw f has children labelled by *distinct* flaws in $B_i^* \subseteq \Gamma(f)$. As each such flaw $g \in \Gamma(f)$ has precisely Z/A_g versions, we can represent the progeny of any node of T as a binary array of size d, setting to one the array entry for each versioned flaw in the progeny, where

$$d := \max_{f\in F} \sum_{g\in\Gamma(f)} \frac{Z}{A_g} \ge \max_{f\in F} \max_{g\in\Gamma(f)} \frac{Z}{A_g} = \max_{f\in F} \frac{Z}{A_f} := \gamma \ ,$$

the equality above following from the fact that, since the causality graph C is strongly connected, for every flaw f there exists at least one flaw g such that $f \in \Gamma(g)$.

[1] If instead of π we had a sequence of permutations $\pi_1, \pi_2, \ldots$, we would simply use I_{π_i} to determine q_i from E_i.

We call the above map $\mathrm{Array}(T)$ and observe that:

- The total number of ones in $\mathrm{Array}(T)$ is $|T|-1$ since we don't write a one for the root.
- The Array map is self-terminating, termination occurring the first time the quantity "ones seen minus arrays read" becomes negative (initially being 0).
- Given the unversioned root of T and $\mathrm{Array}(T)$ we can recover T up to the version of its root.

Therefore, we can recover the versioned Break Forest from its versioned roots and the concatenation of the Array mappings of its trees. Moreover, observe that when the forest has r roots, the concatenation of the Array mappings is a string in $\{0,1\}^{td}$ with exactly $t-r$ ones. Writing $|U(\sigma_1)|=m$, we see that the number of versioned Break Forests is bounded by

$$\sum_{r=0}^{m}\binom{m}{r}\gamma^r\binom{td}{t-r}\ . \tag{5}$$

To estimate (5) we distinguish between $d=1$ and $d\geq 2$. For $d\geq 2$, we have $(t-i)/(td-t+r-i)\leq 1/(d-1)$ for all $0\leq i\leq r-1$ in (6) below and, therefore,

$$\begin{aligned}\gamma^r\frac{\binom{td}{t-r}}{\binom{td}{t}} &= \gamma^r\frac{t!}{(t-r)!}\frac{(td-t)!}{(td-t+r)!}\\ &= \frac{\gamma^r t_{(r)}}{(td-t+r)_{(r)}}\\ &\leq \left(\frac{\gamma}{d-1}\right)^r\ .\end{aligned}\tag{6}$$

Thus, for $d\geq 2$, the number of versioned Break Forests is bounded by

$$\binom{td}{t}\sum_{r=0}^{m}\binom{m}{r}\left(\frac{\gamma}{d-1}\right)^r=\binom{td}{t}\left(1+\frac{\gamma}{d-1}\right)^m\ .$$

Let $h(x)=-x\log_2 x-(1-x)\log_2(1-x)$ be the binary entropy function. For $d\geq 2$, Stirling's approximation yields $\log_2\binom{td}{t}\leq tdh(1/d)$. Thus, the binary logarithm of the probability that the walk has not encountered a flawless state after t steps is at most

$$\begin{aligned}&\log_2|\mathrm{Bad}(t)|-t\log_2 Z\\ \leq\ &\log_2|\Omega|+tdh(1/d)+m\log_2\left(1+\frac{\gamma}{d-1}\right)-t\log_2 Z\\ \leq\ &T_0-(\log_2 Z-dh(1/d))t\ ,\end{aligned}\tag{7}$$

where $T_0\leq\log_2|\Omega|+2m$ for all $d\geq 2$ and $\gamma\in[d]$. When $\gamma=1$, in particular, note that we can take $T_0=\log_2|\Omega|+m$.

From (7) we see that if $\log_2 Z-dh(1/d)>0$, we can make the probability of failure arbitrarily small by making t sufficiently large. The definition of δ yields (8) below and thus

$$\begin{aligned}\log_2 Z-dh(1/d) &\geq \log_2 Z-\log_2(d\cdot\mathrm{e})\\ &= -\log_2\left(\max_{f\in F}\sum_{g\in F}\frac{\mathrm{e}}{A_g}\right)\\ &= -\log_2(1-\delta)\\ &> \delta\ .\end{aligned}\tag{8}$$

Therefore, we can conclude for $t=(T_0+s)/\delta$, the probability that the uniform random walk on D_π does not reach a flawless state within t step is less than 2^{-s}, as desired.

To deal with the case $d=1$ we first note that since for every flaw f there exists at least one flaw g such that $f\in\Gamma(g)$, since C is strongly connected, if $A_f\leq 2$ for every $f\in F$, the amenability condition (3) can only be satisfied if C is empty (in which case, anyhow, the result follows trivially: each step of the walk deterministically eradicates the flaw addressed without introducing any other flaws). Therefore, we can assume that $A_f\geq 3$ for some f, implying $Z\geq 3$. We also note that if $d=1$ then $\delta=1-\mathrm{e}/Z$.

Since $\gamma\leq d$, when $d=1$ we must have $\gamma=1$, in which case (5) equals $\binom{m+t}{t}$. Thus, for $d=1$, the binary logarithm of the number of versioned Break Forests is bounded by $m+t$. Letting $T_0=\log_2|\Omega|+m$ we see that the binary logarithm of the probability that the walk has not encountered a flawless state after $t=(T_0+s)/\delta$ steps is at most

$$\begin{aligned}&\log_2|\Omega|+m+t-t\log_2 Z\\ \leq\ &\log_2|\Omega|+m+(T_0+s)\frac{1-\log_2 Z}{1-\mathrm{e}/Z}\\ <\ &\log_2|\Omega|+m-T_0-s\\ \leq\ &-s\ ,\end{aligned}\tag{9}$$

since for $Z\geq 3$ the fraction in (9) is at most $-2/\ln 2$.

VI. Structured Flaw Selection via Responsibility Digraphs

Let $C=C(\Omega,D,F)$ be the causality digraph for a set of objects Ω, multi-digraph D, and set of flaws F. Let π be an ordering of F, inducing an ordering of the vertices of C. For an ordered set of vertices $v_1<v_2\cdots<v_n$, say that arc $v_i\to v_j$ is *forward* if $i<j$ and *backward* if $i>j$. Let R be a digraph on the same π-ordered vertices as C. We say that R is a *responsibility digraph for D with respect to π* if:

1) Every forward arc of C is a forward arc of R. Also every self-loop of C is a self-loop of R.
2) If a backward arc $v_j\to v_i$ of C does not exist in R, then for each k such that $v_k\to v_j$ exists in R, $v_k\to v_i$ exists in R as well.

The neighbourhood of a flaw f in a responsibility graph R is $\Gamma_R(f)=\{g\in F: f\to g \text{ exists in } R\}$. Recall that $I_\pi:2^F\to F$ is the function that maps each set of flaws to its greatest element according to π, with $I_\pi(\emptyset)=\emptyset$. Consider now the following *recursive* algorithm, noting that for $R=C$ the algorithm is simply the uniform random walk on D_π.

Recursive Walk

```
1: procedure ELIMINATE
2:     Let σ ← σ_1
3:     while U(σ) ≠ ∅ do
4:         ADDRESS (I_π(U(σ)), σ)
5:     return σ
6: procedure ADDRESS(f, σ)
7:     σ ← A uniformly random element of A(f, σ)
8:     g ← I_π(U(σ) ∩ Γ_R(f))      ▷ When R = C this is the
9:     while g ≠ ∅ do               ▷ same as g ← I_π(U(σ))
10:        ADDRESS(g, σ)            ▷ trivializing recursion
```

Definition 7. *D is* amenable under *R if for every $f \in F$,*

$$\sum_{g \in \Gamma_R(f)} \frac{1}{A_g} < \frac{1}{\mathrm{e}} \ . \tag{10}$$

Theorem 2. *Fix any ordering π of the elements of F and any $\sigma_1 \in \Omega$. If R is a responsibility digraph for D with respect to π and D is amenable under R, then the probability that the Recursive Walk on D started at σ_1 does not reach a sink within $t = (T_0 + s)/\delta$ steps is 2^{-s}, where $T_0 = \log_2 |\Omega| + 2|U(\sigma_1)|$.*

A. Proof of Theorem 2

The following captures the correctness of Recursive Walk.

Lemma 1. *For a set of flaws F, an ordering π of F, and a flaw $f \in F$, let S_f denote the set of flaws strictly greater than f according to π. For a state σ and a flaw $f \in U(\sigma)$, let $W(\sigma, f) = U(\sigma) \cap S_f$.*

If we invoke ADDRESS(f, σ) *and it terminates at state σ', then $W(\sigma', f) \subseteq W(\sigma, f)$ and $f \notin U(\sigma')$.*

Given Lemma 1 we show termination by a proof very similar to the one in Section V for the uniform random walk. As in that proof, for the purposes of the analysis, we create a graph D^* in which each arc with label f is replaced by Z/A_f parallel arcs labeled by equally many versions of f. Similarly, we connect each sink vertex of D to σ_1 using Z distinct, parallel arcs (not labeled by elements of F) in order to make the walk ceaseless. Specifically, we require that whenever (the walk on D induced by) the Recursive Walk algorithm reaches a flawless state, rather than terminating it selects one of the Z distinct arcs to σ_1 uniformly and continues from σ_1 by running the Recursive Walk algorithm (with fresh randomness). Thus, exactly as in the proof for the uniform random walk, the probability of each t-trajectory is at most Z^{-t}.

To bound the number of bad t-trajectories of the Recursive Walk we represent each trajectory Σ as a pair (ϕ, V), where $\phi = \phi(\Sigma)$ is the *Recursive Forest* of the trajectory and $V = V(\Sigma)$ is its *Version Sequence*. The Recursive Forest is a labelled, ordered forest with one tree per invocation of procedure ELIMINATE. It has t nodes, one for each invocation of the procedure ADDRESS, each node labeled by the flaw of the invocation. The children of a node v are the nodes that correspond to invocations of ADDRESS made by the invocation corresponding to v. Thus, the preorder traversal of the Recursive Forest outputs the sequence of (non-versioned) flaws addressed in Σ. Identically to the proof for the uniform walk, $V(\Sigma)$ is a sequence of integers in $[\max_{f \in F} Z/A_f]$, the i-th integer corresponding to the version of the i-th flaw addressed in Σ. Thus, from $(\phi(\Sigma), V(\Sigma))$ we can trivially reconstruct the witness $W(\Sigma)$.

We will prove that the root labels in a Recursive Forest that corresponds to a bad t-trajectory are distinct and, similarly, that the children of each node in the forest are distinct (both properties were automatically true for Break Forests). If we can do that then each ⟨Recursive Forest, Version Sequence⟩ pair can be represented as a versioned Break Forest with $d' = \max_{f \in F} \sum_{u \in \Gamma_R(f)} Z/A_u$. The rest of the proof is then identical to that for Break Forests and therefore omitted.

To establish the distinctness of the root labels, observe that each time procedure ELIMINATE is invoked at a state σ, by definition of I_π, we have $S_{I_\pi(\sigma)} = \emptyset$. By Lemma 1, if the invocation returns at state σ', then neither $I_\pi(\sigma)$ nor any of the fauls greater than it are present in σ'. Therefore, ELIMINATE is invoked at most once for each $f \in F$. To see the distinctness of the labels of the children of each node, consider a node v of the Recursion Forest corresponding to an invocation of procedure ADDRESS(f, σ). Whenever this invocation of ADDRESS recursively invokes ADDRESS(g, σ'), where $g \in \Gamma_R(f)$, by definition of I_π, every flaw in $S_g \cap \Gamma_R(f)$ is absent from σ'. Whenever each such invocation returns neither g, nor any of the flaws in $S_g \cap \Gamma_R(f)$ are present, and thus Lemma 1 implies that ADDRESS(f, σ) invokes ADDRESS(g, σ') at most once for each $g \in \Gamma_R(f)$, proving the statement.

Proof of Lemma 1. The execution of ADDRESS(f, σ) generates a recursion tree, each node labeled by the flaw of the invocation. Thus, the root is labelled by f, while each child of the root is labelled by a flaw in $\Gamma_R(f)$. Assume, for the sake of contradiction, that flaw $f^* \in (W(\sigma', f) \setminus W(\sigma, f)) \cup \{f\}$ is present at σ'.

Since ADDRESS(f, σ) terminates, $f^* \notin \Gamma_R(f)$. Let $S_f^+ = S_f \cup \{f\}$ and observe that $\Gamma(f) \cap S_f^+ \subseteq \Gamma_R(f) \cap S_f^+$ since the forward edges and the self-loops of C are a subset of the forward edges and self-loops of R. Therefore, f^* cannot have been introduced by the action taken to address f at the original invocation. For a state τ, let $Q(f, \tau)$ be the set of flaws in $S_f^+ \setminus \Gamma_R(f)$ that are present in τ. We claim that if $g \in \Gamma_R(f)$ and ADDRESS(g, τ) terminates at τ', then $Q(f, \tau') \subseteq Q(f, \tau)$. This suffices to prove the lemma since it implies that when each subtree of the root of the recursion tree returns, no flaws in $S_f \setminus \Gamma_R(f)$ that were not present in σ will be present in σ' and neither is f. In particular f^* can not be present.

To prove the above claim, consider the recursion tree of ADDRESS(g, τ). If $h \in Q(f, \tau')$ and $h \notin Q(f, \tau)$, then there has to be a path $g_1 = g, g_2, \ldots, g_i$ from the root of the recursion tree of ADDRESS(g, τ) to a node g_i such that:

- $h \in \Gamma(g_i)$, Notice we are referring to Γ not $\Gamma_R(g_i)$
- $h \notin \Gamma_R(g_j)$ for each $j \in [i]$.

To see this, notice that since h was absent in τ but is present in τ', it must have been introduced by some flaw g_i addressed

during the execution of ADDRESS(g, τ). But if h belonged in the neighborhood (with respect to R) of any of the flaws on the path from the root to g_i, the algorithm would have not terminated.

However, such a path can not exist. Assume that $i = 1$. Then it must be that all of the following hold, violating the second condition in the definition of responsibility digraphs:

- $h \in \Gamma(g_1)$, The arc $g_1 \to h$ exists in C
- $h \notin \Gamma_R(g_1)$, The arc $g_1 \to h$ does not exist in R
- $g_1 \in \Gamma_R(f)$, The arc $f \to g_1$ exists in R
- $h \notin \Gamma_R(f)$. The arc $f \to h$ does not exist in R

Similarly, for $i > 1$, we need to have $h \in \Gamma(g_i)$, $h \notin \Gamma_R(g_i)$, $g_i \in \Gamma_R(g_{i-1})$ but $h \notin \Gamma_R(g_{i-1})$, violating once again the second property of the responsibility digraphs. □

VII. A First Application
Hamilton Cycles in Hypergraphs

A. Preliminaries

An edge coloring of a hypergraph $H(V, E)$ is a function $\phi : E \to \mathbb{N}$ assigning natural numbers (colors) to its edges. We will say that $e_1 \neq e_2 \in E$ are *adjacent* if $e_1 \cap e_2 \neq \emptyset$. A subhypergraph S is *properly* colored if every two adjacent edges of S receive different colors. If, further, every edge of S receives a different colorwe will say that S is *rainbow*.

For a coloring ϕ and a color $i \in \mathbb{N}$, let $H^i_\phi = H[\phi^{-1}(i)]$ denote the hypergraph induced by the edges of color i in ϕ. We say that ϕ is *r-degree bounded* if H^i_ϕ has maximum degree at most r, for all $i \in \mathbb{N}$. If H^i_ϕ has at most r edges, for all $i \in \mathbb{N}$, we say that ϕ is *r-bounded*.

We investigate properly colored and rainbow Hamilton cycles in colored k-uniform complete hypergraphs for $k \geq 3$. (A hypergraph is *k-uniform* if every edge has size k; it is complete if all k-element subsets of the vertices form edges). For $1 \leq \ell < k$, an *ℓ-overlapping cycle* is a k-uniform hypergraph in which, for some cyclic ordering of its vertices, every edge consists of k consecutive vertices and every two consecutive edges share exactly ℓ vertices. Thus, the number of edges in an ℓ-overlapping cycle with s vertices is $\lfloor s/(k-\ell) \rfloor$. The two extreme cases $\ell = 1$ and $\ell = k - 1$ are referred to as, respectively, *loose* and *tight* cycles.

Remark 2. *If $k - \ell$ divides s, then a tight cycle on s vertices contains an ℓ-overlapping cycle on the same vertex set.*

Given a k-uniform hypergraph H on n vertices where $k-\ell$ divides n, an ℓ-overlapping cycle is called *Hamilton* if it goes through every vertex of H, that is, if $s = n$. We denote such a Hamilton cycle by $C^{(k)}_n(\ell)$. Let $K^{(k)}_n$ denote the complete k-uniform hypergraph on n vertices.

In [8], Dudek, Frieze and Ruciński proved the following.

Theorem 3 ([8])**.** *For every $1 \leq \ell < k$ there is $c = c(k, \ell)$ such that if n is sufficiently large and $k-\ell$ divides n, then any $cn^{k-\ell}$-bounded coloring of $K^{(k)}_n$ contains a rainbow copy of $C^{(k)}_n(\ell)$.*

Theorem 4 ([8])**.** *For every $1 \leq \ell < k$ there is $d = d(k, \ell)$ such that if n is sufficiently large and $k-\ell$ divides n, then any $dn^{k-\ell}$-degree bounded coloring of $K^{(k)}_n$ contains a properly colored copy of $C^{(k)}_n(\ell)$.*

In [7], Dudek and Ferrara strengthened Theorems 3, 4 as follows. Say that a coloring is *(a, r)-bounded* if for each color i, every set of a vertices is contained in at most r edges of color i. A r-degree bounded coloring is, thus, $(1, r)$-bounded and an r-bounded coloring is $(0, r)$-bounded (as it has at most r edges of color i). Thus, Theorem 3 follows from Theorem 5 since for every $1 \leq \ell \leq k$, every $cn^{k-\ell}$-bounded coloring is both $(0, cn^{k-1})$-bounded and $(\ell, cn^{k-\ell})$-bounded. Similarly, Theorem 4 follows from Theorem 6 since for every $1 \leq \ell \leq k$, every $dn^{k-\ell}$-degree-bounded coloring is $(\ell, dn^{k-\ell})$-bounded.

Theorem 5 ([7])**.** *For every $1 \leq \ell < k$ there is a constant $c = c(k, \ell)$ such that if n is sufficiently large and $k-\ell$ divides n, then any $(\ell, cn^{k-\ell})$-bounded coloring of $K^{(k)}_n$ that is $(0, cn^{k-1})$-bounded contains a rainbow copy of $C^{(k)}_n(\ell)$.*

Theorem 6 ([7])**.** *For every $1 \leq \ell < k$ there is a constant $d = d(k, \ell)$ such that if n is sufficiently large and $k-\ell$ divides n, then any $(\ell, dn^{k-\ell})$-bounded coloring of $K^{(k)}_n$ contains a properly colored of $C^{(k)}_n(\ell)$.*

We make Theorems 5 and 6 constructive while also improving the constants from [7].

Theorem 7. *The Hamilton cycles guaranteed by Theorems 5, 6 can be found in time $O(kn^{6k+1} \log n)$.*

B. Proof of Theorem 7

The following proposition follows from results in [8].

Proposition 1. *Fix $1 \leq \ell < k$. Let $\{e, f\}$ be any pair of edges of $K^{(k)}_n$ with $|e \cap f| = \alpha \leq \ell$. Let X be any set of pairs $\{g, h\}$ of edges of $K^{(k)}_n$ satisfying $(e \cup f) \cap (g \cup h) = \emptyset$.*

- *Let $\mathcal{C}(X)$ be the set of all copies C of $C^{(k)}_n(k-1)$ in $K^{(k)}_n$ such that $\{g, h\} \nsubseteq C$ for all $\{g, h\} \in X$.*
- *Let $\mathcal{C}_{e,f}(X) = \{C \in \mathcal{C}(X) : \{e, f\} \subset C\}$.*

There is $\delta = \delta(k, \ell) > 0$ such that if $\mathcal{C}_{e,f}(X) \neq \emptyset$, one can find a disjoint family $\{\mathcal{S}_C : C \in \mathcal{C}_{e,f}(X)\}$ of sets of copies of $C^{(k)}_n(k-1)$ from $\mathcal{C}(X)$ (indexed by the copies $C \in \mathcal{C}_{e,f}(X)$) such that for all $C \in \mathcal{C}_{e,f}(X)$:

1) *$\mathcal{S}_C \cap \mathcal{C}_{e,f}(X) = \emptyset$.*
2) *$|\mathcal{S}_C| \geq \delta n^{2k-2}$, if $\alpha = 0$.*
3) *$|\mathcal{S}_C| \geq \delta n^{2k-\alpha-1}$, if $1 \leq \alpha \leq \ell$.*

Furthermore, a uniformly random element of each set $\mathcal{S}_C$ can be sampled in time $O(n^{2k})$.

Constructive Proof of Theorem 5. Fix $1 \leq \ell < k$ and let ϕ be a coloring of $K^{(k)}_n$. Let M consist of all pairs of edges that have the same color and share at most ℓ vertices, i.e.,

$$M = \{\{e_1, e_2\} : e_1, e_2 \in K^{(k)}_n, \phi(e_1) = \phi(e_2), |e_1 \cap e_2| \leq \ell\}.$$

Let Ω be the set of copies of $C_n^{(k)}(k-1)$ in $K_n^{(k)}$. For each pair of edges $\{e,f\} \in M$ we define the flaw

$$F_{e,f} = \{C \subset K_n^{(k)} : C \sim C_n^{(k)}(k-1) \text{ and } \{e,f\} \subset C\} \ .$$

That is, $F_{e,f}$ consists of all tight Hamilton cycles containing both e and f which, since $\phi(e) = \phi(f)$), means they are improperly colored. A flawless $C \in \Omega$ is, thus, a tight Hamilton cycle whose edges have distinct colors. As $k-\ell$ divides n, Remark 2 implies that any such cycle C contains a rainbow copy of $C_n^{(k)}(\ell)$.

Next we define actions for each flaw. To that end, for each pair $\{e,f\} \in M$ and each integer $0 \le \alpha \le \ell$, we define

$$Y_{e,f}(\alpha) = \{\{e',f'\} \in M : \{e',f'\} \neq \{e,f\}, |e' \cap f'| = \alpha, \text{ and } (e \cup f) \cap (e' \cup f') \neq \emptyset\} \ .$$

Let $Y_{e,f} = \bigcup_{\alpha=0}^{\ell} Y_{e,f}(\alpha)$ and $X_{e,f} = M \setminus (Y_{e,f} \cup \{e,f\})$. For each Hamilton cycle $C \in \Omega$, for each pair of edges $\{e,f\} \in M$ such that flaw $F_{e,f}$ is present in C, we invoke Proposition 1 with e, f and $X = X_{e,f}$. Let $\mathcal{S}_C$ be the set of Hamilton cycles guaranteed by Proposition 1. We let $A(F_{e,f}, C) = \mathcal{S}_C$. To lighten notation, we let $A_{e,f} := A_{F_{e,f}} = \min_{C \in \Omega} |A(F_{e,f}, C)|$. By Proposition 1 we thus have:

- D is atomic since for each flaw we have a disjoint family of sets of cycles (actions).
- If $|e \cap f| = \alpha$, then $A_{e,f} \ge \begin{cases} \delta n^{2k-2} & \text{if } \alpha = 0, \\ \delta n^{2k-\alpha-1} & \text{if } 1 \le \alpha \le \ell. \end{cases}$
- If $F_{g,h} \in \Gamma(F_{e,f})$ then $\{g,h\} \in Y_{e,f}$ since $\{g,h\} \notin X_{e,f}$ and $\{g,h\} \neq \{e,f\}$.

To bound $|Y_{e,f}|$ we use the following fact, established in [7]. For every $c > 0$, if ϕ is $(\ell, cn^{k-\ell})$-bounded and $(0, cn^{k-1})$-bounded, then there exists $n_0 = n_0(c)$ such that for all $n \ge n_0$,

$$\max_{\{e,f\} \in M} |Y_{e,f}(\alpha)| \le \begin{cases} 2ckn^{2k-2} & \text{if } \alpha = 0, \\ 2ck^{\ell+1} n^{2k-\alpha-1} & \text{if } 1 \le \alpha \le \ell \ . \end{cases}$$

Therefore, if $c = \delta(2\mathrm{e}k(1+\ell k^{\ell}))^{-1}$, for each pair of edges $\{e,f\} \in M$ with $|e \cap f| = \alpha$, we have

$$\sum_{F_{g,h} \in \Gamma(F_{e,f})} \frac{1}{A_{g,h}} = \sum_{\alpha=0}^{\ell} \sum_{\{g,h\} \in Y_{e,f}(\alpha)} \frac{1}{A_{g,h}} \le$$

$$\frac{2ckn^{2k-2}}{\delta n^{2k-2}} + \sum_{\alpha=1}^{\ell} \frac{2ck^{\ell+1} n^{2k-\alpha-1}}{\delta n^{2k-\alpha-1}} = \frac{2ck}{\delta}(1+\ell k^{\ell}) < \frac{1}{\mathrm{e}} \ .$$

Thus, by Theorem 1 the uniform random walk on D terminates after $O(|M|^2 \log |\Omega|))$ steps with high probability. Further, we know that $\log_2 |\Omega| \le \log_2 \binom{n}{k}^n \le nk \log_2 n$ and that $|M| \le n^{2k}$. Since at each step of the algorithm we need $O(n^{2k})$ time to find the greatest flaw and $O(n^{2k})$ time to choose an action for it, we have proven the theorem. □

Constructive Proof of Theorem 6 (sketch). Modify the definition of set M in the proof of Theorem 5 so that it contains no pair of disjoint edges. Analogously, define the sets $Y_{e,f}(\alpha)$ only for $\alpha \in [\ell]$, and consequently $Y_{e,f} = \cup_{a=1}^{\ell} Y_{e,f}(\alpha)$. Now for $d = \delta(2\mathrm{e}\ell k^{\ell+1})^{-1}$ Theorem 1 applies. □

ACKNOWLEGEMENTS

DA thanks Christos Papadimitriou for introducing him to the Probabilistic Method a quarter century ago and Mario Szegedy for inspiration to shun the lowest common denominator (Z). We are also grateful to the anonymous reviewers for comments that improved the presentation.

REFERENCES

[1] D. Achlioptas and F. Iliopoulos. The Lovász Local Lemma as a Random Walk. *ArXiv e-prints*, June 2014, 1406.0242.

[2] Noga Alon. A parallel algorithmic version of the local lemma. *Random Struct. Algorithms*, 2(4):367–378, 1991.

[3] József Beck. An algorithmic approach to the Lovász local lemma. I. *Random Structures Algorithms*, 2(4):343–365, 1991.

[4] Karthekeyan Chandrasekaran, Navin Goyal, and Bernhard Haeupler. Deterministic algorithms for the Lovász local lemma. In *SODA*, pages 992–1004. SIAM, 2010.

[5] Artur Czumaj and Christian Scheideler. Coloring non-uniform hypergraphs: a new algorithmic approach to the general Lovász local lemma. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 2000)*, pages 30–39, 2000.

[6] P Diaconis. The cutoff phenomenon in finite markov chains. *Proceedings of the National Academy of Sciences*, 93(4):1659–1664, 1996.

[7] Andrzej Dudek and Michael Ferrara. Extensions of results on rainbow hamilton cycles in uniform hypergraphs. *Graphs and Combinatorics*, pages 1–7, 2013.

[8] Andrzej Dudek, Alan M. Frieze, and Andrzej Rucinski. Rainbow hamilton cycles in uniform hypergraphs. *Electr. J. Comb.*, 19(1):P46, 2012.

[9] Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday), Vol. II*, pages 609–627. Colloq. Math. Soc. János Bolyai, Vol. 10. North-Holland, Amsterdam, 1975.

[10] Paul Erdös and Joel Spencer. Lopsided Lovász local lemma and latin transversals. *Discrete Applied Mathematics*, 30(2-3):151–154, 1991.

[11] Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New constructive aspects of the Lovász local lemma. In *FOCS*, pages 397–406, 2010.

[12] David G. Harris and Aravind Srinivasan. The Moser-Tardos framework with partial resampling. In *FOCS*, pages 469–478, 2013.

[13] David G. Harris and Aravind Srinivasan. A constructive algorithm for the Lovász local lemma on permutations. In *SODA*, pages 907–925. SIAM, 2014.

[14] Kashyap Babu Rao Kolipaka and Mario Szegedy. Moser and Tardos meet Lovász. In *STOC*, pages 235–244. ACM, 2011.

[15] Linyuan Lu, Austin Mohr, and László Székely. Quest for negative dependency graphs. In *Recent Advances in Harmonic Analysis and Applications*, pages 243–258. Springer, 2013.

[16] Michael Molloy and Bruce Reed. Further algorithmic aspects of the local lemma. In *STOC '98 (Dallas, TX)*, pages 524–529. 1999.

[17] Robin A. Moser. A constructive proof of the Lovász local lemma. In *STOC'09—Proceedings of the 2009 ACM International Symposium on Theory of Computing*, pages 343–350. ACM, New York, 2009.

[18] Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2):Art. 11, 15, 2010.

[19] Christos H. Papadimitriou. On selecting a satisfying truth assignment. In *FOCS*, pages 163–169. IEEE Computer Society, 1991.

[20] Wesley Pegden. Highly nonrepetitive sequences: Winning strategies from the local lemma. *Random Struct. Algorithms*, 38(1-2):140–161, 2011.

[21] Joel Spencer. Asymptotic lower bounds for ramsey functions. *Discrete Mathematics*, 20(0):69 – 76, 1977.

[22] Aravind Srinivasan. Improved algorithmic versions of the Lovász local lemma. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 611–620, New York, 2008. ACM.

[23] Mario Szegedy. The Lovász local lemma - a survey. In Andrei A. Bulatov and Arseny M. Shur, editors, *CSR*, volume 7913 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 2013.