Witness

A proof technique

Chunyang Wang
Nanjing University

January 14, 2022

What is witness?

What is witness?

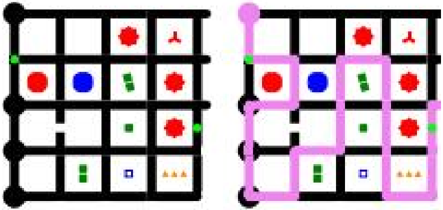


Figure 1 A small Witness puzzle featuring all clue types (left) and its solution (right). (Not from the actual video game.)

[ABC+19]

Witness

- Witness is a kind of proof technique used to make bounds(on running time of the algorithm, size of certain combinatorial structure, etc.).

Witness

- Witness is a kind of proof technique used to make bounds(on running time of the algorithm, size of certain combinatorial structure, etc.).
- Both its name and definition are made up by myself.

Witness

- Witness is a kind of proof technique used to make bounds(on running time of the algorithm, size of certain combinatorial structure, etc.).
- Both its name and definition are made up by myself.
- Not exactly, as I took its name from the "witness tree" term defined in the celebrated result from Moser and Tardos in algorithmic local lemma.

Witness

- Witness is a kind of proof technique used to make bounds(on running time of the algorithm, size of certain combinatorial structure, etc.).
- Both its name and definition are made up by myself.
- Not exactly, as I took its name from the "witness tree" term defined in the celebrated result from Moser and Tardos in algorithmic local lemma.
- Its main idea can be concluded as: if something (contrary to what you want to prove) happens, then there exists a certain kind of combinatorial structure(the "witness") whose existence has very low probability.

Witness

- Witness is a kind of proof technique used to make bounds(on running time of the algorithm, size of certain combinatorial structure, etc.).
- Both its name and definition are made up by myself.
- Not exactly, as I took its name from the "witness tree" term defined in the celebrated result from Moser and Tardos in algorithmic local lemma.
- Its main idea can be concluded as: if something (contrary to what you want to prove) happens, then there exists a certain kind of combinatorial structure(the "witness") whose existence has very low probability.
- Then usually we can take a union bound over all possible such combinatorial structures to finish the proof.

A simple puzzle

In case some of you get lost(which is unlikely but anyway), here's a simple puzzle:

A simple puzzle

In case some of you get lost(which is unlikely but anyway), here's a simple puzzle:

Puzzle

There are n vertices $1, 2, \ldots, n$, arranged in a cycle.

1. Every vertex has exactly one of the two states in $\{0, 1\}$. Initially, all vertices are in state 0.
2. Visit vertices in the cyclical order $1, 2, \ldots, n, 1, \ldots$, each time when we visit a vertex v, we do the following:
   2.1 If v is not adjacent to any other vertex in state 0, then the state of v becomes 1.
   2.2 Otherwise, the state of v becomes 1 with probability $\frac{2}{3}$, and becomes 0 with probability $\frac{1}{3}$.

A simple puzzle

In case some of you get lost(which is unlikely but anyway), here's a simple puzzle:

Puzzle

There are n vertices $1, 2, \dots, n$, arranged in a cycle.

1. Every vertex has exactly one of the two states in $\{0, 1\}$. Initially, all vertices are in state 0.
2. Visit vertices in the cyclical order $1, 2, \dots, n, 1, \dots$, each time when we visit a vertex v, we do the following:
   2.1 If v is not adjacent to any other vertex in state 0, then the state of v becomes 1.
   2.2 Otherwise, the state of v becomes 1 with probability $\frac{2}{3}$, and becomes 0 with probability $\frac{1}{3}$.

The problem is: what's the (asymptotic) expected number of visits before all vertices are in state 1? How to formally prove it?

Topics involved

I will show some topics where this technique is involved, namely the following:

Topics involved

I will show some topics where this technique is involved, namely the following:

- Algorithmic Lovász local lemma

Topics involved

I will show some topics where this technique is involved, namely the following:

- Algorithmic Lovász local lemma
  - Approach predating Moser and Tardos' work [Bec91, Alo91, MR99, Sri08, Mos09]
  - Moser and Tardos's approach [MT10]

Topics involved

I will show some topics where this technique is involved, namely the following:

- Algorithmic Lovász local lemma
  - Approach predating Moser and Tardos' work [Bec91, Alo91, MR99, Sri08, Mos09]
  - Moser and Tardos's approach [MT10]
- Information Percolation for proving rapid mixing of Glauber dynamics[HSZ19, JPV21, HSW21]

Constraint Satisfaction Problems(CSP)

- **Variables**: $V = \{x_1, x_2, \ldots, x_n\}$ where $x_i \in [q]$ for each $i \in [n]$
- (local)**Constraints**: $C = \{C_1, C_2, \ldots, C_m\}$
  - Each C_i is defined on a subset $\mathsf{vbl}(C_i)$ of variables

$$C_i : [q]^{\mathsf{vbl}(C_i)} \to \{\mathsf{True}, \mathsf{False}\}$$

Constraint Satisfaction Problems(CSP)

- **Variables**: $V = \{x_1, x_2, \ldots, x_n\}$ where $x_i \in [q]$ for each $i \in [n]$
- (local)**Constraints**: $C = \{C_1, C_2, \ldots, C_m\}$
  - Each C_i is defined on a subset $\mathrm{vbl}(C_i)$ of variables

$$C_i : [q]^{\mathrm{vbl}(C_i)} \to \{\mathrm{True}, \mathrm{False}\}$$

- Any $\boldsymbol{x} \in [q]^n$ is a CSP solution if it satisfies all $C_1, \ldots, C_m$.

Constraint Satisfaction Problems(CSP)

- **Variables**: $V = \{x_1, x_2, \ldots, x_n\}$ where $x_i \in [q]$ for each $i \in [n]$
- (local)**Constraints**: $C = \{C_1, C_2, \ldots, C_m\}$
  - Each C_i is defined on a subset $\mathsf{vbl}(C_i)$ of variables

$$C_i : [q]^{\mathsf{vbl}(C_i)} \to \{\mathsf{True}, \mathsf{False}\}$$

- Any $\boldsymbol{x} \in [q]^n$ is a CSP solution if it satisfies all $C_1, \ldots, C_m$.
- **Examples:**
  - k-CNF, (hyper)graph coloring, set cover, unique games, ...
  - vertex cover, independent set, matching, perfect matching, ...

Lovász local lemma

- Given a CSP formula $\Phi = (V, C)$, let $\mathcal{P}$ be the product distribution where each $x \in V$ takes a uniform random value from $[q]$.
- For each $c \in C$, define its neighbourhood $\Gamma(c)$ as

$$\Gamma(c) = \{c' \in C | \mathsf{vbl}(c) \cap \mathsf{vbl}(c') \neq \varnothing\}$$

Lovász local lemma

- Given a CSP formula $\Phi = (V, C)$, let $\mathcal{P}$ be the product distribution where each $x \in V$ takes a uniform random value from $[q]$.
- For each $c \in C$, define its neighbourhood $\Gamma(c)$ as

$$\Gamma(c) = \{c' \in C | \mathrm{vbl}(c) \cap \mathrm{vbl}(c') \neq \varnothing\}$$

Lovász Local lemma[EL75]

Let $p \triangleq \max_i \Pr_{\mathcal{P}}[\neg C_i]$ and $\Delta \triangleq \max_i |\Gamma(C_i)|$, then

$$4p\Delta \leq 1 \Longrightarrow \Pr_{\mathcal{P}}[\bigwedge_{i=1}^{m} \neg C_i] > 0$$

Algorithmic Lovász local lemma

- Lovász local lemma guarantees the existence of a CSP solution given only probability guarantees.

Algorithmic Lovász local lemma

- Lovász local lemma guarantees the existence of a CSP solution given only probability guarantees.
- However, it(and its proof) doesn't directly imply how to find one such solution.

Algorithmic Lovász local lemma

- Lovász local lemma guarantees the existence of a CSP solution given only probability guarantees.
- However, it(and its proof) doesn't directly imply how to find one such solution.
- And this is what algorithmic(constructive) Lovász local lemma focuses on.

Algorithmic Lovász local lemma

- Lovász local lemma guarantees the existence of a CSP solution given only probability guarantees.
- However, it(and its proof) doesn't directly imply how to find one such solution.
- And this is what algorithmic(constructive) Lovász local lemma focuses on.

Work on algorithmic Lovász local lemma:

- [Sri08]: One can find a solution in $n^{O(\Delta,k)}$ time if $p\Delta^4 \leq c$ for some constant c.
- [MT10]: One can find a solution in $O(ndk)$ time if $p\Delta \leq c$ for some constant c.

A random assignment approach on algorithmic Lovász local lemma

A random assignment approach on algorithmic Lovász local lemma

- Let H_Φ be the hypergraph representation of $\Phi = (V, C)$, if H_Φ is not connected, we can construct a solution for each connected component independently and then put them together.

A random assignment approach on algorithmic Lovász local lemma

- Let H_Φ be the hypergraph representation of $\Phi = (V, C)$, if H_Φ is not connected, we can construct a solution for each connected component independently and then put them together.
- Take another parameter p' such that $4p'q\Delta \leq 1$ and $c_0 p \Delta^3 \leq p'$ for some constant c_0.

A random assignment approach on algorithmic Lovász local lemma

- Let H_Φ be the hypergraph representation of $\Phi = (V, C)$, if H_Φ is not connected, we can construct a solution for each connected component independently and then put them together.
- Take another parameter p' such that $4p'q\Delta \leq 1$ and $c_0 p\Delta^3 \leq p'$ for some constant c_0.

Algorithm in [Sri08]

Phase One: Randomly assign a uniform random values in $[q]$ to any unassigned variable. When the violation probability of some constraint exceeds p', "freeze" all its unassigned variables, keep assigning until all variables are assigned or frozen.
Phase Two: For each remaining connected component in H_Φ after simplifying, using an exhaustive enumeration to find one solution.

Proof sketch

- One can always find a solution for each connected component after Phase One of the algorithm(guaranteed by $4p'q\Delta \leq 1$ and local lemma).

Proof sketch

- One can always find a solution for each connected component after Phase One of the algorithm(guaranteed by $4p'q\Delta \leq 1$ and local lemma).
- Let's call a constraint "bad" if its violation probability exceeds p' in Phase One. Denote the set of such constraints as B.

Proof sketch

- One can always find a solution for each connected component after Phase One of the algorithm(guaranteed by $4p'q\Delta \leq 1$ and local lemma).
- Let's call a constraint "bad" if its violation probability exceeds p' in Phase One. Denote the set of such constraints as B.
- Let $Lin(H_\Phi)$ be the line graph of H_Φ, and let $L^2(B)$ be the graph whose vertices are B and $B_i \neq B_j \in B$ are adjacent if and only if $dist_{Lin(H_\Phi)}(B_i, B_j) \leq 2$.

Proof sketch

- One can always find a solution for each connected component after Phase One of the algorithm(guaranteed by $4p'q\Delta \leq 1$ and local lemma).
- Let's call a constraint "bad" if its violation probability exceeds p' in Phase One. Denote the set of such constraints as B.
- Let $Lin(H_\Phi)$ be the line graph of H_Φ, and let $L^2(B)$ be the graph whose vertices are B and $B_i \neq B_j \in B$ are adjacent if and only if $dist_{Lin(H_\Phi)}(B_i, B_j) \leq 2$.

Proof sketch

- One can always find a solution for each connected component after Phase One of the algorithm(guaranteed by $4p'q\Delta \leq 1$ and local lemma).
- Let's call a constraint "bad" if its violation probability exceeds p' in Phase One. Denote the set of such constraints as B.
- Let $Lin(H_\Phi)$ be the line graph of H_Φ, and let $L^2(B)$ be the graph whose vertices are B and $B_i \neq B_j \in B$ are adjacent if and only if $dist_{Lin(H_\Phi)}(B_i, B_j) \leq 2$.

Observation

After Phase One, each not satisfied constraint is either bad or share at least one frozen variable with some bad constraint.

Corollary

Distinct connected component in $L^2(B)$ are disconnected in $Lin(H_\Phi)$.

Proof sketch(Cont'd)

- A major observation is that each connected component in $L^2(B)$ after Phase One is small.

Lemma

If $16p\Delta^3 \leq p'$, after Phase One, the probability that $L^2(B)$ has a connected component of size at least L is at most $n\Delta \cdot 2^{-L/\Delta}$.

Proof sketch(Cont'd)

- A major observation is that each connected component in $L^2(B)$ after Phase One is small.

Lemma

If $16p\Delta^3 \leq p'$, after Phase One, the probability that $L^2(B)$ has a connected component of size at least L is at most $n\Delta \cdot 2^{-L/\Delta}$.

- Proof idea: Find some witness that happens with very low probability if L is large.

Proof sketch(Cont'd)

Definition($\{2,3\}$-tree)[Alo91]

Let $G = (V, E)$ be a graph. A set of vertices $T \subseteq V$ is a $\{2,3\}$-tree if

(1) for any $u, v \in T, dist_G(u, v) \geq 2$;

(2) if one adds an edge between every $u, v \in T$ such that $dist_G(u, v) = 2$ or 3, then T is connected.

Proof sketch(Cont'd)

Definition($\{2,3\}$-tree)[Alo91]

Let $G = (V, E)$ be a graph. A set of vertices $T \subseteq V$ is a $\{2,3\}$-tree if

(1) for any $u, v \in T, dist_G(u, v) \geq 2$;

(2) if one adds an edge between every $u, v \in T$ such that $dist_G(u, v) = 2$ or 3, then T is connected.

[Alo91], Lemma 2.1

Let $G = (V, E)$ be a graph with maximum degree Δ. Then, for any $v \in V$, the number of $\{2,3\}$-trees in G of size t containing v is at most $\frac{(e\Delta^3)^{t-1}}{2}$.

Proof sketch(Cont'd)

(corollary for [GLLZ18], Lemma 14)

Let $H = (V, E)$ be a hypergraph such that each hyperedge in E intersects at most Δ other hyperedges (equivalently, the degree of $Lin(H)$ is at most Δ).Let $B \subseteq E(H)$ be a set of hyperedges which induces a connected subgraph in $L^2(H)$, and $e^* \in B$ be an arbitrary hyperedge. There exists a $\{2, 3\}$-tree $T \subseteq B$ such that $e^* \in T$ in $Lin(H)$ and $|T| \geq \frac{|B|}{\Delta}$.

Proof sketch(Cont'd)

(corollary for [GLLZ18], **Lemma 14**)

Let $H = (V, E)$ be a hypergraph such that each hyperedge in E intersects at most Δ other hyperedges (equivalently, the degree of $Lin(H)$ is at most Δ).Let $B \subseteq E(H)$ be a set of hyperedges which induces a connected subgraph in $L^2(H)$, and $e^* \in B$ be an arbitrary hyperedge. There exists a $\{2, 3\}$-tree $T \subseteq B$ such that $e^* \in T$ in $Lin(H)$ and $|T| \geq \frac{|B|}{\Delta}$.

- So for each connected component of size $\geq L$, we can find a $\{2, 3\}$-tree in $Lin(H_\Phi)$ with size $\geq \frac{L}{\Delta}$, by Markov's inequality, such $\{2, 3\}$-tree occurs with probability at most $\left(\frac{p}{p'}\right)^{\frac{L}{\Delta}}$.

Proof sketch(Cont'd)

(corollary for [GLLZ18], **Lemma 14**)

Let $H = (V, E)$ be a hypergraph such that each hyperedge in E intersects at most Δ other hyperedges (equivalently, the degree of $Lin(H)$ is at most Δ).Let $B \subseteq E(H)$ be a set of hyperedges which induces a connected subgraph in $L^2(H)$, and $e^* \in B$ be an arbitrary hyperedge. There exists a $\{2,3\}$-tree $T \subseteq B$ such that $e^* \in T$ in $Lin(H)$ and $|T| \geq \frac{|B|}{\Delta}$.

- So for each connected component of size $\geq L$, we can find a $\{2,3\}$-tree in $Lin(H_\Phi)$ with size $\geq \frac{L}{\Delta}$, by Markov's inequality, such $\{2,3\}$-tree occurs with probability at most $\left(\frac{p}{p'}\right)^{\frac{L}{\Delta}}$.
- As there are at most $|B|(e\Delta^3)^{\frac{L}{\Delta}}$ such trees, taking a union bound leads to the desired answer.

Short summary

- The trick here is to use $\{2,3\}$-tree as a witness for a large connected component. As all constraints in a $\{2,3\}$-tree are independent, we can easily bound its probability of occurrence.

Short summary

- The trick here is to use $\{2,3\}$-tree as a witness for a large connected component. As all constraints in a $\{2,3\}$-tree are independent, we can easily bound its probability of occurrence.
- Although this algorithm was outperformed by Moser-Tardos algorithm that we are going to introduce next, the idea of this random assignment approach still find its use in a harder problem sampling Lovász local lemma where one is required to count the number of solutions(sample a solution uniformly at random) and remains the state of art[JPV20].

The Moser-Tardos algorithm

The Moser-Tardos algorithm

1. Initialize all variables uniformly at random.
2. While there exists an unsatisfied constraint: pick one (various rules) and resample all its variables.

The Moser-Tardos algorithm

The Moser-Tardos algorithm

1. Initialize all variables uniformly at random.
2. While there exists an unsatisfied constraint: pick one (various rules) and resample all its variables.

Theorem[MT10]

If there exists $\alpha_1, \ldots, \alpha_m \in [0, 1)$ s.t.

$$\forall i, \Pr_{\mathcal{P}}[\neg C_i] \leq \alpha_i \prod_{C_j \in \Gamma(C_i)} (1 - \alpha_j),$$

then the Moser-Tardos algorithm terminates within $\sum_{i=1}^{m} \frac{\alpha_i}{1-\alpha_i}$ resamples in expectation.

Execution log

The Moser-Tardos algorithm

1. Initialize all variables uniformly at random.
2. While there exists an unsatisfied clause: pick one (various rules) and resample all its variables.

Execution log

The Moser-Tardos algorithm

1. Initialize all variables uniformly at random.
2. While there exists an unsatisfied clause: pick one (various rules) and resample all its variables.

Execution log

Execution log(Exe-log)Λ of the M-T algorithm:

$$\Lambda_1, \Lambda_2, \ldots, \in C :$$

random sequence of resampled constraints

Witness Tree

The Moser-Tardos algorithm

1. Initialize all variables uniformly at random.
2. While there exists an unsatisfied clause: pick one (various rules) and resample all its variables.

Witness Tree

The Moser-Tardos algorithm

1. Initialize all variables uniformly at random.
2. While there exists an unsatisfied clause: pick one (various rules) and resample all its variables.

Witness Tree

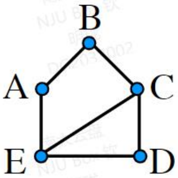
Witness tree $T(\Lambda, t)$: rooted tree, each node u with label $C_{[u]} \in C$.

- Initially, T contains a single root r with Λ_t
- for $i = t - 1$ to 1:
  if $\Lambda_i \in \Gamma^+(C_{[u]})$ for some node $u \in T$
  add child $v \to$ deepest such u, labeled with Λ_i
- $T(\Lambda, t)$ is the resulting T.

Inclusive neighbourhood: $\Gamma^+(C_{[u]}) = \Gamma(C_{[u]}) \cup C_{[u]}$

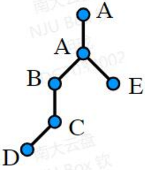
An example

dependency graph:

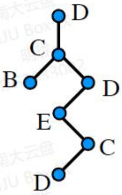


exe-log Λ: D, C, E, D, B, A, C, A, D, ...

$T(\Lambda, 8)$:



$T(\Lambda, 9)$:



Proof sketch

Proposition

$\forall s \neq t,\ T(\Lambda, s) \neq T(\Lambda, t)$.

Proof sketch

Proposition

$\forall s \neq t,\ T(\Lambda, s) \neq T(\Lambda, t)$.

Lemma 1

For any particular witness tree τ:

$$\Pr_{\Lambda}[\exists t, T(\Lambda, t) = \tau] \leq \prod_{u \in \tau} \Pr[\neg C_{[u]}]$$

Proof sketch

Proposition

$\forall s \neq t,\ T(\Lambda, s) \neq T(\Lambda, t)$.

Lemma 1

For any particular witness tree τ:

$$\Pr_{\Lambda}[\exists t, T(\Lambda, t) = \tau] \leq \prod_{u \in \tau} \Pr[\neg C_{[u]}]$$

Lemma 2

For any particular witness tree $\tau \in \mathscr{T}_{C_i}$:

$$\Pr[T_{C_i} = \tau] = \frac{1 - \alpha_i}{\alpha_i} \prod_{u \in \tau} \left[\alpha_{[u]} \prod_{C_j \in \Gamma^+(C_{[u]})} (1 - \alpha_j) \right]$$

Proof sketch(Cont'd)

$$\mathop{\mathbb{E}}_{\#\text{ of } C_i \text{ in } \Lambda} = \sum_{\tau \in \mathscr{T}_{C_i}} \Pr[\exists t, T(\Lambda, t) = \tau]$$

$$(\text{Lemma 1}) \leq \sum_{\tau \in \mathscr{T}_{C_i}} \prod_{u \in \tau} \Pr[\neg C_{[u]}]$$

$$(\text{LLL condition}) \leq \sum_{\tau \in \mathscr{T}_{C_i}} \prod_{u \in \tau} \left[\alpha_{[u]} \prod_{C_j \in \Gamma^+(C_{[u]})} (1 - \alpha_j) \right]$$

$$(\text{Lemma 2}) \leq \frac{\alpha_i}{1 - \alpha_i} \sum_{\tau \in \mathscr{T}_{C_i}} \Pr[T_{C_i} = \tau] \leq \frac{\alpha_i}{1 - \alpha_i}$$

Short summary

- The key idea is the witness tree that captures the independent occurences of various events during the execution of the algorithm.

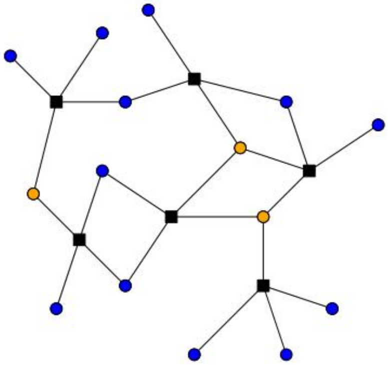
Short summary

- The key idea is the witness tree that captures the independent occurences of various events during the execution of the algorithm.
- Furthermore, M-T algorithm is later shown to be working up to[KS11] and go beyond[HLS21] a tighter condition than LLL which is known as Shearer's bound.

Short summary

- The key idea is the witness tree that captures the independent occurences of various events during the execution of the algorithm.
- Furthermore, M-T algorithm is later shown to be working up to[KS11] and go beyond[HLS21] a tighter condition than LLL which is known as Shearer's bound.
- Moser and Tardos won 2021 Gödel Prize for this contribution.

Hypergraph independent sets



$G = (V, F, E)$

V: vertices(circles).
F: hyperedges(squares).

Degree Δ, Size k

Hypergraph independent set: Every edge has at least one 0.
—Also known as monotone CNF.

Rapid mixing of Glauber dynamics

Glauber dynamics Pick a vertex, flip a coin, set to new value whenver possible.

mixing time $t_{mix} = \min\{t : d_{TV}(\mathbb{P}^t(\sigma, \cdot)) < \frac{1}{4}\}$

Rapid mixing of Glauber dynamics

Glauber dynamics Pick a vertex, flip a coin, set to new value whenver possible.

mixing time $t_{mix} = \min\{t : d_{TV}(\mathbb{P}^t(\sigma, \cdot)) < \frac{1}{4}\}$

Theorem[HSZ19]

For any k-hypergraph with maximum degree $\Delta \leq c2^{k/2}$, the Glauber dynamics mixes in $O(n \log n)$ time.

Coupling

- Let Ω_G be the state space of the Glauber dynamics. Let X^σ be the Markov chain with starting state $\sigma \in \Omega_G$.

Coupling

- Let Ω_G be the state space of the Glauber dynamics. Let X^σ be the Markov chain with starting state $\sigma \in \Omega_G$.
- Let $t_{coup} \triangleq \min\{t : X_t^\sigma = X_t^\tau, \forall \sigma, \tau \in \Omega_G\}$, it follows that

$$t_{mix} \le \min\{T : \Pr[t_{coup} > T] \le 1/4\}$$

Coupling

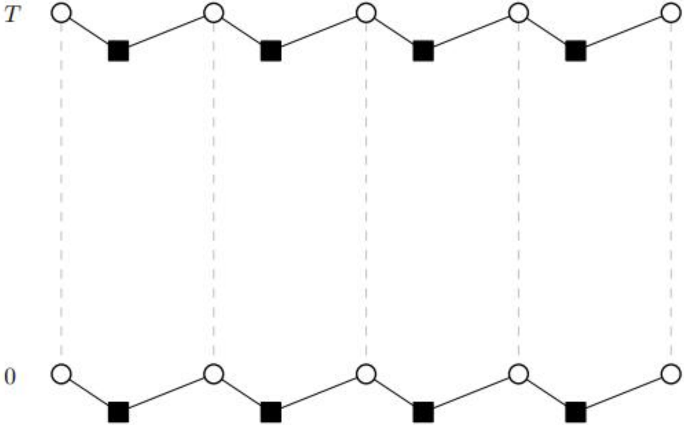
- Let Ω_G be the state space of the Glauber dynamics. Let X^σ be the Markov chain with starting state $\sigma \in \Omega_G$.
- Let $t_{coup} \triangleq \min\{t : X_t^\sigma = X_t^\tau, \forall \sigma, \tau \in \Omega_G\}$, it follows that

$$t_{mix} \leq \min\{T : \Pr[t_{coup} > T] \leq 1/4\}$$

- It is enough to bound $\Pr[t_{coup} > T]$ for $T = O(n \log n)$.

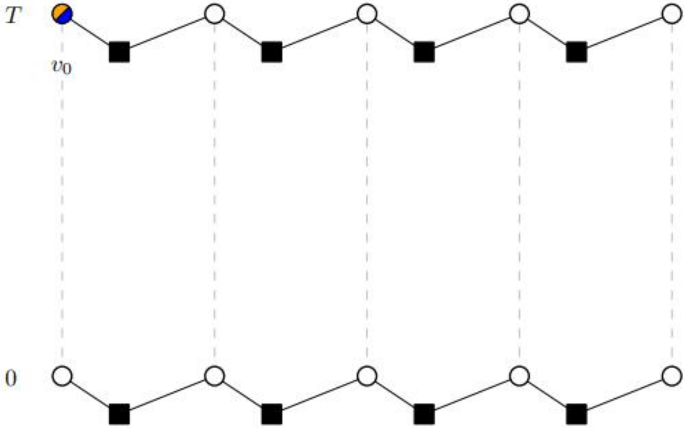
Tracing back the discrepancies

$t_{coup} \geq T$



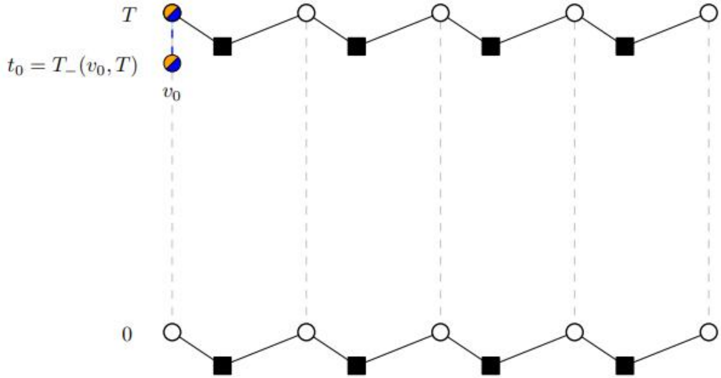
Tracing back the discrepancies(Cont'd)

$t_{coup} \geq T \Longrightarrow \exists \sigma, \tau \in \Omega_G, v_0 \in V$ such that $X_T^{\sigma}(v_0) \neq X_T^{\tau}(v_0)$



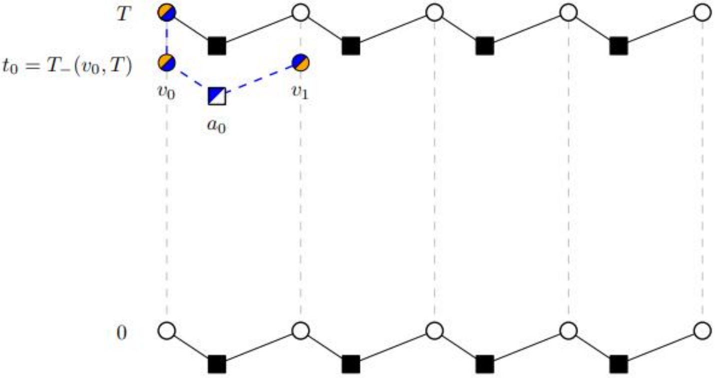
Tracing back the discrepancies(Cont'd)

$t_{coup} \geq T \implies \exists \sigma, \tau \in \Omega_G, v_0 \in V$ such that $X_T^\sigma(v_0) \neq X_T^\tau(v_0)$



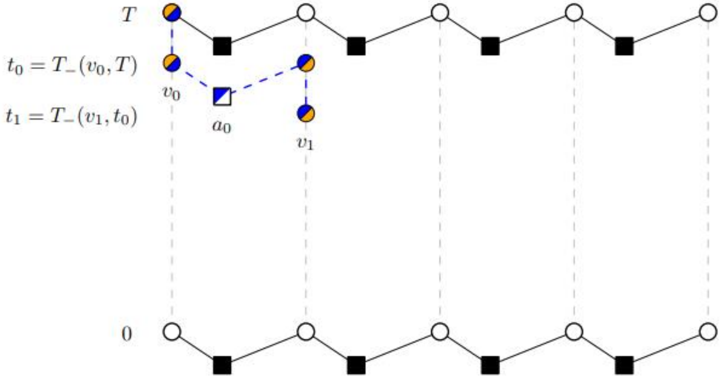
Tracing back the discrepancies(Cont'd)

$t_{coup} \geq T \implies \exists \sigma, \tau \in \Omega_G, v_0 \in V$ such that $X_T^\sigma(v_0) \neq X_T^\tau(v_0)$



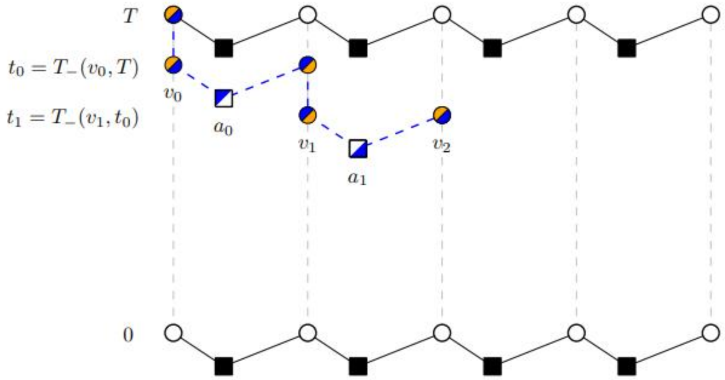
Tracing back the discrepancies(Cont'd)

$t_{coup} \geq T \implies \exists \sigma, \tau \in \Omega_G, v_0 \in V$ such that $X_T^\sigma(v_0) \neq X_T^\tau(v_0)$



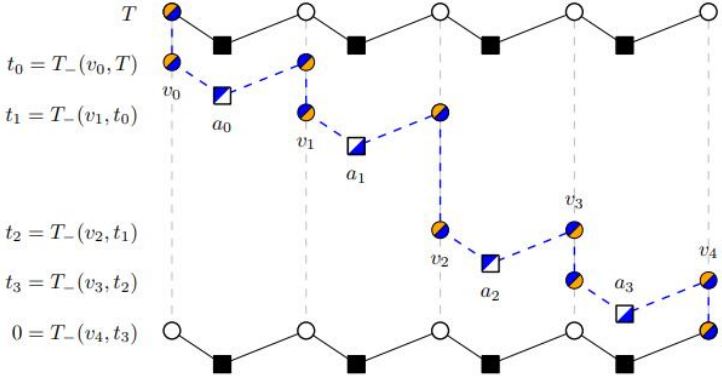
Tracing back the discrepancies(Cont'd)

$t_{coup} \geq T \implies \exists \sigma, \tau \in \Omega_G, v_0 \in V$ such that $X_T^\sigma(v_0) \neq X_T^\tau(v_0)$

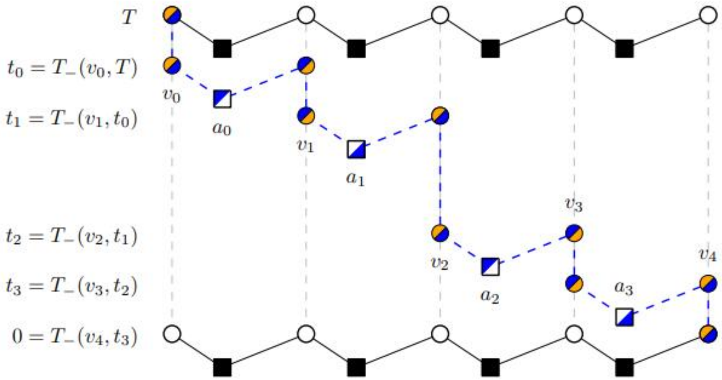


Tracing back the discrepancies(Cont'd)

$t_{coup} \geq T \implies \exists \sigma, \tau \in \Omega_G, v_0 \in V$ such that $X_T^\sigma(v_0) \neq X_T^\tau(v_0)$



Tracing back the discrepancies(Cont'd)



$t_{coup} \geq T \implies \exists((v_l, a_l, t_l))_{0 \leq l \leq L}$ such that for all $0 \leq l \leq L$,

$$(v_l, t_l) \in \text{Updates}, t_l = T_-(v_l, t_{l-1}), X_{t_l}^{\sigma} \vee X_{t_l}^{\tau}(\text{vbl}(a_l) \setminus \{v_l\}) = \mathbf{1}$$

Proof sketch

- We say that $((v_l, a_l, t_l))_{0 \le l \le L}$ is a discrepancy path if

  $$(v_l, t_l) \in \text{Updates}, t_l = T_-(v_l, t_{l-1}), X^{\sigma}_{t_l} \vee X^{\tau}_{t_l}(\text{vbl}(a_l) \backslash \{v_l\}) = \mathbf{1}$$

Proof sketch

- We say that $((v_l, a_l, t_l))_{0 \le l \le L}$ is a discrepancy path if

  $$(v_l, t_l) \in \text{Updates}, t_l = T_-(v_l, t_{l-1}), X^{\sigma}_{t_l} \vee X^{\tau}_{t_l}(\text{vbl}(a_l) \backslash \{v_l\}) = \mathbf{1}$$

- It follows that $t_{coup} > T \implies \exists$ discrepancy path with $t_{-1} = T$

Proof sketch

- We say that $((v_l, a_l, t_l))_{0 \le l \le L}$ is a discrepancy path if

$$(v_l, t_l) \in \text{Updates}, t_l = T_{-}(v_l, t_{l-1}), X^{\sigma}_{t_l} \vee X^{\tau}_{t_l}(\text{vbl}(a_l) \setminus \{v_l\}) = \mathbf{1}$$

- It follows that $t_{coup} > T \implies \exists$ discrepancy path with $t_{-1} = T$
- It remains to bound the probability a certain discrepancy path occurs and take a union bound over all such paths.

Proof sketch

- We say that $((v_l, a_l, t_l))_{0 \le l \le L}$ is a discrepancy path if

$$(v_l, t_l) \in \text{Updates}, t_l = T_-(v_l, t_{l-1}), X^{\sigma}_{t_l} \vee X^{\tau}_{t_l}(\text{vbl}(a_l) \backslash \{v_l\}) = \mathbf{1}$$

- It follows that $t_{coup} > T \implies \exists$ discrepancy path with $t_{-1} = T$
- It remains to bound the probability a certain discrepancy path occurs and take a union bound over all such paths.
- We'll omit this part as it's a bit involved. One may refer to the paper for more details.

Short summary

- The key idea is observing that a long discrepancy path serves as a witness that the two chains fail to couple before time T.

Short summary

- The key idea is observing that a long discrepancy path serves as a witness that the two chains fail to couple before time T.
- This approach gets its name as it is similar to the approach of Information Percolation used to prove cutoff for the Ising model[LS16].

Short summary

- The key idea is observing that a long discrepancy path serves as a witness that the two chains fail to couple before time T.
- This approach gets its name as it is similar to the approach of Information Percolation used to prove cutoff for the Ising model[LS16].
- This approach is later extended to work for analysis of (projected)Glauber dynamics for (possibly non-monotone) atomic CSP in the local lemma regime[JPV21, HSW21].

The End

Bibliography I

Zachary Abel, Jeffrey Bosboom, Michael Coulombe, Erik D. Demaine, Linus Hamilton, Adam Hesterberg, Justin Kopinsky, Jayson Lynch, Mikhail Rudoy, and Clemens Thielen.
Who witnesses the witness? finding witnesses in the witness is hard and sometimes impossible, 2019.

Noga Alon.
A parallel algorithmic version of the local lemma.
In *32nd Annual Symposium on Foundations of Computer Science (San Juan, PR, 1991)*, pages 586–593. IEEE Comput. Soc. Press, Los Alamitos, CA, 1991.

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

Bibliography II

P. Erdős and L. 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. 1975.

Heng Guo, Chao Liao, Pinyan Lu, and Chihao Zhang.
Counting hypergraph colourings in the local lemma regime.
In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 926–939, New York, NY, USA, 2018. Association for Computing Machinery.

Kun He, Qian Li, and Xiaoming Sun.
Moser-tardos algorithm: Beyond shearer's bound, 2021.

Bibliography III

Kun He, Xiaoming Sun, and Kewen Wu.
Perfect sampling for (atomic) lovász local lemma.
CoRR, abs/2107.03932, 2021.

Jonathan Hermon, Allan Sly, and Yumeng Zhang.
Rapid mixing of hypergraph independent sets.
Random Struct. Algorithms, 54:730–767, 2019.

Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong.
Towards the sampling lovász local lemma.
CoRR, abs/2011.12196, 2020.

Vishesh Jain, Huy Tuan Pham, and Thuy-Duong Vuong.
On the sampling lovász local lemma for atomic constraint satisfaction problems, 2021.

Bibliography IV

Kashyap Kolipaka and Mario Szegedy.
Moser and Tardos meet Lovász.
In *STOC'11—Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 235–243. ACM, New York, 2011.

Eyal Lubetzky and Allan Sly.
Information percolation and cutoff for the stochastic Ising model.
J. Amer. Math. Soc., 29(3):729–774, 2016.

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.

Bibliography V

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

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

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. ACM, New York, 2008.