Game Derandomization

Samuel Epstein[*]

April 25, 2023

Abstract

The recently introduced method of derandomization provides bounds on the Kolmogorov complexity of solutions to problems such as k-Sat and Graph-Coloring. This is done by using a simple randomized method that produces a solution with positive probability. This overall method can be applied to games, where if a probabilistic agent beats the environment, then a simple deterministic agent can be shown to win as well. In this paper, we show multiple examples of game derandomization. In one such example, we translate the Algorithmic Lovász Local Lemma of Moser and Tardos into a solution to an interactive k-Sat game.

Introduction

We describe two simplified cybernetic agent models. For the first model, the agent $\mathbf{p}$ and environment $\mathbf{q}$ are defined as follows. The agent is a function $\mathbf{p} : (\mathbb{N}\times\mathbb{N})^* \to \mathbb{N}$, where if $\mathbf{p}(w) = a$, $w \in (\mathbb{N}\times\mathbb{N})^*$ is a list of the previous actions of the agent and the environment, and $a \in \mathbb{N}$ is the action to be performed. The environment is of the form $\mathbf{q} : (\mathbb{N}\times\mathbb{N})^* \times \mathbb{N} \to \mathbb{N}\cup\{\mathbf{W}\}$, where if $\mathbf{q}(w, a) = b \in \mathbb{N}$, then b is $\mathbf{q}$'s response to the agent's action a, given history w, and the game continues. If $\mathbf{q}$ responds $\mathbf{W}$ then the agents wins and the game halts. The agent can be randomized. The game can continue forever, given certain agents and environments. This is called a win/no-halt game.

Theorem 1 ([Eps22]) *If probabilistic agent $\mathbf{p}'$ wins against environment $\mathbf{q}$ with at least probability p, then there is a deterministic agent $\mathbf{p}$ of complexity $\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{K}(\mathbf{p}') - \log p + \mathbf{I}((p, \mathbf{p}', \mathbf{q}); \mathcal{H})$ that wins against $\mathbf{q}$.*

The mutual information term is $\mathbf{I}(x; \mathcal{H}) = \mathbf{K}(x) - \mathbf{K}(x|\mathcal{H})$, where $\mathbf{K}$ is the prefix Kolmogorov complexity, $\mathcal{H} \in \{0,1\}^\infty$ is the halting sequence, and for nonnegative function f, $<^{\log} f$ is defined to be $< f + O(\log(f+1))$. Thus we get the following conclusion.

> *For a given win/no-halt game if there is a good simple randomized player but no simple winning deterministic player, then that game is exotic, in that it has high mutual information with the halting sequence.*

This mirrors derandomization, where if a solution to a problem can be produced easily with a simple probability, but has no simple solutions, then it is exotic. A game instance that has a simple winning probabilistic agent but no simple winning deterministic agent can be found in Example 1.

The second game is modified such that the environment gives a nonnegative rational penalty term to the agent at each round. Furthermore, the environment specifies an end to the game without specifying a winner or loser. This is called a penalty game.

[*]JP Theory Group. samepst@jptheorygroup.org

Theorem 2 ([Eps22]) *If given probabilistic agent $\mathbf{p}'$, environment $\mathbf{q}$ halts with probability 1, and $\mathbf{p}'$ has expected penalty less than $n \in \mathbb{N}$, then there is a deterministic agent $\mathbf{p}$ of complexity $\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{K}(\mathbf{p}') + \mathbf{I}((\mathbf{p}', n, \mathbf{q}); \mathcal{H}) - \log \epsilon$ that receives penalty $(1+\epsilon)n$ against $\mathbf{q}$, for $\epsilon \in (0,1)$.*

Contents

1 Even-Odds

We define the following win/no-halt game, entitled Even-Odds. There are N rounds. At round 1, the environment $\mathbf{q}$ secretly records bit $e_1 \in \{0,1\}$. It sends an empty message to the agent who responds with bit $a_1 \in \{0,1\}$. The agent gets a point if $e_1 \oplus b_1 = 1$. Otherwise the agent loses a point. For round i, the environment secretly selects a bit e_i that is a function of the previous agent's actions $\{a_j\}_{j=1}^{i-1}$ and sends an empty message to the agent, which responds with a_i and the agent gets a point if $e_i \oplus a_i = 1$, otherwise it loses a point. The agent wins after N rounds if it has a score of at least $\sqrt{N}$.

Theorem 3 *For large enough N, there is a deterministic agent $\mathbf{p}$ that can win* Even-Odds *with N rounds, with complexity $\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{I}(\mathbf{q}; \mathcal{H})$.*

Proof. We describe a probabilistic agent $\mathbf{p}'$. At round i, $\mathbf{p}'$ submits 0 with probability 1/2. Otherwise it submits 1. By the central limit theorem, for large enough N, the score of the probabilistic agent divided by $\sqrt{N}$ is $S \sim \mathcal{N}(0,1)$. Let $\Phi(x) = \Pr[S > x]$. A common bound for $\Phi(x)$ is

$$\Phi(x) > \frac{1}{2\pi}\frac{x}{x^2+1}e^{-x^2/2}$$
$$\Phi(1) > \frac{1}{4\pi}e^{-1/2} > \frac{1}{8\pi}.$$

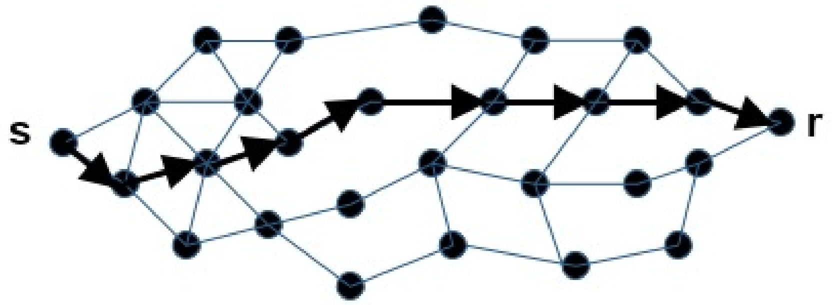

Figure 1: A graphical depiction of a winning deterministic player to the Graph-Navigation game. The player starts at s and chooses a path to reach the goal state r, (assuming $t_G = 8$).

Thus when $S \geq 1$, the score is at least $\sqrt{N}$. Thus $\mathbf{p}'$ wins with probability at least $p = \frac{1}{8\pi}$. Thus by Theorem 1, there exists a deterministic agent $\mathbf{p}$ that can beat $\mathbf{q}$ with complexity

$$\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{K}(\mathbf{p}') - \log p + \mathbf{I}((p, \mathbf{p}', \mathbf{q}); \mathcal{H}) <^{\log} \mathbf{I}(\mathbf{q}; \mathcal{H}).$$

□

2 Graph-Navigation

The win/no-halt game is as follows. The environment $\mathbf{q}$ consists of (G, s, r). $G = (E, V)$ is a non-bipartite graph with undirected edges, $s \in V$ is the starting vertex, and $r \in V$ is the goal vertex. Let t_G be the time it takes for any random walk starting anywhere to converge to the stationary distribution $\pi(v)$, for all $v \in V$, up to a factor of 2.

There are t_G rounds and the agent starts at $s \in V$. At round 1, the environment gives the agent the degree $s \in V$, $\text{Deg}(s)$. The agent picks an number between 1 and $\text{Deg}(s)$ and sends it to $\mathbf{q}$. The agent moves along the edge the number is mapped to and is given the degree of the next vertex it is on. Each round's mapping of numbers to edges to be a function of the agent's past actions. This process is repeated t_G times. The agent wins if it is on $r \in V$ at the end of round t_G. A graphical depiction of this can be seen in Figure 1.

Theorem 4 *There is a deterministic agent* $\mathbf{p}$ *that can win the* Graph-Navigation *game with complexity* $\mathbf{K}(\mathbf{p}) <^{\log} \log |E| + \mathbf{I}((G, s, r); \mathcal{H})$.

Proof. It is well known if G is non-bipartite, a random walk starting from any vertex will converge to a stationary distribution $\pi(v) = \deg(v)/2|E|$, for each $v \in V$.

A probabilistic agent $\mathbf{p}'$ is defined as selecting each edge with equal probability. After t_G rounds, the probability that $\mathbf{p}'$ is on the goal r is close to the stationary distribution π. More specifically the probability is $\overset{*}{>} |E|^{-1}$. Thus by Theorem 1, there is a deterministic agent $\mathbf{p}$ that can find r in t_G turns and has complexity $\mathbf{K}(\mathbf{p}') <^{\log} \log |E| + \mathbf{I}((G, s, t); \mathcal{H})$.

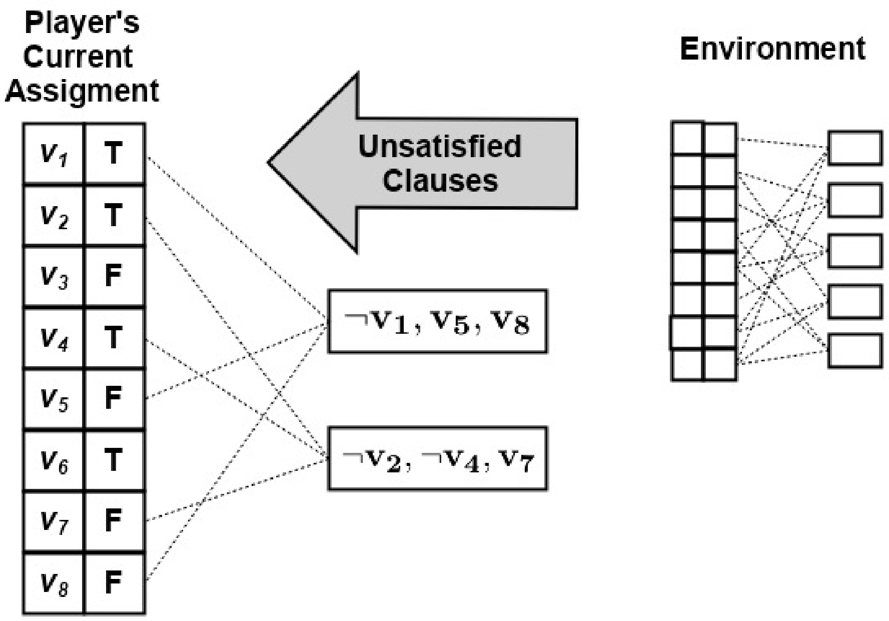

Figure 2: A graphical depiction of a turn in the Interactive-k-Sat game. The player's current assignment of variables has resulted in two unsatisfied clauses, which the environment sends to the player. The environment has a complete picture of the variables and clauses, which are hidden to the player.

3 Interactive-k-Sat

The penalty game Interactive-k-Sat is as follows. The environment $\mathbf{q}$ has access to a hidden k-Sat formula, with n variables and some number of clauses, each containing k literals, which are a variable instance or its negation. Each variable appears in at most $2^k/ke$ clauses. The environment's first action is to send the number of variables to the agent. After this step, the agent, $\mathbf{p}$ has n variables, each initially set to true. At each subsequent round, the environment gives to the agent the clauses which are not satisfied. The player can change up to k variables, and sends these changes to the environment. If the k-Sat formula is satisfied, the game stops. Otherwise the game continues. When the game ends, the penalty is the number of terms. A graphical representation of this game can be seen in Figure 2. Obviously there is a deterministic player of complexity $O(1)$ that can try every possible assignment of variables. This is a winning strategy of at most 2^n turns. However, the following theorem shows that a much more successful player exists without much more complexity bounds.

Theorem 5 *There exists a deterministic player* $\mathbf{p}$ *that can achieve a penalty of* $(1+\epsilon)(n/2k + n/(2^k/e-k))$ *with complexity* $\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{I}(\mathbf{q};\mathcal{H}) - \log\epsilon$*, for* $\epsilon \in (0,1)$.

Proof. We use the Algorithmic Lovász Local Lemma, from [MT10]. The randomized algorithm, when applied to k-Sat is as follows.

1. Given is a random assignment of n variables.
2. While there exists an unsatisfied clause.
   (a) Pick an unsatisfied clause at random.

(b) Reassign the variables of this clause randomly.

3. Return the variable assignment.

It was proved in [MT10] that this algorithm has n/D where D is the size of the dependency between events. So $D = (2^k/e - k)$ expected steps. Thus the goal of this proof is to construct a randomized player $\mathbf{p}'$ that simulates the above randomized algorithm. The player $\mathbf{p}'$ first starts out by randomizing its variables. Thus at each turn in this phase, it selects up to k untouched variables and gives them random assignments. This takes at most $\lceil n/k \rceil$ steps. However, noting that player $\mathbf{p}'$ only needs to update the variables that are set to true, this task takes $n/2k$ expected steps. Once this is complete, whenever the environment sends back unsatsified clauses, the randomized agent chooses one at random and randomly resamples its k variables. Thus, by [MT10], this has $n/(2^k/e - k)$ expected steps before a satisfying assignment is found and the game halts. So the randomized player runs in $(n/2k + n/(2^k/e - k))$ expected steps. Thus by Theorem 2, there is a deterministic player that can find a satisfying assignment in less than $(1+\epsilon)(n/2k + n/(2^k/e - k))$ turns with complexity

$$\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{I}(\mathbf{q}; \mathcal{H}) - \log \epsilon.$$

□

Note that as the game proceeds, the environment reveals more and more information about its hidden k-Sat formula to the player. This leaves open the possibility of a deterministic player of Kolmogorov complexity $O(1)$ that coerces the environment to reveal all the clauses and then manually set the corresponding satisfying assignment. Whether or not this will take less steps than that proved in the above theorem is unknown.

4 Penalty-Tests

An example penalty game is as follows. The environment $\mathbf{q}$ plays a game for N rounds, for some very large $N \in \mathbb{N}$, with each round starting with an action by $\mathbf{q}$. At round i, the environment gives, to the agent, an encoding of a program to compute a probability P_i over $\mathbb{N}$. The choice of P_i can be a computable function of i and the agent's previous turns. The agent responds with a number $a_i \in \mathbb{N}$. The environment gives the agent a penalty of size $T_i(a_i)$, where $T_i : \mathbb{N} \to \mathbb{Q}_{\geq 0}$ is a computable test, with $\sum_{a \in \mathbb{N}} P_i(a) T_i(a) < 1$. After N rounds, $\mathbf{q}$ halts. A graphical representation of this game can be found in Figure 3.

Theorem 6 *There is a deterministic agent* $\mathbf{p}$ *that can receive a penalty* $< (1+\epsilon)N$ *and has complexity* $\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{I}(\mathbf{q}; \mathcal{H}) - \log \epsilon$*, for* $\epsilon \in (0,1)$.

Proof. A very successful probabilistic agent $\mathbf{p}'$ can be defined. Its algorithm is simple. On receipt of a program to compute P_i, the agent randomly samples a number $\mathbb{N}$ according to P_i. At each round the expected penalty is $\sum_a P_i(a) T_i(a) < 1$, so the expected penalty of $\mathbf{p}$ for the entire game is $< N$. Thus by Theorem 2, there is a deterministic agent $\mathbf{p}$ such that

1. The agent $\mathbf{p}$ receives a penalty of $< (1+\epsilon)N$,

2. $\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{I}(\mathbf{q}; \mathcal{H}) - \log \epsilon$.

□

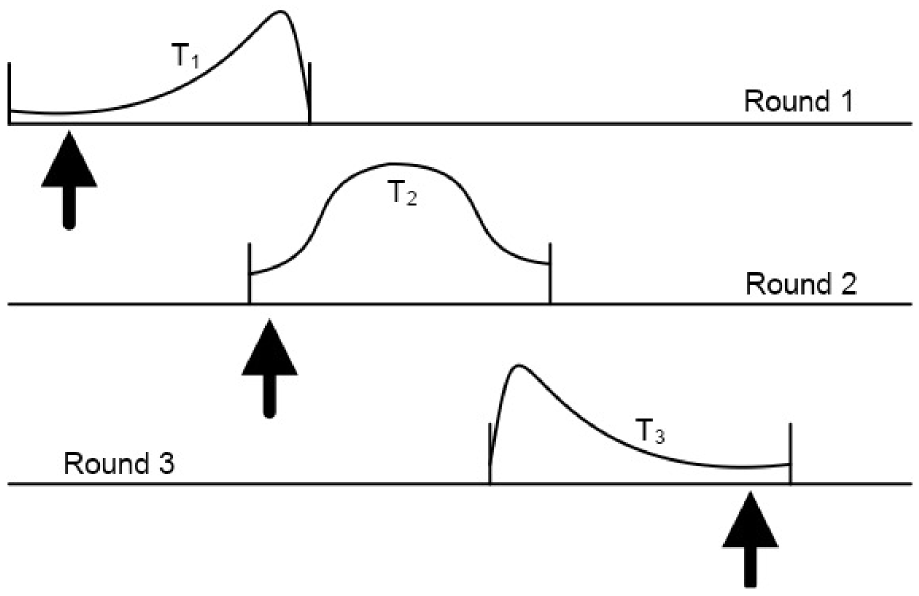

Figure 3: Three rounds of the PENALTY-TEST game. At each round i the probability, P_i, that the environment gives to the player is a uniform measure over a unique interval. The environment has three tests $\{T_1, T_2, T_3\}$, which is not shared with the player, that represent the penalties. In this depiction, the player's moves are numbers in the interval (represented by the arrows) and result in low total penalty.

Example 1 *Let* $\mathbf{q}$ *be defined so that* $P_i(a) = [a \leq 2^i]2^{-i}$ *and* $T_i = [a \leq 2^i]2^{i-\mathbf{K}(a|i)}$*, where* $[A] = 1$ *if* A *is true, and 0 otherwwise. Thus each* T_i *is a randomness deficiency function (to the power of 2). The probabilistic algorithm* $\mathbf{p}'$ *will receive an expected penalty* $< N$*. However any deterministic agent* $\mathbf{p}$ *that receives a penalty* $< 2N$ *must be very complex, as it must select many numbers with low randomness deficiency. Thus, by the bounds above,* $\mathbf{I}(\mathbf{q}; \mathcal{H})$ *must be very high. This makes sense because* $\mathbf{q}$ *encodes* N *randomness deficiency functions.*

5 SET-SUBSET

We define the following win/no-halt game, entitled SET-SUBSET. There are k rounds. At round $i = \{1, \ldots, k\}$, the environment $\mathbf{q}$ gives n numbers $A_i \subset \mathbb{N}$ to the agent $\mathbf{p}$. The environment secretly selects $m \leq n$ numbers $B_i \subseteq A_i$. The player selects a number $a_i \in A_i$. Each A_i and B_i are a function of the player's previous actions. The player wins if for every round, his selection a_i is in the secret set B_i. So for all $i \in \{1, \ldots, k\}$, $a_i \in B_i$. A graphical depiction of this game can be seen in Figure 4.

Theorem 7 *There is a deterministic agent* $\mathbf{p}$ *that win against* SET-SUBSET *environment* $\mathbf{q}$*, where* $\mathbf{K}(\mathbf{p}) <^{\log} k \log(n/m) + \mathbf{I}(\mathbf{q}; \mathcal{H})$*.*

Proof. Let $\mathbf{p}'$ be the randomized the player that selects a member of the given set with A_i with uniform probability. The probability that $\mathbf{p}'$ picks a member of B_i is $|B_i|/|A_i| = m/n$. The probability that $\mathbf{p}'$ picks a member of B_i for all $i \in \{1, \ldots, k\}$ is $(m/n)^k$, which is the probability that $\mathbf{p}'$ wins. $\mathbf{K}(\mathbf{p}') = O(1)$. Thus by Theorem 1, there exists a deterministic player $\mathbf{p}$ that wins against $\mathbf{q}$ with complexity bounded by the theorem statement. $\square$

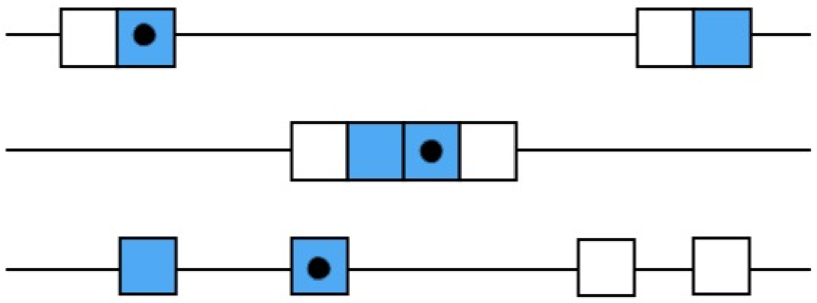

Figure 4: A graphical depiction of the SET-SUBSET game. Each line represents a round of the game. The boxes of the ith line represent A_i, and the filled boxes represent the secret set $B_i \subset A_i$. A winning player is shown, by placing a circle in B_i for each round i.

6 INTERACTIVE-HYPERGRAPH

We define the following penalty game, entitled INTERACTIVE-HYPERGRAPH. The environment has access to a hidden k-regular -hypergraph. A *hypergraph* is a pair $J = (V, E)$ of vertices V and edges $E \subseteq \mathcal{P}(V)$. Thus each edge can connect ≥ 2 vertices. A hypergraph is *k-regular* of the size $|e| = k$ for all edges $e \in E$. A 2-regular hypergraph is just a simple graph. The player has access to a list of vertices and the goal of the player is to produce a valid 2-coloring of the hypergraph. A valid 2-coloring of a hypergraph (V, E) is a mapping $f : V \to \{1, 2\}$ where every edge $e \in E$ is not *monochromatic* $|\{f(v) : v \in e\}| = 2$. We assume that for each edge f of the graph, there are at most $2^{k-1}/e - 1$ edges h such that $f \cap h \neq \emptyset$.

The game proceeds as follows. For the first round, environment gives the number of vertices to the player. The player has n vertices, each with starting color 1. At each subsequent turn, the environment sends to the player the edges which are monochromatic. The player can change the color of to k vertices and sends these changes to the environment. The game ends when the player has a valid 2-coloring of the graph.

Theorem 8 *There exists a deterministic player* $\mathbf{p}$ *that can beat the environment* $\mathbf{q}$ *in* $(1+\epsilon)(n/2k + n/(2^{k-1}/e - 1))$ *turns of complextiy* $\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{I}(\mathbf{q}; \mathcal{H}) - \log \epsilon$.

Proof. We use the Algorithmic Lovász Local Lemma, from [MT10]. The randomized algorithm, when applied to hypergraphs (G, E) is as follows.

1. Given is a random 2-color assignment of n vertices.
2. While there exists a monochromatic edge.
   (a) Pick a monochromatic edge random.
   (b) Reassign the colors of this edge randomly.
3. Return the valid 2-coloring.

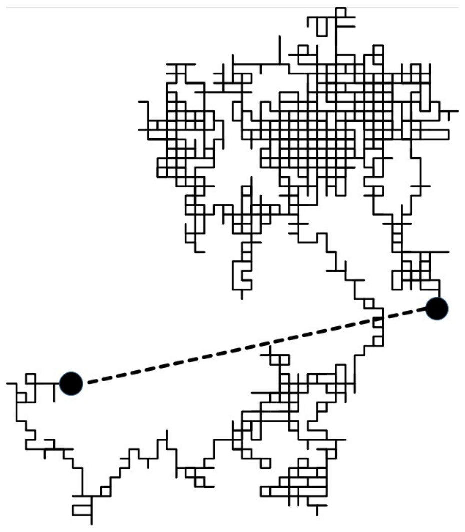

Figure 5: A graphical depiction of a random two dimensional walk. The radius r is the length of the line between the starting and ending points.

It was proved in [MT10] that this algorithm has n/D expected steps, where D is the size of dependency between the events. So $D = 2^{k-1}/e - 1$. Thus the goal of this proof is to construct a randomized player $\mathbf{p}'$ that simulates the above randomized algorithm. The player $\mathbf{p}'$ first starts out by randomizing the color assignments of each vertex. Thus at each turn in this phase, it selects up to k untouched vertices and gives them random assignments. This takes at most $\lceil n/k \rceil$ steps. However, noting that player $\mathbf{p}'$ only needs to update the colors that are set to 2, this task takes $n/2k$ expected steps. Once this is complete, whenever the environment sends back unsatsified edges, the randomized agent chooses one edge at random and randomly recolors its k vertices. Thus, by [MT10], this has $n/D = n/(2^{k-1}/e - 1)$ expected steps before a valid two-coloring is found and the game halts. So the randomized player runs in $(n/2k + n/(2^{k-1}/e - 1))$ expected steps. Thus by Theorem 2, there is a deterministic player that can find a svalid 2-coloring in less than $(1+\epsilon)(n/2k + n/(2^{k-1}/e - 1))$ turns with complexity

$$\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{I}(\mathbf{q}; \mathcal{H}) - \log \epsilon.$$

□

7 Grid-Walk

We define the following win/no-halt game, entitled Grid-Walk. The player starts at the origin of a two dimension grid with integer coordinates. At the start of the turn, the player chooses a number from $\{1, \ldots, 4\}$ and the the environment maps this number to a direction (North, South, East, West) based on a function of the player's previous actions. The enviroment moves the player in the choseen direction and the next round begins. No messages are sent from the environment to the player. The radius of the player is its Euclidean distance to the origin. The player wins if after N rounds, it has a radius at least $\sqrt{N}$. An example random walk can be seen in figure 5.

Theorem 9 *For large enough N, there is a deterministic agent* $\mathbf{p}$ *that can win against* Grid-Walk *environment* $\mathbf{q}$ *such that* $\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{I}((\mathbf{q}, N); \mathcal{H})$.

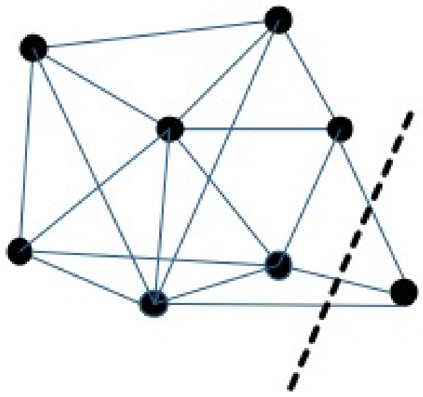

Figure 6: A graphical depiction of a minimum cut. By removing the edges along the dotted line, two components are created.

Proof. We define a probabilistic player $\mathbf{p}'$ that randomly chooses a number from $\{1,2,3,4\}$ with uniform probability. Thus $\mathbf{p}'$ performs a random walk on the two dimensional grid. For large enough N, the probability density function for the radius r of random walker $\mathbf{p}'$ is the Rayleigh density function $P(r) = \frac{2r}{N}e^{-r^2/N}$. The culmulative distribution function is $F(r) = 1 - e^{-r^2/N}$. Thus $c = 1 - F(\sqrt{N}) = 1 - e^{-1}$. Thus with at least constant probability c the radius of $\mathbf{p}'$ is at least $\sqrt{N}$, and $\mathbf{p}'$ wins. So by Theorem 1, there exists a deterministic player $\mathbf{p}$ that wins against $\mathbf{q}$ where

$$\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{K}(\mathbf{p}') - \log c + \mathbf{I}((\mathbf{p}', N, \mathbf{q}); \mathcal{H}) <^{\log} \mathbf{I}((N, \mathbf{q}); \mathcal{H}).$$

□

8 Min-Cut

We define the following win/no-halt game, entitled Min-Cut. The game is defined by an undirected graph G and a mapping ℓ from numbers to edges. At round i, the environment $\mathbf{q}$ sends the number of edges of G. The player responds with a number. The environment maps the number to an edge, and this mapping can be dependent on the player's previous actions. The environment then contracts the graph G along the edge. The game halts when the graph G has contracted into two vertices. The player wins if the cut represented by the contractions is a min cut. A minimum cut of a graph is the minimum number of edges, that when removed from the graph, produces two components. A graphical depiction of a min cut can be seen in Figure 6.

Theorem 10 *There is a deterministic agent* $\mathbf{p}$ *that can win against* Min-Cut *instance* (G, S, ℓ), $|G| = n$, *such that* $\mathbf{K}(\mathbf{p}) <^{\log} 2\log n + \mathbf{I}((G,\ell);\mathcal{H})$.

Proof. We define the following randomized agent $\mathbf{p}'$. At each round, $\mathbf{p}'$ chooses an edge at random. Thus the interactions of $\mathbf{p}'$ and $\mathbf{q}$ represent an implementation of Karger's algorithm. Karger's algorithm has an $\Omega(1/n^2)$ probability of returning a min-cut. Thus $\mathbf{p}'$ has an $\Omega(1/n^2)$ chance of winning. By Theorem 1, there exist a deterministic agent $\mathbf{p}$ and c where $\mathbf{p}$ can beat $\mathbf{q}$ and has complexity $\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{K}(p') - \log c/n^2 + \mathbf{I}(\mathbf{q};\mathcal{H}) <^{\log} 2\log n + \mathbf{I}((G,\ell);\mathcal{H})$. □

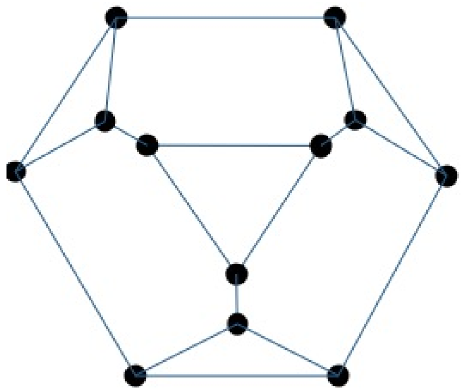

Figure 7: An example vertex-transitive graph.

9 Cover-Time

We define the following interactive penalty game. Let $G = (E, V)$ be a graph consisting of n vertices V and undirected edges E. The environment $\mathbf{q}$ consists of (G, s, ℓ). $G = (E, V)$ is a non-bipartite graph with undirected edges, $s \in V$ is the starting vertex. ℓ is a mapping from numbers to edges to be described later.

The agent starts at $s \in V$. At round 1, the environment gives the agent the degree $s \in V$, $\mathrm{Deg}(s)$. The agent picks a number between 1 and $\mathrm{Deg}(s)$ and sends it to $\mathbf{q}$. The agent moves along the edge the number is mapped to and is given the degree of the next vertex it is on. Each round's mapping of numbers to edges, ℓ, is a computable function of the agent's past actions. The game stops if the agent has visited all vertices and the penalty is the number of turns the agents takes.

Theorem 11 *There is a deterministic agent* $\mathbf{p}$ *that can play against* Cover-Time *instance* (G, S, ℓ), $|G| = n$, *and achieve penalty* $\frac{8}{27}n^3 + o(n^3)$ *and* $\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{I}((G, s, \ell); \mathcal{H})$.

Proof. A probabilistic agent $\mathbf{p}'$ is defined as selecting each edge with equal probability. Thus the agent performs a random walk. The game halts with probability 1. Due to [Fei95], the expected time (i.e. expected penalty) it takes to reach all vertices is $\frac{4}{27}n^3 + o(n^3)$. Thus by Theorem 2 there is a deterministic agent $\mathbf{p}$ that can reach each vertex with a penalty of $\frac{8}{27}n^3 + o(n^3)$ and has complexity

$$\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{K}(\mathbf{p}') + \mathbf{I}((G, s, \ell); \mathcal{H}) <^{\log} \mathbf{I}((G, s, \ell); \mathcal{H}).$$

□

10 Vertex-Transitive-Graph

We describe the following graph based game. The environment $\mathbf{q} = (G, \ell, u, 2k)$ consists of an undirected vertex-transitive graph $G = (V, E)$, a start vertex $u \in V$, the number of rounds $2k$, and a mapping ℓ from numbers to vertices. A vertex-transitive graph $G = (V, E)$ has the property that for any vertices $u, v \in V$, there is an automorphism of G that maps u into v. An example of a vertex transitive graph can be seen in Figure 7. At round 1, the agent starts at vertex $u \in V$ and the environment send to the agent the degree of u. The agent picks a number from 1 to $\mathrm{Deg}(u)$ and the environment moves the agent along the edge specified by the mapping ℓ from numbers to edges. The mapping ℓ can be a function of the agents previous actions. The agent wins if after $2k$ rounds, the agent is back at u.

Theorem 12 *There is a deterministic agent* $\mathbf{p}$ *that can win at the* Vertex-Transitive-Graph *game* $(G = (V, E), \ell, u, k)$*,* $|V| = n$ *with complexity* $\mathbf{K}(\mathbf{p}) <^{\log} \log n + \mathbf{I}((G, \ell, u, k); \mathcal{H})$.

Proof. We define the following randomized agent $\mathbf{p}'$. At each round, after being given the degree d of the current vertex, $\mathbf{p}'$ chooses a number randomly from 1 to d. $\mathbf{K}(\mathbf{p}') = O(1)$. This is equivalent to a random walk on G. Let $P^l(u, v)$ denote the probability that a random walk of length l starting at u ends at v. Then due to [AS04], for vertex-transitive graph G,

$$P^{2k}(u, u) \geq P^{2k}(u, v).$$

So after $2k$ rounds the randomized agent is back at u with probability $P^{2k}(u, u) \geq 1/n$, which lower bounds the winning probability of $\mathbf{p}'$ against $\mathbf{q}$. By Theorem 1, there exists a deterministic agent $\mathbf{p}$ that can beat $\mathbf{q}$ with complexity

$$\mathbf{K}(\mathbf{p}) <^{\log} \mathbf{K}(\mathbf{p}') + n + \mathbf{I}((n, \mathbf{p}', \mathbf{q}); \mathcal{H}) <^{\log} n + \mathbf{I}((G, \ell, u, k); \mathcal{H}).$$

□

References

[AS04] N. Alon and J. Spencer. *The Probabilistic Method.* Wiley, New York, 2004.

[Eps22] S. Epstein. The outlier theorem revisited. *CoRR*, abs/2203.08733, 2022.

[Fei95] U Feige. A tight upper bound on the cover time for random walks on graphs. *Random Struct. Algorithms*, 6(1):51–54, 1995.

[MT10] R. Moser and G. Tardos. A Constructive Proof of the General LováSz Local Lemma. *J. ACM*, 57(2), 2010.