Learning Complexity of Simulated Annealing

Anonymous Author
Anonymous Institution

Abstract

Simulated annealing is an effective and general means of optimization. It is in fact inspired by metallurgy, where the temperature of a material determines its behavior in thermodynamics. Likewise, in simulated annealing, the actions that the algorithm takes depend entirely on the value of a variable which captures the notion of temperature. Typically, simulated annealing starts with a high temperature, which makes the algorithm pretty unpredictable, and gradually cools the temperature down to become more stable.

A key component that plays a crucial role in the performance of simulated annealing is the criteria under which the temperature changes, namely, *the cooling schedule*. Motivated by this, we study the following question in this work: "*Given enough samples to the instances of a specific class of optimization problems, can we design near-optimal cooling schedules that minimize the runtime or maximize the success rate of the algorithm on average when the underlying problem is drawn uniformly at random from the same class?*"

We provide positive results both in terms of *sample complexity* and *simulation complexity* [1]. For sample complexity, we show that $\widetilde{O}(\sqrt{m})$ samples suffice to find an approximately optimal cooling schedule of length m. We complement this result by giving a lower bound of $\widetilde{\Omega}(m^{1/3})$ on the sample complexity of any learning algorithm that provides an almost optimal cooling schedule. These results are general and rely on no assumption. For simulation complexity, however, we make additional assumptions to measure the success rate of an algorithm. To this end, we introduce *the monotone stationary graph* that models the performance of simulated annealing. Based on this model, we present polynomial time algorithms with provable guarantees for the learning problem.

1 Introduction

The goal of this work is to better understand how we can design efficient simulated annealing (SA) algorithms. Simulated annealing is a well-known heuristic method to tackle hard problems. Term annealing originates from thermodynamics, referring to the way that metals cool and anneal. Instead of the energy of the material, simulated annealing utilizes the objective function of an optimization problem. Surprisingly, the implementation of SA is very simple as it is very similar to hill-climbing. The only difference is that instead of picking the best move in every step, simulated annealing picks a random move. If the selected move improves the quality of the solution, then the move is always accepted. Otherwise, the algorithm makes the move anyway with some probability less than 1. The probability decreases exponentially with the *badness* of the move, which is the amount by which the solution is worsened. This is shown by $\Delta(E)$. One example of the annealing criteria is given below:

$$\mathsf{Pr}[\text{accepting downhill move at step } i] \simeq 1 - e^{\Delta(E)/t_i}.$$

where parameter t_i is the temperature of the algorithm at step i which is used to determine this probability. The t_i parameter is analogous to temperature

[1] We call the overall runtime of the algorithm that determines the cooling schedule the simulation complexity

Preliminary work. Under review by AISTATS 2021. Do not distribute.

Sample complexity		
Upper bound:	$\widetilde{O}(\sqrt{m})$	
Lower bound: (for our discretization approach)	$\widetilde{\Omega}(\sqrt{m})$	
Lower bound: (for any learning algorithm)	$\widetilde{\Omega}(m^{1/3})$	
Simulation complexity		
identical paths	separate paths	separate paths + all-satisfied
exact solution in time $\mathsf{poly}(m, n, \lvert T\rvert)$	exact solution in time $\mathsf{poly}(m, n, \lvert T\rvert^n)$	$(O(\log n \lvert T\rvert), 0)$ approximation in time $\mathsf{poly}(m, n, \lvert T\rvert)$

Table 1: Overview of our results. Here, m denote the length of cooling schedule. In the computational results, T is the set of the discretized temperatures and n is the number of samples used in the learning algorithm.

in an annealing system at time step i. At higher values of temperature, downhill moves are more likely to occur. As the temperature tends to zero, they become more and more unlikely, until the algorithm behaves more or less like hill-climbing. In a typical SA optimization, the temperature starts at a high value and is gradually decreased according to a cooling schedule. Simulated annealing is used for a broad class of computational problems ranging from SAT to travelling salesman problem, to VLSI routing, etc. as experiments strongly support the efficiency of simulated annealing in practice [3, 17, 18, 26], and still achieving empirical success in various applications recently [11, 12, 21, 22, 23, 35].

Indeed the efficiency of an SA algorithm significantly depends on its cooling schedule [1, 4, 18, 19, 20, 27, 28, 30, 32]. One simple cooling schedule is to start with a single temperature t_0 and decrease the temperature linearly with a rate of α to obtain lower temperatures gradually. We use this simple cooling strategy to present illustrating examples, nonetheless we consider a more generalized setting in this work. The literature has also gone beyond simple cooling schedules and several non-linear methods have been proposed so far [1, 4, 18, 19, 20, 27, 28, 30, 32]. It is not hard to imagine that even for different instances of the same problem, the optimal cooling schedules may vary significantly.

In our setting, we focus on a family of problem instances (available via sample access to an unknown distribution) and tend to maximize the average score over all instances. This approach is known as the PAC-style modeling which has been used to analyze a variety of other application-specific algorithms, including branch and bound algorithms, center-based/linkage-based clustering, and combinatorial auctions[5, 8, 9, 14]. We refer the readers to [29] for a more comprehensive review of this area. Generally in algorithm design, we need to incorporate an unknown set of instances, since otherwise for any given instance there is an algorithm that has the solution memorized. Typically one does this worst-case over an instance family, and with this analysis we are at least aiming to bring that closer to the specific instances by performing near-optimally with respect to the actual distribution over instances at hand.

Therefore in this work, we take a learning approach towards designing simulated annealing algorithms, using the PAC-style model for data-driven algorithm design introduced in [14] and used to analyze a wide range of important families of algorithms and heuristics in [5, 7, 8, 9]. In brief, we consider a distribution $\mathcal{D}$ over a specific class of instances of a presumably hard problem (such as SAT) and aim to design near-optimal cooling schedules for such instances, analyzing both sample complexity (the number of instances from $\mathcal{D}$ we need to observe) and simulation complexity (runtime) needed for learning. Our approach is particularly motivated by the work of [9].

2 Prelimnaries

As aforementioned, an SA algorithm makes a random walk on the nodes of a search graph. Each node of this graph represents a potential (not necessarily optimal) solution for the underlying problem and the energy of a node is a value reflecting how close this solution is to an optimal solution. We assume that for each node, its energy and neighbors are available via oracle queries. One thing to keep

in mind is that the number of nodes in this huge search graph may be exponentially large and that we only have local views on the nodes of the graph. For instance, when the underlying problem is SAT, we may have 2^k nodes where k is the number of variables in the SAT problem and each node represent an assignment of true/false to the variables.

Crucial to any cooling schedule are the parameters that maximize its performance. This could be as simple as just a real value specifying the cooling rate or as complicated as a sequence of variables determining the exact value of the temperature at every step. Take for instance, the simplest case in which a parameter t_0 and linear cooling rate α formulate the temperature at every step. In this case, at step i, $t_i = t_0(1 - \alpha i)$ formulates the temperature. Therefore, the learning algorithm has to find the optimal pair (t_0, α) that maximizes efficiency. It is an easy exercise to see that the learning problem is actually not very challenging in this case. Although this simple formulation involves infinitely many (t_0, α) pairs that need to be searched over, via careful discretization techniques, one can narrow down the set of possible (t_0, α) pairs to polynomially many candidates and iterate over them to find the optimal cooling schedule[2]. Samples are used to determine how well each cooling schedule performs in practice. More precisely, samples are used to approximate the score of a cooling schedule.

However, we go beyond linear cooling schedules and include more sophisticated systems (*i.e.,* non-linear cooling schedules). Our setting is pretty general: we denote the cooling schedule by a vector $\mathcal{E} = \langle t_1, t_2, \ldots, t_m \rangle$ where m is the number of steps our algorithm takes and t_i specifies the temperature at time i. Any non-increasing sequence of values makes a valid cooling schedule. The problem becomes more challenging with this representation; Even after discretizing the temperatures, still there are exponentially many cooling schedules and determining an approximately optimal schedule is non-trivial.

Recall that each node of the search graph corresponds to a potential solution for the underlying problem. In case of SAT for instance, each node can be an assignment of the true/false values to the variables. We label a subset of nodes in the search graph as *acceptable solution nodes*. These nodes correspond to solutions that are acceptable for the underlying problem. In the case of SAT, a node whose corresponding solution satisfies all of the clauses is a solution node. The score of an SA algorithm with a specific cooling schedule is the likelihood of reaching an acceptable solution node after a fixed number of steps. We would like to point out that although a reasonable energy function for the nodes of the search graph gives higher energies to the acceptable solution nodes, we make no particular assumption on the energies in our setting. We remark that if the cooling schedule is available, it is computationally easy to evaluate the score of the algorithm. We run the SA algorithm according to the cooling schedule and once it terminates we find out if the solution found by the algorithm is acceptable for the problem. By repeating this process enough times, we can estimate the score of the cooling schedule very accurately.

Indeed the optimal parameters may vary for different problems or even for different instances of the same problem and therefore we need to also incorporate the problem instances in our setting. To illustrate the importance of the cooling schedule, consider the example shown in Figure 2. In the example shown in Figure 2, there is one solution node (colored in red) which has an energy of $3n$ and the search graph consists of a clique of vertices with distinct energies one of whose vertices has a path to the solution node. The energies of the nodes of this path are increasing. Let us assume that the initial state of the algorithm is the node colored in green. It is easy to verify that an extreme strategy that never accepts downhill moves has zero chance of reaching the solution node and another extreme strategy that always accepts all downhill moves requires a cubic number of steps to reach the solution node. However, a strategy that accepts each downhill move with probability 1/2 only requires $\mathcal{O}(n^2)$ steps in expectation to reach the solution node.

Motivated by this example, we define our general problem in the following way:

Problem Let $\mathcal{D}$ be a distribution over a specific class of instances[3] of a hard problem (such as SAT). Denote the set of valid (combination of) parameters for the SA algorithm by $\mathcal{F} = \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \}$. Moreover, let for an instance $\mathsf{I} \sim \mathcal{D}$ and a set of parameters $\mathcal{E} \in \mathcal{F}$, $\mathsf{score}(\mathsf{I}, \mathcal{E})$ be a function that reflects how well an SA algorithm with parameters $\mathcal{E}$ works on instance I. This is basically the likelihood of finding a solution, if our SA algorithm uses $\mathcal{E}$ as its cooling schedule. Our goal is to find a set of parameters

[2]This improvement comes with a small error to the quality of the solution.

[3]For instance industrial instances of SAT.

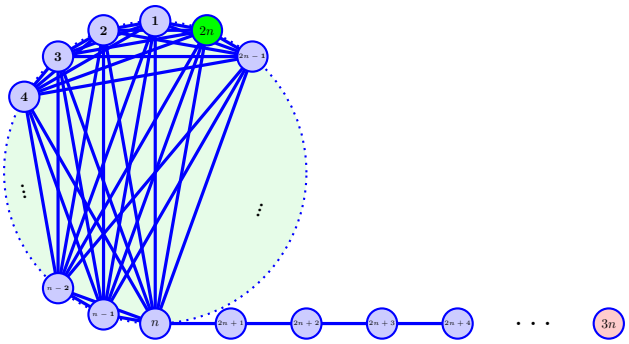


Figure 1: A search graph with $3n$ nodes is illustrated in this figure. The numbers on the nodes show their energy or in other words goodness of the nodes. In this example, we consider the red node to be the only solution of the problem.

$\mathcal{E} \in \mathcal{F}$ that maximizes efficiency. In other words,

$$\mathbb{E}_{\mathsf{I}\sim\mathcal{D}}[\mathsf{score}(\mathsf{I}, \mathcal{E})]$$

is (approximately) maximized. We assume throughout this paper that the scores improve as energy *increases*. Also, a downhill move is a move which hurts the energy of a node and thus is accepted with some probability smaller than 1. However, whenever the score of a node does not hurt in a move, such a move is always made.

We clarify the notation by a simple example. Let us go back to the basic setting in which we formulate the temperature at each step with a pair (t_0, α). In this case, $\mathcal{F} = \mathbb{R}^+ \times (0, 1/m)$ would be the set of all valid parameters. Moreover, a natural example for score is the probability of finding a correct solution after a given (say m) number of steps. This way, the problem is to find a temperature t_0 and a cooling rate α that maximize the success probability after performing m moves of SA. Our attention in this work is focused on an even more general setting. We denote the cooling schedule by a sequence of non-increasing temperatures $\langle t_1, t_2, \ldots, t_m\rangle$ for a fixed m. Thus, in our setting we have $\mathcal{F} \subseteq (\mathbb{R}^+)^m$ subject to the temperatures being non-increasing. For simplicity, and without loss of generality, we assume that the energies of the nodes are integer numbers in range $\{1, 2, \ldots, \mathsf{e}_{\max}\}$.

Any SA algorithm basically makes a random walk on a search graph in which every node represents a potential/partial solution for the problem. For instance, when the underlying problem is SAT, every node of the search graph is a true/false assignment to the variables of the program. The energy of each node is a local guess on how well that solution satisfies the goals of the problem. For the case of SAT for instance, one simple energy function for a node is the number of clauses the corresponding solution satisfies. In our setting, we make no assumption on the energy of the nodes though in practice we expect that a higher energy signals a better solution. Since an SA algorithm makes a random walk, its state at every step can be shown via a distribution over the nodes of the search graph. Initially, this distribution shows the likelihood of each node being used as the starting solution and as the algorithm proceeds, the distribution changes based on the criteria of the random walk. The final state of the algorithm represents the likelihood of each node reported as the final solution. Thus, we wish the final distribution of our algorithm to be highly concentrated on the solution nodes.

We evaluate our learning algorithm based on two quantities: *sample complexity* and *simulation complexity*[4]. The former measures the number of samples one needs in order to find an (approximately) optimal cooling schedule and the latter measures the runtime of the learning algorithm in order to find an optimal cooling schedule.

Our main results are concerned with the sample complexity of the learning problem. As a typical challenge for learning problems, we have to face the issue that the space of the problem is infinitely large as there are infinitely many cooling schedules for an SA algorithm. In order to prove a bound on the sample complexity, the first step is to show that by losing a small additive error, we can bound the space of the solutions to a finite set. We begin by explaining this in Section 3. We also propose a computational model for evaluating the simulation complexity of the problem and design efficient algorithms with theoreteical guarantees in Section 4.

3 Sample Complexity

In this section, we give an analysis for the sample complexity of the problem. Recall that, for any problem instance I and any sequence of m temperatures $\mathcal{E} = \langle t_1, t_2, \ldots, t_m\rangle$, we define $\mathsf{score}(\mathsf{I}, \mathcal{E})$ to be the probability of finding an acceptable solution of I using temperatures in $\mathcal{E}$. We say $\mathcal{E}$ is ε-approximately optimal, if $\mathbb{E}_{\mathsf{I}\sim\mathcal{D}}\mathsf{score}(\mathsf{I}, \mathcal{E}) \geq \sup_{\mathcal{E}'} \mathbb{E}_{\mathsf{I}\sim\mathcal{D}}\mathsf{score}(\mathsf{I}, \mathcal{E}') - \varepsilon$. That is, no other cool-

[4]This is equivalent to the notion of running time if we assume that our SA procedure halts after a polynomial number of steps.

ing schedule of the same length can achieve a significantly higher success rate. Our goal is to provide upper and lower bounds on the number of i.i.d. samples from $\mathcal{D}$ required for learning an ε-approximately optimal cooling schedule.

3.1 Warm-Up: A Crude Upper Bound

We first show that although the space of the problem is infinitely large, only a polynomial number of samples suffice to approximate the optimal solution within desirable guarantees. This step is quite classic as discretization is the typical approach to bound the solution set. One of the difficulties in finding near-optimal cooling schemes is that there are infinitely many options available. We show that by discretizing the temperatures into $\widetilde{O}(m/\varepsilon)$ different values, we only lose an additive error of ε in the success rate when running the algorithm on any instance of the problem. Note that, we are not making any assumptions yet: we only rely on the fact that the algorithm is evaluated based on the success rate. Discretizing the temperature makes designing efficient algorithms possible too as we will show in Section 4. Our main result is an upper bound of $\widetilde{O}(\sqrt{m})$ for the sample complexity which is explained in details later in Section 3.3. Here we start as a warm-up by giving an upper bound of $\widetilde{O}(m)$.

Theorem 3.1. *The sample complexity of computing an ε-approximately optimal cooling schedule with length m is bounded by $O\left(\varepsilon^{-2}\left(m\log(\frac{m\mathsf{e}_{max}}{\varepsilon})\right)\right)$.*

Roughly speaking, the total number of samples we need in order to approximate the optimal cooling schedule is logarithmic in terms of the number of *candidate solutions* we have. Initially, the space of cooling schedules is infinitely large, however, a discretization technique can reduce the space of candidate solutions to $2^{\widetilde{O}(m)}$ many. More precisely, we define a discretized temperature set T whose size is $\widetilde{O}(m)$ and show that there is an almost optimal solution that only uses the temperatures in T. This reduces the space of candidate solutions to $(O(m))^m \leq 2^{\widetilde{O}(m)}$ which implies that the sample complexity is bounded by $\widetilde{O}(m)$.

The only non-trivial part of the above analysis is to show that a discretized set of temperatures with size $\widetilde{O}(m)$ is enough to approximate the optimal cooling schedule within an arbitrarily small additive error. Let us fix an $\epsilon > 0$ and assume that the goal is to construct a discretized set of temperatures T such that there is a cooling schedule that only uses the temperatures of T and its score is at most ϵ smaller than the optimal solution. One convenient way to construct such a set is to make sure for each $t > 0$ there is a $t' \in T$ such that for any $1 \leq x \leq \mathsf{e}_{\max}$ we have $|e^{-x/t} - e^{-x/t'}| \leq \epsilon/m$. Then we can imply that if we replace every temperature t_i of the optimal solution with its corresponding t'_i of the discretized set, each step we make a different decision with probability at most ϵ/m and thus the total error is bounded by ϵ. That is, with probability $1-\epsilon$ our algorithm traverses the exact same path as had we not modified the optimal cooling schedule. It is not hard to prove that such a condition can be met by having $O(m\log \mathsf{e}_{\max})$ elements in $|T|$ which gives us an almost linear bound on the sample complexity.

Up to this point we show that an almost linear number of queries is sufficient for approximating an optimal cooling schedule. This raises two questions: i) Can we improve the bound such that the dependence on m is subpolynomial? In particular, do polylogarithmically many samples suffice for our purpose? ii) If the answer to the first question is negative, can we prove a linear lower bound on the sample complexity? As we show in the following, the answer to both questions is negative!

3.2 A Polynomial Lower Bound

We present a negative answer to the first question above. Although this section gives us a lower bound, our improved upper bound in the next section is actually inspired by this lower bound. The first attempt to prove a lower bound is to understand the limit of the discretization technique explained above. Therefore, we ask the following question: "assuming that our algorithm first constructs a discretized set of temperatures and then seeks to find an optimal solution that only uses the discretized temperatures, how many samples do we need?" Indeed, the answer to this question does not imply a lower bound in general, but it does give us an insight into the problem which leads to a general lower bound.

To answer the above question, we need to understand what is the smallest set T of temperatures that can be used to make a cooling schedule whose score is very close to the optimal solution? The search graph shown in Figure 2 proves that $|T|$ should be at least as large as $\widetilde{\Omega}(\sqrt{m})$, otherwise the guarantee may not hold.

In the search graph of Figure 2, we set $m' = m/100$. For a fixed τ, we set x in a way that $e^{-x/\tau} = 1/2$, that is if the temperature is equal to τ the prob-

ability of making a downhill move is exactly equal to $1/2$[5]. The goal of this search graph is to start the SA algorithm from the initial node and the only acceptable solution node is the final node.

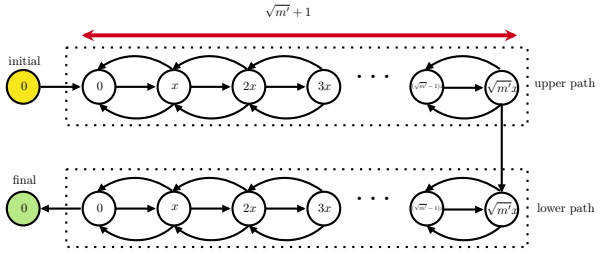


Figure 2: The search graph is depicted for a fixed temperature τ. $x = \tau \ln 0.5$ is chosen in a way that $e^{-x/\tau} = 1/2$ holds.

The search graph of Figure 2 is particularly interesting because of the following observations: i) A cooling schedule of length m only having temperature τ is guaranteed to reach the final node with high probability. ii) A cooling schedule of length m that does not contain any temperature in range $[\tau(1-\widetilde{\Omega}(m^{-1/2})), \tau(1+\widetilde{\Omega}(m^{-1/2}))]$ has very little chance to reach the final node. As a consequence, if the multiplicative distance between two consecutive temperatures in our discretized set is more than $1+\widetilde{\Omega}(m^{-1/2})$, one can delicately design such a search graph for which our discretization performs poorly while the optimal solution gets a score close to 1. This implies that the size of the discretized set has to be at least $\widetilde{\Omega}(\sqrt{m})$ to prove a bound.

While the above argument shows that our specific algorithm definitely needs $\widetilde{O}(\sqrt{m})$ samples[6], it does not give a lower bound for general algorithms beyond our discretization approach. However, we show in Section B with a slightly more advanced analysis that any algorithm requires at least $\widetilde{\Omega}(m^{1/3})$ samples from the distribution in order to guarantee a non-trivial bound. While the heart of the proof is based on the same search graph, in order to extend the observation to all algorithms, we slightly lose on the exponent of m in the lower bound.

Theorem 3.2. *Even if* $\mathsf{e}_{max} = 2^{\widetilde{\Theta}(1)}$*, any learning algorithm requires at least* $\widetilde{\Omega}(m^{1/3})$ *samples from the distribution in order to obtain an additive error less than* 0.5.

Before proceeding to the main result ($\widetilde{O}(\sqrt{m})$ sample complexity upper bound), we would like to note an implication of this result in the context of simulated annealing. There have been several attempts in the literature to understand the complexity of simulated annealing. One question asked in the literature from both theoretical and practical standpoints is if there is a meaningful difference between Simulated Annealing and the Metropolis Algorithm [15, 36]. Metropolis is a special case of simulated annealing where the temperature does not change by time. That is the cooling schedule repeats a single temperature m times. While this observation was made previously, our lower bound also implies that (from a theoretical standpoint) there is a meaningful difference between the two algorithms as Metropolis can be learned with much fewer samples which shows there are cases for which simulated annealing performs much better. Another example is when the temperature drops linearly for which the sample complexity is small. More generally, this lower bound actually shows a gap between SA and any special case of SA whose cooling schedule has complexity smaller than $m^{1/3}$. For instance, it shows that an extended version of Metropolis that uses $m^{1/3-\epsilon}$ many different temperatures in the cooling schedule is not competitive with the general SA algorithm.

3.3 Tightening the Upper Bound

Perhaps the more surprising result of this paper is that the sample complexity can be improved to $\widetilde{O}(\sqrt{m})$:

Theorem 3.3. *The sample complexity of computing an ε-approximately optimal cooling schedule with length m is bounded by $O(\varepsilon^{-2}\sqrt{m}\log(\frac{m\mathsf{e}_{max}}{\varepsilon}))$.*

Our algorithm is almost identical to the one explained in Section 3.1 except that we construct a smaller set T whose size is bounded by $\widetilde{O}(\sqrt{m})$. Then we argue that the total number of cooling schedules with this temprature set is bounded by $2^{\widetilde{O}(\sqrt{m})}$ which leads to sample complexity $\widetilde{O}(\sqrt{m})$.

Below we provide a sketched version of the proof. A complete version can be found in appendix. The first pointer to this result is that there is no clear way to improve the lower bound of Section 3.2. Keep in mind that for the lower bound, we construct a search graph for which a particular cooling schedule works well, but if we multiply (or divide) each temperature by a small factor $1+\widetilde{\Omega}(m^{-1/2})$, the score of the algorithm drops significantly. Obviously, if one comes up

[5]For now, we assume x can be an arbitrary real number but this comes without loss of generality.

[6]See the proof of Theorem 3.3 for more details.

with a better search graph for which a multiplicative factor of $1+\widetilde{\Omega}(m^{-1/2-\epsilon})$ breaks the solution, then it shows that it is impossible to obtain an upper bound of $\widetilde{O}(\sqrt{m})$ with the discretization technique. Failure to make a better bad instance brings us to the possibility that maybe massaging each temperature by a multiplicative factor of $1+\widetilde{O}(m^{-1/2})$ cannot hurt the score of the cooling schedule significantly. We show that this is indeed the case!

Recall that in Section 3.1, in order to prove that the discretized cooling schedules perform almost optimally, we show that there is a discretized cooling schedule that behaves the same as the optimal cooling schedule with probability $1-\epsilon$. That is, in the unlikely event of making a different decision (we call it a *mistake*) we give 0 credit to our discretized cooling schedule, yet we prove that the score is pretty close to that of the optimal. Clearly, this is a loose upper bound as we do not expect to lose too much by making a single mistake.

We illustrate the idea with a toy problem. Consider a complete binary tree of depth m. The root has depth 0 and the leaves have depth m. Each leaf is attributed to a score which is either 0 or 1. The score of each non-leaf node is the average of the scores of its children. In other words, if we make a random walk towards the leaves with equal probability of going to each child, the score of a node is equal to the probability of reaching a leaf with score 1 using the random-walk. Let us call this *even-random-walk* and consider a different type of random-walk, namely *uneven-random-walk*. The uneven-random-walk is pretty much the same as the even random walk, except that at some depth i uniformly drawn from $[1, m]$, an adversary may change the decision of which child to go to. The toy-problem is to understand how much the score of a node hurts by replacing even-random-walk by uneven-random-walk.

To study this, we attribute to each node a *deviation value* which is equal to the absolute value of the difference between the scores of it children. This roughly captures an upper bound of the score we lose, if we traverse the edges of that node with a different criteria (other than $1/2, 1/2$). Thus, we need to know what is the average deviation values of the nodes in an even-random-walk? This roughly tells us how much we lose in the score, if an adversary changes the criteria of the walk at some random point!

The upper bound on the answer is $O(1/\sqrt{m})$ no matter how the leaves are scored. It goes beyond the scope of this paper, but we mention the idea in the hope that it helps a mindful reader decipher some of steps that we take in the proof of Lemma C.1. Define a deviation function $f(x) : [0,1] \to [0, 0.25] = x - x^2$. One can show by induction that starting from each node v of depth i, the average sum of deviations in a random walk is bounded by $O((f(s_v) + (m-i)/m)\sqrt{m})$ where s_v is the score of node v (obtained via even-random-walk).

The toy problem illustrates that in the event that our optimal solution makes decisions with probability $1/2, 1/2$ (which is indeed the case for our lower bound), we can afford to make $O(\sqrt{m})$ mistakes and not lose much in the average score. This does not hold if the decisions are made with different probabilities. To see this, consider the case that only the rightmost leaf has a score 1 and the rest of the leaves have scores 0. Moreover, the probability of going to the right child in the random walk is $1-\epsilon/m$ and the probability of going to the left child is ϵ/m. In this case, the average deviation is $\Omega(1)$ when we start from the root and make a random walk according to the probabilities.

The next observation is that when the decisions are not necessarily $1/2, 1/2$ say $p, 1-p$, multiplying the temperature by a factor of $1+x$ changes the probabilities by at most $\max\{\ln 1/p, 1\} \min\{p, 1-p\}x$ (see Observation C.5). That is, as the probabilities deviate from $1/2$, the probability of making a "mistake" drops linearly. More precisely, the multiplicative term $\min\{p, 1-p\}$ gives us extra power to deal with these situations. For instance, if $p < 1/\sqrt{m}$ or $p > 1 - 1/\sqrt{m}$ then the probabilities change by an additive error of $\widetilde{O}(1/m)$ when we multiply the temperature by a factor of $1+\widetilde{O}(m^{-1/2})$. This error is tolerable since we can afford to have an error of ϵ/m for each decision we make.

The proof is based on the above ideas but the analysis is quite involved and rather cryptic by nature. We show in Section C that if the temperatures in the discretized set are at most $1+\widetilde{O}(m^{-1/2})$ away from each other (multiplicative), then one can make a cooling schedule by the discretized temperatures whose score is arbitrarily close to that of the optimal solution. This then can be used to obtain an upper bound of $\widetilde{O}(\sqrt{m})$ on the sample complexity of the problem.

4 Simulation Complexity

The second part of the paper is concerned with the computational aspects of the learning problem. Although we prove that the sample complexity is poly-

nomial without any assumptions, it seems that extra assumptions are necessary for the runtime concerns. Notice that we make no assumption on the underlying problem and the only information available to us when we sample an instance of the problem is a huge search graph containing exponentially many vertices. Even if we bring the underling problem into the setting, it is not clear how we can make use of the conditions of a problem such as SAT to find the right cooling schedule. Keep in mind that the complexity of the underlying problem is the reason we use simulated annealing in the first place. Therefore, we introduce a stylized model to make the problem more tractable. We call our model *the monotone stationary graph*. Although the model relies on extra assumptions, it features nice properties that make it particularly suitable for our purpose.

First, it gives a compact representation for every instance of the problem. Up to this point, we treated each problem instance as a huge search graph with exponentially many vertices which is too big to store in the memory let alone optimizing the solution over it. Our model represents the search graphs in a more efficient way. Next, notice that even if we fix a well-defined representation for a search graph, one should be able to recover the new representation of a problem instance without spending too much time (and of course without taking a complete look at the already exponentially large search graph). Our model makes it possible to recover the stationary graph in polynomial time. Finally, the any model used for our problem has to give us enough structure so that finding an approximately optimal cooling schedule becomes polynomially tractable in the new setting. This is the most important feature of our model.

In our model, we represent each instance of the problem as a graph. Vertices of this graph correspond to the temperatures in our discretized set. Intuitively, for a temperature $t \in T$, its corresponding vertex in the graph represent the state of an SA algorithm that runs infinitely many steps with temperature t. Thus, when the state of our algorithm is close to such a stationary distribution, we assume that our algorithm is pointing at the corresponding vertex in the monotone stationary graph. We draw edges between the vertices to specify how many steps we need to take in the SA algorithm to move between the stationary distributions. Since in our model, the state of an algorithm can be approximated with a node in this graph, we can also determine its score by examining the corresponding stationary distribution.

Therefore, given n instances of the underlying problem, our goal is to find a cooling schedule that obtains the highest average score for these instances by our model. We consider the following three settings and provide a solution for each one of them:

(i). identical-paths: in this setting, we assume that the optimal cooling schedule traverses the same path for all n instances.

(ii). separate-paths: in this setting, we allow the optimal solution to use different paths for different instances.

(iii). separate-paths + all-satisfied: This is a special case of the second setting where we know that there exists a cooling schedule of length m that is optimal for all instances and brings us to the last node for each monotone stationary graph.

We refer the readers to Section D for the details of the computational model. To obtain polynomial time solutions, we introduce the notion of an (α, ϵ)-approximate cooling schedule. In such a solution we allow the cooling schedule to violate the size constraint by a factor of α with the promise that its score is no more than ϵ smaller than the score of the optimal cooling schedule of length m. With this notation, we present computational results below (also summarized in Table 1):

(i). identical-paths: This is the simplest one among the three settings - we achieve an *exact* algorithm which runs in $\mathsf{poly}(m, n, |T|)$ and maximizes the average score for the n instance.

(ii). separate-paths: in this setting, we again achieve an *exact* algorithm which runs in $\mathsf{poly}(m, |T|)$ for any fixed n. However, the runtime is exponential in n.

(iii). separate-paths + all-satisfied: To overcome the exponential dependency in n, we design an efficient approximation algorithm which runs in $\mathsf{poly}(m, n, |T|)$, but achieve an $O(\log(n|T|), 0)$ approximation instead of exact solution in the previous settings. The algorithm is based on LP relaxation of an integer program.

Due to the space limitations, we refer the readers to Section E in the appendix for the details of the algorithm and analysis.

5 Conclusion

In this paper, we proposed a PAC-Learning framework for estimating a near optimal cooling sched-

ule in simulated annealing. We provided non-trivial upper and lower bounds on the sample complexity. Our techniques may also be relevant to other problems with a random walk on search space. We also introduced the monotone stationary graph model for which we are able to find a near-optimal cooling schedule in polynomial time based on rounding linear programs.

We conclude with two open questions due to the limitations of this work. First, finding near optimal cooling schedules in poly-time (beyond our monotone stationary graph model) is an important question in both theory and practice. Our statistical results showed that the sample complexity of learning a good schedule is polynomial in m, however, the analysis is based on a discretization over the search space which has a size exponential in m and it's unclear how to design an efficient algorithm based on the large discretization. This is, in fact, a common issue in the data-driven algorithm design community: the sample complexity scales logarithmically with the search space, while the search space may be exponentially large. See e.g. Page 5 of [6], and Theorem 17 of [9] where their sample complexity is $O(n)$, but computational complexity is $\Omega(n^2 3^{2n})$. In certain simpler settings, it's possible to polynomially upper bound the search space and get efficient algorithms, e.g. it was shown in [6] that learning a single hyperparameter in linkage-based clustering can be solved in $O(n^8)$, but even in their setting the algorithm is based on enumeration. Since simulated annealing is a very complicated problem and it's unlikely to have efficient learning algorithm for the general setting, we seek to find approximation algorithms under additional assumptions and restrictions in the monotone stationary graph model. Designing efficient algorithms beyond this model is definitely an important future direction, and perhaps of interest broadly to the data-driven algorithm design community as well.

Second, designing a better energy function for simulated annealing remains an open question. Throughout the paper, we assumed that the energy function is given apriori, and our goal is to learn a cooling schedule using the given energy function so that the score of learned schedule is at least $\text{OPT} - \varepsilon$. While the sample complexity bounds derived in this paper depend on $\mathrm{e}_{\max}$, which varies from different choices of energy function, we believe that the choice of energy function plays a more important role, as it decides the optimal score OPT among all possible cooling schedules $\mathcal{E}$. Consequently, we should expect OPT to be smaller (and empirical performance to be better) when energy function is chosen carefully. Since we are only considering additive approximation to OPT, the value of OPT itself has no effect on the sample complexity, but it is definitely a useful quantity in practice. Thus, finding a better energy function (with a smaller OPT) is an interesting direction for future works.

Acknowledgement We thank the annonymous reviewers for many helpful suggestions for improving the exposition of this work. Part of this work was done when CD and SS were visiting students at TTIC. This work was supported in part by the National Science Foundation under grants CCF-1733556, CCF-1800317, and CCF-1815011.

References

[1] E. Aarts and J. Korst. Simulated annealing and boltzmann machines. 1988.

[2] E. H. Aarts et al. Simulated annealing: Theory and applications. 1987.

[3] C. Aragon, D. Johnson, L. McGeoch, and C. Schevon. Simulated annealing performance studies. In *Workshop on Statistical Physics in Engineering and Biology*, pages 865–892, 1984.

[4] N. Azizi and S. Zolfaghari. Adaptive temperature control for simulated annealing: a comparative study. *Computers & Operations Research*, 31(14):2439–2451, 2004.

[5] M.-F. Balcan, D. DeBlasio, T. Dick, C. Kingsford, T. Sandholm, and E. Vitercik. How much data is sufficient to learn high-performing algorithms? *arXiv preprint arXiv:1908.02894*, 2019.

[6] M.-F. Balcan, T. Dick, and M. Lang. Learning to link. *arXiv preprint arXiv:1907.00533*, 2019.

[7] M.-F. Balcan, T. Dick, and M. Lang. Learning to link. In *ICLR*, 2020.

[8] M.-F. Balcan, T. Dick, T. Sandholm, and E. Vitercik. Learning to branch. In *International Conference on Machine Learning*, pages 353–362, 2018.

[9] M.-F. Balcan, V. Nagarajan, E. Vitercik, and C. White. Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems. In *Conference on Learning Theory*, pages 213–274, 2017.

[10] R. Eglese. Simulated annealing: a tool for operational research. *European journal of operational research*, 46(3):271–281, 1990.

[11] A. E.-S. Ezugwu, A. O. Adewumi, and M. E. Frîncu. Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem. *Expert Systems with Applications*, 77:189–210, 2017.

[12] A. M. Fathollahi-Fard, K. Govindan, M. Hajiaghaei-Keshteli, and A. Ahmadi. A green home health care supply chain: New modified simulated annealing algorithms. *Journal of Cleaner Production*, 240:118200, 2019.

[13] D. A. Freedman et al. On tail probabilities for martingales. *the Annals of Probability*, 3(1):100–118, 1975.

[14] R. Gupta and T. Roughgarden. A PAC approach to application-specific algorithm selection. *SIAM Journal on Computing*, 46(3):992–1017, 2017.

[15] B. Hajek. Cooling schedules for optimal annealing. *Mathematics of operations research*, 13(2):311–329, 1988.

[16] D. Henderson, S. H. Jacobson, and A. W. Johnson. The theory and practice of simulated annealing. In *Handbook of metaheuristics*, pages 287–319. Springer, 2003.

[17] L. Ingber. Simulated annealing: Practice versus theory. *Mathematical and computer modelling*, 18(11):29–57, 1993.

[18] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[19] J. Lam and J.-M. Delosme. An efficient simulated annealing schedule: derivation. *Yale University, New Haven, Connecticut, Technical Report*, 8816, 1988.

[20] J. Lam and J.-M. Delosme. Performance of a new annealing schedule. In *Proceedings of the 25th ACM/IEEE Design Automation Conference*, pages 306–311. IEEE Computer Society Press, 1988.

[21] F. Lu, H. Bi, M. Huang, and S. Duan. Simulated annealing genetic algorithm based schedule risk management of it outsourcing project. *Mathematical Problems in Engineering*, 2017, 2017.

[22] M. M. Mafarja and S. Mirjalili. Hybrid whale optimization algorithm with simulated annealing for feature selection. *Neurocomputing*, 260:302–312, 2017.

[23] J. Matejka and G. Fitzmaurice. Same stats, different graphs: generating datasets with varied appearance and identical statistics through simulated annealing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 1290–1294, 2017.

[24] D. Mitra, F. Romeo, and A. Sangiovanni-Vincentelli. Convergence and finite-time behavior of simulated annealing. *Advances in applied probability*, 18(3):747–771, 1986.

[25] M. Mitzenmacher and E. Upfal. *Probability and computing: randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

[26] M. Nieto-Vesperinas, R. Navarro, and F. Fuentes. Performance of a simulated-annealing algorithm for phase retrieval. *JOSA A*, 5(1):30–38, 1988.

[27] Y. Nourani and B. Andresen. A comparison of simulated annealing cooling strategies. *Journal of Physics A: Mathematical and General*, 31(41):8373, 1998.

[28] J. D. Nulton and P. Salamon. Statistical mechanics of combinatorial optimization. *Physical Review A*, 37(4):1351, 1988.

[29] T. Roughgarden. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2020.

[30] M. Sacco. Stochastic relaxation, gibbs distributions and bayesian restoration of images. 1990.

[31] P. Serafini. Simulated annealing for multi objective optimization problems. In *Multiple criteria decision making*, pages 283–292. Springer, 1994.

[32] E. Triki, Y. Collette, and P. Siarry. A theoretical study on the behavior of simulated annealing leading to a new cooling schedule. *European Journal of Operational Research*, 166(1):77–92, 2005.

[33] J. Tropp et al. Freedman's inequality for matrix martingales. *Electronic Communications in Probability*, 16:262–270, 2011.

[34] P. J. Van Laarhoven, E. H. Aarts, and J. K. Lenstra. Job shop scheduling by simulated annealing. *Operations research*, 40(1):113–125, 1992.

[35] L. Wang, M. Goh, R. Ding, and L. Pretorius. Improved simulated annealing based risk interaction network model for project risk response decisions. *Decision Support Systems*, 122:113062, 2019.

[36] I. Wegener. Simulated annealing beats metropolis in combinatorial optimization. In *International Colloquium on Automata, Languages, and Programming*, pages 589–601. Springer, 2005.

A Proof of Theorem 3.1

Recall that m is the length of the random walk in the search graph. We will show that $\widetilde{O}(m)$ samples from the distribution suffice to approximate the optimal cooling schedule within a small additive error. Notice that $\widetilde{O}$ hides the polylogarithmic factors (both in terms of m and $\mathsf{e}_{\max}$). This basically gives an almost linear upper bound on sample complexity based on the number of steps of the algorithm.

Theorem 3.1, restated: The sample complexity of computing an ε-approximately optimal cooling schedule with length m is bounded by $O\left(\varepsilon^{-2}\left(m\log(\frac{m\mathsf{e}_{\max}}{\varepsilon})\right)\right)$.

Proof. Recall that we assume that the energies of the nodes are in set $\{0, 1, 2, \cdots, \mathsf{e}_{\max}\}$. The proof can be divided into the following steps:

- We start by showing that it is possible to discretize the temperatures to $T = \{d_1, d_2, d_3, \ldots, d_{|T|}\}$, such that for any sequence of m temperatures $\mathcal{E} = \langle t_1, t_2, \ldots, t_m\rangle$, there exists a sequence of m discrete temperatures $\mathcal{E}' = \langle t'_1, t'_2, \ldots, t'_m\rangle \in T^m$, such that

  $$|\mathsf{score}(\mathsf{I}, \mathcal{E}) - \mathsf{score}(\mathsf{I}, \mathcal{E}')| \leq \varepsilon/3$$

  for any instance I of the problem. In other words, the discretized temperatures preserve the score approximately.

- Then, we show the sample complexity of learning an $\varepsilon/3$-approximately optimal temperature $\mathcal{E}_{\mathsf{OPT}}$ in T^m is polynomial. This is achieved by standard concentration results in finite hypothesis space since T^m has only a finite number of cooling schedules.

- Finally, we conclude that $\mathcal{E}_{\mathsf{OPT}}$ is ε-approximately optimal in $\mathbb{R}^m_{\geq 0}$.

Define a parameter δ in a way that $(1-\delta)^m = 1 - \varepsilon/3$, that is, $\delta = \Theta(\frac{\varepsilon}{m})$. We first construct our discretized temperatures as $T = T_1 \cup T_2 \cup \cdots \cup T_{\mathsf{e}_{\max}}$, where

$$T_j = \{\frac{j}{\ln(1/(i\delta))} \quad | \quad \forall \textit{ integer } 1 \leq i \leq \lceil 1/\delta\rceil\}. \tag{1}$$

Roughly speaking, this discretization has the nice property that for any $1 \leq j \leq \mathsf{e}_{\max}$, set $\{e^{j/t} \quad | \quad \forall t \in T_j\}$ evenly divides $[0, 1]$. Therefore, for each temperature t, we can find a nearest neighbor $\widetilde{t}$ in T defined as

$$\widetilde{t} := \arg\min_{t' \in T} |t - t'|,$$

which implies $\left|e^{j/\widetilde{t}} - e^{j/t}\right| \leq \delta$ for any $0 \leq j \leq \mathsf{e}_{\max}$. Notice that the value of $\Delta(E)$ in our SA algorithm is always in range $\{0, 1, \ldots, \mathsf{e}_{\max}\}$ and thus for t and $\widetilde{t}$, $e^{\Delta(E)/t}$ and $e^{\Delta(E)/\widetilde{t}}$ are always within an additive range of δ regardless of the value of $\Delta(E)$. Our key observation is that for any sequence $\mathcal{E} = \langle t_1, t_2, \ldots, t_m\rangle \in \mathbb{R}^m_{\geq 0}$, there exists a sequence of m temperatures $\mathcal{E}' = \langle t'_1, t'_2, \ldots, t'_m\rangle \in T^m$, such that running the simulated annealing algorithms with discrete temperature in $\mathcal{E}'$ keeps the trajectories the same as $\mathcal{E}$ with probability at least $1 - \delta$. To this end, we define $t'_i = \widetilde{t_i}$. Assuming the two runs share the same randomness, then these two runs are the same at each step with probability at least $1 - \delta$. We only need to check the correctness for two cases, when a move is a downhill move or a uphill move.

For an uphill move, the correctness is obvious since both runs accept the move with probability 1. For a downhill move, the accepting probabilities are $e^{\Delta(E)/t_i}$ and $e^{\Delta(E)/t'_i}$, respectively. By choosing $t'_i = \widetilde{t_i}$, the difference is at most

$$|e^{\Delta(E)/t_i} - e^{\Delta(E)/t'_i}| \leq \delta. \tag{2}$$

Therefore, we have proved that for each step, the two runs are the same with probability $1 - \delta$, hence they remain the same at all steps with probability at least $(1 - \delta)^m = 1 - \epsilon/3$. Assuming the score function is

bounded in $[0,1]$, then the scores are different with at most 1 when the two runs are different. Hence, the expectation of difference is upper bounded by

$$|\mathsf{score}(\mathsf{I},\mathcal{E}) - \mathsf{score}(\mathsf{I},\mathcal{E}')| \le 1-(1-\delta)^m = \frac{\varepsilon}{3}$$

Next, we will show that finding a near-optimal cooling schedule in T^m requires polynomial sample complexity. The technique is based on standard Hoeffding and union bounds. We define n as the upper bound on the number of samples and let $\mathsf{I}_1, \mathsf{I}_2, \cdots, \mathsf{I}_n$ be n problem instances sampled i.i.d. from $\mathcal{D}$ and $\mathcal{S}$ be a uniform distribution over $\{\mathsf{I}_1, \mathsf{I}_2, \cdots, \mathsf{I}_n\}$. For a given sequence of temperatures $\mathcal{E} \in T^m$, by Hoeffding's Inequality we have [7]

$$|\mathbb{E}_{\mathsf{I}\sim\mathcal{D}}\mathsf{score}(\mathsf{I},\mathcal{E}) - \mathbb{E}_{\mathsf{I}\sim\mathcal{S}}\mathsf{score}(\mathsf{I},\mathcal{E})| \le \frac{\varepsilon}{3}$$

with probability at least $1-e^{-\frac{2}{9}n\varepsilon^2}$. Therefore, by union bound, this inequality holds for all $\mathcal{E}\in T^m$ with probability at least $1-|T|^m e^{-\frac{2}{9}n\varepsilon^2}$. Since we would like this event to happen with high probability, we wish to give a value to n to make sure

$$1-|T|^m e^{-\frac{2}{9}n\varepsilon^2} \ge 1-m^{-10} \tag{3}$$

Define $\mathcal{E}_{\mathsf{OPT(S)}} = \arg\min_{\mathcal{E}\in T^m} \mathbb{E}_{\mathsf{I}\sim\mathcal{S}}\mathsf{score}(\mathsf{I},\mathcal{E})$ be the empirically best discretized cooling schedule, and $\mathcal{E}_{\mathsf{OPT(D)}} = \arg\min_{\mathcal{E}\in T^m} \mathbb{E}_{\mathsf{I}\sim\mathcal{D}}\mathsf{score}(\mathsf{I},\mathcal{E})$ be the population best discretized cooling schedule. Conditioning on the two events, we have:

$$\begin{aligned}
&\mathbb{E}_{\mathsf{I}\sim\mathcal{D}}\mathsf{score}(\mathsf{I},\mathcal{E}_{\mathsf{OPT(S)}}) - \sup_{\mathcal{E}\in\mathbb{R}^m_{\ge 0}} \mathbb{E}_{\mathsf{I}\sim\mathcal{D}}\mathsf{score}(\mathsf{I},\mathcal{E}) \\
=\ & \mathbb{E}_{\mathsf{I}\sim\mathcal{D}}\mathsf{score}(\mathsf{I},\mathcal{E}_{\mathsf{OPT(S)}}) - \mathbb{E}_{\mathsf{I}\sim\mathcal{S}}\mathsf{score}(\mathsf{I},\mathcal{E}_{\mathsf{OPT(S)}}) \\
&+ \mathbb{E}_{\mathsf{I}\sim\mathcal{S}}\mathsf{score}(\mathsf{I},\mathcal{E}_{\mathsf{OPT(S)}}) - \mathbb{E}_{\mathsf{I}\sim\mathcal{S}}\mathsf{score}(\mathsf{I},\mathcal{E}_{\mathsf{OPT(D)}}) \\
&+ \mathbb{E}_{\mathsf{I}\sim\mathcal{S}}\mathsf{score}(\mathsf{I},\mathcal{E}_{\mathsf{OPT(D)}}) - \mathbb{E}_{\mathsf{I}\sim\mathcal{D}}\mathsf{score}(\mathsf{I},\mathcal{E}_{\mathsf{OPT(D)}}) \\
&+ \mathbb{E}_{\mathsf{I}\sim\mathcal{D}}\mathsf{score}(\mathsf{I},\mathcal{E}_{\mathsf{OPT(D)}}) - \sup_{\mathcal{E}\in\mathbb{R}^m_{\ge 0}} \mathbb{E}_{\mathsf{I}\sim\mathcal{D}}\mathsf{score}(\mathsf{I},\mathcal{E}) \\
\ge & -\frac{\varepsilon}{3} + 0 - \frac{\varepsilon}{3} - \frac{\varepsilon}{3} \ge -\varepsilon
\end{aligned}$$

Hence, we have proved

$$\mathbb{E}_{\mathsf{I}\sim\mathcal{D}}\mathsf{score}(\mathsf{I},\mathcal{E}_{\mathsf{OPT(S)}}) \ge \sup_{\mathcal{E}\in\mathbb{R}^m_{\ge 0}} \mathbb{E}_{\mathsf{I}\sim\mathcal{D}}\mathsf{score}(\mathsf{I},\mathcal{E}) - \varepsilon \tag{4}$$

In other words, $\mathsf{OPT(S)}$ is ε-approximately optimal.

Sample complexity: we need to set n in a way that meets Inequality (3), i.e. $|T|^m e^{-\frac{2}{9}n\varepsilon^2} \le m^{-10}$. Therefore, we have

$$n = \Theta\left(\varepsilon^{-2} m \log(|T|)\right) = \Theta\left(\varepsilon^{-2} m \log(\frac{m\mathsf{e}_{\max}}{\varepsilon})\right).$$

□

The dependence of the above bound on $\mathsf{e}_{\max}$ is logarithmic which is loose when $\mathsf{e}_{\max}$ can obtain exponentially large values. We show that this can be further improved. More precisely, we show this by a more careful construction of T, such that $|T| = \Theta(\frac{m\log(\mathsf{e}_{\max})}{\delta}) = \Theta(\frac{m^2\log(\mathsf{e}_{\max})}{\varepsilon})$. Therefore, we can improve the upper bound on the sample complexity to

$$n = O\left(\varepsilon^{-2} m \log(|T|)\right) = O\left(\varepsilon^{-2} m \log\left(\frac{m\log(\mathsf{e}_{\max})}{\varepsilon}\right)\right).$$

We construct the discretized temperatures as $T = \bigcup_{j\in J} T_j$, where $J = \{1, (1+\delta), (1+\delta)^2, (1+\delta)^3, \cdots, \mathsf{e}_{\max}\}$ and T_j defined as (1). In order to improve the sample complexity, it suffices to show that for each temperature t there is a $\widetilde{t}$ in T such that

$$\left| e^{\Delta(E)/\widetilde{t}} - e^{\Delta(E)/t} \right| \le O(\delta)$$

[7] Hoeffding's Inequality: Let $X_1, X_2, \cdots, X_n \sim_{i.i.d.} P$ and $X_i \in [0,1]$, then $|\frac{1}{n}\sum_{i=1}^n X_i - \mathbb{E}[\frac{1}{n}\sum_{i=1}^n X_i]| \le \varepsilon$ holds with probability at least $1-2e^{-2n\varepsilon^2}$

holds for all and $\Delta(E) \in [0, \mathsf{e}_{\max}]$.

By the definition of J, there always exists a $j^* \in J$, such that $\Delta(E) \leq j^* \leq (1+\delta)\Delta(E)$. Recall that our discretization has the nice property that for any $j \in J$, set $\{e^{j/t} \quad | \quad \forall t \in T_j\}$ evenly divides $[0,1]$. Therefore, there exists a $\widetilde{t} \in T_{j^*}$, such that $|e^{j^*/t} - e^{j^*/\widetilde{t}}| \leq \delta$.

Using Observation C.4, now we can bound the difference $\left|e^{\Delta(E)/\widetilde{t}} - e^{\Delta(E)/t}\right|$:

$$\begin{aligned}
&\left|e^{\Delta(E)/\widetilde{t}} - e^{\Delta(E)/t}\right| \\
\leq &\left|e^{\Delta(E)/\widetilde{t}} - e^{j^*/\widetilde{t}}\right| + \left|e^{j^*/\widetilde{t}} - e^{j^*/t}\right| + \left|e^{j^*/t} - e^{\Delta(E)/t}\right| \\
\leq &O(\frac{j^*}{\Delta(E)} - 1) + \delta + O(\frac{j^*}{\Delta(E)} - 1) \\
= &O(\delta)
\end{aligned}$$

and the rest of the proof remains the same.

Corollary A.1 (of Theorem 3.1)**.** *The sample complexity of computing an ε-approximately optimal cooling schedule with length m is bounded by $O\left(\varepsilon^{-2}\left(m\log(\frac{m\log \mathsf{e}_{max}}{\varepsilon})\right)\right)$.*

B Proof of Theorem 3.2

This section is dedicated to proving a lower bound for the sample complexity of any algorithm. Similar to our upper bound, our lower bound is also very general and without any assumptions. We show that any algorithm that approximates the optimal schedule within a small additive error requires at least $\widetilde{\Omega}(m^{1/3})$ samples from the distribution.

The overall idea of the proof is summarized in the following. We construct $l = |L| = \widetilde{\Omega}(m^{1/3})$ different search graphs $L = \{s_1, s_2, \ldots, s_l\}$. Our construction has a nice property that each search graph requires a certain sequence of temperatures to be present in the cooling schedule in order to find a desirable solution after at most m steps. We refer to such sequences as *keys*. For each search graph, having its key in the cooling schedule guarantees that the search graph is traversed successfully with high probability when we use that cooling schedule. However, the length of each key is smaller than m which allows us to bring multiple keys in an almost optimal solution. The keys are designed in a way that they do not share any elements in common. That is, a temperature used for a key specific to a search graph offers little benefit to the other search graphs. Our distribution $\mathcal{D}$ is a uniform distribution over a subset $L_{\mathcal{D}} \subseteq L$ which contains $\widetilde{\Theta}(l)$ (but much smaller than l) search graphs from L. The crux of the argument is that by knowing $L_{\mathcal{D}}$, one can construct a sequence of size m which includes all the keys of the search graphs in $L_{\mathcal{D}}$ that achieves a score close to 1 on average. However, $L_{\mathcal{D}}$ is unknown to the learner and if we draw fewer than $\widetilde{\Omega}(l)$ samples, there is no hope to get any score more than 0.1. Therefore, any learning scheme needs at least $\widetilde{\Omega}(l) = \widetilde{\Omega}(m^{1/3})$ samples from the distribution to report an approximate solution.

Let $c = 100$ be a large constant. Recall that m is the length of the optimal solution. We define a parameter $m' = \widetilde{\Theta}(m^{2/3})$ which determines both the width of each gadget and the size of the key for each gadget. More precisely, we set the width of each gadget to $\sqrt{m'}$ and the size of the key for each gadget to $2cm'$. Let us first explain how each gadget is constructed and then show how the gadgets can be used to prove a lower bound on the sample complexity.

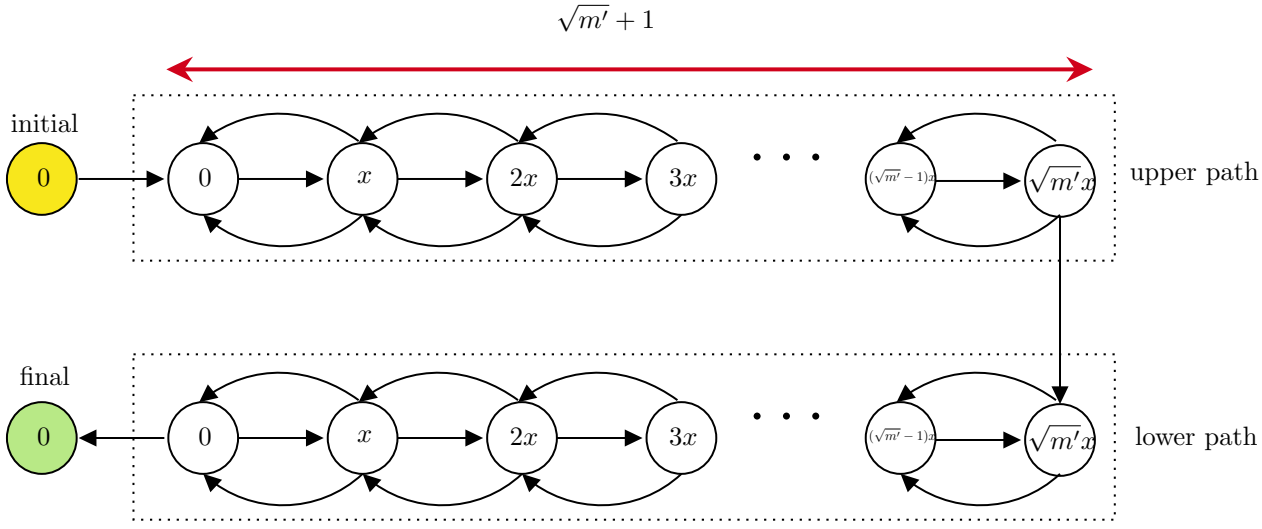


Figure 3: The search graph is depicted for a fixed temperature τ. $x = \tau \ln 0.5$ is chosen in a way that $e^{-x/\tau} = 1/2$ holds.

Each gadget is made for a specific temperature. We fix the temperature to be τ and construct the corresponding gadget, namely $\mathcal{G}(\tau)$ in the following way: As shown in Figure 3, our gadget is constructed of two identical paths. In the upper path, the first node has an energy of 0 and has an outgoing edge to the second vertex. For the next $\sqrt{m'} - 2$ vertices, vertex $i + 1$ has an energy of ix and three outgoing edges: 1) two edges to vertex i and one edge to vertex $i + 2$. Finally, the last node has an energy of $\sqrt{m'}x$ and has two outgoing edges to vertex $\sqrt{m'}$. $x = (\ln 1/2)\tau$ is set in a way that when the temperature is equal to τ the

probability of accepting a downhill move is exactly equal to 1/2.

The lower path is constructed exactly the same way as the upper path. To connect the two paths together, we put an edge from the last node of the upper path to the last node of the lower path. Finally we add two dummy nodes to the search graph. The first dummy node has a single outgoing edge to the first vertex of the upper path and the second dummy node has a single incoming edge from the first node of the lower path. The goal of the gadget is to start from the first dummy node and reach the second dummy node. We call the first and the second dummy nodes the *initial* and *final* nodes respectively.

We define the key $\mathcal{K}(\tau)$ to be a sequence of size $2cm'$ only containing temperature τ. As shown in Lemma B.1, starting from an arbitrary node of $\mathcal{G}(\tau)$ and running the SA algorithm on cooling schedule $\mathcal{K}(\tau)$ our algorithm ends at the final node with probability at least 0.9.

Before bringing the proof, we state an observation for which we provide a proof in the appendix.

Observation B.1. *Let $c = 100$, $x_0 = 0$ and $x_1, x_2, \ldots, x_k$ be k variables constructed in the following way:*

$$\begin{cases} x_{i-1} - 1 & \text{with probability } p_b, \\ x_{i-1} & \text{with probability } p_s, \\ x_{i-1} + 1 & \text{with probability } p_f. \end{cases}$$

Then we have:

(i). For $p_b = p_s = p_f = 1/3$ we have $\max\{x_i\} \geq \sqrt{k/c} + 2$ with probability at least 0.95.

(ii). For $p_b = p_s = p_f = 1/3$ we have $\max\{x_i\} < \sqrt{ck}\log k$ with probability at least $1 - 1/k^2$.

(iii). For any $k' \leq k$, $p_b \geq 1/3 + \frac{c\log k'}{\sqrt{k'}}$, $p_f \leq 1/3 - \frac{c\log k'}{\sqrt{k'}}$, $p_s = 1 - p_b - p_f$ we have $\max\{x_i\} < \sqrt{k'}/2$ with probability at least $1 - 1/k'^2$.

Lemma B.1. *An SA algorithm that starts from any node of $\mathcal{G}(\tau)$ and runs on cooling schedule $\mathcal{K}(\tau)$ ends at the final node with probability at least 0.9.*

Proof. We prove the lemma for an SA algorithm that starts from the initial node. Indeed this implies the lemma for any other starting node since in order to reach the final node, one needs to traverse all nodes of the search graph starting from the initial node.

To this end, we show that after cm' steps our SA algorithm reaches the last node of the lower-path with probability at least 0.95. With a similar analysis, one can show that starting from the last node of the lower-path, after cm' steps our algorithm reaches the final node with probability at least 0.95 after cm' steps. Then, by applying the union bound, we imply that after $2cm'$ steps, our algorithm reaches the final node with probability at least 0.9.

From here on, our aim is to prove that starting from the initial node, our algorithm reaches the last node of the lower-path with probability at least 0.95 after cm' steps. Notice that since the temperature is always equal to τ, in every step, our node in the search graph gets closer to the destination with probability at least 1/3 and get farther from the destination with probability at most 1/3. Due to Observation B.1 (item (i)) after cm' steps, with probability at least 0.95 at some point the number of times we go forward is at least $\sqrt{m'} + 2$ more than the number of times we go backward which means we reach the last node of the lower-path. This implies that with probability at least 0.95 our algorithm reaches the last node of the lower-path after cm' steps. A similar analysis proves that the next cm' steps take us to the final node with probability at least 0.9 which implies that $2cm'$ steps suffices to reach the final node with probability at least 0.9. □

We also show that any cooling schedule needs a certain amount of temperatures close to τ to reach the final node with a considerable probability.

Lemma B.2. *Let $\mathcal{E}$ be a cooling schedule of length m containing no more than $\frac{m'}{4c\log^2 m'}$ temperatures in range $[\tau\frac{\sqrt{m'}-c^2\log m'}{\sqrt{m'}}, \tau\frac{\sqrt{m'}+c^2\log m'}{\sqrt{m'}}]$. If an SA algorithm starts from the initial node and runs with cooling schedule $\mathcal{E}$, the probability that it reaches the final node is at most 0.1.*

Proof. The intuition behind the proof is the following: For the upper-path, we would like to go to the right and thus a low temperature is desirable. For the lower-path however, since we would like to go to the left, we would like the temperature to be as high as possible. The key point is that in the cooling schedule, the temperatures are decreasing, thus either all the temperatures we use for traversing the upper-path are at least τ or all of the temperatures we use for traversing the lower-path are bounded by τ. Any one of the two events makes it unlikely to get a high score.

We assume w.l.o.g that we would like to traverse the upper-path with temperatures higher than τ. Notice however that except for $\frac{m'}{4c\log^2 m'}$ temperatures, all the rest are more than τ by a multiplicative factor of $\frac{\sqrt{m'}+c^2\log m'}{\sqrt{m'}}$. Since we strictly favor lower temperatures, the most desirable cooling schedule in this case is a sequence of $m-\frac{m'}{4c\log^2 m'}$ temperatures $\tau\frac{\sqrt{m'}+c^2\log m'}{\sqrt{m'}}$ followed by $\frac{m'}{4c\log^2 m'}$ temperatures τ. We show that it is still very unlikely to traverse the upper-path using this sequence.

To keep the analysis simple, we avoid the edge cases and assume that the goal is to start from the second vertex and never go back to the first vertex. This way, the probability of going forward or going backward only depends on the temperature and does not depend on the current vertex. If the temperature is equal to τ then with probability $p_f = 1/3$ we go forward and with probability $p_b = 1/3$ we go backward. If the temperature is $\tau\frac{\sqrt{m'}+c^2\log m'}{\sqrt{m'}}$ we go backward and with probability at least $p_b \geq 1/3+\frac{c\log m'}{\sqrt{m'}}$ we go forward with probability at most $p_f \leq 1/3-\frac{c\log m'}{\sqrt{m'}}$. Due to Observation B.1, if we proceed $\frac{m'}{4c\log^2 m'}$ steps with temperature τ or m steps with temperature $\tau\frac{\sqrt{m'}+c^2\log m'}{\sqrt{m'}}$, our position does not improve by more that $\sqrt{m'}/2$ with probability at least $1-\widetilde{O}(1)/m'^2$. Thus, in total the amount of improvement is bounded by $\sqrt{m'}$ with probability at least $1-\widetilde{O}(1)/m'^2$.

The above analysis fails when we bring in to the setting the first node of the upper-path since the probability of going to the right at this node is more than other nodes. However, we make the following argument: in order to traverse the upper-path, at some point we reach the second node of the upper-path and never go back. Let us say this happens at step i. Thus, from step i on, we never go backwards and therefore all the probabilities are only a function of the temperature (and not the current node). The downside however, is that there are m different possible choices for i which multiplies the bad event probability by m. However, since we show in the above that increasing the position by an additive term of $\sqrt{m'}$ is not possible with probability $1-\widetilde{O}(1)/m'^2$, we can imply by union bound that starting from any position i, increasing the position by an additive term $\sqrt{m'}$ is not possible with probability at least $1-\widetilde{O}(m)/m'^2 << 0.1$ (for a large enough choice of m) which completes the proof. □

Now we are ready to prove the lower bound using Lemmas B.1 and B.2.

Theorem 3.2, restated. *Any learning algorithm that approximates the solution within an additive error of* 0.5 *needs at least* $\widetilde{\Omega}(m^{1/3})$ *samples from the distribution.*

Proof. As mentioned earlier, we have $l, |L_{\mathcal{D}}| = \widetilde{\theta}(m^{1/3})$ and $m' = \widetilde{\Theta}(m^{2/3})$. To be more precise, we set $l = 40cm^{1/3}\log m$, $m' = m^{2/3}\log m/2c$ and $|L_{\mathcal{D}}| = m^{1/3}/\log m$.

Assume for now that we have l different temperatures $1 \leq \tau_1 < \tau_2 < \ldots < \tau_l$ such that their multiplicative distance is at least $\frac{\sqrt{m'}+10c^2\log m'}{\sqrt{m'}}$.

As outlined earlier, $L_{\mathcal{D}}$ is a uniform distribution over $m^{1/3}/\log m$ search graphs corresponding to temperatures $\tau_1, \tau_2, \ldots, \tau_l$. Each combination has equal probability of forming $L_{\mathcal{D}}$. Distribution $\mathcal{D}$ is a uniform distribution over the search graphs corresponding to the elements of $L_{\mathcal{D}}$. The optimal solution consists of the keys for all the search graphs corresponding to the temperatures of $L_{\mathcal{D}}$. Since the size of the key for each search graph is $2cm' = m^{2/3}\log m$ and $|L_{\mathcal{D}}| = m^{1/3}/\log m$, this makes a cooling schedule of size m. Lemma B.1 implies that the score of such a cooling schedule is at least 0.9 on average.

On the other hand, after drawing fewer than $m^{1/3}/(100\log m)$ samples, we can get a score of 1 for at most a 0.01 fraction of the search graphs of $L_{\mathcal{D}}$ but the average score for the rest of the instances would be smaller than 0.2 by Lemma B.2 (Notice that the gap between the temperatures is large enough). Thus,

$\Omega(m^{1/3}/\log m)$ samples are necessary to obtain an additive error smaller than 0.5.

To construct the temperatures we do the following: We set $x_1 = 1$ and for $1 < i \leq l$ we set $x_i = \lceil x_{i-1} \frac{\sqrt{m'}+10c^2 \log m'}{\sqrt{m'}} + 1 \rceil$. Finally we set $\tau_i = x_i / \ln 2$ to obtain $e^{-x_i/\tau_i} = 0.5$. To make sure all the energies are non-zero, we add 1 to the energy of all nodes in all gadgets. □

C Proof of Theorem 3.3

We show in this section that the bound of Theorem 3.1 can be significantly improved: **Theorem** 3.3, restated. *The sample complexity of computing an ε-approximately optimal cooling schedule with length m is bounded by $O_\epsilon(\sqrt{m}(\log m + \log \mathsf{e}_{max}))$.*

The proof is based on two observations: 1) first we show that the discretized set of temperatures can be made smaller while keeping the additive error small and 2) the proof can be modified to improve the sample complexity using the new discretized set. We first start by explaining the former.

Our discretization is very similar to that of Theorem 3.1 except that in the construction of the temperatures we allow for a multiplicative error of $\widetilde{\Theta}(m^{-1/2})$ instead of $\Theta(1/m)$. This implies that the multiplicative distance between consecutive elements of T is bounded by $1 + \widetilde{\Theta}(m^{-1/2})$ (instead of $1 + \Theta(1/m)$). This obviously leaves us with a smaller set of temperatures which later can be used to improve the sample complexity but the crucial part of the analysis is to show this smaller set suffices to bound the error by a small ϵ. We prove that for any sequence of temperatures $\mathcal{E} = \langle t_1, t_2, \ldots, t_m \rangle$, there exists another sequence $\mathcal{E}' = \langle t'_1, t'_2, \ldots, t'_m \rangle$ such that $t'_i \in T$ for all $1 \leq i \leq m$ and that the scores of $\mathcal{E}$ and $\mathcal{E}'$ are very close for every search graph. Obviously we set t'_i as the largest element of T which is not greater than t_i. Therefore we have $1 \leq t_i/t'_i \leq 1 + \widetilde{O}(m^{-1/2})$.

Let us introduce a *deviation function* $f(x) : [0,1] \to [0, 0.25] = x - x^2$ which plays an important role in the proof of Lemma C.1. The proof of this section is rather mathematical and unintuitive. For more intuition and as to why such a strange function is necessary for the proof we encourage the reader to review Section 1. Before proceeding to the proof of Lemma C.1, we state some properties of function f as auxiliary observations as well as some mathematical inequalities which are used in the proof of the bound. We defer the proofs of these observations to the appendix.

Observation C.1. *Let $x, y \in [0,1]$ be two real values and $0 \leq p \leq 1$ be a multiplicative factor. Then we have:*

$$pf(x) + (1-p)f(y) \leq f(px + (1-p)y) - \min\{p, 1-p\}(x-y)^2.$$

Since $(x-y)^2$ is always non-negative therefore Observation C.1 implies that $pf(x)+(1-p)f(y) \leq f(px+(1-p)y)$ always holds. By recursing on this inequality we can extend it to the case of more than two variables.

Observation C.2 (as a corollary of Observation C.1)**.** *Let $p_1, p_2, \ldots, p_k$ be non-negative probabilities whose total sum is equal to 1 and $x_1, x_2, \ldots, x_k$ be k real values in range $[0,1]$. Then we have:*

$$\sum p_i f(x_i) \leq f(\sum p_i x_i).$$

Also, we show that for two real numbers $0 \leq x, y \leq 1$ we have $|x - y| \geq |f(x) - f(y)|$.

Observation C.3. *For any two real numbers $0 \leq x, y \leq 1$ we have $|x - y| \geq |f(x) - f(y)|$.*

Observation C.4. *For any $0 \leq p \leq 1$ and any $0 \leq x$ we have*

$$p - p^{1+x} \leq x.$$

We also present a slightly modified version of Observation C.4 which provides a better bound for limited p.

Observation C.5. *For any $0 < p < 1$ and any $0 \leq x \leq 1$ we have*

$$p - p^{1+x} \leq \max\{\ln 1/p, 1\} \min\{p, 1-p\}x.$$

Now we are ready to prove Lemma C.1.

Lemma C.1. *Let I be an instance of the underlying problem and $\mathcal{E} = \langle t_1, t_2, \ldots, t_m \rangle$ and $\mathcal{E}' = \langle t'_1, t'_2, \ldots, t'_m \rangle$ be two cooling schedules such that $1 \leq t_i/t'_i \leq 1 + \frac{\epsilon m^{-1/2}}{4 \log m}$ for some $\epsilon > 0$. Then we have*

$$\mathsf{score}(\mathsf{I}, \mathcal{E}') \geq \mathsf{score}(\mathsf{I}, \mathcal{E}) - \epsilon.$$

Proof.

Our proof is based on induction. Define $\mathcal{E}^{+k}$ ($\mathcal{E}'^{+k}$) to be a cooling schedule starting from element $k+1$ of $\mathcal{E}$ ($\mathcal{E}'$) ($\mathcal{E}^{+0} = \mathcal{E}$ and $\mathcal{E}'^{+0} = \mathcal{E}'$). We denote the vertices of the search graph by $u_1, u_2, \ldots$ (their number may be exponentially large) and define $\mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}^{+k})$ as the average score we obtain if we initiate the search on node u_i and run the algorithm using cooling schedule $\mathcal{E}^{+k}$. When $k = m$, then $\mathcal{E}^{+k}$ is empty which means $\mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}^{+k})$ is either equal to 0 or 1 depending on whether u_i is an acceptable solution node in the search graph. A similar notation also holds for $\mathcal{E}'$. Our aim is to prove that for any u_i and k we have $\mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}'^{+k}) \geq \mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}^{+k}) - \epsilon$ which immediately implies $\mathsf{score}(\mathsf{I}, \mathcal{E}') \geq \mathsf{score}(\mathsf{I}, \mathcal{E}) - \epsilon$. However, to use induction, we strengthen the hypothesis. We show that

$$\mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}'^{+k}) \geq \mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}^{+k}) - \epsilon' \left[f(\mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}^{+k})) + \frac{m-k}{m} \right], \tag{5}$$

where $\epsilon' = \epsilon/2$. Notice that since the value of f is always in range $[0, 0.25]$, Inequality (5) is already stronger than what we wish to prove in the end. The base case is when $k = m$ which means the random walk has terminated and that $\mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}'^{+k}) = \mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}^{+k})$. Thus, for a fixed $k < m$, provided that Inequality (5) holds for any vertex u_i and $k' = k+1$, we show Inequality (5) holds for any pair (u_i, k).

Recall that in every step of the SA algorithm, we first randomly draw an outgoing edge of the current node and then decide whether we traverse through that edge or not. Therefore

$$\mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}'^{+k}) = \mathbb{E}_{u_j \sim N(u_i)}[\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}'^{+k})],$$

where $N(u_i)$ denotes the set of neighbors of vertex u_i and $\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}'^{+k})$ is the score of node u_i for the event that the drawn edge is (u_i, u_j).

Let us first fix an edge (u_i, u_j) and introduce an edge variant of Inequality (5), namely Inequality (6) for which we give a proof in the following.

$$\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}'^{+k}) \geq \mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k}) - \epsilon' \left[f(\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k})) + \frac{m-k}{m} \right]. \tag{6}$$

For simplicity of notation, let us define $a = \mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}^{+(k+1)})$ and $b = \mathsf{score}_{u_j}(\mathsf{I}, \mathcal{E}^{+(k+1)})$. Similarly, define $a' = \mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}'^{+(k+1)})$ and $b' = \mathsf{score}_{u_j}(\mathsf{I}, \mathcal{E}'^{+(k+1)})$. If the energy of node u_j is more than the energy of node u_i then the decision is deterministic regardless of the temperature and we have

$$\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k}) = b,$$

and

$$\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}'^{+k}) = b'.$$

This implies that

$$\begin{aligned}
\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k}) - \mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}'^{+k}) &= b - b' \\
&\leq \epsilon' \left[f(b) + \frac{m-k-1}{m} \right] \qquad (7) \\
&= \epsilon' \left[f(\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k})) + \frac{m-k-1}{m} \right] \\
&\leq \epsilon' \left[f(\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k})) + \frac{m-k}{m} \right],
\end{aligned}$$

where Inequality (7) follows from the induction hypothesis. This basically means that

$$\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}'^{+k}) \geq \mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k}) - \epsilon' \left[f(\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k})) + \frac{m-k}{m} \right],$$

which is desired. Thus, it only remains to prove Inequality (6) for the cases that the energy decreases. This is the only case where $\mathcal{E}$ and $\mathcal{E}'$ behave differently. In this case, depending the temperatures t_{k+1} and t'_{k+1}

our SA algorithm moves to node u_j or stays at node u_i. Let p be the probability of rejecting the downhill move to node u_j when the temperature is equal to t_{k+1} and p' the same probability for the case that the temperature is t'_{k+1}. Recall that the acceptance probabilities are equal to $1-e^{-\Delta(E)/t_{k+1}}$ and $1-e^{-\Delta(E)/t'_{k+1}}$ (for $\mathcal{E}$ and $\mathcal{E}'$ respectively) where $\Delta(E)$ is the difference between the energies of nodes u_i and u_j. Thus, $p = e^{-\Delta(E)/t_{k+1}}$ and $p' = e^{-\Delta(E)/t'_{k+1}}$ and since $1 \leq t_{k+1}/t'_{k+1} \leq 1 + \frac{\epsilon m^{-1/2}}{4 \log m}$ then we have

$$p^{1+\frac{\epsilon m^{-1/2}}{4 \log m}} \leq p' \leq p.$$

Note that $\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k})$ and $\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}'^{+k})$ can be formulated as

$$\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k}) = pa + (1-p)b \tag{8}$$

and

$$\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}'^{+k}) = p'a' + (1-p')b' \tag{9}$$

due to the acceptance probabilities. Thus, we have:

$$\begin{aligned}
\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}'^{+k}) =\ & p'a' + (1-p')b' \\
\geq\ & p' \left[a - \epsilon'[f(a) + \frac{m-k-1}{m}]\right] && \text{by induction hypothetis} \\
& + (1-p') \left[b - \epsilon'[f(b) + \frac{m-k-1}{m}]\right] \\
=\ & p' \left[a - \epsilon' f(a)\right] \\
& + (1-p') \left[b - \epsilon' f(b)\right] \\
& - \epsilon' \frac{m-k-1}{m} \\
=\ & p \left[a - \epsilon' f(a)\right] \\
& + (1-p) \left[b - \epsilon' f(b)\right] \\
& - \epsilon' \frac{m-k-1}{m} \\
& - (p-p')([a - \epsilon' f(a)] - [b - \epsilon' f(b)]) \\
\geq\ & p \left[a - \epsilon' f(a)\right] && p \geq p' \\
& + (1-p) \left[b - \epsilon' f(b)\right] \\
& - \epsilon' \frac{m-k-1}{m} \\
& - (p-p')(|a-b| + \epsilon'|f(a) - f(b)|) \\
\geq\ & p \left[a - \epsilon' f(a)\right] && p \geq p' \\
& + (1-p) \left[b - \epsilon' f(b)\right] && \text{and } \epsilon' \leq 1 \\
& - \epsilon' \frac{m-k-1}{m} \\
& - (p-p')(|a-b| + |f(a) - f(b)|) \\
\geq\ & p \left[a - \epsilon' f(a)\right] && p \geq p' \\
& + (1-p) \left[b - \epsilon' f(b)\right] && \text{and } |a-b| \geq |f(a) - f(b)| \\
& - \epsilon' \frac{m-k-1}{m} && \text{(Observation C.3)} \\
& - 2(p-p')|a-b| \\
\geq\ & [pa + (1-p)b] - \epsilon' f([pa + (1-p)b]) && \text{Observation C.1} \\
& + \epsilon' \min\{p, 1-p\}(a-b)^2
\end{aligned}$$

$$
\begin{aligned}
&- \epsilon' \frac{m-k-1}{m} \\
&- 2(p-p')|a-b| \\
= \;& \mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k}) - \epsilon' f(\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k})) && \text{by Equation (8)} \\
&+ \epsilon' \min\{p, 1-p\}(a-b)^2 \\
&- \epsilon' \frac{m-k-1}{m} \\
&- 2(p-p')|a-b| \\
= \;& \mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k}) - \epsilon' \left[f(\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k})) + \frac{m-k}{m} \right] \\
&+ \epsilon' \left[\min\{p, 1-p\}(a-b)^2 + 1/m \right] \\
&- 2(p-p')|a-b|,
\end{aligned}
$$

which is exactly the same as (6) except for additional additive expressions of the last two lines. Thus, to complete the proof of Inequality (6) we need to show

$$
\epsilon' \left[\min\{p, 1-p\}(a-b)^2 + 1/m \right] \geq 2(p-p')|a-b|. \tag{10}
$$

Based on the values of p and $a-b$ we consider the following three cases separately:

(i). $0 \leq |a-b| \leq m^{-1/2}$

(ii). $0 \leq p \leq m^{-1/2}$

(iii). $m^{-1/2} \leq |a-b| \leq 1$ and $m^{-1/2} \leq p \leq 1$

Case (i): $0 \leq |a-b| \leq m^{-1/2}$: By Observation C.4 and the fact that $p^{1+\frac{\epsilon m^{-1/2}}{4 \log m}} \leq p' \leq p$ we can imply $p - p' \leq \frac{\epsilon m^{-1/2}}{4 \log m}$. Therefore the right hand side of Inequality (10) is bounded by

$$
\begin{aligned}
2(p-p')\big|a-b\big| &\leq 2\frac{\epsilon m^{-1/2}}{4 \log m}\big|a-b\big| \\
&= \frac{\epsilon m^{-1/2}}{2 \log m}\big|a-b\big| \\
&\leq \frac{\epsilon m^{-1/2}}{2 \log m} m^{-1/2} && \text{since } |a-b| \leq m^{-1/2} \\
&= \frac{\epsilon}{2m \log m} \\
&= \frac{\epsilon'}{m \log m} \\
&\leq \frac{\epsilon'}{m}.
\end{aligned}
$$

which implies Inequality (10) since the left hand side is at least ϵ'/m.

Case (ii): $0 \le p \le m^{-1/2}$**:** Let us first give a bound on the value of $p - p'$.

$$\begin{aligned}
p - p' \le & p - p^{1+\frac{\epsilon m^{-1/2}}{4 \log m}} \\
\le & \max\{\ln 1/p, 1\} \min\{p, 1-p\} \frac{\epsilon m^{-1/2}}{4 \log m} && \text{by Observation C.5} \qquad (11) \\
\le & (\ln \sqrt{m}) m^{-1/2} \frac{\epsilon m^{-1/2}}{4 \log m} && (11) \text{ is maximized for } p = m^{-1/2} \\
\le & m^{-1/2} \frac{\epsilon m^{-1/2}}{4} && \text{since } \log m \ge \ln \sqrt{m} \\
= & \frac{\epsilon}{4m} \\
= & \frac{\epsilon'}{2m} && \text{since } \epsilon' = \epsilon/2.
\end{aligned}$$

Also, $|a - b|$ is bounded by 1 so the the right hand side is bounded by ϵ'/m. Since the left hand side is at least ϵ'/m then Inequality (10) holds.

Case (iii): $m^{-1/2} \le |a - b| \le 1$ **and** $m^{-1/2} \le p \le 1$**:** In this case, we leverage Observation C.5 to show that

$$\begin{aligned}
p - p' \le & p - p^{1+\frac{\epsilon m^{-1/2}}{4 \log m}} \\
\le & \max\{\ln 1/p, 1\} \min\{p, 1-p\} \frac{\epsilon m^{-1/2}}{4 \log m} && \text{by Observation C.5} \\
\le & (\ln \sqrt{m}) \min\{p, 1-p\} \frac{\epsilon m^{-1/2}}{4 \log m} && \text{since } p \ge m^{-1/2} \\
\le & \min\{p, 1-p\} \frac{\epsilon m^{-1/2}}{4} && \text{since } \log m \ge \ln \sqrt{m} \\
= & \min\{p, 1-p\} \frac{\epsilon' m^{-1/2}}{2} && \text{since } \epsilon' = \epsilon/2.
\end{aligned}$$

Therefore, the right hand side of Inequality (10) can be bounded by

$$\begin{aligned}
2(p - p')|a - b| \le & 2 \min\{p, 1-p\} \frac{\epsilon' m^{-1/2}}{2} |a - b| \\
= & \min\{p, 1-p\} (\epsilon' m^{-1/2}) |a - b| \\
\le & \min\{p, 1-p\} (\epsilon' m^{-1/2}) |a - b| \frac{|a - b|}{m^{-1/2}} && \text{since } |a - b| \ge m^{-1/2} \\
= & \epsilon' \min\{p, 1-p\} |a - b|^2 \\
= & \epsilon' \min\{p, 1-p\} (a - b)^2.
\end{aligned}$$

which proves Inequality (10) since the left hand side is lower bounded by $\epsilon' \min\{p, 1-p\}(a-b)^2$.

So far, we have proven that Inequality (6) holds for every pair of vertices (u_i, u_j). All that remains is to show that Inequality (6) implies Inequality (5). To show this, we point out that by definition we have

$$\mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}'^{+k}) = \mathbb{E}_{u_j \sim N(u_i)}[\mathsf{score}_{u_i, u_j}(\mathsf{I}, \mathcal{E}'^{+k})].$$

By applying Inequality (6) we obtain:

$$\mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}'^{+k}) = \mathbb{E}_{u_j \sim N(u_i)} \left[\mathsf{score}_{u_i, u_j}(\mathsf{I}, \mathcal{E}'^{+k})\right]$$

$$
\begin{aligned}
&\geq \mathbb{E}_{u_j \sim N(u_i)} \left[\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k}) - \epsilon'[f(\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k})) + \frac{m-k}{m}] \right] \\
&= \mathbb{E}_{u_j \sim N(u_i)} \left[\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k}) \right] \\
&\quad - \mathbb{E}_{u_j \sim N(u_i)} \left[\epsilon'[f(\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k})) + \frac{m-k}{m}] \right] \\
&= \mathbb{E}_{u_j \sim N(u_i)} \left[\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k}) \right] \\
&\quad - \epsilon' \mathbb{E}_{u_j \sim N(u_i)} \left[f(\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k})) \right] \\
&\quad - \epsilon' \frac{m-k}{m} \\
&\geq \mathbb{E}_{u_j \sim N(u_i)} \left[\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k}) \right] \\
&\quad - \epsilon' f\left(\mathbb{E}_{u_j \sim N(u_i)} [\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k})] \right) \\
&\quad - \epsilon' \frac{m-k}{m} \\
&= \mathbb{E}_{u_j \sim N(u_i)} \left[\mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k}) - \epsilon' \mathsf{score}_{u_i,u_j}(\mathsf{I}, \mathcal{E}^{+k}) \right] \\
&\quad - \epsilon' \frac{m-k}{m} \\
&= \mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}^{+k})) - \epsilon' \left[f(\mathsf{score}_{u_i}(\mathsf{I}, \mathcal{E}^{+k})) + \frac{m-k}{m} \right] \qquad ,
\end{aligned}
\tag{12}
$$

which implies Inequality (5). Inequality (12) follows from Observation C.2. □

Lemma C.1 suggests that we can have a discretized temperature set T with size $|T| \leq O(\sqrt{m} \log m \log \mathsf{e}_{\max})$ that can make an almost optimal cooling schedule for any search graph. If we naively count the number of possible cooling schedules, then we obtain a bound of $(\sqrt{m} \log m \log \mathsf{e}_{\max})^m$ which gives us the same upper bound as Corollary A.1. However, a better analysis can show that the number of possible cooling schedules limited to the temperatures in T is bounded by

$$
(\sqrt{m} \log m \log \mathsf{e}_{\max})^{\sqrt{m} \log m \log \mathsf{e}_{\max}} \binom{m}{\sqrt{m} \log m \log \mathsf{e}_{\max}}
$$

which gives us a sample complexity of $O_\epsilon(\sqrt{m}(\log m + \log \mathsf{e}_{\max}))$. Since $|T| << m$ and the temperature is non-increasing over time, there are at most $|T|$ steps in which the temperature changes. Thus, there are at most $\binom{m}{|T|}$ choices for the places where the temperature changes. Moreover, each change has at most $|T|$ different possibilities since there are at most $|T|$ distinct temperatures. Thus, the total number of cooling schedules is bounded by

$$
\binom{m}{|T|} |T|^{|T|},
$$

and we get the desired bound by using the upper bound $|T| \leq O(\sqrt{m} \log m \log \mathsf{e}_{\max})$. Therefore we have completed the proof.

D A Computational Model to Evaluate SA Algorithms

In this section, we introduce a model to evaluate the performance of an SA algorithm. The purpose of this model is to study the computational aspects of finding an optimal cooling schedule. We call this model *the monotone stationary graph*. For simplicity, (and indeed without loss of generality as we show in Section A[8]), we narrow down the space of the temperatures used in any algorithm to a finite set $T = \{d_1, d_2, d_3, \ldots, d_{|T|}\}$. Therefore from here on, we focus our attention on the discretized temperatures in T and assume that any algorithm (including any optimal solution) only uses temperatures in set T. Recall that every instance I of the underlying problem translates to a search graph for our SA algorithm. The goal of the monotone stationary graph is to represent the search graph in a compact manner so that we can evaluate the performance of a cooling schedule on each instance. Thus, the monotone stationary graph is made by the search graph and may differ between different instances of the problem.

Recall that every state of an SA algorithm $\mathcal{A}$ corresponds to a distribution $\mathcal{R}^{\mathcal{A}}$ over the vertices of the search graph. Initially, $\mathcal{R}^{\mathcal{A}}$ is the same for all algorithms and shows the probability distribution over the vertices on which our algorithm initiates the search. One example is when our algorithm starts with a fixed node of the search graph in which case $\mathcal{R}^{\mathcal{A}}$ is a deterministic distribution. Alternatively, $\mathcal{R}^{\mathcal{A}}$ may be a uniform distribution when our algorithm starts with a random node of the search graph. As we perform more steps of the algorithm, $\mathcal{R}^{\mathcal{A}}$ changes based on the criteria of the random walk and we hope that the correlation between $\mathcal{R}^{\mathcal{A}}$ and the energy of the nodes becomes stronger. Ideally, we would like our algorithm to end up with a distribution $\mathcal{R}^{\mathcal{A}}$ highly concentrated on the solution nodes.

Let us for every temperature $t \in |T|$, define a stationary distribution $\mathcal{S}_t$ which is a distribution of probabilities over the nodes of the search graph that an SA algorithm converges to after infinitely many steps of running on temperature t. Stationary distributions of simulated annealing are important and have been subject to a plethora of studies in the past decades [2, 10, 16, 24, 31, 34]. Intuitively, stationary distributions have positive correlation with the score of the nodes and as the temperature drops we expect the stationary distributions to provide higher (average) scores. Thus, the ideal case is when the state of our algorithm is very close to the stationary distribution for the lowest temperature for which the average score is the highest. The computational barrier is the *convergence rate* of the distributions. An algorithm that starts from an initial distribution and runs on a temperature t may need exponentially many steps to converge to the stationary distribution $\mathcal{S}_t$ whereas an algorithm that first reaches a stationary distribution $\mathcal{S}_{t'}$ for a higher temperature and then attempts to reach $\mathcal{S}_t$ may only need a small number of steps. This is perhaps best shown by the work of Wegener [36] wherein the author showed that for the minimum spanning tree problem, a cooling schedule that gradually decreases the temperature is exponentially faster than a cooling schedule that repeats a certain temperature. Thus, moving to intermediate stationary distributions may significantly improve the convergence rate of the algorithm.

Motivated by the above argument, we consider a model in which the states of any algorithm move between the stationary distributions. Let $d_1 > d_2 > d_3 > \ldots > d_{|T|}$ be all the distinct temperatures in T. We construct a graph with $|T| + 1$ nodes $v_0, v_1, v_2, \ldots, v_{|T|}$ such that node v_i corresponds to the set of all states close enough to the stationary distribution of temperature d_i. Also, v_0 is a special node corresponding to the initial distribution of the starting nodes. We assume that for every node v_i, the distances to the stationary distribution of temperature d_i are so small such that the difference in the performance is negligible. Due to this assumption, our model features monotonicity. More precisely, a cooling schedule that repeats a temperature t for 100 times is no better than the same cooling schedule that repeats t for 101 times.

In our model, we add edges between the nodes to denote transitions between stationary distributions. The labels of these edges indicate the number of steps needed for transition between a node v_i to a node v_j.

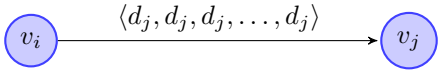


Figure 4: A transition is shown between two graph nodes.

[8] A loss of $\epsilon > 0$ is incurred to the score of any algorithm in the discretized setting.

Finally, we make one more assumption to complete the notion of monotonicity. If we have three temperatures $d_i > d_j > d_k$ the length of the edge from v_i to v_k is not smaller than the length of the edge from node v_j to node v_k. Another interpretation of this property is the following: in order to reach the stationary distribution of a temperature d_k, it is easier to start from the stationary distribution of a temperature closer to d_k rather than a temperature with a much higher difference. Although for some very delicately constructed examples this may fail, the assumption is along the common perception for the behavior of the SA algorithms [1, 2].

With the above definition, every path in the monotone stationary graph corresponds to a sequence of temperature which is made by the concatenation of the labels of the edges. A path can be traversed with a sequence of temperatures $\mathcal{E}$ if its corresponding label is a subsequence of $\mathcal{E}$. Given a sequence of temperatures $\mathcal{E} = \langle t_1, t_2, \ldots, t_m \rangle$, one can travel from node v_0 of the stationary distribution graph to a set of nodes via $\mathcal{E}$. In order to model the score of a cooling schedule $\mathcal{E}$, we assume that it takes us to the right most node v_i such that there is a path from v_0 to v_i whose label is a subsequence of $\mathcal{E}$. Implicit to our model is the assumption that stationary distributions become better[9] as the temperature drops. Thus, the scoring function gives us higher scores for lower temperatures.

For our computational results, we assume that the score of each cooling schedule is evaluated based on the above model. We compete against an optimal cooling schedule that uses a sequence of at most m moves. Thus, we can assume w.l.o.g that the length of every (existing) edge is bounded by m. This along with the monotonicity property of our model implies that there is a trivial cooling schedule with $|T|m$ many moves that performs at least as well as the optimal schedule with m steps. That is, in our model, a cooling schedule that contains m copies of each temperature performs always as well as any cooling schedule of length m. Although we allow the size constraint to be violated by a small factor, our aim is to keep the length of our approximately optimal cooling schedule close to m.

Our model may raise a concern for a thoughtful reader. We only incorporate the types of algorithms whose states move between the stationary distributions. What if the optimal solution never gets close enough to some of the stationary distributions, yet moves towards them in order to reach the stationary distributions for lower temperatures (see Figure 5)?

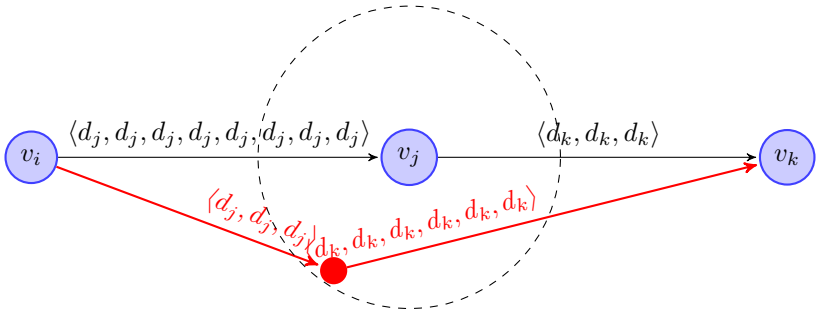


Figure 5: Red edges show the cooling schedule of the optimal solution. In this case, the optimal solution moves toward the stationary distribution of temperature d_j without getting close enough to its stationary distribution.

Although this may very well be the case in practice, the goal of this model is competing with the optimal algorithm that moves between the stationary distributions (and thus such a scenario is ruled out). We justify our model by the following intuitive argument: If moving towards a stationary distribution $\mathcal{S}_{d_j}$ makes a significant difference in the convergence rate for stationary distribution $\mathcal{S}_{d_k}$, it should be the case that a considerable portion of the path to the stationary distribution of $\mathcal{S}_{d_j}$ is already traversed. Thus, if we multiply the number of d_j steps of the algorithm by a small constant, this algorithm should reach the stationary distribution of $\mathcal{S}_{d_j}$. In other words, the optimal algorithm that adheres to our model may not necessarily be the optimal algorithm, however, if we allow for more steps (by a multiplicative constant factor), we expect that the optimal algorithm of our model performs as well as the optimal algorithm in the

[9] More concentration on the solution nodes.

unrestricted setting.

D.1 Computational Results

Although our model is general, we use the SAT problem to explain the terminology. Assume that the search graph contains 2^k vertices where every vertex is a true/false assignment to k variables of the underlying problem. Every node of the search graph is associated with a value which we refer to as energy. This concept reflects how close this node is to a solution. One example of such energy function is the amount of clauses satisfied by that solution. Also, the score of a cooling schedule $\mathcal{E}$ is equal to the probability of finding a solution for the problem via simulated annealing using $\mathcal{E}$ as a cooling schedule. We model this quantity with the monotone stationary graph.

Recall that we are given a distribution $\mathcal{D}$ over a class of SAT instances and our aim is to design a learning algorithm that computes/approximates a cooling schedule with the highest average score. In other words, our goal is to find a cooling schedule $\mathcal{E}$ for simulated annealing that maximizes

$$\mathbb{E}_{\mathsf{I}\sim\mathcal{D}}[\mathsf{score}(\mathsf{I},\mathcal{E})].$$

We model the performance of a simulated annealing algorithm by the monotone stationary graph explained previously. We compete against the score of the optimal cooling schedule with at most m steps subject to our model. Notice that, the optimal cooling schedule may in fact get a higher score than what our model suggests but we only give credit to that schedule based on our model and not the actual likelihood of finding a solution. Nonetheless, our hope is that the difference between the practical results and our model is negligible.

We assume throughout this paper that the number of steps of the optimal cooling schedule is equal to m. However, in order to compete with the optimal solution, we allow more steps for our algorithm. We define an algorithm $\mathcal{A}$ to be (α,ϵ)-approximate, if the number of steps of $\mathcal{A}$ is bounded by αm and the average score of $\mathcal{A}$ differs from the optimal solution by at most an additive error of ϵ.

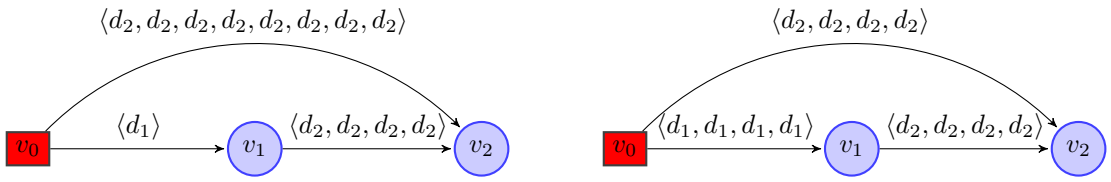
We show in Section A that from a sample-complexity standpoint, a learning algorithm only needs $\widetilde{O}(\sqrt{m})$ samples from $\mathcal{D}$ (Theorem 3.3). This result is indeed not dependent on the monotone stationary graph. However, the computational complexity of the solution requires more assumption on the score of cooling schedules. To this end, we define four scoring systems and analyze each of the systems separately.

Our results in Section A show that if we draw $n = \widetilde{O}(\sqrt{m})$ samples from $\mathcal{D}$ and find the solution that maximizes the average score on these n samples, the objective is approximately maximized for $\mathcal{D}$. Therefore, in the computational results, we assume that n problem instances $\mathsf{I}_1, \mathsf{I}_2, \ldots, \mathsf{I}_n$ are given and our goal is to find a sequence that maximizes the average score for those instances. We consider the following two settings for our problem:

- separate paths: For each instance I, the optimal cooling schedule runs on a sequence of temperatures that move between the nodes of the stationary distribution graph. However, the sequence of stationary nodes may vary between different instances.
- identical paths: The optimal cooling schedule chooses a sequence $v_{a_1}, v_{a_2}, \ldots, v_{a_x}$ of the nodes and does the following: starts and runs the algorithm by temperature d_{a_1} so long as **all** instances reach stationary distribution v_{a_1}. Then, proceeds with applying temperature d_{a_2} until all input instances reach stationary distribution v_{a_2} and so on. In this case, the path taken in the stationary distribution graph is the same for all instances of the problem.

We bring an example to illustrate the difference of the two models. Consider a distribution $\mathcal{D}$ of the SAT instances which returns instances I_1 and I_2 with equal probabilities. Let us assume that the monotone stationary graphs of the two instances are as shown in Figure 6. In the separate paths setting, the optimal sequence of temperatures that reaches the lowest stationary distribution for both instances is $\langle d_1, d_2, d_2, d_2, d_2\rangle$. Notice that in this case, for I_1 the path to v_2 is through v_1 but for I_2 the path consists of a direct edge from v_0 to v_2.

However, the choice of separate paths is not allowed in the identical paths model. Therefore, in the identical paths setting, the optimal solution is $\langle d_1, d_1, d_1, d_1, d_2, d_2, d_2, d_2\rangle$ which is through v_1 for both instances.



(a) Monotone stationary graph for I_1

(b) Monotone stationary graph for I_2

Figure 6: An example to show the difference between the separate paths setting and identical paths setting.

Moreover, we also study a more restricted setting, in which in the optimal solution, all instances of the problem reach the stationary distribution for the lowest temperature. We call this setting the all-satisfied setting.

For each combination of the settings we provide an algorithm along with its analysis. Table D.1 summarizes the time complexity of our algorithm in each setting.

identical paths	separate paths	separate paths + all-satisfied
exact solution in time $\mathsf{poly}(m, n, \lvert T\rvert)$	exact solution in time $\mathsf{poly}(m, n, \lvert T\rvert^n)$	$(O(\log n\lvert T\rvert), 0)$ approximation in time $\mathsf{poly}(m, n, \lvert T\rvert)$

One last thing to keep in mind before we go to the technical discussion is that monotone stationary graphs are not available to our algorithms. Therefore, the first step is to learn such a graph for a given instance I of the problem. We begin by explaining this in Section D.2 and then bring our algorithms in Section E.

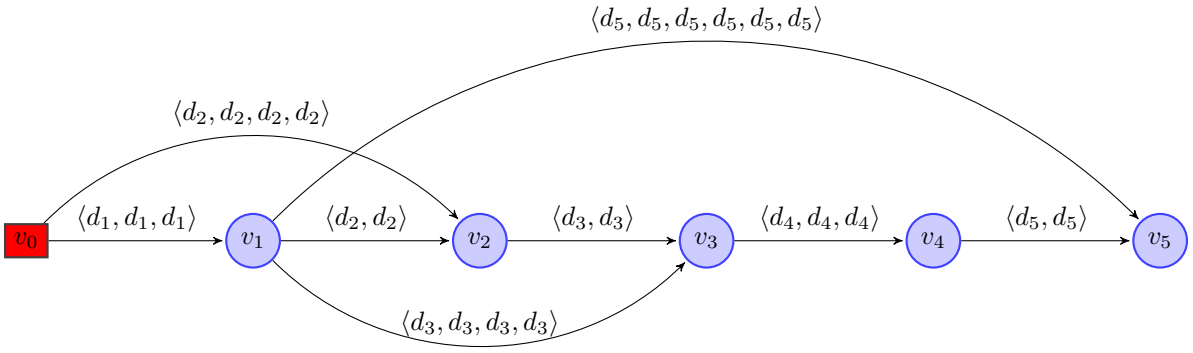


Figure 7: An example of the monotone stationary graph is shown. Only non-trivial edges are shown in this figure. For instance, an edge of length 6 from v_3 to v_5 can be implied from the edge (v_1, v_5).

To remind the reader of our assumptions, we bring a list of assumptions that we make for the model and the results:

- (for the model): The state of any algorithm moves between stationary distributions.
- (for the model): For $i < j < k$ the length of the edge from v_i to v_k is not smaller than that of v_j to v_k.
- (for the model): For any path P, a cooling schedule that contains the labels of the edges of the path as subsequence can take us to the end vertex. The score of a cooling schedule is equal to that of the best stationary distribution reachable via that schedule.

- (for the model): The score improves as i increases for v_i.
- (in order to learn the monotone stationary graph): For every instance of the problem, there is a cooling schedule of length m that takes us to node $v_{|T|}$.
- (in order to learn the monotone stationary graph): There is a noticeable difference between the scores of the nodes of the monotone stationary graph. That is, by running $\mathsf{poly}(n, m, |T|)$ experiments we can tell whether two cooling schedules $\mathcal{E}_1$ and $\mathcal{E}_2$ take us to the same node or not.

D.2 Learning A Monotone Stationary Graph

In this section, we show how one can learn the monotone stationary graph for a particular instance of the problem in polynomial time. Recall that $\mathsf{score}(\mathcal{E}, \mathsf{I})$ denotes the success probability of finding a solution to the problem. We make two assumptions to learn the monotone stationary graph. The first assumption is that for every instance there exists a sequence of length m that takes us to the optimal node (corresponding to the lowest temperature) in the monotone stationary graph. The second assumption is that there is a noticeable difference between the score of the nodes. That is, for any two cooling schedules $\mathcal{E}_1$ and $\mathcal{E}_2$ we can tell whether they take us to the same node in the monotone stationary graph or they take us to different nodes by running the SA algorithm several times and comparing their success ratios.

The discretization of the the temperatures is w.l.o.g as we show in Section A. Also, we ignore all edges whose sizes are more than m. This obviously does not hurt the optimal solution since its length is bounded by m.

Observation D.1. *Given a sequence $\mathcal{E}$ of temperatures, we can verify in polynomial time whether $\mathcal{E}$ takes us to $v_{|T|}$. That is, we can answer in polynomial time whether $\mathcal{E}$ is at least as good as any other sequence or not.*

Proof. By the above assumptions, a sequence that contains m repetitions of each temperature has to take us to node $v_{|T|}$ (otherwise there is no path of length m to $v_{|T|}$). Thus, by running the algorithm on this sequence, we can learn the average score of that node in time $\mathsf{poly}(m, |T|)$. Now, for a sequence $\mathcal{E}$, we just need to run the algorithm several times and verify whether the success rate is close to $s_{|T|}$ or not. □

Using Observation D.1, we can construct the monotone stationary graph for an instance I of the problem.

Lemma D.1. *Given an instance I of the problem, one can construct the underlying graph $\mathcal{G}(\mathsf{I})$ in time $\mathsf{poly}(m, |T|)$.*

Proof. We assume that there is a path of length m from node v_0 to node $v_{|T|}$. Thus, a sequence containing m repetitions of each temperature takes us there. Now, imagine we wish to answer the following question:

"*Is there an edge from node v_i to node $v_{|T|}$ with label $\overbrace{\langle d_{|T|}, d_{|T|}, \ldots, d_{|T|}\rangle}^{k}$?*"

To answer the above question, we can construct a sequence of temperatures that contains m repetitions of all temperatures $d_1, d_2, \ldots, d_i$. Next, we add k repetitions of temperature $d_{|T|}$ to the end of this sequence. If this sequence takes us to node $v_{|T|}$, then there is an edge from v_i to $v_{|T|}$ with a label that contains at most k copies of $d_{|T|}$. Thus, we can answer the query with a binary search.

Using the above machinery, we can extract all the edges that end at node $v_{|T|}$. Based on this information, we can find the smallest path that takes us from node $v_{|T|-1}$ to node $v_{|T|}$ and then recursively solve the problem for node $v_{|T|-1}$. With the same argument, we can discover all of the edges for all vertices of the graph. □

E Computing/Approximating the Optimal Cooling Schedule

The problem that we are concerned with in this section is computing (or approximating) the optimal cooling schedule for a set of problem instances. More precisely, let $\mathsf{I}_1, \mathsf{I}_2, \ldots, \mathsf{I}_n$ be n instances of the problem whose

monotone stationary graphs are available. The goal here is to find a cooling schedule whose size is close to m and and whose average score for the n instances is close to the optimal solution.

E.1 Identical Paths

The easier setting that we study is identical paths. In this setting, we compete with the optimal solution that chooses the same path for all instances. In other words, in such solutions, we fix a set of nodes $v_{a_1}, v_{a_2}, \ldots, v_{a_k}$ and find a cooling schedule that takes all instances through this path. More precisely, we put enough temperatures d_{a_1} to make sure all instance reach vertex v_{a_1}. Next, we proceed by doing the same thing for v_{a_2} and so on.

We show that in this setting the problem of finding the optimal cooling schedule reduces to shortest path. Construct a graph G with the same vertex set as the monotone stationary graphs. We put a directed edge from vertex v_i to vertex v_j of G, if and only if such an edge exists in the corresponding monotone stationary graphs of all instance. Moreover, we set the length of this edge as the largest length in all the graphs. Finally, we find the lowest temperature d_i (meaning that i is maximized) such that vertex v_i is reachable from vertex v_0 via a path of length at most m. We prove that this algorithm is optimal.

Algorithm 1: Exact algorithm for the identical paths setting.

1: **for** $1 \leq k \leq n$ **do**
2: Let $r_{i,j}^{(k)}$ be the minimum number of repetitions for temperature d_j to make a transition from node v_i to v_j on the k-th monotone stationary graph.
3: $r_{i,j} := \max_{k \in [n]} r_{i,j}^{(k)}$.
4: Let $r_{i,j} = 0$ when $i \geq j$.
5: Construct a graph with vertices $v_0, \cdots, v_{|T|}$ and pairwise distances $r_{i,j}$.
6: Find the the shortest paths from v_0 to all nodes of the graph.
7: Find the largest q such that node v_q is within a distance of m from v_0.
8: Output the optimal sequence of temperatures made by this path.

Theorem E.1. *Given n monotone stationary graphs for n instances of the problem, one can find in polynomial time a cooling schedule of length m that maximizes the average score for the n instances in the same paths setting*

Proof. The proof is based on the fact that the path is the same for all instances. Thus, in order to make a transition from vertex v_i to vertex v_j, one needs to add as many copies of temperature d_j as the size of the largest label among all instances. Thus, the furthest we can get from node v_0 is a node v_i whose distance from v_0 is bounded by m on the graph we make. □

E.2 Separate Paths

The more challenging setting is when we allow the instances to have different paths in the optimal solution. In this case, the problem is much harder since we have to consider n different monotone stationary graphs and solve the problem with respect to all of them. However, it is not hard to see that if n is constant, one can find the optimal cooling schedule of length m in polynomial time.

Lemma E.2. *Given n monotone stationary graphs for n instances of the problem, one can find in time $\widetilde{O}(m|T|^{n+1})$ a cooling schedule of length m that maximizes the average score for the n instances in the separate paths setting*

Proof. The proof is similar to the proof of Theorem E.1. However, since we may traverse a different path for every graph, we need to construct our graph more carefully. To this end, our vertex set would be the multiplication of the vertex sets for the monotone stationary graphs. That is, we put $(|T| + 1)^n$ different vertices in our graph such that every vertex shows one combination of the nodes for the instances.

Every vertex has $O(m|T|)$ different edges that shows how the combination changes by adding $1 \leq i \leq m$ copies of temperature d_j to the sequence. Finally, we compute the distances of all vertices from node

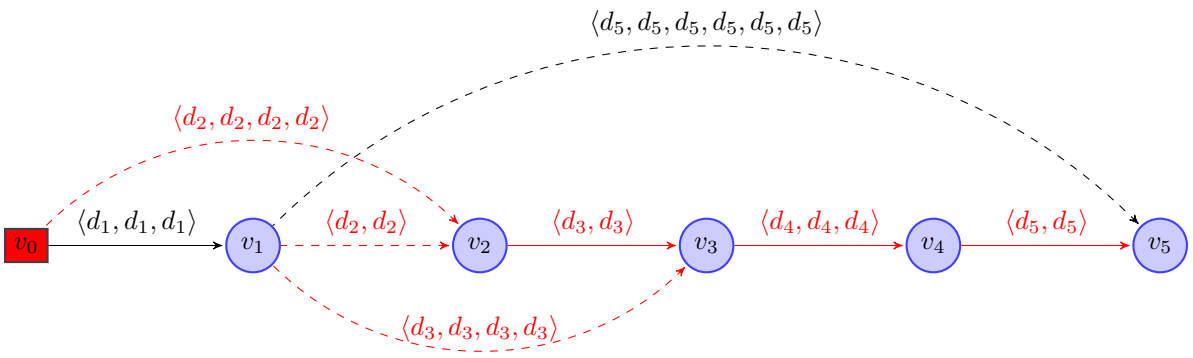


Figure 8: All the dashed edges are crossing for vertex v_2. Moreover, sequence $\langle d_1, d_1, d_2, d_2, d_2, d_2, , d_3, d_3, d_3, d_3, , d_4, d_4, d_4, d_4, d_5, d_5 \rangle$ is an acceptable sequence that encompasses all the red edges. The edges that are not drawn are implied by the edges depicted in the figure.

$(v_0, v_0, \ldots, v_0)$ and find the one whose distance is bounded by m and its score is maximized. Then, we recover the path to that node and report it. □

Obviously, the runtime of Lemma E.2 is not polynomial when n is super constant. Therefore, for asymptotically larger n's, we present a polynomial time algorithm that approximates the solution. Our algorithm works for the all-satisfied setting, which means that there is an optimal solution that brings all instances to the vertex corresponding to the lowest temperature. Our algorithm loses a polylogarithmic factor in the size of the sequence but obtains the same score as the optimal solution with high probability.

Let us assume for simplicity that the score for each instance I_k is equal to 1 if and only if our sequence takes us to node $v_{|T|}$ in its monotone stationary graph. Otherwise the score is equal to 0. Our algorithm is **not** dependent on this assumption, yet it makes the explanation much simpler. We begin by an observation that translates the definition of score into the set cover setting.

We say a cooling schedule $\mathcal{E}$ is an *acceptable cooling schedule* for an instance I of the problem if $\mathcal{E}$ takes us all the way to node $v_{|T|}$ in its monotone stationary graph. Define an edge from a vertex v_i to a vertex v_j to be *crossing* for a vertex v_k if $i < k \leq j$ holds. Moreover, we say a sequence $\mathcal{E}$ *encompasses* an edge from v_i to v_j from a particular monotone stationary graph if $\mathcal{E}$ contains at least as many repetitions of temperature d_j as the label of the edge from v_i to v_j. An example of the definitions is shown in Figure 8.

Now, we are ready to state an observation that plays an important role in our algorithm.

Observation E.1. *A sequence of temperatures $\mathcal{E}$ is acceptable for an instance I of the problem if and only if for every $1 \leq i \leq |T|$, $\mathcal{E}$ encompasses at least one crossing edge with v_i.*

Proof. The necessity of the condition is trivial. If $\mathcal{E}$ does not encompass a crossing edge for a vertex v_i, then $\mathcal{E}$ cannot reach vertex v_i in monotone stationary graph. The vice versa also holds. Suppose for the sake of contradiction that a sequence $\mathcal{E}$ encompasses a crossing edge for every vertex but it does not take us to node $v_{|T|}$. In this case, there exists a vertex v_i, such that all vertices v_{i-1} is reachable but none of the vertices v_j is reachable for $j \geq i$ are reachable using $\mathcal{E}$. This means that $\mathcal{E}$ does not encompass an edge that crosses vertex v_i otherwise we could have reached vertex v_i using $\mathcal{E}$. □

We are now ready to state the main theorem of this section.

Theorem E.3. *Let $\mathsf{I}_1, \mathsf{I}_2, \ldots, \mathsf{I}_n$ be n monotone stationary graph with the guarantee that there exists a cooling schedule of length m that is acceptable for all instance. One can find in polynomial time a cooling schedule for the SA algorithm whose average score is equal to that of the optimal cooling schedule of size m. Our algorithm is randomized and gives a solution with probability at least $1 - e^{-100}$. Also, the average size of the cooling schedule is bounded by $O(m(\log |T| + \log n))$.*

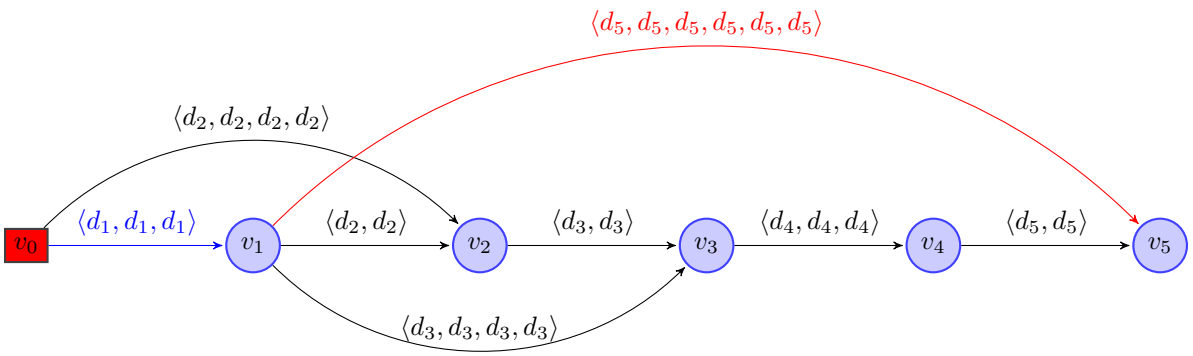


Figure 9: For $m = 9$ the optimal sequence of temperatures is $\langle d_1, d_1, d_1, d_5, d_5, d_5, d_5, d_5, d_5 \rangle$. All the edges skipped in the figured can be implied from the edges shown by monotonicity.

Proof. Observation E.1 gives us a strong tool to analyze the solution. Let OPT be the optimal cooling schedule of size m which is acceptable for all instances. Due to Observation E.1, OPT encompasses at least one crossing edge for all vertices of all monotone stationary graphs. To formalize this, define a set

$$\begin{aligned} S = &\{\langle d_1 \rangle, \langle d_1, d_1 \rangle, \ldots, \overbrace{\langle d_1, d_1, \ldots, d_1 \rangle}^{m}\} \cup \\ &\{\langle d_2 \rangle, \langle d_2, d_2 \rangle, \ldots, \overbrace{\langle d_2, d_2, \ldots, d_2 \rangle}^{m}\} \cup \\ &\vdots \\ &\{\langle d_{|T|} \rangle, \langle d_{|T|}, d_{|T|} \rangle, \ldots, \overbrace{\langle d_{|T|}, d_{|T|}, \ldots, d_{|T|} \rangle}^{m}\} \end{aligned}$$

to be the set of all possible repetitions for all temperatures and for each element $e \in S$, define $\ell(e)$ to be the size of e. In addition to this, for each element $e \in S$, define $c_e = \{0, 1\}$ to be equal to 1 if and only if OPT contains $\ell(e)$ repetitions of the character corresponding to e.

To clarify the definitions, consider an example with only a single instance shown in Figure 9. In this case, $m = 9$ and the optimal sequence of temperatures is $\mathsf{OPT} = \langle d_1, d_1, d_1, d_5, d_5, d_5, d_5, d_5, d_5 \rangle$. In this case S contains $45 = m|T|$ elements out of which only $c_{\langle d_1, d_1, d_1 \rangle}$ and $c_{\langle d_5, d_5, d_5, d_5, d_5, d_5 \rangle}$ are equal to 1. Moreover, $\ell(\langle d_1, d_1, d_1 \rangle) = 3$ and $\ell(\langle d_5, d_5, d_5, d_5, d_5, d_5 \rangle) = 6$ hold.

For a vertex v_i in graph I_k, define $\mathsf{crossingI}(\mathsf{I}_k, v_i)$ to be the set of elements in S that correspond to the crossing edges of v_i. This way, the optimal solution of the problem can be formulated via the following integer feasibility program:

$$\begin{array}{lllll} \text{constraints:} & \sum \ell(e) c_e & \leq m & & \\ & \sum_{e \in \mathsf{crossingI}(v_i, \mathsf{I}_k)} c_e & \geq 1 & \forall 1 \leq k \leq n \text{ and } 1 \leq i \leq |T| & (13) \\ & c_e & \in \{0, 1\} & \forall e \in S, & \end{array}$$

where the variables of the program are c_e's. Indeed by relaxing the conditions of IP 13 we can obtain LP 14.

$$\begin{array}{lllll} \text{constraints:} & \sum \ell(e) c_e & \leq m & & \\ & \sum_{e \in \mathsf{crossingI}(v_i, \mathsf{I}_k)} c_e & \geq 1 & \forall 1 \leq k \leq n \text{ and } 1 \leq i \leq |T| & (14) \\ & 0 \leq c_e \leq 1 & & \forall e \in S & \end{array}$$

Now we solve LP 14 and construct a solution as follows: for each element $e \in S$, we add e to our solution independently with probability $\min\{\alpha c_e, 1\}$, where $\alpha = 100(\log |T| + \log n)$.

First, it's easy to see the expected length of our solution is bounded by αm:

$$\mathbb{E}[\text{length}] = \sum_{e \in S} \Pr[e \text{ is picked}]\ell(e) = \sum_{e \in S} \min(\alpha c_e, 1)\ell(e) \leq \sum_{e \in S} \alpha c_e \ell(e) \leq \alpha m,$$

where the last step is due to the constraint in LP.

Next, we will show that with high probability, the resulting sequence is acceptable for each instance I_k. Consider the case when the resulting sequence is not acceptable for I_k. By Observation E.1, there exists a v_i such that none of the edges in $\mathsf{crossingI}(v_i, \mathsf{I}_k)$ were encompassed in our solution. By union bound, the probability of this bad event can be upper bounded by

$$\Pr[\mathsf{I}_k \text{ is not satisfied}] \leq \sum_{i=1}^{|T|} \Pr[\forall e \in \mathsf{crossingI}(v_i, \mathsf{I}_k) \text{ wasn't picked}]. \tag{15}$$

Now we focus on the probability inside the summation. Since each element was selected independently, this probability equals to

$$\begin{aligned}\Pr[\forall e \in \mathsf{crossingI}(v_i, \mathsf{I}_k) \text{ wasn't picked}] &= \prod_{e \in \mathsf{crossingI}(v_i, \mathsf{I}_k)} \Pr[e \text{ wasn't picked}] \\ &= \prod_{e \in \mathsf{crossingI}(v_i, \mathsf{I}_k)} \max(1 - \alpha c_e, 0).\end{aligned}$$

If $\alpha c_e \geq 1$ for some $e \in \mathsf{crossingI}(v_i, \mathsf{I}_k)$, then the probability is 0. Otherwise, since $1 - x \leq e^{-x}$, we have

$$\prod_{e \in \mathsf{crossingI}(v_i, \mathsf{I}_k)} \max(1 - \alpha c_e, 0) \leq e^{-\sum_{e \in \mathsf{crossingI}(v_i, \mathsf{I}_k)} \alpha c_e} \leq e^{-\alpha}.$$

Therefore, by 15

$$\Pr[\mathsf{I}_k \text{ is not satisfied}] \leq |T| e^{-\alpha}$$

Using union bound again, we have

$$\begin{aligned}\Pr[\text{any instance } \mathsf{I}_k \text{ is not satisfied}] &\leq \sum_{1 \leq k \leq n} \Pr[\mathsf{I}_k \text{ is not satisfied}] \\ &\leq |T| n e^{-\alpha} \\ &\leq e^{-100}.\end{aligned}$$

Hence, we proved that with probability $1 - e^{-100}$, the resulting sequence is acceptable for each instance I_k.□

F Omitted Proofs of Section B

Proof of Observation B.1: For Observation B.1.(i), define $\tau_r = \{\arg\min_{i\geq 0} : x_i = r\}$. We would like to prove that,

$$\Pr[\tau_{\sqrt{k/c}} \leq k] \geq 0.95. \tag{16}$$

By the Markov property, τ_r is the sum of r independent copies of τ_1. Let the probability generating function of τ_r be $F_r(z) := \mathbb{E}[z^{\tau_r}] = \sum_{j=0}^{\infty} \Pr(\tau_r = j) z^j$, then we have $F_r(z) = F_1(z)^r$. Furthermore, we have the following recurrence about $F_1(z)$:

$$F_1(z) = \frac{z}{3}\left(1 + F_1(z) + F_1(z)^2\right). \tag{17}$$

Hence,

$$F_1(z) = \frac{(3-z) - \sqrt{3(z+3)(1-z)}}{2z}. \tag{18}$$

One important property about $F_1(z)$ is that for all $z \in [0,1]$,

$$F_1(z) \leq 1 - \sqrt{1-z}. \tag{19}$$

By Markov's inequality,

$$\Pr[\tau_r \geq k] \leq \inf_z \frac{\mathbb{E}[z^{\tau_r}]}{z^k} \tag{20}$$

$$= \inf_z \frac{(1-\sqrt{1-z})^r}{z^k} \tag{21}$$

$$(z := 1 - \frac{1}{k}) = (1 - \frac{1}{\sqrt{k}})^r (1 - \frac{1}{k})^{-k} \tag{22}$$

$$(r := c\sqrt{k}) \leq \exp(-c - 1). \tag{23}$$

Hence we have completed the proof.

For Observation B.1.(ii) and (iii), we need a classical result in martingale concentration inequalities, the Freedman's inequality for scalar martingales [13, Thm. (1.6)], see also [33, Thm. (1.1)].

Theorem F.1 (Freedman)**.** *Consider a real-valued martingale* $\{Y_k : k = 0, 1, 2, \dots\}$ *with difference sequence* $\{X_k : k = 1, 2, 3, \dots\}$*. Assume that the difference sequence is uniformly bounded:*

$$|X_k| \leq R \quad \textit{almost surely} \quad \textit{for } k = 1, 2, 3, \dots.$$

Define the predictable quadratic variation process of the martingale:

$$W_k := \sum\nolimits_{j=1}^{k} \mathbb{E}_{j-1}(X_j^2) \quad \textit{for } k = 1, 2, 3, \dots.$$

Then, for all $t \geq 0$ *and* $\sigma^2 > 0$*,*

$$\Pr[\exists k \geq 0 : Y_k \geq t \quad \textit{and} \quad W_k \leq \sigma^2] \leq \exp\left\{-\frac{t^2/2}{\sigma^2 + Rt/3}\right\}.$$

When the difference sequence $\{X_k\}$ consists of independent random variables, the predictable quadratic variation is no longer random. In this case, Freedman's inequality reduces to the usual Bernstein inequality.

To prove B.1, we let $y_k = x_k + k\gamma$ where $\gamma = p_b - p_f$, then y_k is a martingale since the difference sequence $\Delta_k = y_k - y_{k-1}$ has expectation zero. Furthermore, the difference sequence Δ_k is uniformly bounded with $|\Delta_k| \leq R := 1 + |\gamma|$, and $W_k = \sum_{j=1}^{k} \mathbb{E}_{j-1}(\Delta_k^2) \leq kR^2$. By the Freedman's inequality,

$$\Pr[\max_k y_k \geq t] \leq \exp\left\{-\frac{t^2/2}{kR^2 + Rt/3}\right\}. \tag{24}$$

Since we have $R = 1 + |\gamma|$ and $y_k = x_k + k\gamma$, (24) is equivalent to

$$\Pr[\max_k x_k \geq t - k\gamma] \leq \exp\left\{-\frac{t^2/2}{kR^2 + tR/3}\right\}. \tag{25}$$

Let $t = 3R\sqrt{k \log k}$, we have

$$\frac{t^2/2}{kR^2 + Rt/3} = \frac{t^2/(2R^2)}{k + t/(3R)} \geq \frac{t^2/(2R^2)}{k + k} = \frac{t^2}{4kR^2} \geq 2 \log k.$$

Rearranging terms gives

$$\Pr[\max_k x_k \geq 3(1 + |\gamma|)\sqrt{k \log k} - k\gamma] \leq \frac{1}{k^2}. \tag{26}$$

When $p_f = p_b = p_s = \frac{1}{3}$, we have $\gamma = 0$ and the above inequality is equivalent to

$$\Pr[\max_k x_k \geq 3\sqrt{k \log k}] \leq \frac{1}{k^2}. \tag{27}$$

This proves (ii). For (iii), we note that $\gamma = p_b - p_f \geq \frac{2c \log k'}{\sqrt{k'}}$ and we only needs to prove that

$$3(1 + |\gamma|)\sqrt{k \log k} - k\gamma \leq \frac{\sqrt{k'}}{2}. \tag{28}$$

Note that

$$3(1 + |\gamma|)\sqrt{k \log k} - k\gamma \leq 3\sqrt{k \log k} - \frac{k}{2}\gamma. \tag{29}$$

When $\gamma \geq \frac{6\sqrt{\log k}}{\sqrt{k}}$, RHS is negative so the inequality holds trivially. Otherwise, we have $\gamma = 2c\frac{\log k'}{\sqrt{k'}} < \frac{6\sqrt{\log k}}{\sqrt{k}}$, hence $k' \geq O(1)\ \frac{k}{\log^2 k} > \sqrt{k}$. By AM-GM inequality,

$$3\sqrt{k \log k} \leq \frac{k}{2}\gamma + \frac{9 \log k}{\gamma}, \tag{30}$$

Therefore, we have

$$3(1 + |\gamma|)\sqrt{k \log k} - k\gamma \leq \frac{9 \log k}{\gamma} = \frac{9 \log k}{2c \log k'}\sqrt{k'} \leq \frac{9}{c}\sqrt{k'}. \tag{31}$$

Hence, letting $c = 9$ completes the proof. □

G Omitted Proofs of Section C

Proof of Observation C.1: The proof is given below:

$$\begin{aligned} pf(x)+(1-p)f(y) =& p\left[x-x^2\right]+(1-p)\left[y-y^2\right] \\ =& (px+(1-p)y)-(px+(1-p)y)^2-\left[(p-p^2)(x^2+y^2-2xy)\right] \\ =& (px+(1-p)y)-(px+(1-p)y)^2-\left[(p-p^2)(x-y)^2\right] \\ \leq& (px+(1-p)y)-(px+(1-p)y)^2-\left[\min\{p,1-p\}(x-y)^2\right] \qquad (32) \\ =& f(px+(1-p)y)-\left[\min\{p,1-p\}(x-y)^2\right]. \end{aligned}$$

Inequality (32) follows from the fact that both p and $1-p$ are in range $[0,1]$ and thus $p(1-p)\leq \min\{p,1-p\}$. □

Proof of Observation C.3:

$$\begin{aligned} |f(x)-f(y)| =& |[x-x^2]-[y-y^2]| \\ =& |(x-y)-(x^2-y^2)| \\ =& |(x-y)-(x-y)(x+y)| \\ =& |(x-y)(1-(x+y))| \\ =& |x-y||(1-(x+y))| \\ \leq& |x-y| \end{aligned}$$

where the last inequality holds since $0\leq x+y\leq 2$ and therefore $-1\leq 1-(x+y)\leq 1$ which implies $0\leq |1-(x+y)|\leq 1$. □

Proof of Observation C.4: To prove the observation, we take the first derivative of $p-p^{1+x}$ which is equal to

$$\frac{d}{dp}\left[p-p^{1+x}\right]=1-(1+x)p^x$$

which means that the function is maximized (or minimized) at $p_0=(1+x)^{-1/x}$. It is easy to see that since $p-p^{1+x}$ is non-negative in range $[0,1]$ and is equal to 0 at both $p=0$ and $p=1$ then the expression should be maximized at p_0. Thus, the maximum value for $p-p^{1+x}$ is bounded by

$$\begin{aligned} p_0-p_0^{1+x} &= p_0(1-p_0^x) \\ &\leq 1-p_0^x \\ &= 1-((1+x)^{-1/x})^x \\ &= 1-(1+x)^{-1} \\ &= 1-1/(1+x) \\ &= x/(1+x) \\ &\leq x. \end{aligned}$$

□

Proof of Observation C.5: We first show the proof for the case of $p\leq 1/2$. We start by the famous inequality $1+y\leq e^y$ [25] which holds for any $y\in\mathbb{R}$. Therefore, we have $1-e^y\leq -y$. By setting $y=-x\ln 1/p$ we obtain

$$1-e^{-x\ln 1/p}\leq x\ln 1/p$$

Notice that $e^{-x \ln 1/p}$ can be written as $(e^{-\ln 1/p})^x = (e^{\ln p})^x = p^x$. Thus, we have

$$1 - p^x \leq x \ln 1/p$$

Multiplying both sides by p gives us

$$p - p^{1+x} \leq p(\ln 1/p)x$$

which proves the observation for $p \leq 1/2$. Next, we show the statement for $p \geq 1/2$. In this case, we prove $p - p^{1+x} \leq (1-p)x$ which implies the observation. Our goal here is to prove $p - p^{1+x} - (1-p)x \leq 0$ for any $p \in [0.5, 1]$ and any $0 \leq x \leq 1$. Thus, we take the derivative of p to bound its maximum value.

$$\frac{d}{dp}\left[p - p^{1+x} - (1-p)x\right] = 1 - (1+x)p^x + x$$

which is equal to 0 only at $p_0 = 1$. At p_0 we have $p - p^{1+x} - (1-p)x = 0$ which is not greater than 0. Also, since for $p = 0$ the expression $p - p^{1+x} - (1-p)x$ is equal to $-x$ which is negative, it means that the function is maximized at $p = 1$. Thus, $p - p^{1+x} - (1-p)x$ is always upper bounded by 0 which means $p - p^{1+x} \leq (1-p)x$. □