Bocconi University

Empirical Exploration of Search Heuristics: A Comparative Study of WalkSAT and Simulated Annealing for Random K-SAT Problems

Student: Parolari Elia

Supervisor: Carlo Lucibello

Bachelor Thesis

Abstract

Satisfiability is a well-known problem and is connected to numerous issues, many of which hold significant practical importance. This research compares WalkSAT, a common algorithm for solving random K-SAT instances, with simulated annealing. The comparison shows that both algorithms have similar performance in terms of solved instances, but the less greedy strategy of simulated annealing reveals it to be slightly faster close to the phase transition between SAT and UNSAT. This reflects the fact that the problem remains NP-complete, and at any given moment, there is no general optimal strategy for making a decision that reliably approaches the problem's solution.

Contents

1 Introduction

The thesis focuses on empirically comparing the performance of two algorithms, Walk-SAT and Simulated Annealing, on satisfiability problems. This research is primarily motivated by personal curiosity towards simulated annealing, particularly its use of concepts from statistical physics to implement a generic procedure applicable to various optimization problems.

The document is divided into two main parts. The first part addresses fundamental theoretical aspects necessary to understand the analysis, starting from the rigorous definition of a satisfiability problem to the theory required to comprehend the logic behind the algorithms and their presentation.

The second section delves into an analysis of the data derived from comparing different instances, considering various comparison metrics.

Satisfiability problems necessitate exponential time to solve. Therefore, due to limited computational resources, this study focuses on relatively simple instances. Every conclusion drawn from the experiments pertains only to such types of instances. Hence, it's crucial to exercise caution before generalizing to larger problems.

2 Theoretical background

2.1 Satisfiability

This section explores the concept of satisfiability. The content, notation, and terminology used are drawn from [5].

2.1.1 What is a K-SAT problem?

To define an instance of satisfiability problem we need N boolean variables and M constraints between them. A single variable is denoted x_i with $i \in \{1, ..., N\}$ and takes values in $\{1, 0\}$. We denote the negation of this variable with $\bar{x}_i = 1 - x_i$. We refer to a variable or its negation as a literal or spin [1]. A clause a, instead, is a constraint between K_a literals in the form of

[1] The reason for this name will come clear later.

an **OR** logical function. Hence depending on the assignment (or state) of the literals a clause can either be satisfied or not. Denoting by ∂a the subset $\{i_1^a, ..., i_{K_a}^a\} \subseteq \{1, ..., N\}$, i.e. the indices of the literals contained in clause a, we can refer to clause a as C_a (here one should be more precise since there is no information about the negation of the literals in the index). Finally we can write an instance of a satisfiability problem in **conjunctive normal form** (*cnf*) as:

$$F = C_1 \wedge C_2 \wedge ... \wedge C_M \tag{1}$$

The goal of an algorithm is to find among the 2^N possible assignments of the literals a configuration that satisfies F, such a solution is called a **SAT-assignment**, instead if no solution exists we refer to the problem as **UNSAT**.

Remark: Usually instead of having clauses with different lengths, a fixed length K for each clause is used, in this case we refer to such a problem as K-SAT.

Illustrative example of a K-SAT problem with $N = 6$, $K = 3$, $M = 2$.

$$F = (x_1 \vee \bar{x}_3 \vee x_5) \wedge (x_3 \vee \bar{x}_4 \vee \bar{x}_6) \tag{2}$$

For example, any assignment where $x_3 = x_4 = 0$ satisfies the expression.

Satisfiability was the first problem to be proved NP-complete, hence in general it cannot be solved in polynomial time. A special case is 2-SAT, for which polynomial time algorithms exist, but already for K=3 the problem becomes hard.

2.1.2 Random K-SAT and phase transitions

Although satisfiability is NP-complete, empirical studies have shown that in many cases a solution is easy to find. Therefore it's very convenient to characterize ensembles of problems that are easy and separate them from harder ones. To achieve this we introduce a particular class of satisfiability problems, called **random K-SAT**. Fixing N, K and M we denote $SAT_N(K, M)$ the ensemble of all K-SAT problems with such parameters, and we draw uniformly at random one instance. In other terms we select M random clauses from the $\binom{N}{K}2^K$ possible ones. It

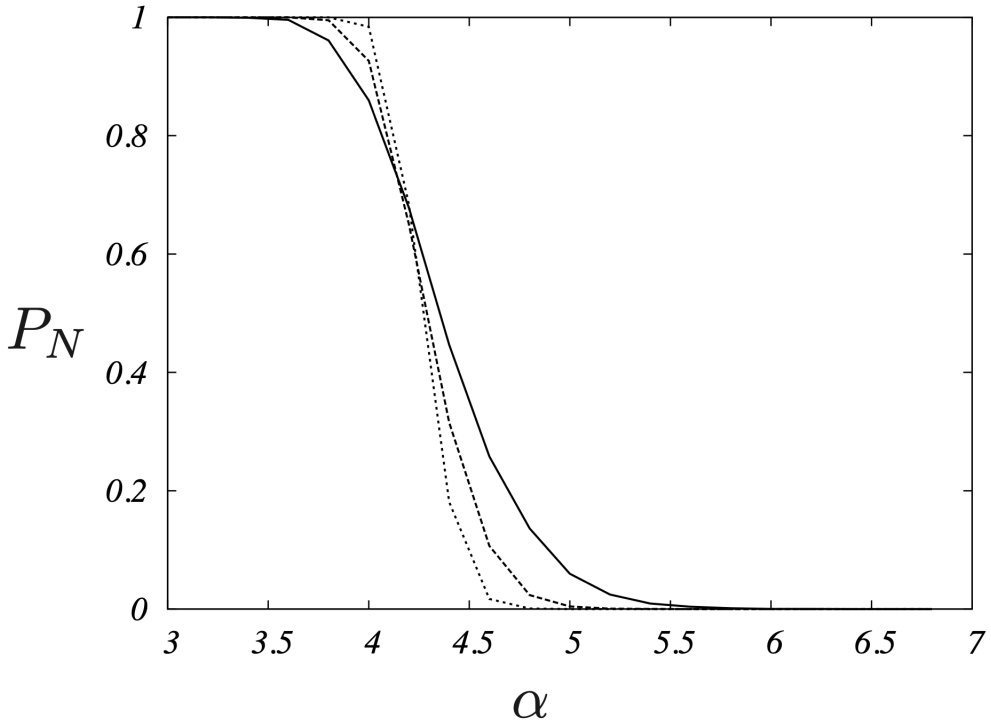


Figure 1: Plot showing the relationship between α and P for $K = 3$ as presented in [5]. Different lines are for different values of N. The separation is sharper for larger values of N.

turns out that the parameter the defines 'how hard' is the problem is the ratio between N and M denoted as α, that can be thought of as a density of clauses, i.e. how many clauses we have per literal. Denote with $P_N(K, \alpha)$ the probability that an instance of random K-SAT is solvable (i.e. there is at least one SAT-assignment). Numerical studies (see fig.1) have shown that for $K \geq 2$ in the large N limit there is value of α that determines a *phase transition* from SAT to UNSAT. This idea is formalized in the following conjecture:

Conjecture 2.1. *For any $K \geq 2$ there exists a threshold $\alpha_S(K)$ such that:*

$$\lim_{N\to\infty} P_N(K, \alpha) = \begin{cases} 1 & \text{if } \alpha < \alpha_S(K) \\ 0 & \text{if } \alpha > \alpha_S(K) \end{cases} \tag{3}$$

A phase transition refers to a qualitative change of the properties of a system due to the modification of one of the parameters of the system, in this case α. The value of the parameter at which the transition takes place is usually referred to as *critical value*. Phase transitions are very common phenomena in nature, one of the most used example is the transition of water

from solid to liquid at 0°C. In random K-SAT problems there can be other phase transitions than the one just described, as pointed out in [3]. This other transitions regard the structure of the set of solutions. What has been shown in particular is the existence of $\alpha_d(K) < \alpha_S(K)$ for problems with $K \geq 4$ such that for values of α below $\alpha_d(K)$ the subspace of solutions is compact, while for values above the threshold the solutions forms disjoint small subsets. This is an important fact that has direct consequences on the performance of algorithms, as it will be discussed later.

2.1.3 Relevance

It's important to consider that satisfiability connects to numerous issues, many of which hold significant practical importance. It stands as a crucial problem within mathematical logic, computing theory, and artificial intelligence but its applications span a wide spectrum, from integrated circuit design (including modeling, placement, routing, and testing) to computer architecture (encompassing compiler optimization, scheduling, and task partitioning), as well as extending to fields like computer graphics and image processing. Having algorithms that can solve or approximate solutions is of crucial importance for the development of this fields.

2.2 Markov Chains

Both algorithms used in this research (particularly simulated annealing) are based on the concept of Markov chains, hence its crucial to give an introduction to the topic. This section refers to [6]. Markov chains are a broad topic, here we discuss the fundamental aspects needed to understand the subsequent sections.

2.2.1 Discrete time stochastic processes and the Markov property

Definition 2.1. *(Stochastic process). A stochastic process is a sequence of random variables denoted* $(X_t)_{t\in T}$*, with* $X_t : \Omega \to \mathcal{X}$ $\forall t \in T$*.* $\mathcal{X}$ *is called the state space and* T *the index.*

Throughout this research it is assumed $T = \mathbb{N}$ and $|\mathcal{X}| < \infty$. This particular type of processes are called *discrete time stochastic processes with discrete state space*. In general to

characterize the behaviour of this processes one can study the joint distribution of $(X_0, ..., X_t)$, but this can be a very complicated mathematical object to analyze. In Markov chains, the structure is simplified, focusing on a subclass of stochastic processes.

Definition 2.2. *(Markov chain). A discrete time stochastic process* $(X_t)_{t\geq 0}$ *has the Markov property if*

$$P(X_{t+1} = x_{t+1}|X_t = x_t, ..., X_0 = x_0) = P(X_{t+1} = x_{t+1}|X_t = x_t) \tag{4}$$

$\forall t \geq 0$ *and for all* $(x_0, ..., x_{t+1}) \in \mathcal{X}^{t+1}$.

Intuitively this means that what happens at time $t + 1$ only depends on the present time t and not on the whole history. Additionally time homogeneity is assumed to hold, i.e.

$$P(X_{t+1} = x_{t+1}|X_t = x_t) = P(X_1 = x_1|X_0 = x_0) \quad \forall t \geq 0 \tag{5}$$

Time homogeneity implies that the probability of transitioning from a state to another does not depend on when the transition takes place in time. This allows a convenient way of describing the process using *transition probabilities*, namely:

$$P_{ij} = P(X_{t+1} = j|X_t = i) \quad \forall i, j \in \mathcal{X} \tag{6}$$

This probabilities can be stored in a $|\mathcal{X}| \times |\mathcal{X}|$ matrix P called *transition matrix*. Note that P satisfies:

$$\begin{cases} P_{ij} \geq 0 & \forall i, j \in \mathcal{X} \\ \sum\limits_j P_{ij} = 1 & \forall i \in \mathcal{X} \end{cases} \tag{7}$$

2.2.2 Stationarity and convergence

A nice property of Markov chains is that they often converge to a stationary distribution regardless of the initial conditions.

Definition 2.3. *(Stationary distribution). A probability measure* π *on* $\mathcal{X}$ *is stationary for a*

Markov chain with transition probability matrix P if:

$$\pi P = \pi \tag{8}$$

or equivalently:

$$\pi_j = \sum_i \pi_i P_{ij} \quad \forall i, j \in \mathcal{X} \tag{9}$$

To introduce the next notion, denote P_{ij}^t the probability of going from i to j in t steps.

Definition 2.4. *(Limiting distribution). A probability measure π on $\mathcal{X}$ is limiting for a Markov chain with transition probability matrix P if:*

$$\lim_{t\to\infty} P_{ij}^t = \pi_j \quad \forall i, j \in \mathcal{X} \tag{10}$$

What we are interested is to understand what are the conditions in order for a Markov chain to have a stationary distribution. Moreover, when is it the case that the chain also converges to this distribution? Before answering we need to introduce two important properties that a Markov chain can have: irreducibility and aperiodicity.

Definition 2.5. *(Irreducible MC.) A MC with transition matrix P is irreducible if every state is accessible from every other state, more formally:*

$$P_{i\to j} > 0 \quad \forall i, j \in \mathcal{X} \tag{11}$$

Note that $P_{i\to j} \neq P_{ij}$, what is meant here is that there is a trajectory in the state space from i to j that can be taken with positive probability.

Definition 2.6. *(Aperiodic MC). A MC with transition probability P is aperiodic if $\forall i \in \mathcal{X}$ the greatest common divisor of the set $\{t \geq 1 : P_{ii}^t\} = 1$*

Intuitively irreducibility implies that all the states communicates with each other; aperiodicity instead makes sure that the chain is 'stochastic enough', i.e. it avoids the case of going

back to a particular state in regular time intervals. The following theorems (given without proofs) answer the previous questions.

Theorem 2.1. *(Existence and uniqueness of a stationary distribution). Let $(X_t)_{t\geq 0}$ be a MC.*

1. *if $|\mathcal{X}| < \infty$ then there is always a stationary distribution π.*

2. *if the MC is irreducible there is at most one stationary distribution.*

Theorem 2.2. *(Convergence theorem). Let $(X_t)_{t\geq 0}$ be an irreducible, aperiodic, π-stationary Markov chain, then:*

$$\lim_{t\to\infty} P_{ij}^t = \pi_j \quad \forall i, j \in \mathcal{X} \tag{12}$$

A last useful remark is to remind that the condition in (8) (also known as *global balance*) is not used in practice. To find a stationary distribution or to check that a distribution is indeed stationary one often use the so called *detailed balance equation*.

Definition 2.7. *(Detailed balance). A MC with transition matrix P is π-reversible if it satisfies the detailed balance equation:*

$$\pi_i P_{ij} = \pi_j P_{ji} \quad \forall i, j \in \mathcal{X} \tag{13}$$

Theorem 2.3. *Detailed balance implies global balance.*

2.2.3 Metropolis–Hastings algorithm

Suppose X is a random variable with some distribution P and one is interested in sampling from it. In many cases if X is high dimensional it cannot be done with naive sampling techniques and this is where Metropolis [10] comes to help. The idea behind the algorithm is to generate a Markov chain with a stationary distribution equal to P and ensuring that the chain converges to it. This is achieved by initially starting at a random state (i.e. a possible realization of X) and then iteratively proposing a new state, deciding whether to accept the proposal or not. Eventually if the former conditions are satisfied the state generated by the algorithm is distributed according to P.

Let x and x' be two different states and denote $q(x \to x')$ the probability of moving from x to x' (the proposal). There are two main versions of the algorithm, one where q is symmetric and one where q it's not symmetric. In this research the symmetric one is used, i.e. $q(x \to x') = q(x' \to x)$. If $P(x') \geq P(x)$ the move is always accepted, otherwise if $P(x') < P(x)$ it is accepted with probability $\frac{P(x')}{P(x)}$. All the moves that bring to a more probable state are accepted, while the algorithm is more 'cautious' accepting moves the diminish the probability.

Algorithm 1: Metropolis (P, q)

```
[1]  initialize x randomly;
[2]  repeat
[3]      let x' proposed move according to q;
[4]      if P(x') ≥ P(x) then
[5]          x = x', accept the move;
[6]      else
[7]          draw r uniformly at random in [0, 1];
[8]          if r < P(x')/P(x) then
[9]              x = x', accept the move;
[10]         else
[11]             do not accept the move;
[12]         end
[13]     end
[14] until convergence;
[15] return x
```

The following result provides a formal proof of the correctness of the procedure used by Metropolis.

Theorem 2.4. *Metropolis satisfies detailed balance.*

Proof. Denote by $Q(x \to x')$ the probability of going from x to x' during one iteration of the algorithm, and denote by $A(x \to x')$ the probability of accepting the move proposed according

to q. The detailed balance condition reads:

$$\begin{aligned} \pi_x Q(x \to x') &= \pi_{x'} Q(x' \to x) \\ \pi_x q(x \to x') A(x \to x') &= \pi_{x'} q(x' \to x) A(x' \to x) \\ \pi_x A(x \to x') &= \pi_{x'} A(x' \to x) \end{aligned} \tag{14}$$

Suppose $P(x) > P(x')$ then $\pi_x \frac{P(x')}{P(x)} = \pi_{x'}$ (analogous result in the other case).

$$1 = \sum_{x'} \pi_{x'} = \sum_{x'} \pi_x \frac{P(x')}{P(x)} = \frac{\pi_x}{P(x)} \implies \pi_x = P(x) \tag{15}$$

Moreover one can choose q according to 2.2 to ensure convergence. □

2.3 Statistical physics

As shown later, the simulated annealing algorithm is a variation of Metropolis with a particular target distribution, the Boltzmann distribution. This is one, if not the most, important concept from statistical physics. In this section the distribution formula is derived and some comments are made on its properties. This section refers to [9].

2.3.1 Microcanonical ensemble and entropy

The *microstate* of a system is defined as a particular configuration of a physical system composed of a large number of degrees of freedom, usally on the order of 10^{23}. For example, consider a gas in a container; then, a microstate of the system is given by the positions and momenta of all the particles. Another example is the case of a collection of N spins. A microstate in this case is a particular configuration where each spin can be up or down (only the eigenstates of the Hamiltonian are considered). The focus of this section will be more on the latter, i.e. on quantum systems with a discretized state space, although everything that is going to be presented can be extended also to classical systems. Now consider one of this systems and fix its energy E. Such system is called a microcanonical ensemble and because of the high degree of freedom there can be a lot of microstates of the system with the same

energy E. The fundamental assumption of statistical mechanics is the following:

For an isolated system in equilibrium, all accessible microstates are equally likely.

If we denote $\Omega(E)$ the number of microstates at energy E then the probability for the system of being in a specific state n is just $P(n) = \frac{1}{\Omega(E)}$. A key quantity related to the energy of a system is the *entropy*, defined as:

$$S(E) := k_B \; log \; \Omega(E) \tag{16}$$

Where k_B is the Boltzmann constant. Usually $\Omega(E) \sim e^N$ hence the entropy is proportional to the number of of particles in the system. Entropy can be interpreted as a measure of 'disorder', meaning that if its high the system can be found in a lot of possible microstates. Another crucial quantity is the temperature T, defined as:

$$\frac{1}{T} = \frac{\partial S}{\partial E} \tag{17}$$

2.3.2 Canonical ensemble and Boltzmann distribution

Consider a system S in contact with a much larger system called reservoir R. The reservoir is at a fixed temperature T. R is larger then S from an energy point of view, i.e. $E_S << E_R$ and the reservoir can accept or donate energy to S without changing its temperature. The total system is isolated so that $E_{TOT} = E_S + E_R$ is fixed. If S is in a particular microstate n then the energy of the reservoir must be $E_{TOT} - E_n$. The number of possible configurations is given by:

$$\Omega(E_{TOT}) = \sum_n \Omega_R(E_{TOT} - E_n) = \sum_n exp\left(\frac{S_R(E_{TOT} - E_n)}{k_B}\right) \tag{18}$$

Where the sum is over the possible states of S and the last equality comes from the inverse formula of the entropy. Since $E_{TOT} >> E_n$ we can Taylor expand the exponent, obtaining:

$$\Omega(E_{TOT}) = \sum_n exp\left(\frac{S_R(E_{TOT})}{k_B} - \frac{\partial S_R}{\partial E_{TOT}}\frac{E_n}{k_B}\right) = e^{\frac{S_R(E_{TOT})}{k_B}} \sum_n e^{-\frac{E_n}{k_B T}} = \delta \sum_n e^{-\frac{E_n}{k_B T}} \quad (19)$$

From this last formulation it's easy to see that the number of states of the total system where S is in the state n is $\delta e^{-\frac{E_n}{k_B T}}$. Using the fundamental postulate of statistical mechanics:

$$P(n) = \frac{\textit{states where S is in } n}{\textit{total number of states}} = \frac{e^{-\frac{E_n}{k_B T}}}{\sum_m e^{-\frac{E_m}{k_B T}}} = \frac{1}{Z} e^{-\frac{E_n}{k_B T}} \quad (20)$$

This is the Boltzmann distribution and Z, the normalization factor, is called the partition function.

Remark: The key takeaway from this section is that the temperature of the reservoir imposes a distribution on the states of the smaller system S. In particular:

1. For high temperatures all states of S are probable.

2. For small temperatures only states of S with low energy are probable, and in the limit only the ground state (state of lowest energy) is probable.

3 Algorithms

There are two types of algorithms to solve K-SAT instances: complete search and incomplete search algorithms. The first type of solvers either find a SAT-assignment or they prove that such an assignment doesn't exist, providing a so-called **UNSAT certificate**. On the other hand, incomplete search algorithms are based on heuristic procedures and usually have a threshold on the number of iterations; If no solution is found they return *I don't know*. The most used complete search algorithm is DPLL (and its variations). Regarding incomplete approaches WalkSAT and its variation are widely adopted, and most recently also the *survey propagation* algorithm from [1] seems promising.

What is the need of introducing simulated annealing [2] and what does simulated annealing have to do with satisfiability problems? There are several reasons. Firstly, an approach based on statistical physics such as survey propagation has shown excellent results. Since simulated annealing also relies on concepts from statistical mechanics, this is an indicator that it is a valid approach. Secondly, there are already researches taking this direction. In particular, in [8] annealing is compared to GSAT, an algorithm similar to walkSAT [7], and the authors have shown that simulated annealing performs at least as well as GSAT. Although walkSAT is mentioned in the paper as an empirically better solution to GSAT, there is no comparison with the annealing. Therefore, it could be interesting to see if the results they have obtained are applicable to walkSAT as well.

The DPLL and WalkSAT pseudo code used in this section is adapted from [5] while the annealing refers to [4].

3.1 DPLL

The name DPLL comes from the initials of their inventors, Davis, Putnam, Logemann and Loveland. In fact with DPLL we do not refer to a single algorithm but to a class of algorithms that all implements the core idea of exploring possible solutions in a recursive-like manner.

Denote with F a satisfiability instance in cnf (1), with A a partial assignment of the literals and with V the set of indices of unassigned literals. Moreover define $F|\{x_i = 0\}$ to be the formula obtained from F by assigning $x_i = 0$ and removing all clauses that contain the literal $\bar{x}_i$ (since under this assignment they are satisfied). Same reasoning applies to $F|\{x_i = 1\}$. Call the following recursive procedure starting from $F = F$, $A = \emptyset$, $V = \{1, ..., N\}$.

Algorithm 2: DPLL (F,A,V)

```
[1]  if V ≠ ∅ then
[2]      Choose an index i ∈ V ;
[3]      B = DPLL(F|{x_i = 0}, A ∪ {x_i = 0}, V\i);
[4]      if B=UNSAT then
[5]          B = DPLL(F|{x_i = 1}, A ∪ {x_i = 1}, V\i);
[6]      else
[7]          return A ∪ {x_i = 0} ∪ B;
[8]      end
[9]      if B=UNSAT then
[10]         return B;
[11]     else
[12]         return A ∪ {x_i = 1} ∪ B;
[13]     end
[14] else
[15]     if F has no clause then
[16]         return A;
[17]     else
[18]         return UNSAT;
[19]     end
[20] end
```

The algorithm can be represented as a walk in the decision tree. When it finds a contradiction, i.e. it reaches an 'UNSAT' leaf of the tree, it backtracks and searches a different branch. Deciding on which variable the next branching will be done is not trivial, and can result in strongly varying performances. Whenever the DPLL procedure does not return a SAT-assignment, the formula is UNSAT.

3.2 WalkSAT

WalkSAT is an incomplete search algorithm that implements a random walk in the space of all possible assignments. The random walker starts from a random configuration, then (if the current configuration is not a SAT-assignment) with probability p a random spin in an unsatisfied clause is flipped and with probability $1-p$ the literal leading to the largest positive increase in the number of satisfied clauses is flipped. This procedure is repeated for a fixed

number of iterations. If no SAT-assignment is found, the algorithm returns *I don't know*.

Denote with $\underline{x}$ the assignment vector (or state of the system) $\underline{x} = (x_1, ..., x_N)$, with $E(\underline{x})$ the number of clauses violated by $\underline{x}$ and with $\underline{x}^{(i)}$ the assignment $\underline{x}$ where the i-th literal is flipped.

Algorithm 3: WalkSAT (F, number of flips, p)

```
[1]  initialize x randomly;
[2]  for number of flips do
[3]      if x satisfies F then
[4]          return x;
[5]      else
[6]          draw r uniformly at random in [0, 1];
[7]          if r < 1 − p then
[8]              let Δ_i = E(x) − E(x^(i));
[9]              flip x_i for which Δ_i is maximal;
[10]         else
[11]             choose violated clause a uniformly at random;
[12]             flip a literal in a uniformly at random;
[13]         end
[14]     end
[15] end
[16] return IDK;
```

This strategy works reasonably well if p is properly optimized. The greedy step of choosing the literal minimizing the difference drives the assignment toward 'quasi-solutions' (i.e. low number of unsatisfied clauses), while the other term allows to escape from local minima. This idea of trade-off between greedy steps and escaping minima is implemented differently in the simulated annealing, as explained in the next section.

3.3 Simulated annealing

3.3.1 Introduction to the algorithm

Simulated annealing, (often abbreviated as SASAT in the context of satisfiability), shares similarities with walkSAT as an incomplete search method. It employs an annealing algorithm, a versatile approach applicable to various combinatorial optimization challenges, such as the

Traveling Salesman Problem (TSP) among others. For clarity, the notation for random K-SAT problems is used in this section.

The algorithm is based on the analogy between the simulation of the annealing of solids and the problem of solving large combinatorial optimization problems. For this reason the algorithm became known as *simulated annealing*. In condensed matter physics, annealing denotes a physical process in which a solid in a *heat bath* is heated up by increasing the temperature of the heat bath to a maximum value at which all particles of the solid randomly arrange themselves in the liquid phase, followed by cooling through slowly lowering the temperature of the heat bath. In this way, all particles arrange themselves in the low energy ground state of a corresponding lattice, provided the maximum temperature is sufficiently high and the cooling is carried out sufficiently slowly. More specifically, at each temperature value T, the solid is allowed to reach thermal equilibrium, characterized by a probability of being in a state with energy E given by the *Boltzmann distribution* (20).

As the temperature decreases, the Boltzmann distribution concentrates on the states with lowest energy and finally, when the temperature approaches zero, only the minimum energy states have a nonzero probability of occurrence. If the cooling is too rapid, i.e. if the solid is not allowed to reach thermal equilibrium for each temperature value, defects can be frozen into the solid and metastable amorphous structures can be reached rather than the low energy crystalline lattice structure.

The simulated annealing algorithm implements this idea using a sequence of Metropolis algorithms (1) with target distribution equal to the Boltzmann distribution, evaluated at a sequence of decreasing values of the temperature. At each temperature Metropolis ensures that the state of the system is drawn from the Boltzmann distribution, and the cooling schedule makes sure that the system does not get stuck in any local minima, so that eventually the returned state should be the ground state of the system, i.e. the state of minimum energy.

3.3.2 SASAT

All the necessary notation and notions to apply the algorithm to the specific case of random K-SAT problems are now introduced. As before $\underline{x} = (x_1, ..., x_N)$ denotes the state of the

system (the literal assignment), and $E(\underline{x}) =$ *number of violated clauses* its *energy*, so that minimizing the energy corresponds to finding a SAT-assignment. if $\underline{x}$ and $\underline{x}'$ are two different states with $P_B(\underline{x}) < P_B(\underline{x}')$, this implies that $E(\underline{x}) > E(\underline{x}')$ and the Metropolis acceptance probability becomes:

$$exp\left\{-\frac{1}{k_B T}[E(\underline{x}) - E(\underline{x}')]\right\} \tag{21}$$

In practice we can drop the Boltzmann constant since it gets absorbed by the temperature. The last ingredient is the proposal probability q. There are multiple choice that one can make, but a common one is to propose uniformly at random a state that differs only for one literal. Note that this choice ensures that the underlying Markov chain is irreducible and aperiodic.

Algorithm 4: SASAT (F, *temperature schedule*)

```
[1]  initialize x randomly;
[2]  repeat
[3]      repeat
[4]          if x satisfies F then
[5]              return x;
[6]          else
[7]              choose i ∈ {1, ..., N} uniformly at random;
[8]              if E(x^(i)) ≤ E(x) then
[9]                  x = x^(i), accept the move;
[10]             else
[11]                 draw r uniformly at random in [0, 1];
[12]                 if r < exp{−(1/T)[E(x^(i)) − E(x)]} then
[13]                     x = x^(i), accept the move;
[14]                 else
[15]                     do not accept the move;
[16]                 end
[17]             end
[18]         end
[19]     until convergence;
[20]     update T;
[21] until final temperature;
[22] return IDK;
```

4 Experiments

4.1 Experimental setup

4.1.1 Generate random K-SAT instances

A key step for the comparison is to generate good random instances. The term 'good' refers to instances being uniformly sampled from the space of all possible instances. (having fixed N,M,K). As explained in 2.1.2 the number of possible instances can become very large. In order to avoid any bias introduced by naive sampling techniques the Metropolis algorithm has been employed to generate instances (In this case every proposal is accepted because the target distribution is uniform). This can be formalized with the two procedures 5 and 6. The results are shown in fig.2.

Algorithm 5: Initial formula (K, N, M)

```
[1]  initialize an empty formula F;
[2]  repeat
[3]  |  select uniformly at random K literals;
[4]  |  select uniformly at random K negations;
[5]  |  create the clause a;
[6]  |  if a not in F then
[7]  |  |  add a to F;
[8]  |  else
[9]  |  |  continue;
[10] |  end
[11] until |F| has length M;
[12] return F;
```

Algorithm 6: Random instance (initial formula F)

```
[1] repeat
[2] |  select a clause a uniformly at random;
[3] |  propose a new clause a' not in F
[4] |  accept the move;
[5] until convergence;
[6] return F
```

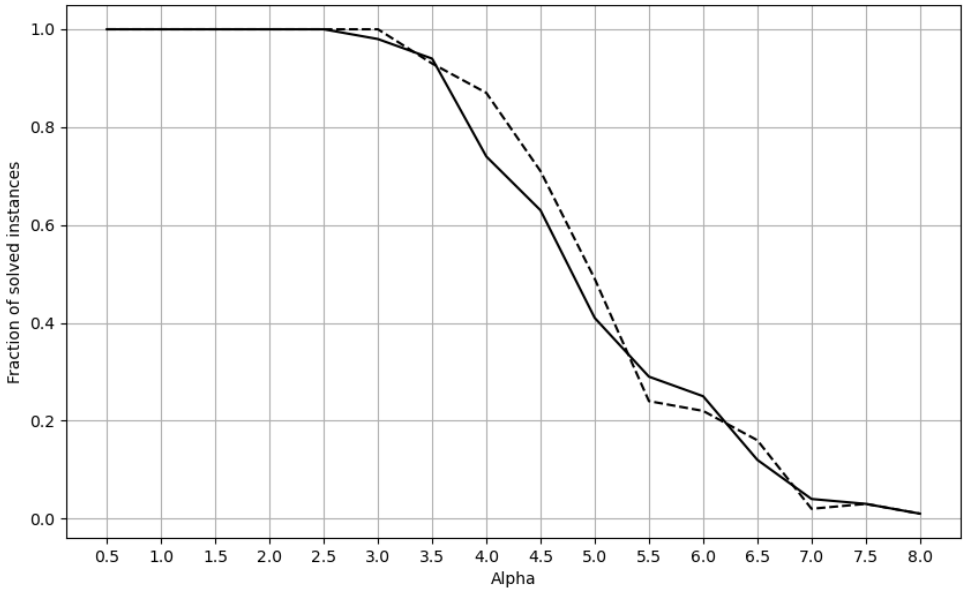


Figure 2: Plot showing the performance of DPLL on random instances with $K = 3$. The dashed line corresponds to $N = 10$ while the solid one to $N = 5$. The plot is obtained averaging over 100 trials, and is qualitatively very similar to the one in fig.1. The phase transition is less sharp because of the lower number of literals. This result ensures that the sampling procedure behaves correctly.

4.1.2 Ensuring a fair comparison

The second step is to decide a comparison metric and a way to ensure that the comparison between walkSAT and SASAT is fair. Regarding the former a natural approach is to use the average number of solved instances, i.e. the algorithm is run over I instances for fixed values of N,K,M (or α) and an average of how many time the algorithm returned a SAT-assignment is used as a metric. More specifically denote $S^{(i)}$ the solution of a general algorithm on instance i, with $S^{(i)} = 1$ whenever the the algorithm has returned a SAT-assignment and $S^{(i)} = 0$ otherwise, then the performance metric used is:

$$\frac{1}{I}\sum_{i=1}^{I}\mathbb{1}(S^{(i)} = 1) \tag{22}$$

Concerning the fair comparison problem instead, one needs to consider the time complexities of walkSAT and SASAT. Given a particular state of the system, computing its energy is $O(MK)$, since it implies counting the number of unsatisfied clauses, and each clause contains

K literals. In the worst case walkSAT performs N of this operations in one iteration during the greedy step, while SASAT only computes the energy of the new proposed configuration. If one lets walkSAT run for T iterations, to ensure a fair comparison SASAT must be run for NT iterations in order for both algorithms to have the same complexity overall.

The previously computed bounds can be reached only if the energy is computed efficiently. Specifically, apart from the initial iteration, both algorithms must calculate energy differences between configurations with a flipped spin rather than computing the energy of each configuration. Knowing all the clauses where a specific literal appears allows for focusing solely on changes within those clauses when the literal is flipped, rather than reassessing the entire formula. This is crucial for obtaining faster algorithms.

4.1.3 Parameters

All the plots in the subsequent sections are made using the following parameters that were revealed to be empirically the bests or achieved best trade-off between performance and running time. **Remark:** a correct choice of p and of the temperature range are crucial to achieve good performances.

walkSAT		
Parameter	**Description**	**Value**
n_{iter}	Number of iterations	$4N$
p	Greedy choice probability	0.5

SASAT		
Parameter	**Description**	**Value**
n_{iter}	Number of iterations	$4N^2$
T_i	Initial temperature	1.0
T_f	Final temperature	0.0
t^{eq}	Equilibrium time (i.e. number of iterations at fixed temperature)	1
$T(i)$	Temperature at i-th iteration	fig.3

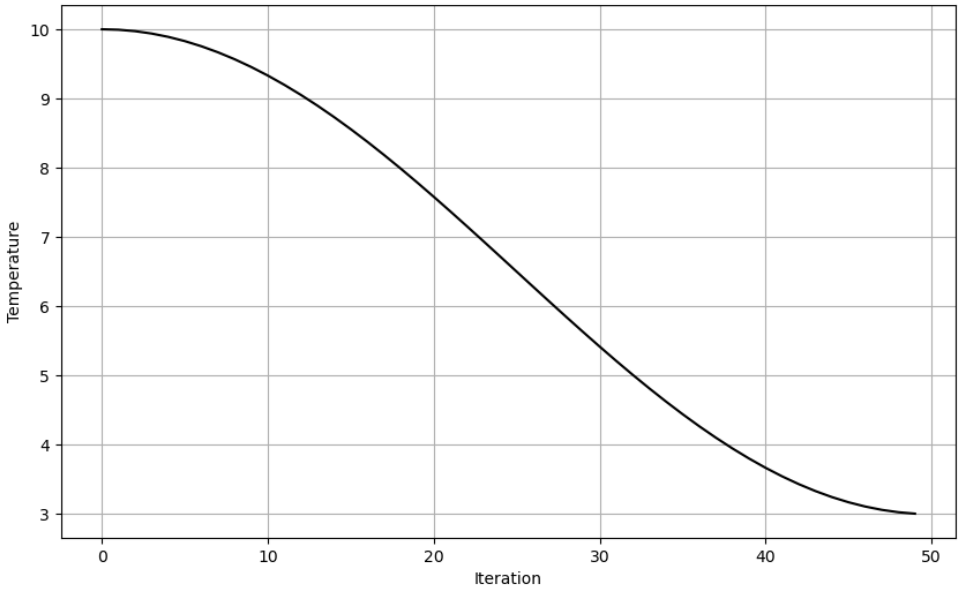


Figure 3: Temperature schedule function: $T(i) = \frac{T_i+T_f}{2}\cos\left(\frac{\pi}{n_{iter}}i\right) + \frac{T_i-T_f}{2}$. In the plot the values used are $T_i = 10$, $T_f = 3$, $n_{iter} = 50$.

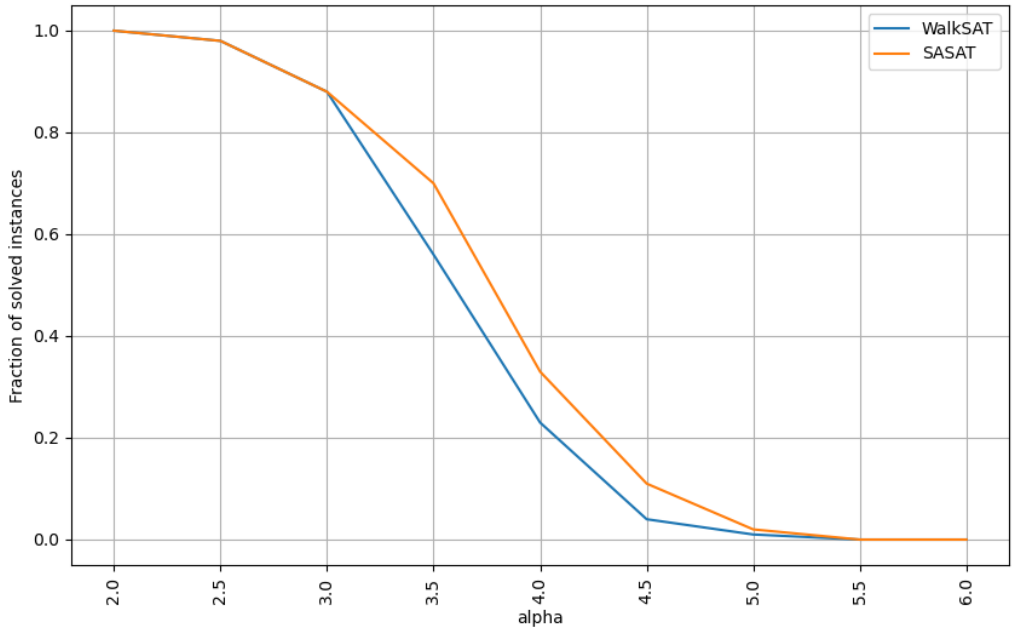
4.2 Comparison on general instances

The comparison yielded the results depicted in fig.4 and fig.5. Interestingly, a minimal discrepancies in the performance between walkSAT and SASAT is observed. The phase transition from satisfiable (SAT) to unsatisfiable (UNSAT) instances is clearly discernible across all plots.

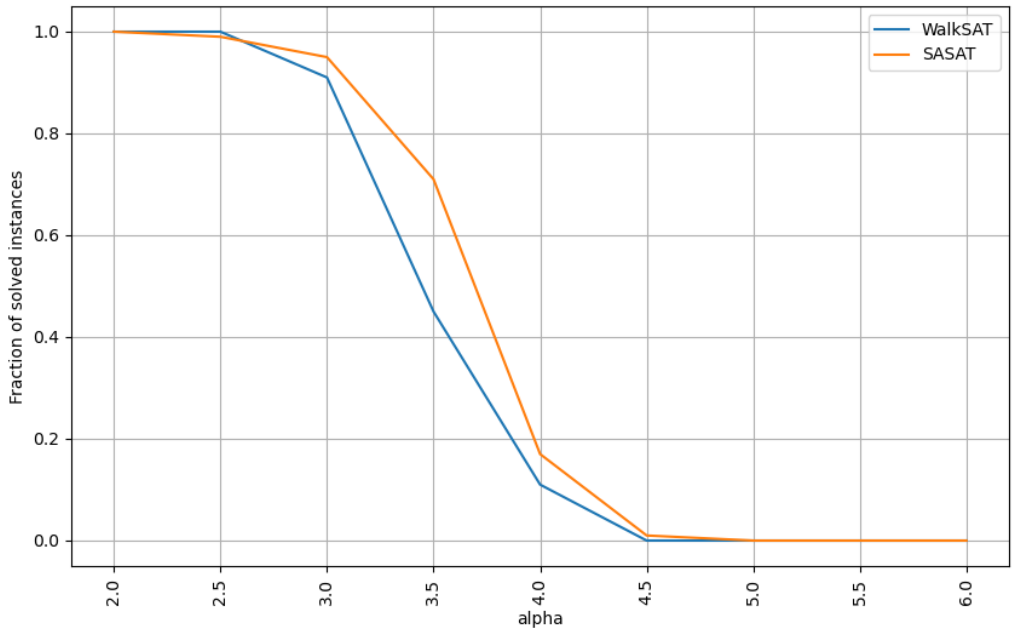
However, the transition from compact solutions to scattered ones it's not visible. This could be attributed to the focus on time performance rather than the algorithm's capability to find a solution. Even if the solutions aren't compact within the space of all assignments, the algorithm can still discover them. Nonetheless, it might require more time. Given the utilization of a relatively high number of iterations, this potential time-related issue it's not evident.

To clarify this last point a time analysis for the case $N = 100$ and $K = 3$ has been performed, obtaining the result in fig.6. Actually in the figure the number of steps of walkSAT are normalized taking into account the the cost of a step, obtaining an overall cost proportional to $n[p + (1 - p)N]$, with n being the number of iterations.

As before, in terms of number of solved instances, they both perform very similarly, but

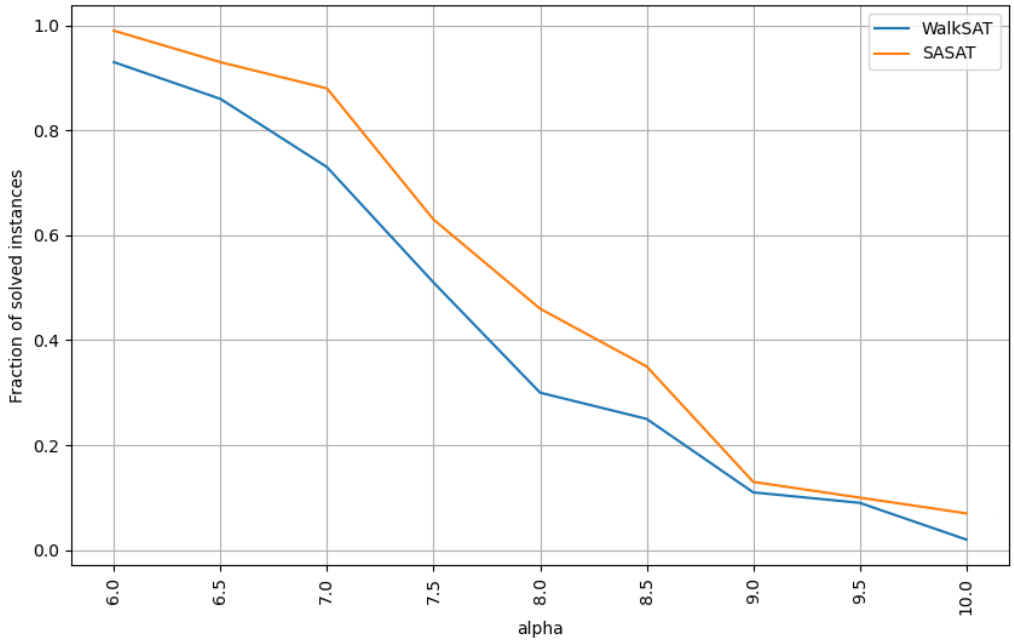


(a) $N = 50,\ K = 3$

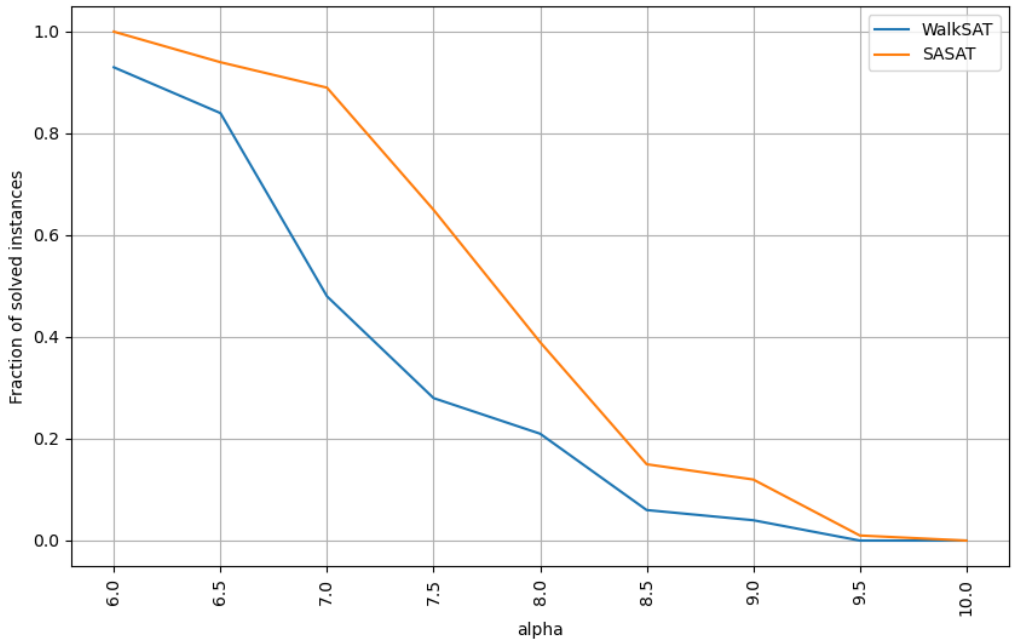


(b) $N = 100,\ K = 3$

Figure 4: 3-SAT



(a) $N = 50,\ K = 4$



(b) $N = 100,\ K = 4$

Figure 5: 4-SAT

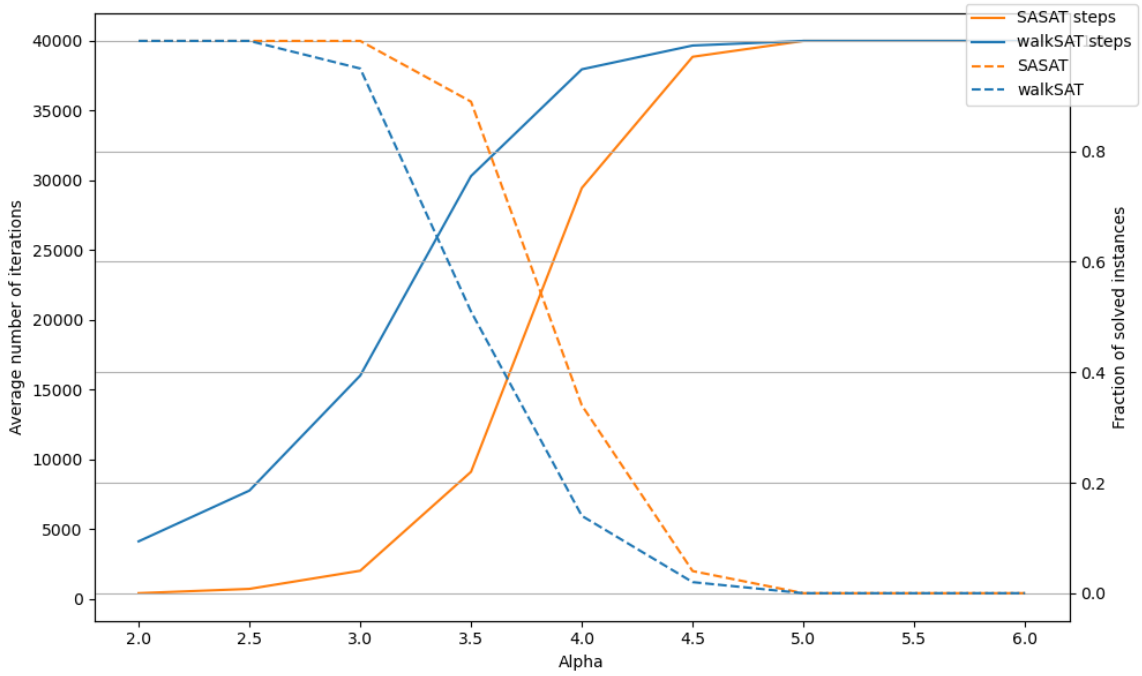


Figure 6: Time comparison with with $N = 100$, $K = 3$. Solid lines: average number of iterations of walkSAT and SASAT on 100 trials. Dashed lines: fraction of solved instances.

SASAT seems to converge faster. From the data one can conclude that the cost of taking the greedy step it's not enough to provide a real computational benefit compared to the annealing.

4.3 Comparison close to phase transition

The most interesting instances are the one close to the phase transition between SAT and UNSAT problems. In this section the focus is on such instances. In particular, all the following analysis are made on problems with a single solution.

Two different paths have been taken. A first approach tries to explain why the greedy step of walkSAT doesn't work especially close to the transition. To do so an instance is generated and the shortest path from any node to the (unique) solution is found (nodes correspond to states). Then for every shortest path the probability of taking it is calculated and an average is performed over all paths. This procedure is done considering the transition probabilities of walkSAT and SASAT to compare them. This gives a rough measure on the 'power' of a greedy strategy. The results are reported in fig.7. As N increases the probability of taking the shortest path from a node to a solution becomes the same for both algorithms. Moreover

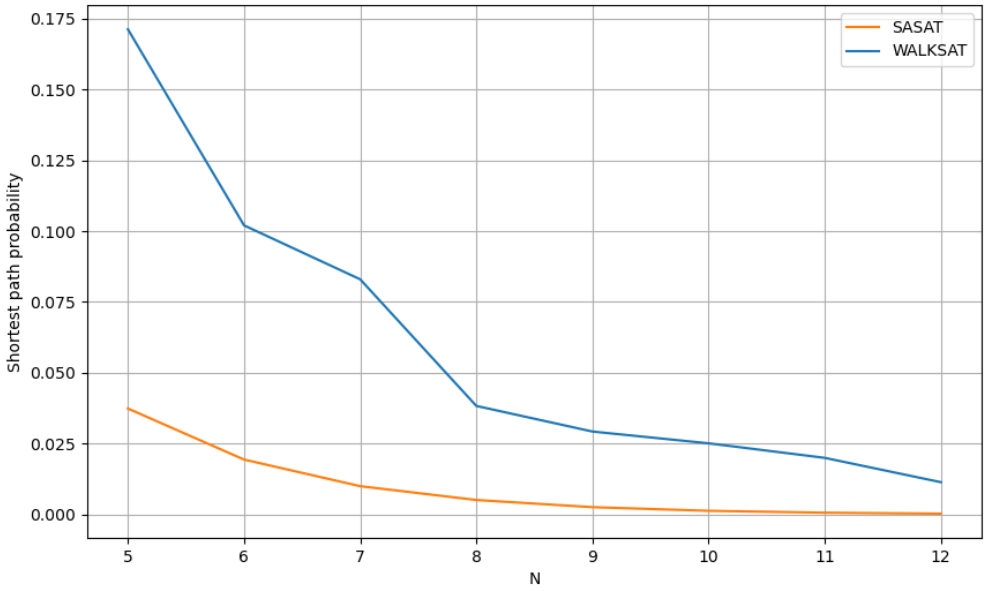


Figure 7: Probability of taking the shortest path from a node to the single SAT-assignment for different values of N with $K = 3$.

by giving unlimited iterations to both algorithm, and computing the average cost, it appears that walkSAT is always slightly above SASAT (fig.8). From a practical point of view as N increases one should pick bigger values for the parameter p, and avoid wasting resources on the greedy procedure.

The second approach is a more general approach to study the convergence behaviour of Markov chains. In fact the two algorithms implement two random walkers in the space of possible configurations. The space is the same for both walkers, what changes are the transition probabilities of moving from a state to another which depends on the specific algorithm.

Let P be a general transition matrix, thanks to *Perron–Frobenius theorem* the maximum eigenvalue is 1, and all other eigenvalues have norm smaller then one. Assuming that P is diagonalizable a general initial distribution d can be written as:

$$d = c_1 v_1 + c_2 v_2 + ... + c_n v_n \tag{23}$$

where $v_1, ..., v_n$ are eigenvectors with eigenvalues $\lambda_1, ..., \lambda_n$ in decreasing order. A stationary distribution correspond to the left eigenvector of eigenvalue 1. Hence if we simulate the chain

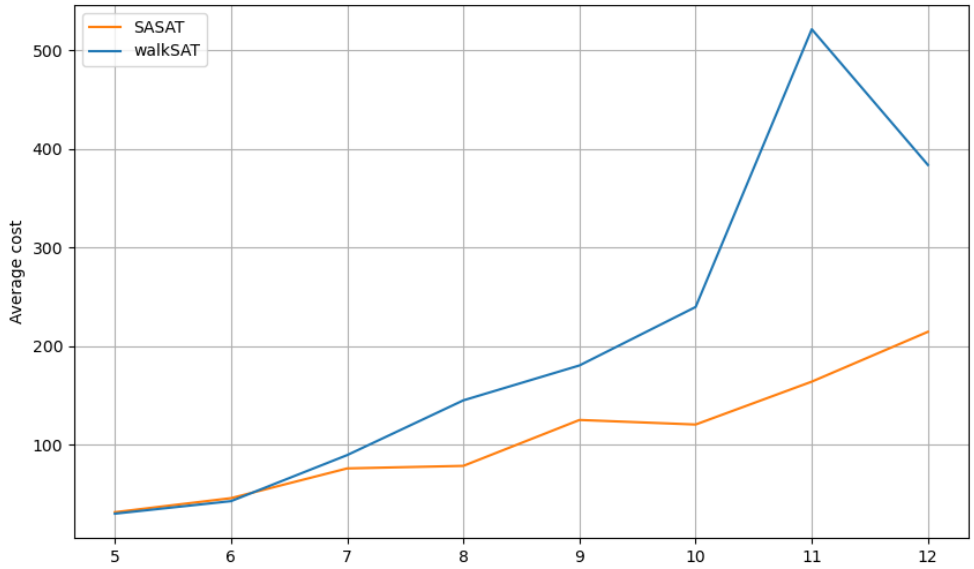


Figure 8: Average computational cost over 100 instances as a function of N with $K = 3$.

by iteratively applying P to the initial distribution at time t one obtains:

$$d(t) = c_1\lambda_1^t v_1 + c_2\lambda_2^t + ... + c_n\lambda_n^t v_n = v_1 + c_2\lambda_2^t + ... + c_n\lambda_n^t v_n \tag{24}$$

Since $|\lambda_i| < 1$ for $i \neq 1$ the convergence depends on the second larges eigenvalue of P. Denote by $P^{walkSAT}$ and P^{SASAT} the transition matrices of the relative algorithms, with a fixed temperature for the annealing to impose time homogeneity. By generating instances and calculating the norm of the second eigenvalues for both cases one obtains the result in fig.9 which shows that they both have similar convergence rate as N increases.

5 Conclusions and future research

The purpose of the research was to produce a comparison between walkSAT and simulated annealing on random satisfiability problems. The research was conducted on instances with a relatively small number of variables due to computational limitations, and as already mentioned in the introduction, each conclusion cannot be automatically applied to problems with many variables.

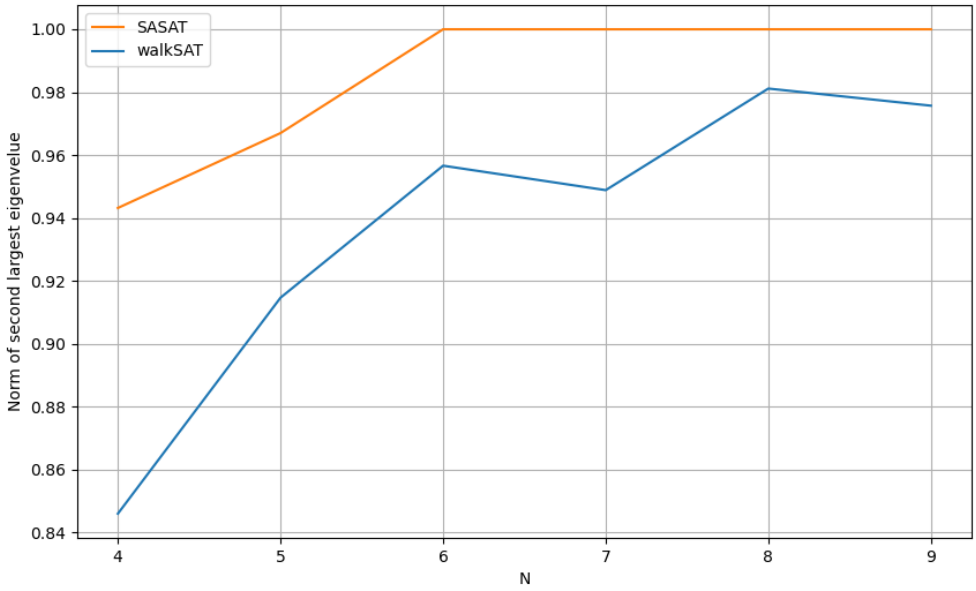


Figure 9: Norm of the second largest eigenvalue for different values of N with $K = 3$.

The main results are three. First, in terms of number of solved instances the algorithms performs very similarly. On the other hand, when one considers also the time and computational cost for convergence SASAT is slightly faster, in particular near the phase transition between SAT and UNSAT. This is likely due to the fact that near critical alpha values, the energy landscape is highly irregular, and changing even a single spin can significantly change the energy, making it more difficult to apply a greedy strategy. This reflects the fact that the problem remains NP-complete, and at any given moment, there is no general optimal strategy for making a decision that reliably approaches the problem's solution.

A third remark is that both algorithm are very sensible to the parameter p and temperatures, hence depending on the problem one should carefully tune them. This gives inspiration for further research on techniques to determine a priori good values for this parameters.

References

[1] A. Braunstein, M. Mezard, and R. Zecchina. *Survey propagation: an algorithm for satisfiability*. 2006. arXiv: cs/0212002 [cs.CC].

[2] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing". In: *Science* 220.4598 (1983), pp. 671–680. DOI: 10.1126/science.220.4598.671. eprint: https://www.science.org/doi/pdf/10.1126/science.220.4598.671. URL: https://www.science.org/doi/abs/10.1126/science.220.4598.671.

[3] Florent Krzakała et al. "Gibbs states and the set of solutions of random constraint satisfaction problems". In: *Proceedings of the National Academy of Sciences* 104.25 (June 2007), pp. 10318–10323. ISSN: 1091-6490. DOI: 10.1073/pnas.0703685104. URL: http://dx.doi.org/10.1073/pnas.0703685104.

[4] Peter J. M. van Laarhoven and Emile H. L. Aarts. "Simulated annealing". In: *Simulated Annealing: Theory and Applications*. Dordrecht: Springer Netherlands, 1987, pp. 7–15. ISBN: 978-94-015-7744-1. DOI: 10.1007/978-94-015-7744-1_2. URL: https://doi.org/10.1007/978-94-015-7744-1_2.

[5] Marc Mezard and Andrea Montanari. *Information, Physics, and Computation*. USA: Oxford University Press, Inc., 2009. ISBN: 019857083X.

[6] J. R. Norris. *Markov Chains*. Cambridge, UK: Cambridge University Press, 1997.

[7] Bart Selman, Hector Levesque, and David Mitchell. "A New Method for Solving Hard Satisfiability Problems". In: July 1992.

[8] William Spears. "Simulated Annealing for Hard Satisfiability Problems". In: (July 1998).

[9] David Tong. "Lectures on Statistical Physics". In: (2012). URL: https://www.damtp.cam.ac.uk/user/tong/statphys.html.

[10] Wikipedia contributors. *Metropolis–Hastings algorithm — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-April-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Metropolis%E2%80%93Hastings_algorithm&oldid=1215964544.