**Computer Programming Assignment 2024**

Vittorio Manfriani – 3245185 – BEMACS

**Introduction**

In this report, we investigate the use of Simulated Annealing, a stochastic optimization technique inspired by the physical process of annealing in metallurgy, to solve the k-SAT problem. This study explores the performance of simulated annealing in solving k-SAT instances of varying complexity, characterized by parameters such as the number of variables (N), the number of clauses (M), and setting the clause size (k) equal to three. Key points relate around:

1. Choosing the parameters that optimize both accuracy and computational cost.
2. Evaluating the empirical probability of finding a satisfying solution for different problem sizes and annealing parameters.
3. Analyzing the impact of algorithmic parameters such as the number of Monte Carlo steps, annealing schedule, and cooling rate on solution quality and computational efficiency.
4. Investigating the scaling behavior of the algorithm, including the relationship between the number of variables and the algorithmic threshold, where the transition from solvable to unsolvable instances occurs.

**Parameters selection**

For the k-SAT problem, the key parameters that require attention are the initial and final temperatures ( and ), the number of annealing steps, and the number of Monte Carlo steps (MCMC steps) at each temperature. Each parameter plays a distinct role in balancing exploration and exploitation, making their selection critical for achieving high solution quality and computational efficiency.

* is set to 1 to ensure that, at the start of the annealing process, the acceptance rate consistently falls within an acceptable range of 30% to 80%, regardless of the specific values of N and M on which the algorithm operates.
* is set to 10.
* **MCMC** is set to 200 to minimize running time and keep accuracy at a sufficient level.
* **Annealing steps** are set to 20 to minimize running time and keep accuracy at a sufficient level.
* **Seed** is set to45.

Below is the plot of the evolution of the acceptance rate for N = 200 and M = 200.

![A graph with a line

Description automatically generated]()

It is important to note that, starting from state number 3, the cost reaches 0. From this point onward, all subsequent states have equal cost. For completeness, the plot below illustrates the results for N = 200 and M = 800 with no states that have equal cost.

**![A graph with a line

Description automatically generated]()**

**Ability to find solution as M grows large**

As M increases, the Simulated Annealing algorithm does not consistently succeed in solving the 3-SAT problem or minimizing the cost to zero. Analyzing the algorithm’s performance across multiple values of M (up to 1000) while keeping N fixed at 200 reveals a significant trend: beyond a certain point (specifically, M > 400), the algorithm becomes unable to always reliably find a solution.

To give a better picture of the limits of the simulated annealing, we computed the empirical probability of of solving a random instance, at fixed N and M over 30 trials. Below is the plot of how this probability changes for different values of M and N = 200 for fixed parameters of the annealing algorithm.

A graph with a line

Description automatically generated

As the graph clearly illustrates, the Simulated Annealing algorithm struggles to consistently find a solution when M is sufficiently large. Moreover, for N > 800, it becomes completely incapable of finding a solution.

**Algorithmic Threshold**

The algoritmic treshold () is the value of M at which the at which the empirical probability of solving a clause with Simulated Annealing is equal to 0.5. To find the value of for different values of N, I implemented a binary search algorithm, which I evaluated to be the best choice to work with the monotonically decreasing function in terms of both accuracy and computational efficiency. Below are reported in the graph the results I got.

A graph with a line

Description automatically generated

The graph shows that the Algorithmic Threshold, , increases with  N , which is intuitive because a larger number of variables  N  provides more flexibility to satisfy the constraints, making it easier for Simulated Annealing to find a solution. However, as  M increases, the problem becomes harder to solve. For larger  N , the system can handle more clauses before the solution space collapses, meaning    must be higher to reach a probability of 0.5 of solving the problem. This reflects the balance between the increased degrees of freedom provided by more variables and the tightening constraints introduced by additional clauses, with  transitioning sharply near    as solvability shifts from likely to unlikely.

**Behaviour of around the Algorithmic Threshold**

A graph of a graph of a graph

Description automatically generated with medium confidence