**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 importance**

**[paragraph on parameters and table]**

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

[picture]

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.

**[picture]**

**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.

[picture]

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.

**Parameters selection for Probability Analysis**

In the next section, we will run Simulated Annealing for various values of M and N . To ensure the performance comparisons are meaningful, we will use fixed parameters throughout the experiments. The selected parameter values are as follows:

* 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 500 to keep a reasonable running time and maintain accuracy at a sufficient level.
* **Annealing steps** are set to 20 to keep a reasonable running time and maintain accuracy at a sufficient level.

**Algorithmic Threshold**

The algorithmic threshold is the value of M at which the empirical probability of solving a clause with Simulated Annealing equals 0.5. To determine for different values of , I implemented a binary search algorithm. This approach was chosen for its ability to efficiently handle the monotonically decreasing nature of ensuring both accuracy and computational efficiency. Referring to the chart below:

[picture 1]

For , as M approaches , the solution space shrinks significantly due to the increasing number of constraints, resulting in a rapid decline in This sharp drop illustrates the system’s critical transition from a solvable to an unsolvable phase. The analysis was extended to different values of , and the graph below shows how the value of algorithmic threshold changes as increases.

A graph with a line

Description automatically generated

The graph shows that the Algorithmic Threshold, , increases with  , which is intuitive because a larger number of variables    provides more flexibility to satisfy the constraints, making it easier for Simulated Annealing to find a solution. However, as  increases, the problem becomes harder to solve. For larger  , 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.

Also looking at different plots of demonstrate this same interplay between the solution space, constrained by the number of clauses , and the degrees of freedom introduced by the variables . As approaches , again transitions sharply, with solvability dropping rapidly from likely to unlikely. However, the effect is smoother for larger , meaning the slope of the transition becomes less steep. This happens because larger provides more degrees of freedom, meaning the system has more flexibility to satisfy the constraints imposed by . When is small, each added clause significantly reduces the solution space, causing to drop quickly as increases. In contrast, for larger , the solution space is much larger initially, so the constraints imposed by reduce it more gradually. As a result, the transition from solvable to unsolvable is less abrupt, making the slope of the curve near appear smoother. This reflects how the balance between variables and constraints governs the behavior of .

[picture 1]

**Curve Collapsing**