Project 1 - Simulated Annealing for Constructing Covering Arrays

Yukta Batra

***Abstract*—This report presents an implementation of simulated annealing (SA) to generate covering arrays, a crucial element in combinatorial testing. The objective is to efficiently generate test cases that encompass various parameter combinations. The study examines the effectiveness of SA in optimizing covering arrays under different parameter constraints.**

I. Introduction

A. Motivation

Ensuring software reliability requires robust testing strategies. One such approach is *combinatorial testing*, which systematically evaluates different input combinations. Covering Arrays (CAs) serve as a structured methodology for generating these combinations. This project explores the use of simulated annealing to construct CAs, leveraging an optimization-based approach to enhance testing efficiency.

B. Objective

This project aims to evaluate the performance of simulated annealing in generating optimal covering arrays for systems with varying parameters. The goal is to develop a structured method for identifying test cases that minimize missing combinations, thereby improving test coverage.

II. Background

A. System Model

The system under evaluation consists of three parameters: *Operating System (OS), Web Browser, and Database*. Each parameter can take on two possible values, yielding a total of $2^3 = 8$ combinations.

TABLE I
System with 3 Parameters, Each with 2 Values

OS	Browser	Database
Linux	Firefox	MySQL
Windows	Chrome	Oracle

B. Covering Arrays

A *Covering Array (CA)*, denoted as $CA(N; t, k, v)$, is a combinatorial matrix where each $N \times k$ structure ensures that every $N \times t$ subarray includes all possible combinations of v^t values at least once. In this study, we construct CAs with:

- Strength $t = 2$ (each pairwise combination must be covered)
- Parameter values $v = 2$
- Variable number of parameters k

The objective is to construct an optimal CA while keeping N (the number of rows) as low as possible.

III. Proposed Approach

A. Simulated Annealing Overview

Simulated Annealing (SA) is a metaheuristic optimization algorithm inspired by the annealing process in metallurgy. It starts with a random initial solution and progressively explores neighboring solutions, occasionally accepting worse solutions to escape local optima. The algorithm gradually lowers the temperature, shifting from exploration to exploitation, ensuring convergence to a near-optimal solution.

B. Implementation Methodology

The SA algorithm follows these steps:

1) **Initialization:** A random matrix is generated as a potential covering array.
2) **Objective Function:** The number of missing combinations in $N \times t$ subsets is calculated.
3) **Neighborhood Exploration:** A candidate solution is created by modifying a random parameter in the matrix.
4) **Acceptance Probability:** If the new solution reduces the objective function, it is accepted. Otherwise, it is accepted with a probability dependent on the temperature (T).
5) **Cooling Schedule:** Temperature decreases geometrically, controlling the exploration-exploitation balance.
6) **Termination Criteria:** The process stops if an optimal solution is found, the temperature reaches a threshold, or the solution stagnates for a prolonged period.

IV. Experimental Results

A. Experiment Setup

To evaluate the effectiveness of the algorithm, experiments were conducted for $k = 5, 6, 7$. Each case was executed 30 times, yielding a total of 90 runs. The experiment parameters include:

- **Initial Temperature** $T = k$
- **Cooling Rate** 0.99
- **Frozen Factor** $v^t \times (kt)$

B. Observations

- The algorithm successfully generated covering arrays for most test cases.
- Higher values of k required longer execution times to reach an optimal solution.

- In cases where the algorithm froze, increasing N marginally improved results.
- The success rate varied across different values of k, with smaller k values achieving higher convergence rates.

V. Conclusion

A. Key Findings

The simulated annealing approach effectively generates covering arrays under different parameter settings. The ability of SA to escape local optima and progressively improve solutions makes it a viable method for combinatorial testing. However, the choice of cooling parameters and neighborhood functions plays a crucial role in determining the success rate.

B. Future Scope

Future research can focus on:

- **Adaptive Cooling Strategies** to dynamically adjust temperature decay.
- **Hybrid Approaches** combining SA with genetic algorithms for enhanced performance.
- **Scalability Testing** to evaluate the feasibility of SA on larger parameter spaces.