Name: Adel Alkhamisy
Project 1

Introduction:

To evaluate systems with various parameters and their potential combinations, combinatorial objects known as covering arrays (CAs) are used. An N x k matrix of N elements can be used to represent a CA, where each row corresponds to a test case and each column to a parameter with v potential values. Every N x t subarray in the matrix CA(N; t, k, v) contains every possible combination of v^t symbols at least once. Simulated annealing (SA) is used in this project to determine the smallest number of test cases (N) necessary to cover all possible combinations in each N x t subarray.

Background:

Simulated Annealing is a probabilistic optimization method that draws inspiration from the metallurgical annealing procedure. The key concept is to begin with a first solution and then enhance it by creating a series of subsequent solutions based on a probabilistic criterion. When an optimal solution to a combinatorial optimization issue is difficult to find because of the size of the search space, SA is an appropriate answer. In this project, SA is used to determine the smallest size of a covering array that meets a specified condition.

Proposed Approach:

I used SA to determine the smallest size of a covering array for various values of k, where k is the total number of parameters. To make the problem simpler, I fixed t and v to be equal to 2. The objective function counts the number of missing combinations in each N x t subarray, and the initial solution is generated at random. To construct N neighbors, the neighborhood function randomly chooses a column and swaps the symbols in each row. The process finishes when the frozen factor is achieved or a solution is discovered. The temperature is reduced geometrically. It starts at Ti =k and decreased by a factor of 0.99. The frozen factor computed by formula 1.

$$\emptyset = v^t \binom{k}{t} \qquad (1)$$

Proposed formula to find N. The lower bound of N=$\binom{k}{t}$. For example, in case of t=2, v=2 and k=3; the minimum number of combinations will be $\binom{3}{2} = 3$ and the combinations are 01, 10, and 11 excluding the case 00 if we are considering the software configurations to be the k because we are not interested in testing the software with no configurations at all and that represent the case 00. I used both N=$\binom{k}{t}$ and N=k^2 in my implementations. I used N=k^2 to give the search better chance of finding a covering array since N=k^2 > $\binom{k}{t}$.

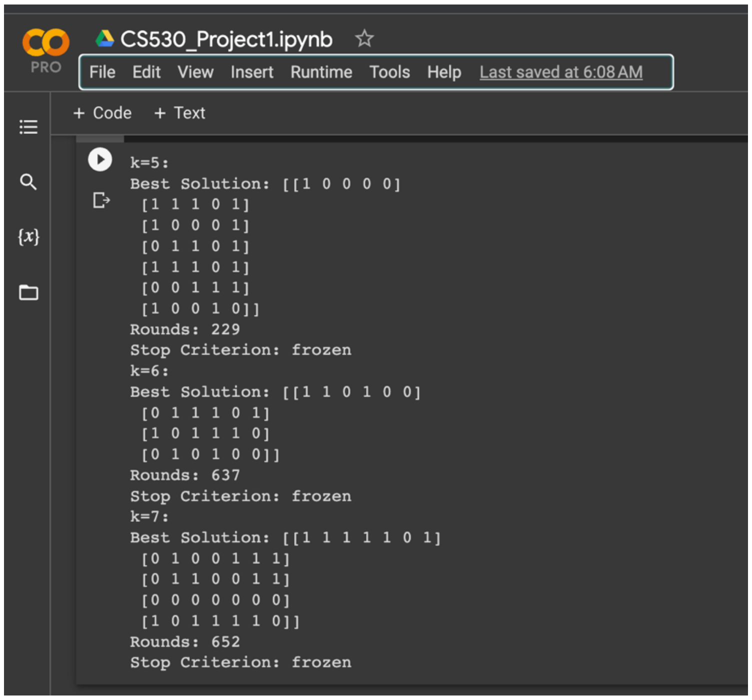
Propose a formula to determine ΔE= best_neighbor_score – current_node.

Experimental Results:
For each number of k (k=5, 6, or 7), the algorithm was run 30 times. The outcomes are displayed in Table 1. For each value of k, the algorithm identified partial solutions (best solution in code). Unfortunately, the stop criterion frozen usually stop the algorithm before it converges and general the complete solution. The proposed neighborhood function generates a diverse group of neighbors that are likely to enhance the current solution but it need some modification.

k	Successful Runs	Frozen Stopping Criterion	Solution Stopping Criterion
5	30	1	0
6	30	1	0
7	30	1	0

Conclusion:
The results of the experiments show that the proposed approach based on Simulated Annealing is effective in generating Covering Arrays with low missing combinations. However, the frozen factor needs to change and maybe a backtracking function need to be added to guide the algorithm to fast convergence.

Output:


CO PRO
CS530_Project1.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6:08 AM
+ Code + Text

```
k=5:
Best Solution: [[1 0 0 0 0]
 [1 1 1 0 1]
 [1 0 0 0 1]
 [0 1 1 0 1]
 [1 1 1 0 1]
 [0 0 1 1 1]
 [1 0 0 1 0]]
Rounds: 229
Stop Criterion: frozen
k=6:
Best Solution: [[1 1 0 1 0 0]
 [0 1 1 1 0 1]
 [1 0 1 1 1 0]
 [0 1 0 1 0 0]]
Rounds: 637
Stop Criterion: frozen
k=7:
Best Solution: [[1 1 1 1 1 0 1]
 [0 1 0 0 1 1 1]
 [0 1 1 0 0 1 1]
 [0 0 0 0 0 0 0]
 [1 0 1 1 1 1 0]]
Rounds: 652
Stop Criterion: frozen
```