REPORT G01369862

Introduction:

Covering Array:

A covering array is a combinatorial testing technique used to test the interactions between input parameters or factors in a system. It is used to ensure that all possible combinations of these factors are tested, without testing all possible combinations of values for each factor. A covering array is typically denoted as:

$CA(N; k, t, v)$

Where:
N is the number of test cases
k is the number of factors
t is the strength of the array
v is the number of values that each factor can take

A covering array ensures that each t-way factor combination happens at least once in the collection of N test instances. This means that all interactions between up to t factors will be tried, which is far more efficient than testing all potential values for each factor individually.

Background:

In this problem, we want to construct the covering array so the number of rows(N) is the lower bound size that can tentatively cover the combinations in each Next sub-array. CA is built using simulated annealing. A Covering array was built using SA code. Several criteria have been considered for terminating the process.

As soon as the solution is found, or the frozen rate is used, or the final temperature is reached, the process would stop.

Proposed Approach:

Simulated annealing technique has been used to solve the problem.

Following are the steps to build a covering array using simulated annealing:

Step 1: Define the issue and its limits.
Step 2: Create a first solution.
Step 3: Determine the original solution's cost.
Step 4: Create a temperature schedule.
Step 5: Define the function for generating moves.
Step 6: Define the likelihood of acceptance function.
Step 7: Put the simulated annealing algorithm to use.
Step 8: Please return the final solution.

K	Result	Algorithm	Stopping Criteria
5	[[1 0 1 0 0] [1 0 0 1 1] [0 0 1 0 0] [1 0 0 1 1]]	success	Cost = 0
6	[[1 0 1 0 0 0] [0 1 1 0 1 0] [0 0 1 0 1 0] [1 0 0 0 0 1]]	success	Cost = 0
7	[[1 1 1 0 0 1 1] [1 1 1 0 0 0 0] [1 0 1 0 0 0 1] [1 1 0 0 1 1 1]]	success	Cost = 0

Formula to obtain N based on the number of parameters k:

N = v^k /C (t, 1) * (v – 1) ^t

Formula to calculate ΔE:
Δ E = E(new) - E(old)

Where: E(new) = number of uncovered combinations in the current solution. E(old) = number of uncovered combinations in the new solution.

Conclusion:

Using the Simulated Annealing approach, an optimal solution can be obtained. At initially, the solution is accepted at random, but as the temperature drops, it gets more eager to find the optimal solution. In this example, we used it to construct the covering array so that the resulting number of rows(N) is the lower bound size that can cover the required combinations in each subsequent sub-array provisionally**.**

Simulated annealing determines whether to accept a new solution based on whether its acceptance probability and cost function value are less than those of the present solution.

The code for SA to build CA was built in Python, and all the stopping criteria are defined as frozen rate, if solution discovered, or final temperature. The algorithm was successful, and the result was obtained.