CS 580: Introduction to Artificial Intelligence
Project 1: Using SA to Construct Covering Arrays

Project 1 Report

Table of Content:

1.1	Introduction
2.1	Background
3.1	Proposed Approach
3.1.1	Formula to Determine N
3.1.2	Formula to Determine delta E
3.1.3	Stop Criteria
4.1	Experiment Results
5.1	Conclusion

CS 580: Introduction to Artificial Intelligence
Project 1: Using SA to Construct Covering Arrays

1.1 Introduction:

Covering array is a mathematical object used in the field of software testing, it can be used to maintain different combinations of the software to be tested. Covering Array can be defined a combinatorial object, denoted by CA(N;t,k,v) which can be described as a matrix with N X k elements, such that every N x t subarray contains all possible combinations of vt symbols at least once.

There are a lot of possible ways for construction of a covering array, here we use simulates annealing to obtain a covering array with minimal value of N

Initially we will take a randomly generated covering array with random values of 0's and 1's, this array contains N rows and K columns, in our case K = [5,7]. Our goal is to construct a covering array with all possible combinations with **minimal and optimal value of N** by using Simulated Annealing.

2.1 Background:

Simulated Annealing is a technique in AI that can be used to arrive at the global optimum, the major advantage of using simulated annealing over other approaches is that it will not get stuck at a local minimum. Simulated annealing is considered as one of the powerful tools to find the global optimum and hence we are using this approach to find the Covering array with all possible combinations vt of an N x T subarray.

This method is better among other methods for the following reasons, it has to ability to find global optima, escape the local minima, handles constraints, handles noisy data, finds solutions to problems that are difficult or impossible to solve using other methods.

3.1 Proposed Approach:

Simulated Annealing technique is proposed to generate a covering array with minimum value of N. It follows a random selection of neighbors along with the search process.

Initially all the candidates are assigned with objective functions and the algorithm decides to pick a neighbor based on the improvement of the current solution.

Simulated Annealing algorithm has been applied on an initial randomly generated covering array of size N x k this process is followed by picking a random column and then obtaining the neighbor states, an objective function is applied on the initial states and the neighbor states to check if the current solution cost has been improved or not if the cost shows improvement, we make it our current state.

We keep track of the best state and initially it is also the current state, best state achieved so far is also modified if the current cost is better than the available best cost.

Accepting bad moves is the major part of simulated annealing if there is no improvement in the objective function of the neighbors we take a bad move, this bad move is based on the acceptance probability which is (-ΔE/T). Initially when the temperature is higher a bad move is taken and the neighbor with no cost improvement is selected as the new current state. Moreover, our best state is not updated when we make a bad movement. The probability of selecting a bad movement is reduced as the temperature reduces.

During each iteration of the loop the temperature is reduced by a factor of T = 0.99*T, when the best so far solution does not show any kind of improvement after $\Phi = v^t \binom{k}{t}$, iterations the algorithm returns no solution this state is called the frozen state.

3.1.1 Formula to determine the N based on the number of parameters (k):

For any Given t (strength), k (number of parameters) and v (#values), there exist a minimum value of N that generates a covering array with all the possible combinations, in our case v and t are 2 and K = [5,7], the minimum value of N can be found using the given formula.

CAN (t, k, v) = min {N: there exists a CA (N; t, k, v)}.

CAN (t = 2, k, v = 2) = {min N: $\binom{N-1}{N/2}$, ≥ k} = log k (1 + o (1)) **(Katona 1973, Kleitman and Spencer 1973)**

3.1.2 Formula to determine ΔE:

ΔE can be defined as the change in energy from the current state to the proposed neighbor state.

It determines if we need to accept of reject a solution if ΔE is negative then we have obtained a best possible solution so we accept the state otherwise we still accept the state based on the acceptance probability

ΔE = Cost (Neighbor State) – Cost (Current State) **(as proposed in the simulated annealing algorithm)**

If (ΔE is negative) then the best solution is arrived so we make the neighbor state as the current and the best state.

If (ΔE is positive) we still accept the solution based on the acceptance probability.

Acceptance_Probability = (-ΔE/T)

If Acceptance_Probability > random (0,1) we make the neighbor state as the current state, the best state is not updated.

3.1.3 Stop Criteria that the algorithm implements:

When the Current Cost is 0, we presume that we have arrived at the best solution and return the solution, if the Temperature has become 0 then we can infer that there exist no solution, if the frozen state is achieved then we return that no solution exist.

4.1 Experimental Results:

These are the results for a covering array when k = 5, t, v = 2 & N = 6:

(Please Refer Output.pdf)

Sl.no	Stop Criterion for CA (6;2,5,2)	Execution Times
1	Solution Arrived	12
2	Solution Arrived	17
3	Solution Arrived	22
4	Solution Arrived	46
5	Solution Arrived	13
6	Solution Arrived	40
7	Solution Arrived	24
8	Solution Arrived	11
9	Solution Arrived	15
10	Solution Arrived	32
11	Solution Arrived	13
12	Solution Arrived	18
13	Solution Arrived	6
14	Solution Arrived	11
15	Solution Arrived	4
16	Solution Arrived	14
17	Solution Arrived	10
18	Solution Arrived	4
19	Solution Arrived	4
20	Solution Arrived	11
21	Solution Arrived	20
22	Solution Arrived	5
23	Solution Arrived	8
24	Solution Arrived	15
25	Solution Arrived	7
26	Solution Arrived	9
27	Solution Arrived	32
28	Solution Arrived	15
29	Solution Arrived	9
30	Solution Arrived	23

These are the results for a covering array when k = 6, t, v = 2 & N = 6:

(Please Refer Output.pdf)

Sl.no	Stop Criterion for CA (6;2,6,2)	Execution Times
1	Solution Arrived	11
2	Solution Arrived	48
3	Solution Arrived	21
4	Solution Arrived	7
5	Solution Arrived	21
6	Solution Arrived	17
7	Solution Arrived	23
8	Solution Arrived	55
9	Solution Arrived	22
10	Solution Arrived	57
11	Solution Arrived	24
12	Solution Arrived	25
13	Solution Arrived	20
14	Solution Arrived	47
15	Solution Arrived	20
16	Solution Arrived	42
17	Solution Arrived	10
18	Solution Arrived	34
19	Solution Arrived	13
20	Solution Arrived	38
21	Solution Arrived	26
22	Solution Arrived	18
23	Solution Arrived	42
24	Solution Arrived	30
25	Solution Arrived	18
26	Solution Arrived	37
27	Solution Arrived	36
28	Solution Arrived	34
29	Solution Arrived	15
30	Solution Arrived	67

These are the results for a covering array when k = 7, t, v = 2 & N = 6:

(Please Refer Output.pdf)

Sl.no	Stop Criterion for CA (7;2,6,2)	Execution Times
1	Solution Arrived	71
2	Solution Arrived	42
3	Solution Arrived	84
4	Solution Arrived	58
5	Solution Arrived	33
6	Solution Arrived	40
7	**Frozen (No Solution arrived)**	-
8	Solution Arrived	32
9	Solution Arrived	87
10	Solution Arrived	28
11	Solution Arrived	126
12	Solution Arrived	46
13	Solution Arrived	62
14	Solution Arrived	62
15	Solution Arrived	44
16	Solution Arrived	49
17	Solution Arrived	89
18	Solution Arrived	34
19	Solution Arrived	30
20	Solution Arrived	21
21	Solution Arrived	30
22	**Frozen (No Solution arrived)**	-
23	Solution Arrived	37
24	Solution Arrived	61
25	Solution Arrived	44
26	Solution Arrived	36
27	Solution Arrived	94
28	Solution Arrived	19
29	Solution Arrived	94
30	Solution Arrived	43

5.1 Conclusion:

There are many ways to deduce a covering array. Simulated annealing is one among them and it is a greedy approach, it is like stochastic hill climbing approach but is more efficient than hill climbing as it do not get confined with the local minimum, by taking bad moves it randomly moves to most of the possible states, to find the covering array. In the following experiment we have constructed a covering array with minimum value of N and have determined the acceptance probability to make a bad movement to choose a neighbor, further we return the best solution. The value of N and Delta E has been determined which are the 2 main factors in implementation of the algorithm.