Sequential Testing with Precedence Constraints

Sabancı Üniversitesi FACULTY OF ENGINEERING AND NATURAL SCIENCES

PURE

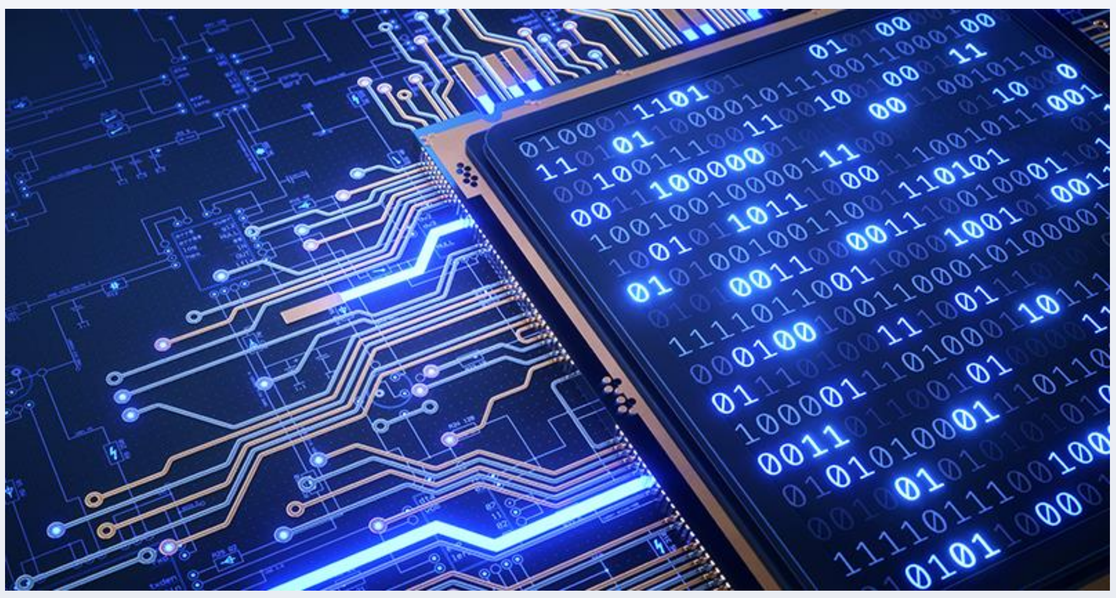
Student(s)

Duygu Ay
Cankut Coşkun

Faculty Member(s)

Tonguç Ünlüyurt

ABSTRACT



We consider the problem of minimum cost sequential testing (diagnosis) of a series (or parallel) system under precedence constraints. The model of the problem is a nonlinear integer program. We develop two different approach to solve the problem. Firstly, we develop and implement a Simulated Annealing algorithm for the problem. And, we compare the performance of the Simulated Annealing algorithm with optimal solutions. The simulated annealing algorithm is particularly effective as the problem size gets larger. Secondly, we develop a General Tree Type data search algorithm which finds and sorts all feasible permutations under precedence constraints in order to find the cost minimizing solution.

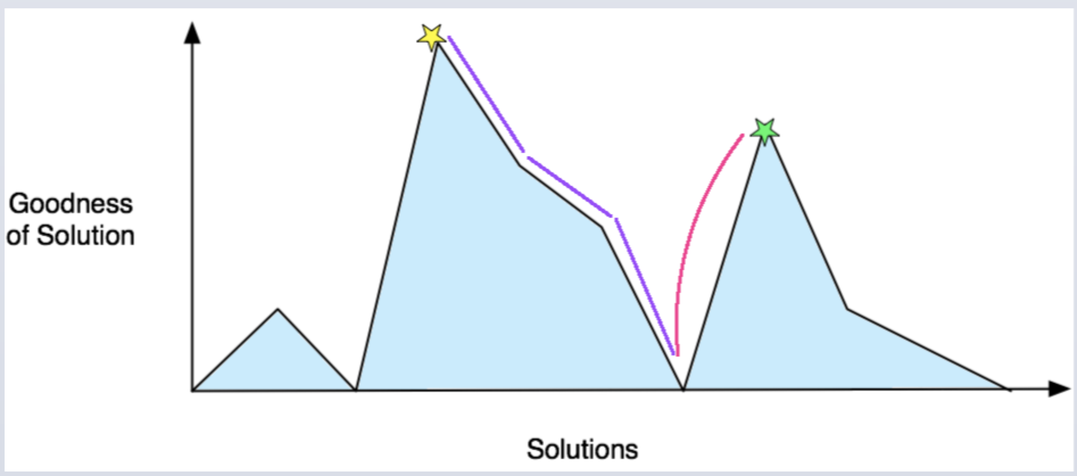
Keywords: Simulated Annealing; Tree Data Structure; Sequential Testing.

OBJECTIVES

1. For the problem of sequential testing, a faster heuristic algorithm was developed by using simulated annealing for efficiency concerns.

2. In order to find the cost minimizing permutation for the problem, enumeration of all possible solutions was developed by using a tree data structure.

PROJECT DETAILS



Simulated Annealing Algorithm Graph

A Faster Heuristic Algorithm Using Simulated Annealing: Simulated annealing is a method for finding a good (not necessarily perfect) solution to an optimization problem. It accepts worse neighbors in order to avoid getting stuck in local optima; It can find the global optimum if run for a long enough amount of time. The problem of sequential testing with precedence constraints is a good example: we are looking to visit all components sequentially by looking precedence constraints in order to minimize the total expected cost. As the size of components gets large, it becomes too computationally intensive to check every possible sequence. At that point, we need an algorithm.

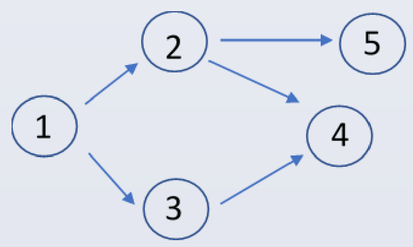
For our project, we implement Simulated Annealing algorithm in R programming language. In the implementation, we have three function: Simulated annealing, Generate a neighbor sequence and Calculating the cost of sequence. The key part is to generate a neighbor sequence. This function is recursive. It takes a feasible initial sequence, swaps two random consecutive points which have not a precedence relations and creates a new feasible sequence. We found good solutions close the global optimum at 100 iteration. The algorithm reads 450 txt files (our data) and writes the results in a excel file.

N	Average of gap1_T1	Average of gap2_T5000	Average of gap3_T10000
12	0.046862061	0.041887763	0.043707631
20	0.04906011	0.051389186	0.049505382
50	0.03284961	0.042362008	0.03583971
100	0.081736001	0.078531407	0.076992747
200	0.041722509	0.058489951	0.053930083
Grand Total	0.050446058	0.054532063	0.051995111

N	Average of gap1_T1	Average of gap2_T5000	Average of gap3_T10000
12	0.053078477	0.052592884	0.052489446
20	0.058871529	0.056236679	0.060872264
50	0.067879709	0.06362445	0.068714526
100	0.106441551	0.100481755	0.109619567
200	0.058871338	0.070088152	0.070498499
Grand Total	0.069028521	0.068604784	0.07243886

Tables of Average Optimality gap with Simulated Annealing Parameters for each Data size

After the implementation, we change simulated annealing parameters that are temperature and iteration number. We compare the results. Then, we take the optimality gap with optimal solutions for each 450 data and take the average optimality gap for each parameter. We realized that as iteration number increases, the results are closer to optimal solutions, but there is no significant change with temperature factor. Tables at the top shows the average optimality gap. Iteration numbers are 200 vs 100 from top to bottom.

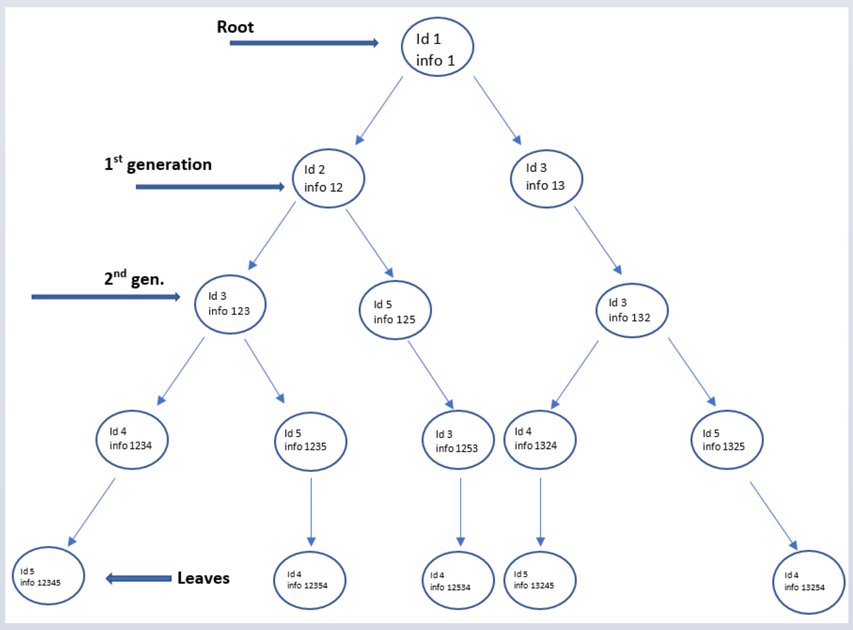


A series system with 5 components

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	1	1
3	0	0	0	1	0
4	0	0	0	0	0
5	0	0	0	0	0

Adjacency matrix of the system

Enumeration of All Possible Solutions Using a Tree: We develop an enumeration algorithm which finds and sorts all feasible permutations under precedence constraints in order to find the cost minimizing solution. Since not all permutations are feasible under precedence constraints the first main objective of our algorithm is the enumeration of all feasible permutations. These constraints are described by an acyclic directed graph where an arc from i to j means j cannot be tested in the absence of i. This relationship can be represented by an adjacency matrix.



Tree Search of the System

In order to enumerate all feasible solutions we build a General Tree Type data search algorithm and implement in C++. Each tree starts with a root which does not have any precedence constraint and add all possible members of the next generation. Thus the second level of the tree is created. Each second generation child add it's own children and create the third level. It goes on like this until the Nth generation. The Nth generation is called as leaves of the tree. Each leaf is a possible solution which follows a unique path. When a column of the matrix is consists of all zeroes, that column is the root of a tree. After chosen root we extract the column and the row belongs to that component. Simply we are extracting Minor (M i x i) of that component. All zero columns of minor will give us children for next generation. Until the minor becomes 1 x 1 matrix algorithm keep on extracting and adding new generations. When it reaches to end genetic info of leaves keeps a unique solution.

CONCLUSIONS

- We realized that as iteration number increases, the results are closer to optimal solutions, but there is no significant change with temperature factor for simulated annealing heuristic algorithm.
- Implementation of the enumeration algorithm has not finished yet. Total cost of each leaf will be calculated and stored in a container data type. After enumeration succeeds, it will find the feasible solution with minimum cost.