THINK INDIA JOURNAL

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

Comparitive Study of The Performance of Ew And Genetic Algorithm on Dynamic Cmst

Sudhans Shekhar Pandey
Department of Computer Science and Engineering
Lovely Professional University, Phagwara, India

Gunjan Gupta
Department of Computer Science and Engineering
IIMT Engineering College, Ganga Nagar, Meerut

Abstract: - There are many network topologies designs related problem which include CMST, this tells us about the finding the minimum cost that in a given spanning tree.There is a central node and someother nodes which are connected with central node and they have their capacity constraints. Each link is having their cost which is totally based on the distance, quality of lines, capacity etc. and one main property that every node associate as a demand value with it. Basically, most of the networks are static in CMST, but there are situations when it needs to change for adding or deletinglinks of this type of connections is called dynamic network and such type of cases is called DCMST (dynamic CMST).EW algorithm is abasic method to solve the DCMST algorithm issues, but it may not always provide optimal output. we are using genetic algorithm to find the best result than EW algorithm to manage dynamic CMST.

Keywords: Esau-William's Algorithm**,** Capacitated minimum spanning tree,

Introduction: -In general, during evaluation of weight of each edge and the capacity of node design, problems identified duetocapacity constraint and/or other constraints such as the time for communication or building, the complexity for construction etc. Sobesides the cost for connections near cities or terminalsand even the reliability are all valuable points and always have to be taken into consideration

Objectives: -Objectives of this research work are:

- To analysis and Comparison of EW and GA for dynamic capacitated minimum spanning tree.
- To compare GA with two encodings Netkey and Prufer for the capacitated minimum spanning tree.

EW (Esau & Williams) Algorithm: -

As it's known that for a Minimum Spanning Tree problem *Kruskal'*salgorithm [14] start with n subtrees, each containing a single node and then merging them until only one tree is left. In *this algorithm* the merging is based on the proximity of the subtrees and for the CMST problem this*Kruskal's* Algorithm is slightly modified to find the feasible CMST for a given problem. One of the modifications is to connect the subtree directly to the root or central node if the sum of vertex or node weight is equal to the capacity constraint k.

In a CMST problem a communication line is capable of transmitting only a limited amount of traffic and still satisfy the application requirements, so this desired configuration will be determined, in part, by calculating the amount of traffic at each remote location. This method to describe this assumes that the *line-loading* capabilities are given. The cost is proportional to distance will suffice as the basic variable in the construction of a most-economical configuration. In the explanation of the algorithm, the cost is proportional to distance; the algorithm assumes that a maximum distance configuration is one in which each remote location is connected to the center by a single link.

The EW (Esau & Williams) algorithm [5], [9] and [18] is based on the savings. Instead of merging two subtrees based on the proximity, EW introduces a new concept of savings to merge any two subtrees.

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

For connecting two subtrees EW compute savings for all subtrees and join those all subtrees, which produce the greatest savings.

The distance between two nodes, dist (i, j) represents distance between i and j. The distance between all nodes are stored in an n ×n matrix, on the bases of this distance matrix the n ×n saving matrix will be computed using the formula as:

Savings (i, j) = dist (i, j) –dist (i, r)

r is the central node/vertex. The subtree i and j, which shows the best savings will be connected if it also fulfills the capacity constraint.

Let's take an *example*: For a given fully connected graph, compute CMST through EW. First calculate the saving for each node and then select the best saver link to connect. A is the root node, total with capacity k= 3, and the demand for each node is 1 except c, where demand = 2.

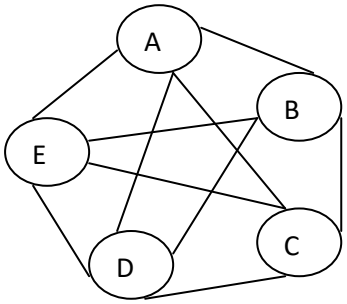


Fig 1: Fully Connected graph

Nodes	B	C	D	E
A	5	2	10	6
B		8	7	11
C			9	15
D				1

Table 1: Cost matrix for 5 nodes

ISTIteration

Tradeoff B= (B, D) = dist (B, D) – dist (B, A) = 7-5= 2
Tradeoff C= (C, D) = dist(C, D) – dist(C, A) = 9-2= 5
Tradeoff D= (D, E)= dist(D, E) – dist(D, A) = 1-10= -9

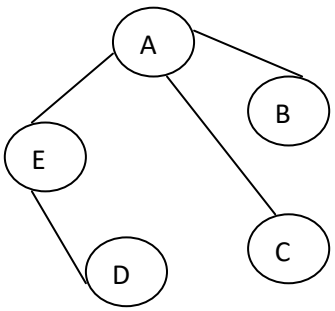
Here the most negative value for D which means the best saving selects that and continue to II$_{nd}$ Iteration and for that recomputed the tradeoff to get the next best saving link.

IInd Iteration

Tradeoff B= (B, D) = dist (B, D) – dist (B, A) = 7-5= 2 unchanged
Tradeoff C= (C, D) = dist(C, D) – dist(C, A) = 9-2= 5 unchanged
Tradeoff D= (D, E)= dist(D, E) – dist(D, A) = 1-10= -9
Tradeoff E= (E, D)= dist(E, D) – dist(E, A) = 1-6= -5 unchanged
Recomputed Tradeoff for D
Tradeoff D= (D, B) = 7-6 = 1

Now most negative is ED but it's already connected so recomputed the tradeoff of E. Tradeoff E= (E, B) = 11-6= 5

Now because no negative value left so all subtrees will be connected to the root node, and the CMST cost is 14

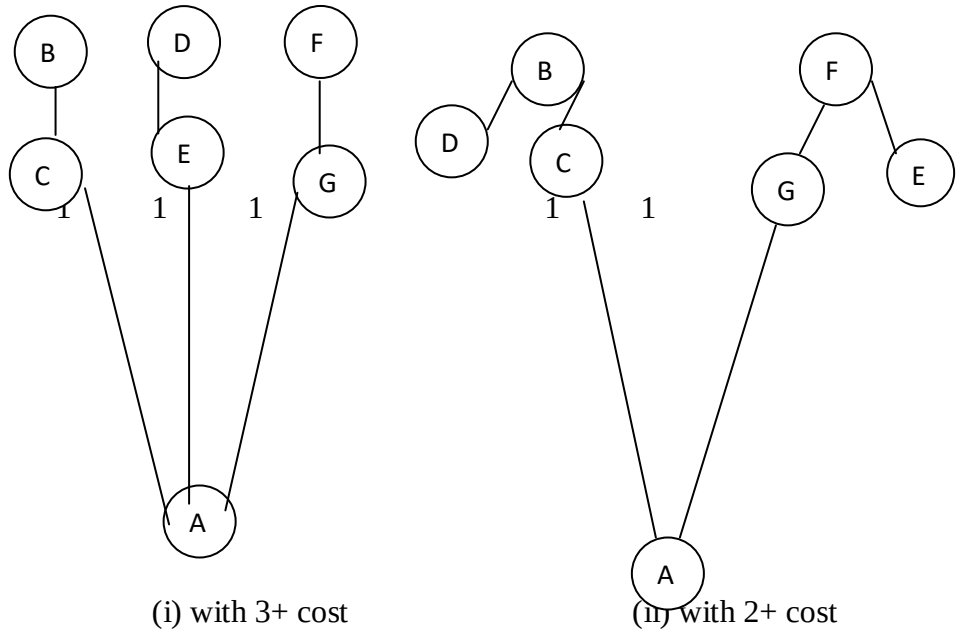


ISSN: 0971-1260
Vol-22-Issue-3-July-2019

Fig 2: CMST through EW

Problems with Esau-William's Algorithm:

The problem with this algorithm [4] is that, it does not provide the optimal result for the CMST problem in all the cases. Consider the case when during the algorithm, the weight must be k/2 which is coming when all sub tree has been formed, as we know that k is the capacity constraints, as there is a difficulty when the we are merging the 2-subtree, EW will connect root with the sub tree directly. This will give us twice optimal solution as number of sub tree as shown in figure (2).



(i) with 3+ cost (ii) with 2+ cost

Fig 3 (i) EW solution (ii) Optimal solution

Related Work:

Dynamic CMST problem lies in many network design problem where timely updation of network is required for efficient utilization of network design. For solving dynamic CMST two different criteria is there, firstly recomputed the CMST whenever any updation occurred in a network design. Secondly, updating network design without affecting the existing CMST. Different solutions [3] are available to solve this problem dynamically or online which means updating the existing CMST without affecting the entire CMST like:

1) The closest-or-direct (COD) heuristic:

Here the new node m is added to the node n which is more closure to m in the CMST T that also check the sum of demand of vertex m and the other nodes in that sub-tree where n lies is at most k, and also the distance between the new node and n node is less than the distance between the new node and the root node. If this condition will not fulfill, then the new node is directly attached to the root node r. In this heuristic [3] the computation time is linear;thus, the overall complexity of this heuristic is linear.

2) The closest feasible or direct (CFOD) heuristic:

In this heuristic [3] the new node m is connected to the closest node n in the existing CMST when these two solutions are fulfilled

a) Distance between the new node and the node n is less than the distance between the new node and the root node.

b) The sum of new vertex weight and the n vertex weight in the subtree is at most k.

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

When first condition is not satisfied, then the new node m is directly connected to root node, otherwise repeat step (a) with next closest node and so on until new node m is not added to the existing CMST.

3) The Best savings (BSA) Heuristic:

In this heuristic [3] the new node m is added to the node n in the CMST T, which gives the greatest savings while adding a new node. This heuristic is based on the EW (Esau and Williams). In this heuristic savings are computed on the bases of the distance between the new node and the other node, then the node with which link of new node gives the best saving will be selected and the new node is added to the particular node.

Genetic Algorithms

Genetic algorithm is a method to find out the solution on the basis of analyzing strings in a proper way. It tries to find out the solution of a problem. This algorithm makes new pattern by given pattern. It chooses a specific way to find out the optimum solution of a given problem. It creates various groups of data sets bit by bit and perform some operations by including some external data. The results obtained by genetic algorithm used to analyses the large number of data.

The requirement of this approach is plays very vital role because this approach does not need any deep knowledge of problem type. Its success depends various factors like complexity, parallelism, independence and combination of nature of the given problem.

Genetic Algorithm wasintroduced by *John Holland* his team.

Biological Background:

1. Chromosomes:

All live body on earth are made up of small cells. A cell is a collection of chromosomes which are made up of DNA pattern strings. A DNA serves as a base foundation for a living body. It's mostly unique in every one. Actually, its relevance only matches with parent cells.

Each DNA cell is made up of specific combination of protein particles. A DNA is responsible for various features of a human body for example eye color can be decided by it.

2. Reproduction:

A DNA play very important role in reproduction of a new pattern by various cross over schemes. Parent chromosomes combine to make a uniquely new type of pattern to form a new DNA. This process is called mutation. Mutation is a process in which original DNA performs some specific operations to generate a new unique DNA

3. Fitness:

The word fitness in terms of chromosomes defines the essential physical representations of the information's to specify new pattern for reproduction. It's really tough to define the fitness function as its accuracy depends of desireless of process. Most of the time it is efficient to compute its value to reach the solution as optimum value can be decided on the basis of it.

4. Evolution:

Evolution specify the sub sequential enhancement in pattern then previous one. It actually defines the adoptability of the chromosomes which need to change frequently due to environment changes.

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

5. Search Space:

The feasible solution of genetic algorithm is called state space as by taking this approach one need to find the best solution. Each single point of this space is evaluated on the basis of fitness function. This is the main reason to find an optimal solution we need to go up to extreme in maximum or minimum values in search space. Each point generates a possible solution to the given problem.

Genetic Algorithm-Basic steps:

This algorithm is focused on adoptability of search space points. It works on Darwin's Theory of survival of the best.

Following steps are followed:

- Make an elementary population of resolutions in an encode scheme.
- Verify if populations are predicted solutions, if not try to regenerate them.
- Calculate the fitness of every DNA by considering a fitness status.
- Accidently obtain few pairs of patterns, by mean of their value positions of fitness.
- calculate crossover value available on the pattern with applying probability.
- Perform, and mutation to the chromosomes.
- Calculate fitness.

Simple Genetic Algorithm Pseudo code:

```
Simple GA ()
{
    initialize population1;
    evaluate population1;
    While (termination criterion not achieved)
    {
        calculate solution for population (reproduction);
        perform crossover1 and mutation1 (recombination);
        evaluate population1;
    }
}
```

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

Strengths of GA:

1) The nature of the finding solution by genetic algorithm is mostly parallel processing. Many algorithms are operational serially and so they are limited to explore a solution space up to some extent. A serial approach is processed in one direction of solution while a parallel approach tries to perform its operation in either direction. If a serial approach found a sub optimal solution then it stops by delivering result but in parallel if one path leads to finish even then we can hope for solution.
2) A parallel approach permits to evaluate many schemas at once, A non-linear problem mostly suited for genetic algorithm approach. In a non-linear approach most of the problems having various fitness factors and they need to be treated independently.
3) Genetic algorithms does not need previous knowledge about the problem to process them.

Limitations of Genetic Algorithm:

As we know that genetic algorithms are the very powerful in the solving the problems they are not treated as panacea genetic algorithms are very efficient also. as it has many advantages as well as there are some limitations also. However, all of these can be overcome and biological evolution validity not bear by it.

1. For the showing candidate solution, the specified solution must be durable means that if there are some random changes then it must tolerate so that if there are fatal errors comes than it should not give consistently result.
2. When the fitness function is taking into account one must think that the higher fitness is considerable and calculate the better solution for the given problem. If we choose the fitness function in very lazy way or poorly then the GA will result very poorly and the result will not come as we are desired, it may be the chosen of fitness function is wrong then solution will be wrong.

To make a better choice of fitness prediction, the other parameters of Genetic Algorithm - the size of the population1, the prediction of mutation, crossover and the type and value of selection - must be taken with care. The population value should be large enough so that, the genetic algorithm may not explore enough of the solution space to consistently find good solutions. If the rate of genetic enhancement is too fast or the selection scheme is chosen carelessly, beneficial schema may be disrupted and the population may enter error catastrophe, changing too fast for selection to ever bring about convergence.

Proposed Solution:

As from the detail study of EW algorithm, it's found that EW fails to give optimal solution to the Dynamic CMST problem in some situations, for example in a process if all subtree is generated possess a weight more than k/2, k is capacity constant, As merging two sub trees is a tough task process has over the value of capacity constant. EW algorithm always joins two sub trees by their root nodes, so sometimes final result contains double number of sub tree as it should be counted in optimal result.

Here I am describing my proposed Genetic algorithm approach to rectify the DCMST.I first introduced Prufer encoding algorithm and then Netkey encoding algorithm and then at last I tried to show Genetic Algorithm way to solve the problem. I described various functions and data sets to show the comparisons.

Implementation:

This section describes the data structures and algorithms applied to get to the solution.

Data structure for Disjoint-Set:for a given set of elements (nodes), it is often useful to breaking up or partitioning them into a number of isolated, non-overlapping sets. A data structure that keeps track of such a partitioning is known as Disjoint-Set Data Structure. Programming language construct ArrayList, is used to represent each disjoint set.

Union Find Algorithm: This algorithm evaluates three useful operations on disjoint-set data structure,

Create: Create a set and assign to it a element (or node)

*Find:*Evaluatethe set with a particular element (node) is in. it is also useful for

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

determining if two elements are in the same set.

Union: Combining or merging two sets into a single set.

Union Find Algorithm as used in the implementation:

Input: set of edges say E1= (n1, n2), E2= (n1, n5), E3=(n2,5), etc.

Step1: Create a new ArrayList for each node, using Create method.

Step2: While all edges are not processed or total edges accepted is one less then total number of nodes {

-Get an unprocessed edge, E(n1, n2)

- if (find(n1) == find(n2)), {//this means cycle is formed if E is added to the tree

Edge E (n1, n2) rejected

}

Else {

if (findCapacity(find(n1), find(n2)) < = Max Capacity

)

{

Edge E(n1,n2) accepted

}

Else {

Edge E(n1,n2) rejected

}

}

}//end of while

A part form these three standard operations, one more operation is added for our implementation

Find Capacity: This will add the capacity of elements (nodes) of two input sets. This is used to determine capacity violation during tree formation.

Union-find algorithm is predominantly used in the implementation. This is used to determine the cycle formation and capacity violation while forming a feasible tree during initial population generation, crossover and mutation stages of Genetic Algorithm.

Below is the example to demonstrate the use of Union-Find algorithm, while initial population generation for prufer encoding.

Let's take a Prufer code for a spanning tree and check for that code whether it is feasible or not. Suppose there are 9 nodes in a fully connected graph having unit demand and the total capacity is 5. Suppose the code for spanning tree is (1, 4, 4, 5, 7, 7, and 6)

Now, first make a spanning tree from this code, which is like:

Prufer code: 1, 4, 4, 5, 7, 7, 6

P': 2, 3, 8, 9

Edge: 1, 2

Prufer code: 4, 4, 5, 7, 7, 6

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

P': 1, 3, 8, 9

Edge: 4, 1

Prufer code: 4, 5, 7, 7, 6

P': 3, 8, 9

Edge: 4, 3

Prufer code: 5, 7, 7, 6

P': 4, 8, 9

Edge: 5, 4

Prufer code: 7, 7, 6

P': 5, 8, 9

Edge: 7, 5

Prufer code: 7, 6

P': 8, 9

Edge: 7, 8

Prufer code: 6

P': 7, 9

Edge: 6, 7

Prufer code: Empty

P': 6, 9

Edge: 6, 9

So, the set of edges for a spanning tree are: [(6, 7), (6, 9), (7, 8), (4, 5), (5, 7), (3, 4), (1, 4), (1, 2)]

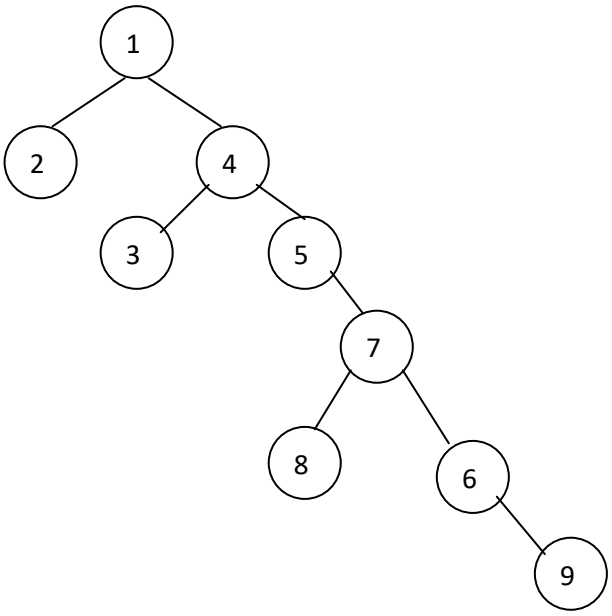


Fig 4: Spanning Tree generated through Prufer

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

Now to check this spanning tree about the feasibility, means whether it follows all constraint (capacity, cycle) or not, an algorithm named union-find, is used as:

First make a set of all nodes as a set of different sub-trees like:

[(1), (2), (3), (4), (5), (6), (7), (8), (9)]

Now take an edge and check its feasibility as:

Edge: (6, 7)

Here total capacity = 1 + 1 = 2

Capacity constrained followed so set becomes:

[(1), (2), (3), (4), (5), (6, 7), (8), (9)]

Edge: (6, 9)

Here total capacity = 2 + 1 = 3

Capacity constrained followed so the set becomes:

[(1), (2), (3), (4), (5), (6, 7, 9), (8)]

Edge: (7, 8)

Here total capacity = 3 + 1 = 4

Capacity constrained followed and set:

[(1), (2), (3), (4), (5), (6, 7, 8, 9)]

Edge: (5, 4)

Here total capacity = 1 + 1 = 2

Capacity constrained followed and set:

[(1), (2), (3), (4, 5), (6, 7, 8, 9)]

Edge: (5, 7)

Here total capacity = 2 + 4 = 6

Capacity constrained violated as is cross the limit of total capacity which is 5 so this is an infeasible tree. Continue to the next tree.

This union-find algorithm is done for each and every tree to check its feasibility for Prufer and Netkey both encoding schemes.

As specified in this chapter propose solution takes array-list as data structure and implemented to solve the Dynamic CMST problem through Genetic Algorithm which optimize the cost with respect to minimum cost with capacity constrained.

Experimental Results

After discussing the proposed solution, the methodology and the implementation details in the previous chapter, this chapter describes the parameters used and the test results obtained. The results obtained from EW and GA algorithm are analyzed and compared; also the results obtained from two encoding sachems are discussed.

Parameters:

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

In any project certain parameters should be specified on the basis of which the project is implemented. Criteria to stop, population size, and crossover and mutation probability are the example. From all them, stopping criteria is the most important to control the GA performance which includes number of generations, time taken for computing, and the fitness criteria. This thesis has taken different network size with following parameters:

Network size	GA parameter			
	Population	Selection	Crossover	Mutation
16	100	Tournament	Edge crossover	Edge mutation
25	100	Tournament	Edge crossover	Edge mutation
50	300	Tournament	Edge crossover	Edge mutation
100	700	Tournament	Edge crossover	Edge mutation

Table 4: Parameters

a) 1500 generations, varying population size based on the network size.

b) Crossover probability is 0.7, Mutation probability is 0.08

c) For unit and non-unit demand both.

A number of test results are generated for all the network size with different capacity constraint and the test bench for all these network sizes is given below:

999999,1616,1909,246,622,829,1006,2237,399,1717,632,1191,2116,824,1336,1519,1243
1616,999999,2996,1419,2217,1213,2046,3753,1516,1180,1997,552,3622,2423,1367,862,200
1909,2996,999999,1893,1543,1792,2785,1362,1667,3556,2332,2446,1248,1508,3233,3287,5
246,1419,1893,999999,799,593,1188,2369,242,1670,857,962,2243,1004,1348,1425,575
622,2217,1542,799,999999,1230,1253,1625,761,2301,801,1748,1509,206,1873,2119,68
829,1213,1792,593,1230,999999,1758,2597,480,1883,1449,663,2463,1420,1701,1573,478
1006,2046,2785,1188,1253,1758,999999,2703,1399,1470,454,1849,2612,1350,960,1470,62
2237,3753,1362,2369,1625,2597,2703,999999,2238,3922,2304,3231,137,1442,3476,3743,85
399,1516,1667,242,761,480,1399,2238,999999,1889,1029,1009,2108,959,1586,1628,456
1717,1180,3556,1670,2301,1883,1470,3922,1889,999999,1693,1437,3808,2480,511,340,35
632,1997,2332,857,801,1449,454,2304,1029,1693,999999,1685,2206,909,1205,1603,78
1191,552,2446,962,1748,663,1849,3231,1009,1437,1685,999999,3098,1952,1429,1100,793
2116,322,1248,2243,1509,2463,2612,137,2108,3808,2206,3098,999999,1331,3368,3624,142
824,2423,2508,2004,206,1420,1350,1442,959,2480,909,1952,1331,999999,2038,2309,3524
1336,1367,3233,1348,1873,1701,960,3476,1586,511,1205,1429,3368,2038,999999,578,463
1519,862,3287,1425,2119,1573,1470,3743,1628,340,1603,1100,3624,2309,578,999999,6
1243,200,564,575,68,478,62,85,456,35,78,793,1425,3524,463,6,999999

Table 5: Cost matrix for 16 node graphs

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

999999,159,351,161,314,257,52,229,106,77,5,276,243,380,204,413,224,266,232,157,198,334,408,449
159,999999,325,453,368,367,495,250,434,127,74,106,149,400,307,480,483,121,21,8,265,86,309,374,52
351,325,999999,233,240,147,290,334,454,112,477,56,19,302,106,72,312,182,9,166,10,362,303,14,115,
161,453,233,999999,162,120,198,96,343,380,383,172,230,313,244,118,181,307,117,302,129,34,240,13
314,368,240,162,999999,365,400,472,293,117,371,259,182,231,63,368,428,366,169,451,53,274,188,25
257,367,147,120,365,999999,484,106,419,263,401,311,80,323,479,145,115,358,334,49,57,301,121,328
52,495,290,198,400,484,999999,280,415,275,372,8,336,290,475,461,83,253,300,370,500,136,360,198,3
229,250,334,96,472,106,280,999999,64,234,10,105,150,127,484,20,138,263,134,447,490,131,106,191,3
106,434,454,343,293,419,415,64,999999,65,24,384,159,359,231,50,31,495,45,381,251,146,345,355,270
77,127,112,380,117,263,275,234,65,999999,95,340,385,16,197,343,476,479,139,40,380,121,215,366,30
5,74,477,383,371,401,372,10,24,95,999999,299,162,2,329,436,393,420,361,260,262,294,166,244,454,3276,106,56
172,259,311,8,105,384,340,299,999999,26,135,253,130,296,4,218,189,229,280,326,196,27,
243,149,19,230,182,80,336,150,159,385,162,26,999999,145,360,36,30,152,126,55,275,450,397,338,205
380,400,302,313,231,323,290,127,359,16,2,135,145,999999,12,248,117,498,470,186,37,365,438,267,23
204,307,106,244,63,479,475,484,231,197,329,253,360,12,999999,228,272,373,317,409,286,431,125,13
413,480,72,118,368,145,461,20,50,343,436,130,36,248,228,999999,339,385,47,71,474,435,101,390,497
224,483,312,181,428,115,83,138,31,476,393,296,30,117,272,339,999999,106,478,62,18,105,42,62,13,3
266,121,182,307,366,358,253,263,495,479,420,4,152,498,373,385,106,999999,144,120,9,448,218,378,3
232,21,9,117,169,334,300,134,45,139,361,218,126,470,317,47,478,144,999999,369,234,334,380,329,84
157,8,166,302,451,49,370,447,381,40,260,189,55,186,409,71,62,120,369,999999,136,342,64,347,293,1198,265,10
129,53,57,500,490,251,380,262,229,275,37,286,474,18,9,234,136,999999,386,62,350,287,3
334,86,362,34,274,301,136,131,146,121,294,280,450,365,431,435,105,448,334,342,386,999999,373,10
408,309,303,240,188,121,360,106,345,215,166,326,397,438,125,101,42,218,380,64,62,373,999999,431
449,374,14,138,253,328,198,191,355,366,244,196,338,267,133,390,62,378,329,347,350,10,431,999999
238,52,115,3,409,276,386,324,270,301,454,27,205,234,224,497,13,385,84,293,287,319,423,162,999999
379,129,204,7,339,196,83,490,449,423,311,199,198,388,266,19,314,28,192,374,28,272,277,188,999999

Table 6: Cost matrix for 25 node graphs

THINK INDIA JOURNAL

ISSN: 0971-1260

Vol-22-Issue-3-July-2019

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

Table 7: Cost matrix for 50 node graphs

Results:

This thesis has used different test benches as shown in the previous section for Dynamic CMST problem and also shows the results of all EW and GA (Prufer & Netkey encoding) for varying network sizes. Below is the different test results obtained:

Parameters				Results					
				Before Addition of a node			After Addition of a node		
Sr.No.	Nodes	Capacity	demand	EW	Prufer	NetKey	EW	Prufer	NetKey
1	16	9	Unit	11280	8526	8797	6071	5073	5378
2	16	9	N-unit	12850	8624	8750	6775	6044	6070
3	16	25	Unit	12145	8520	8639	4087	3860	3940
4	16	25	N-unit	12145	8536	8706	4087	3878	4183
5	25	15	Unit	505	450	444	542	455	439
6	25	15	N-unit	770	522	518	676	548	538
7	25	30	Unit	511	439	430	521	450	405
8	25	30	N-unit	524	420	410	534	414	409
9	50	30	Unit	840	650	610	925	665	615
10	50	30	N-unit	865	689	658	882	697	663
11	50	60	Unit	786	563	532	794	570	546
12	50	60	N-unit	763	557	512	768	564	514
13	50	80	Unit	786	560	520	794	563	510
14	50	80	N-unit	786	553	504	794	601	541
15	100	85	Unit	1043	1005	985	1170	1010	997
16	100	85	N-unit	1058	1015	949	1081	1040	993
17	100	150	Unit	1009	998	909	1000	984	943
18	100	150	N-unit	1009	987	953	1000	990	913

Table 8: Collection of results for different network size

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

5.3 Analysis of result:

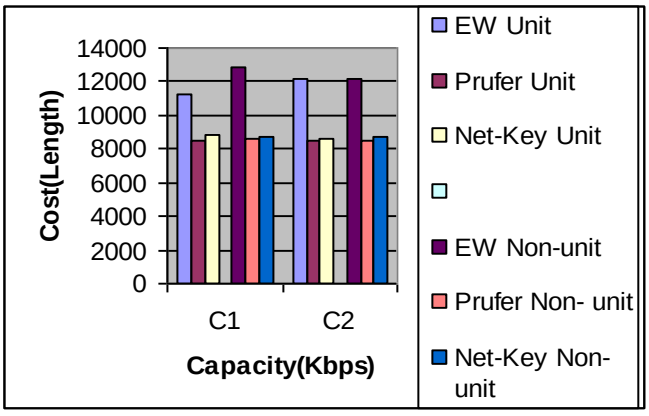
Here is the result of cost optimization (improved percentage) for our considered algorithm for genetic algorithm over EW in table 9. This table result shows that Net key is 0.082 percentage better than Prufer.

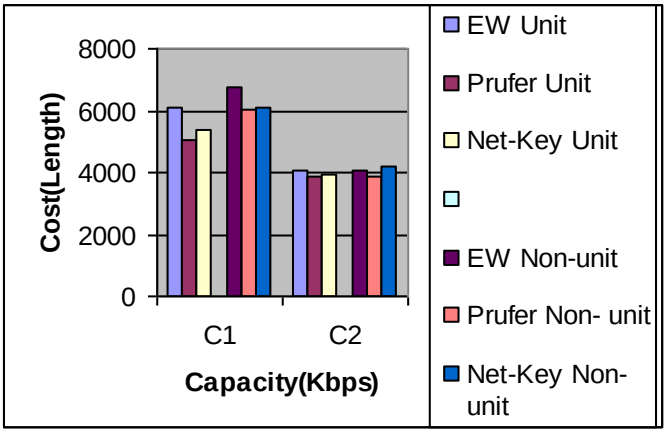
Parameters				Analysis				
				Result			Result of cost optimization (%)	
Sr. No.	Nodes	Capacity	Demand	EW	Prufer	NetKey	Prufer	NetKey
1	16	9	Unit	11280	8526	8797	0.24%	0.22%
2	16	9	Non-unit	12850	8624	8750	0.33%	0.32%
3	25	15	Unit	505	450	444	0.12%	0.12%
4	25	15	Non-unit	770	522	518	0.32%	0.33%
5	50	30	Unit	865	689	658	0.20%	0.24%
6	50	30	Non-unit	786	563	532	0.28%	0.32%
7	100	85	Unit	1043	1005	985	0.04%	0.06%
8	100	85	Non-unit	1058	1015	949	0.04%	0.10%

Table 9: Analysis of results

5.4 Graph generated:

Result graph for Prufer and Net Key shows almost all cases and genetic algorithm gives good result when comparing with EW algorithm.



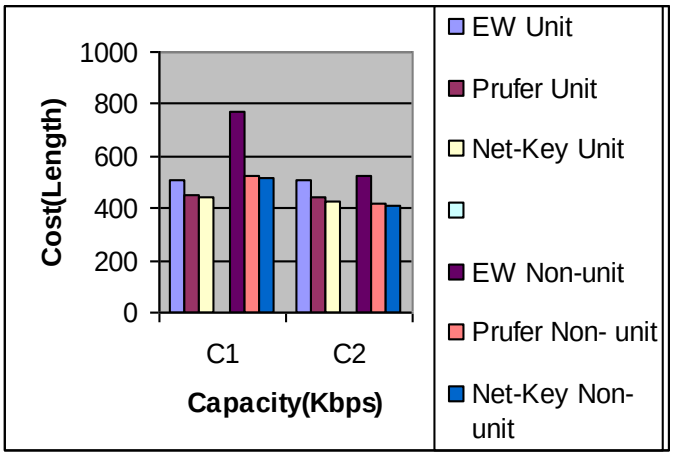


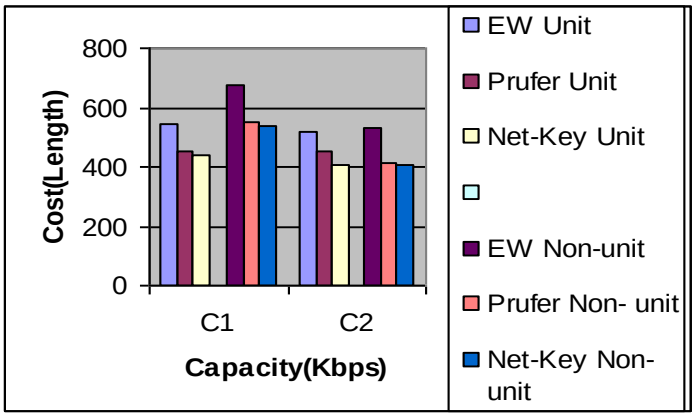
(a) inserting a node (prior CMST) (b) inserting a node (after DCMST)

Fig 28: a and b network size are 16

There are 16 nodes in network and its capacity for C1 is 9 and C2 is15. While EW algorithm gives the max. cost and the Prufer gives the min. cost in the size of n/w is 16 nodes.

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

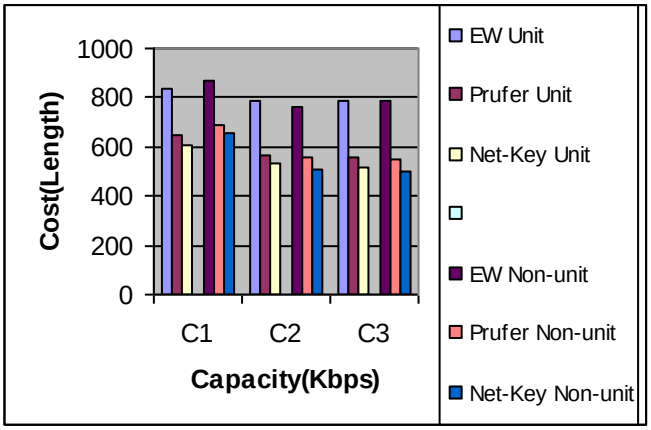


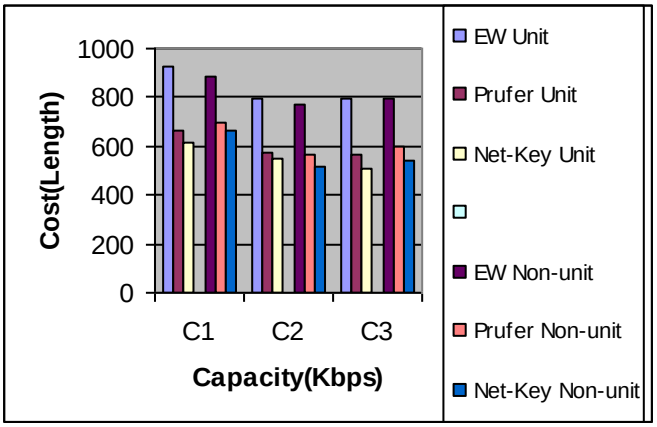


a) inserting a node (prior CMST) (b) inserting a node (after DCMST)

Fig : a and b network size are 25

There are nodes are more than 16 Netkey gives good output if we take 25 nodes with capacity c1is 15 and c2 is 30.

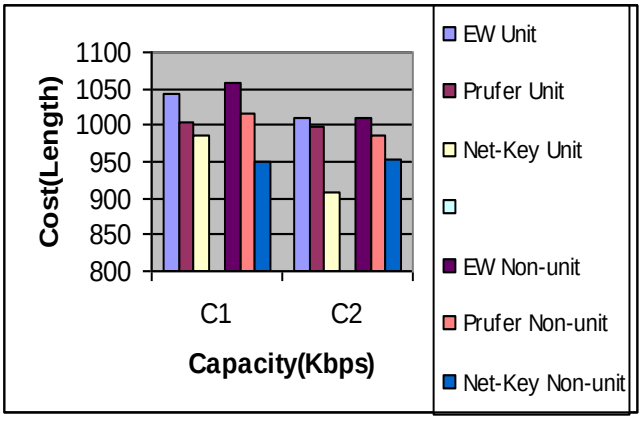


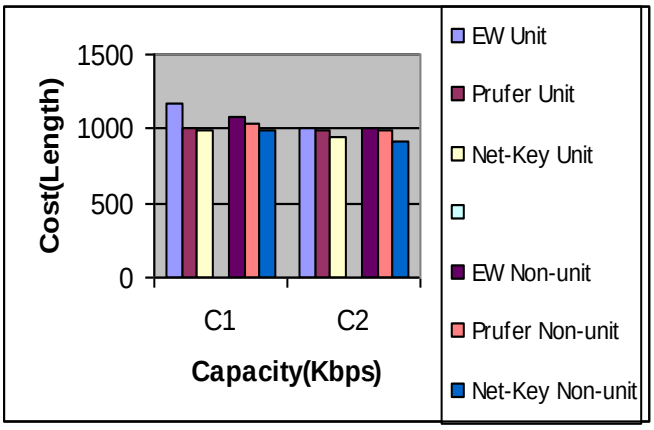


a) inserting a node(prior CMST) (b) inserting a node(after Dcmst)

Fig : network size is 50

For the DCMST having C1 is 30 AND C2 is 60 AND C3 is 80. Net key gives good output with network size =50 nodes.





a) inserting a node (prior CMST)(b) inserting a node(after Dcmst)

Fig : network size is 100

If nodes are 100 Netkey gives good output with capacity c1is 85 and c2 is 150.

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

5.5 Comparisons of EW and Genetic Algorithm

EW is a well-known algorithm for DCMST problem. It gives the solution by re computing the CMST, on addition of a dynamic node. Usually it provides optimal solution but in some situation like, when all the subgraph has been constructed and the weight is exceed than k/2 and the k is capacity constraints it is very costly and difficult to join 2 subgraph because when these two subgraph going to merge then the capacity constraint will grow, this is the main reason when we merge two subgraph for that EW algorithm reduce this problem by linking the subgraph with the root directly. This will give us best solution or optimal solution we can say.

Genetic Algorithm provides the better solution than EW, as seen in the experimental results obtained but GA takes more time than EW to produce the result.

It has seen from the results obtained that GA with Netkey encoding improves results by0.082% over Prufer encoding for varying network size.

REFERENCES

[1]Gengui Zhou. Zhenyu Cao. Jian Cao, 2006,**"*A Genetic Algorithm Approach on Capacitated Minimum Spanning Tree Problem*"**, University of Technology, China, pages 725-729.

[2] LixiaHanr and Yuping Wang, July 2006**"*A Novel Genetic Algorithm for Degree-Constrained Minimum Spanning Tree Problem*"**, *International Journal of Computer Science and Network Security,* VOL.6 No.7A, pages 50-57.

[3] Raja Jothi and BalajiRaghavachari, 2004**"*Dynamic Capacitated Minimum Spanning Tree Problem*"**, Proc. 3rd*IEEE International Conference on Networking (ICN)).*

[4] Raja Jothi and BalajiRaghavachari, 2004 **"*Revisiting Esau-Williams Algorithm: On the Design of Local Access Networks*"**, Department of Computer Science, University of Taxas at Dallas, pages 104-107.

[5] Raja Jothi and BalajiRaghavachari, 2003 "***Design of local access networks*"**, appear in 15th IASTED Inst. Conf. on *Parallel and Distributed Computer and Systems (PDCS),* pages 883-888.

[6] Melanle Mitchell, **"*An Introduction to genetic algorithms*"**, prentice Hall India, 2002 Edition.

[7] Günter R. Raidl, Christina Drexel, 2002 **"*A Predecessor Coding in an Evolutionary Algorithm for the Capacitated Minimum Spanning Tree Problem*"**, Institute of Computer Graphics, Vienna University of Technology, Austria, pages 309-316.

[8] Barbara Schindler, Franz Rothlauf and Hans-JsefPesch, 2002 **"*Evolution Strategies, Network Random Keys, and the One-Max Tree Problem*"**, S.Cagnoni et al. (Eds.): EvoWorkshops 2002, LNCS 2279, pp. 143-152, 2002. ©Springer-Verlag Berlin Heidelberg.

[9] R. Patterson and H. Pirkul, January 2000 **"*Heuristic Procedure Neural Networks for the CMST Problem*"**, School of Management, University of Texas, Dallas, Taxas, vol. 27, pages 1171-1200.

[10] H.chen, Y. Wang, 2000 **"*An efficient algorithm for generating Prufer codes from labeled trees*"**, Theory of Computing Systems, vol. 33, pp. 97-105.

[11] Günter R. Raidl, November 2000 **"*An Efficient Evolutionary Algorithm for the Degree-Constrained Minimum Spanning Tree Problem*"**, Institute of Computer Graphics, Vienna University of Technology, Austria, pages 104 – 111.

[12] Franz Rothlauf, Armin Heinzl and David E. Goldberg, July 2002 **"*Network Random Keys – A Tree Network Representation Scheme for Genetic and Evolutionary Algorithms*"**, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, pages 143-152.

[13] R.C. Prim, 2004 **"*Shortest connection networks and some generalizations*"**, Bell Systems Tech. Journal, vol. 36, pages 1389-1401.

[14] B. Kruskal Jr., 1956 **"*On the shortest spanning subtree of a graph and the traveling salesman problem*"**, In Proc. American Math Soc, volume 7, pages 48-50.

[15]David E. Goldberg, 2009 **"*Genetic algorithms in search, optimization & Machine learning*"**, Pearson Education, Second Edition.

[16] Holland, J.H., **"*Adaptation in Natural and Artificial Systems*"**, MIT Press, 1975.

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

[17]VibhavVineet, Pawan Harish, 2009 **"Fast minimum spanning tree for large graph"** ACM New York pages 167-171.

[18] L.R. Esau and K.C. Williams, ***"On teleprocessing system design",*** IBM Systems Journal 5(1966), pages 142-147.

[19] J.C. Bean, June 1992 ***"Genetics and random keys for sequencing and optimization",*** Ann arbor, MI: Department of Industrial and Operations Engineering, University of Michigan, Technical Report, pages 43-92.

Abbreviations:

EW: Esau and Williams Algorithm
CMST: Capacitated minimum spanning tree
DCMST: Dynamic CMST
BSA: Best saving algorithm
COD: Closet or Direct Heuristic
CFOD: Closet Feasible or Direct Heuristic
TSP: Traveling salesman problem
DNA: Deoxy-ribonucleic Acid