# A Solution of Degree Constrained Spanning Tree Using Hybrid GA with Directed Mutation

Sounak Sadhukhan, and Samar Sen Sarma

Department of Computer Science and Engineering
University of Calcutta, Kolkata, India
sounaksju@gmail.com, sssarma2001@yahoo.com

**Abstract.** It is always an urge to reach the goal in minimum effort i.e., it should have a minimum constrained path. The path may be shortest route in practical life, either physical or electronic medium. The scenario can be represented as a graph. Here, we have chosen a degree constrained spanning tree, which can be generated in real time in minimum turn-around time. The problem is obviously NP-complete in nature. The solution approach, in general, is approximate. We have used a heuristic approach, namely hybrid genetic algorithm (GA), with motivated criteria of encoded data structures of graph and also directed mutation is incorporated with it and the result is so encouraging, that we are interested to use it in our future applications.

**Keywords:** NP-complete, degree constrained spanning tree, graphical edge-set representation, graphical edge-set crossover, and graphical edge-set directed mutation.

## 1 Introduction

Many real-world situations can be described by a set of points, inter-connected through handshaking theorem (i.e., graphical structure) [1]. Graph plays a vital role in computer science. The *Degree Constrained Spanning Tree* (DCST) is one of the classic combinatorial graph problems. It is a NP-complete problem, so, till now, we do not have an exact algorithm which solves it in polynomial time. Obviously, we now seek guidelines for solving this problem around in a way to compromise. The strategies are usually and roughly divided into two classes: Approximation algorithm and heuristic approaches [2]. Heuristic approaches usually find reasonably good solutions reasonably fast. GA, is one of the famous heuristic search technique, tries to emulate Darwin's Evolutionary process. It deals efficiently, with digital encoded problem, and DCST problem is in perfect unison. Our proposed solutions may not be optimal but acceptable in some sense. After running several times, GA will converge each time, possibly at different optimal chromosomes and the schemata, which promise convergence, are actually indicative of the regions in the search space where good chromosomes may be found. So, therefore, the GA is coupled with a random Depth First Search (DFS) mechanism to find the optimal chromosome in a region. This is a kind of hybridization of our algorithm. The data structure employed here is graphical edge-set representation [3]. In addition, it is also shown, how a problem dependent

M.K. Kundu et al. (eds.), *Advanced Computing, Networking and Informatics - Volume 1*,
Smart Innovation, Systems and Technologies 27,
DOI: 10.1007/978-3-319-07353-8_75, 

hill-climbing approach can be effectively incorporated into the mutation operator, so that, it gives a better solution.

The following section describes the DCST problem. The Solution methodology is explained in Section 3, which includes generation of initial populations, graphical edge-set crossover, graphical edge-set directed mutation; and an experimental comparison to other optimization techniques is given in Section 4. Some concluding remarks are made in Section 5.

## 2 Degree Constrained Spanning Tree

**Problem definition**: A simple, symmetric and connected graph G = ($V$, $E$), where $V$ is the set of vertices and $E$ denotes the set of edges with positive integer $N \bullet |V|$. The problem is to find out a Spanning Tree (ST) T in G, which contains no vertices whose degree is larger than $N$, where $N > 0$ [4].

Narula and Ho, first proposed this problem in designing electrical circuits [5]. Hamiltonian path problem is a special edition of this problem, where degree of every vertices of the spanning tree has the upper bound 2. Finding such a path is related to the travelling salesman problem. Hamiltonian path, travelling salesman problem all are the examples of NP-complete problem [4]. In this case, a spanning tree may have degree up to $|V| - 1$. Finding a DCST in a graph is usually a hard task.

## 3 Solution Methodology

### 3.1 Representation of Chromosomes in GA in ST Problems

Representation of each individual is very important in GA. Here each individual must representa spanning tree of a graph. In literature, there are several encoding schemes available for representing a spanning tree namely edge encoding, Pr fer encoding, vertex encoding and graphical edge-set representation etc. [3], [6], [7], [8], [9]. These representations have both advantages and disadvantages over another.

There are lots of advantages to use graphical edge-set to represent a spanning tree, over the other encoding schemes. Using this representation, only feasible solutions (spanning tree) are always generated. Each spanning tree can be represented by a unique chromosome and vice-versa. GA operators like crossover and mutation are computationally efficient and incorporated with this representation.

### 3.2 Generation of Initial Population

In order to create initial population, this initial population generation algorithm always produces the valid ST. We have used two algorithms to generate initial population; one is random algorithm and another is random DFS algorithm. Actually, this mimics classical hybridization of graph optimization algorithms. Random DFS generates 20% individuals of the initial populations. The elitist model of GA is used here.

In the random algorithm, $E_T$ is the subset of $E$, contains the elements, which forms an individual (spanning tree). $V$ is the set of vertices in a graph G. Initially set $E_T$ is

empty and each vertices of $V$, formed individual components like, $\{V_1\}$, $\{V_2\},\ldots,\{V_n\}$.Fig. 1shows two individuals (spanning tree) are created from a graph G with seven vertices and twelve edges. The random algorithm for generating initial population is given below. Complexity of this algorithm is O(|$E$|).

**Algorithm Initial_Population(G($V$, $E$))**
**Step 1:** $E_T$ • , $S$•$E$.
**Step 2:** For all edges ($i$,$j$) • $S$ in random order, do
Check whether $\{E_T \cup (i,j)\}$ is a circuit
i) To check the circuit we find the end vertices of the edge ($i$,$j$).
ii) Assume $V_i$and $V_j$are the end vertices of the edge ($i$,$j$). Then check whether $V_i$and $V_j$belong to same or different connected component. If $V_i$and $V_j$are belong to different connected component, then there would be no circuit at all, otherwise a circuit would be present.
**Step 3:** If no circuit is present then, $E_T$ • $\{E_T \cup (i,j)\}$, $S$ • $S$ – ($i$,$j$).
**Step 4:** If $|V_T| = |V| - 1$, Return $E_T$.
**Step 5:** Else if circuit present then ignores the edge and $S$ • $S$ – ($i$,$j$).
**Step 6:** End loop.
**Step 7:** Stop.

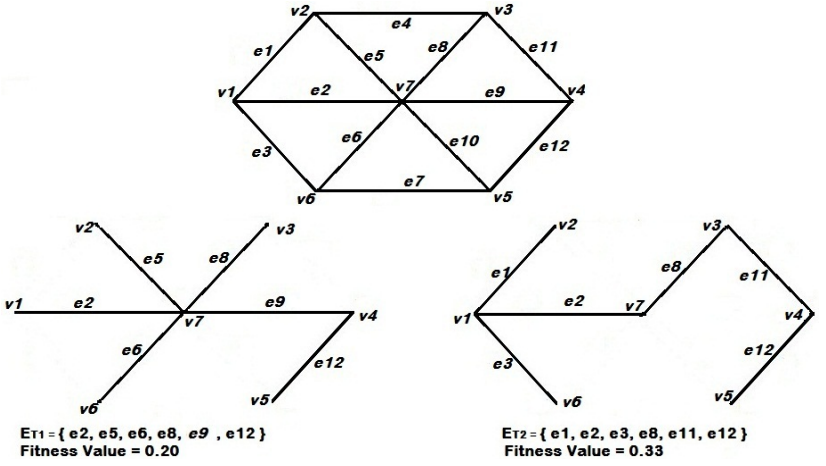


**Fig. 1.** A graph G and its two individuals are created from the above graph

Fitness function gives a value for each chromosome, which will help us to identify the good individual among the population. Definition of fitness function is varies problem to problem, depending upon the objective function. In this problem, fitness function is a fraction, represented by the reciprocal of maximum degree of each chromosome (degree of tree). ST with lowest degree represents the fittest individual among population.

### 3.3 Graphical Edge-Set Crossover

After selecting two individuals $T_1$and $T_2$from the population, this algorithm tries to develop two new offsprings $C_1$ and $C_2$if possible. Like the initialization procedure, in this algorithm, edges are processed one after the another in a random order and only those edges, that do not form a cycle, are included. Fig. 2 shows that two valid offsprings are genarated from the two individuals shown in Fig. 1.This algorithm takes O(|*E*|) time.

**Algorithm Graphical_Edge-set_Crossover($T_1$, $T_2$, |*V*|, *E*)**
**Step 1:** Select any two individuals $T_1$and $T_2$from population depending on their fitness values using Roulette Wheel selection.
**Step 2:** Make some edge sets like, $A \bullet E_{T1}$ U $E_{T2}$, $B \bullet E_{T1} \bullet E_{T2}$, $C \bullet E - A$ (*E*is the edge set of graph G), $D \bullet A - B$, $C_1\bullet$ and $C_2\bullet$ .
**Step 3:** Take edges from *B* one by one randomly and add into $E_{C1}$ till all the edges of *B* are explored and $|E_{C1}| \bullet |V| - 1$. If any edge creates cycle, discard the edge.
**Step 4:** If $|E_{C1}| < |V| - 1$, take edges from *C*,do the same as Step 3.
**Step 5:** If $|E_{C1}| < |V| - 1$, take edges from *D*,do the same as Step 3.
**Step 6:** If $|E_{C1}| = |V| - 1$, then return $C_1$,if $C_1$ is valid. Otherwise discard it.
**Step 7:** Develop $E_{C2,}$ as same way of $E_{C1,}$ from the set *D* and *C* respectively.
**Step 8:** If two offsprings are valid, two individuals from the population are replace depending on their fitness values.
**Step 9:** Stop.

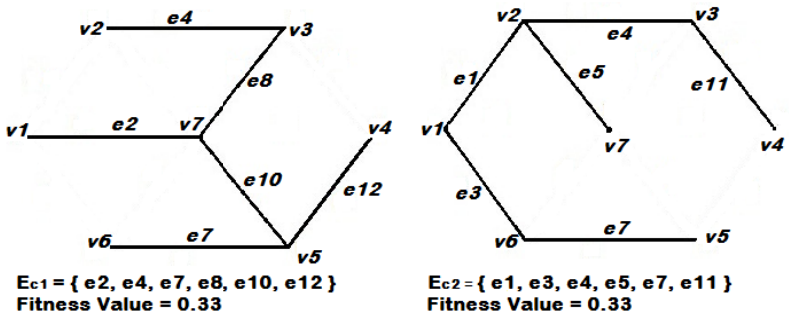


**Fig. 2.** Two individuals are created through graphical edge-set crossover from the individuals

### 3.4 Graphical Edge-Set Directed Mutation

The mutation operator brings diversity in the population of the potential solutions and inhibits premature convergence. Mutation alone induces a random walk through the search space. In this problem, we have used directed mutation which uses hill-climbing approach. The directed mutation is based on the principle of addition of an

edge from a particular set and deletion an edge from a created cycle in the tree depending on the objective function, which leaves a new valid tree with better or same fitness than the previous one. Fig. 3 gives an idea that how a new individual is created from the old one using graphical edge-set directed mutation algorithm and it has been also shown that how its degree would be better or same as the old individual.The computational effort for the procedure is $O(|E|)$.

**Algorithm Graphical_Edge-Set_Directed_Mutation ($E_T$, $|V|$, $E$)**
**Step 1:** Identify the vertices with maximum degree and second maximum degree in the tree $E_T$ and put them in the set S (Provided, if second maximum degree is 1, then take only the vertices with maximum degree).
**Step 2:** Remove all edges adjacent to those vertices of set $S$ and removed edges are put into set $R$ and put the neighbors of $S$ in the set $P$.
**Step 3:** The tree $E_T$ becomes $E_T$• (disconnected). Randomly search an edge from the set ($E$ - $E_T$) whose end vertices are in $P$ and add it to $E_T$• ($E$ is the set of all the edges of graph G).
**Step 4:** Add those removed edges from set $R$ in $E_T$• one by one in random order, till $|V_T| \bullet |V| - 1$, in such a way that no circuit has been created.
**Step 5:** If addition of any edge creates cycle, discard that edge. $E_T$ is replaced with $E_T$•, if $E_T$• is a valid tree.
**Step 6:** Stop.

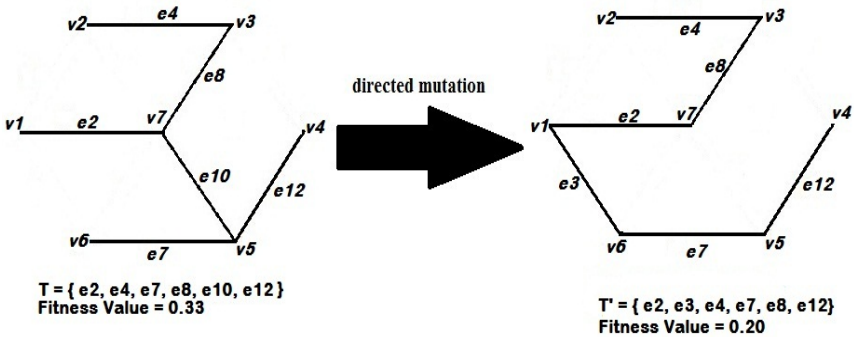


**Fig. 3.** An individual is created (*right*) through mutation from another individual (*left*)

**Lemma 1. The above directed mutation algorithm always creates a feasible solution with the fitness value never greater than the fitness value of the previous.**

**Proof:** Assume the tree contains $n$ vertices. Assume, $n_1$ and $n_2$are number of vertices with maximum and second maximum degree respectively and number of deleted edges is $e_d$. So, number of remaining edges will be = $(n - 1 - e_d)$. After addition of an edge among the components, the number of edges will be $(n - e_d)$. Then, removed edges are added one after another randomly. If all the removed edges are added, then the number of edges in the tree will be, $(n - e_d + e_d) = n$. Then, there must exist a cycle which contains the newly

added edge. The created cycle is also included the vertices with the maximum degree or second maximum degree. To remove the cycle, this algorithm removes an edge, which is adjacent with a maximum degree vertex or a second maximum degree vertex. Such an edge after being removed leaves a valid tree and degree of the adjacent vertices remains either same or reduced by one. It proves the lemma.

## 4 Experimental Result and Comparison

We have compared the result of this proposed algorithm with an existing approximation algorithm and genetic algorithm with normal mutation in Table 1. This experiment has been done on random graphs and each of the procedures was executed 10 times on each random graph. Simulation has been done in MATLAB 7.0.1. Then, we have calculated the percentage of errors with respect to the best value we get among the three procedures and plotted it as a graph to compare the percentage of errors in terms of the size of random graphs in Fig. 4. Here we consider, initial population size = 40, $\boldsymbol{p_c}$= 0.5 and $\boldsymbol{p_m}$= 0.4, total no. of iterations = 500.

**Table 1.** The performances in terms of quality of solutions of the three procedures

| \|V\| | \|E\| | **Procedure A** | | | | **Procedure B** | | | | **Procedure C** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $B_A$ | $W_A$ | $AVG_A$ | $E_A$ | $B_B$ | $W_B$ | $AVG_B$ | $E_B$ | $B_C$ | $W_C$ | $AVG_C$ | $E_C$ |
| 20 | 36 | 3 | 3 | 3.0 | 0.0 | 3 | 3 | 3.0 | 0.0 | 3 | 4 | 3.4 | 13.33 |
| 30 | 150 | 3 | 3 | 3.0 | 0.0 | 3 | 4 | 3.4 | 13.33 | 3 | 5 | 3.8 | 26.60 |
| 40 | 220 | 3 | 4 | 3.5 | 16.66 | 3 | 4 | 3.7 | 23.33 | 4 | 5 | 4.1 | 36.66 |
| 50 | 260 | 3 | 4 | 3.9 | 30.00 | 4 | 4 | 4.0 | 33.33 | 3 | 5 | 4.0 | 33.33 |
| 60 | 369 | 4 | 4 | 4.0 | 0.0 | 4 | 5 | 4.1 | 2.50 | 4 | 4 | 4.0 | 0.0 |
| 70 | 491 | 4 | 5 | 4.1 | 2.50 | 4 | 5 | 4.2 | 5.00 | 4 | 5 | 4.3 | 7.50 |
| 80 | 215 | 3 | 4 | 3.9 | 30.00 | 4 | 4 | 4.0 | 33.33 | 3 | 6 | 4.5 | 50.00 |
| 90 | 186 | 3 | 4 | 3.9 | 30.00 | 4 | 4 | 4.0 | 33.33 | 4 | 7 | 4.7 | 56.66 |
| 100 | 275 | 3 | 4 | 3.9 | 30.00 | 4 | 5 | 4.5 | 50.00 | 4 | 6 | 5.2 | 73.33 |
| 120 | 184 | 3 | 4 | 3.9 | 30.00 | 4 | 5 | 4.7 | 56.66 | 4 | 6 | 5.0 | 66.66 |
| 140 | 2381 | 4 | 4 | 4.0 | 0.0 | 5 | 5 | 5.0 | 25.00 | 4 | 6 | 5.3 | 32.50 |
| 160 | 800 | 4 | 4 | 4.0 | 0.0 | 5 | 5 | 5.0 | 25.00 | 4 | 6 | 5.0 | 25.00 |
| 180 | 1200 | 4 | 4 | 4.0 | 0.0 | 4 | 5 | 4.9 | 22.50 | 4 | 6 | 5.1 | 27.50 |
| 200 | 1600 | 4 | 4 | 4.0 | 0.0 | 5 | 5 | 5.0 | 25.00 | 5 | 7 | 5.7 | 42.50 |
| 250 | 2091 | 4 | 5 | 4.1 | 2.50 | 5 | 6 | 5.1 | 27.50 | 5 | 6 | 5.1 | 27.50 |

**Procedure A**, **Procedure B** and **Procedure C** denote Genetic Algorithm with directed mutation, Genetic Algorithm with crossover and normal mutation and Approximation algorithm respectively. $\mathbf{B}_i$, $\mathbf{W}_i$ and $\mathbf{AVG}_i$ denotes the best, worst and average result produced by procedure $\boldsymbol{i}$, $\mathbf{E}_i$= % of error produced by $\boldsymbol{i}$ = [{$AVG_i$– min($B_A$, $B_B$, $B_C$)}/ min($B_A$, $B_B$, $B_C$)] × 100.

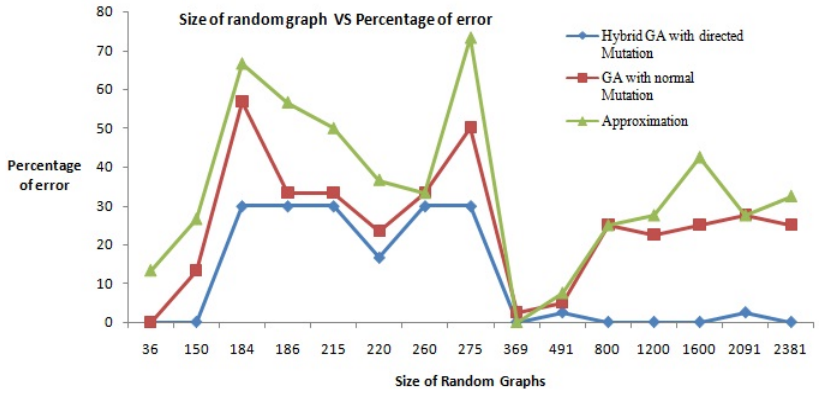


**Fig. 4.** Percentage of error (*in average*) respect to the size

## 5 Conclusion

This paper describes an approach to solve degree constrained spanning tree problem using hybrid GA. This method gets the optimal solution, through randomly generating a series of chromosomes, which represents a spanning tree in form of edge-set, and being operated in the way of genetic algorithms.

Our results show that, this approach is competitive with existing traditional approximation algorithm. The research work shows that, it is very important and crucial to choose the encoding and selection strategy in optimizing combinatorial problems and the graphical edge-set representation is really a fine encoding scheme for generating a tree for this problem. Most importantly, our algorithm always produces only feasible candidate solutions, if possible. Initial population, crossover and directed mutation is done in O(|*E*|) time. After each iteration, weaker individuals are obsolete by the more fit individuals, so, the average fitness value of the population is also increased.

This problem has huge importance in wireless ad-hoc network or mobile network at the time of routing. If it is possible to generate a spanning tree dynamically at the time of routing and whose degree is minimum, then noise and congestion will be reduced and routing will be faster. Similarly, it can be used in traffic signaling system, air traffic control, electrical network etc. We are optimistic that our approach will open up a vista of new solutions to a fascicle of practical problems.

**Acknowledgements.** We acknowledge Prof. Samiran Chattopadhyay of Jadavpur University and Prof. Malay Kumar Kundu of Indian Statistical Institute, Kolkata, for encouragement and help. We also thank Jadavpur University and University of Calcutta for providing necessary support.

## References

1. Rosen, K.H.: Discrete Mathematics and its Application.TMH Edition (2000)
2. SenSarma, S., Dey, K.N., Naskar, S., Basuli, K.: A Close Encounter with Intractability. Social Science Research Network (2009)
3. Raidl, R.G.: An Efficient Evolutionary Algorithm for the Degree Constrained Minimum Spanning Tree Problem. In: Proceedings of the 2000 Congress on Evolutionary Computation, pp. 104–111 (2000)
4. Garey, M.R., Johnson, D.S.: Computers and Intractability – A guide to the Theory of NP Completeness. W.H. Freeman and Company (1999)
5. Narula, S.C., Ho, C.A.: Degree-constrained minimum spanning tree. Computers and Operations Research 7, 239–249 (1980)
6. Zhou, G., Gen, M., Wut, T.: A New Approach to the Degree-Constrained Minimum Spanning Tree Problem Using Genetic Algorithm. In: IEEEInternational Conference on Systems, Man and Cybernetics, pp. 2683–2688 (1996)
7. Knowles, J., Corne, D.: A New Evolutionary Approach to the Degree-Constrained Minimum Spanning Tree Problem. IEEE Transaction on Evolutionary Computation 4, 125–134 (2000)
8. Zeng, Y., Wang, Y.: A New Genetic Algorithm with Local Search Method for Degree-Constrained Minimum Spanning Tree problem. In: IEEE International Conference on Computational Intelligence and Multimedia Applications, pp. 218–222 (2003)
9. Zhou, G., Meng, Z., Cao, Z., Cao, J.: A New Tree Encoding for the Degree-Constrained Spanning Tree Problem. In: IEEE International Conference on Computational Intelligence and Security, pp. 85–90 (2007)