A HYBRID GENETIC ALGORITHM FOR THE STEINER TREE PROBLEM IN GRAPHS TOWARDS OPTIMIZATIONS FOR WIRELESS SENSOR NETWORKS

Tang Phu Quy Le, Bao Nhan Ho Sy, Tan Phu Quoc Hoang, Cong Phap Huynh*, Dai Tho Dang*

The University of Danang - Vietnam-Korea University of Information and Communications Technology, Vietnam

*Corresponding author: hcphap@vku.udn.vn; ddtho@vku.udn.vn

(Received: April 10, 2024; Revised: May 05, 2024; Accepted: May 16, 2024)

Abstract - The Steiner tree problem in graphs (SPG) is one of the most studied problems in combinatorial optimization because of its theories and applications. It is one of the foundations to develop a Wireless Sensor Network (WSNs), such as multicast and topology design. SPG is an NP-Hard problem, and many heuristic and approximation algorithms have been proposed. Thus, this study proposes a Hybrid Genetic algorithm (HGA) to solve SPG. This study is the binary string representation for a set of chosen edges. To increase the diversity of the population and avoid falling into local optimization, we use a 2-longest Distance strategy, dynamic crossover rate, and chosen solutions must differ by at least 5%. The experiment results show that the HGA algorithm's running time equals 153.83% of the GA algorithm's, the deviation found by HGA, and the optimal distance only equals 65% that of GA.

Key words - Steiner tree problem; Genetic algorithm; Hybrid genetic algorithm; Node placement optimization in wireless sensor networks

1. Introduction

SPG is a combinatorial optimization problem that has received considerable research attention for a long time. This problem attracts theoretical research because it combines the shortest path and minimum spanning tree. In addition, regarding practical applications, many problems are modeled into SPG or closely related problems in many fields, such as computer science and networks. In WSNs, SPG is a base for many problems, such as node placement optimization, multicast, and topology design [1], [2].

Given an undirect, weighted graph $G = (V, E)$ and a cost function c mapping $c: E \rightarrow R^+$. A Steiner Tree for (G, R), in which $R \subseteq V$ is the set of terminal vertices, is a subgraph $T = (V(Y), Y)$ that spans R (i.e, $\forall s, t \in R$, there exists a path from s to t in T) with $Y \subseteq E$ is the subset of chosen edges and by $V(Y)$ we denote the set of vertices incident to the edge set Y. The Steiner Minimal tree problem asks to find the tree $T*$ with minimum sum of cost: $T* = arg\ min\ c(T)$ with

$$c(T) = \sum_{e \in Y} c(e) \quad (1)$$

Traditionally, the set $S = V \setminus R$ containing additional vertices is called the Steiner set, and any $u \in S$ is called a Steiner node [3].

SPG is NP-complete; thus, cracking it demands a computational time that grows exponentially with the problem size. Therefore, approximation and heuristic algorithms have been proposed. The GA algorithm has proven to be effective in solving this problem. Therefore, this study proposes a HGA [1] for SPG.

The main contributions of this study are as follows:

We propose the binary string representation for a set of chosen edges instead of vertices.

We propose using the 2-Longest Distance strategy to increase the diversity of the population and avoid falling into local optimization.

We propose using the dynamic crossover rate and the solutions chosen for the next generation to increase the diversity of the population.

2. Related Works

SPG has been of interest in research for a long time. The first approximation algorithm was introduced by Gilbert and Pollak in 1968 [4]. Over the next twenty years, no better approximation algorithm was found than this one [5]. A simple greedy algorithm, called 3-Steiner trees, was proposed by Zelikovsky [6]. Then, this approach was extended by Berman and Ramaiyer using k-Steiner trees. Since then, many approximation algorithms to SPG have applied Zelikovsky's idea [5]. In [7], Karpinski and Zelikovsky introduced the concept of a Steiner tree's loss. They use the general framework with a choice function that minimizes the weighted sum of the length and the loss of a Steiner tree. Bahiense et al. [8] developed an algorithm based on a Lagrangian relaxation of a multi-commodity low formulation of the problem. An extension of the subgradient algorithm, the volume algorithm, was used to receive lower bounds and evaluate primal. Chen [9] presents an efficient two-phase with eves an approximation ratio of 1.4295.

In [10], Kapsalis et al. introduced a classical GA to deal with STP. It uses a simple bit string representation, where a 1 or 0 corresponds to whether or not a node is included in the solution tree. In [11], Hesser et al. introduced a GA algorithm. The crossover probability is chosen based on the Grefenstette study, and the population size is chosen based on the Goldberg study. In [12], Nguyen and Nguyen introduced a parallel genetic algorithm that uses the distance network heuristic to evaluate individuals' fitness, which is implemented in parallel using a global population model. In [13], Dong et al. introduced a genetic algorithm. Because it focuses on narrowing the search range, the search speed is faster, and the number of iterations is low. In [14], Zhang et al. proposed a new crossover mechanism for this problem. This mechanism generates legal offspring by exchanging partial parent chromosomes, requiring neither global network link information, encoding/decoding, nor repair operations.

3. Algorithms

This study proposes using the binary string representation for a set of chosen edges instead of a set of vertices, as in previous studies. We apply binary string encoding. It follows that objective value computation is the sum of chosen edges. Thus, the decoding has no ambiguity. Time per evaluation is linear to the number of edges. Compared to approaches that use binary strings to represent the corresponding set of Steiner nodes, as seen in [10], [15], the proposed approach cannot omit any part of the search space.

Solutions generated in initialization, hybridization, and mutation may not be connected. It is very common that many solutions are not connected or have excess edges, so they need to be processed. We propose a Self-correction strategy for this task.

For each individual who is not connected, we connect its disconnected parts using the shortest path. We use the algorithm Algorithm 1 (in Figure 1) to determine the shortest spanning subtree in [16].

Algorithm 1 `make_span`

Data: Subgraph Y
Result: Modified Y, spans S
$C \leftarrow$ list of connected components of Y;
$T \leftarrow$ random tree of $\{0, 1, 2, \ldots, |C| - 1\}$;
foreach $(u, v) \in T$ **do**
  $x \leftarrow$ random node $\in C_u$;
  $y \leftarrow$ random node $\in C_v$;
  $Y \leftarrow Y \cup$ `shortest_path`(x, y)
end
return Y

Figure 1. *Pseudocode for determining the shortest-spanning subtree*

This often results in excess edges, so we need to trim them. Algorithm 2 (Figure 2), introduced in [17], is used for this task.

Thus, Self-correction includes two steps:

- Step 1: Running Algorithm 1
- Step 2: Running Algorithm 2

Note that the output of Algorithm 1 is the input of Algorithm 2.

Algorithm 2 : `reduce`:

Data: Subgraph Y that spans S
Result: a valid Steiner tree $T \subseteq Y$
$T \leftarrow$ `randomized_mst`(Y);
$Q \leftarrow \emptyset$; `/* contains leaves of tree T */`
while $Q \neq \emptyset$ **do**
  $u \leftarrow Q_0$;
  $Q \leftarrow Q \setminus \{u\}$;
  if $u \notin S$ **then**
    $v \leftarrow \delta(u)_0$;
    $T \leftarrow T \setminus \delta(u)$;
    if *v is now leaf of T* **then**
      $Q \leftarrow Q \cup \{v\}$
    end
  end
end
return T

Figure 2. *Pseudocode for trimming excess edges*

3.1. Genetic Algorithm (GA)

This algorithm is the basic genetic algorithm. The GA is shown in Figure 3.

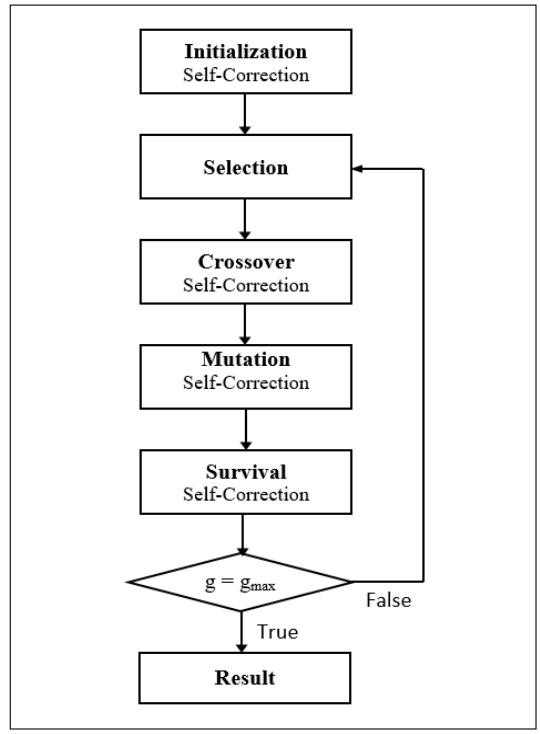

Figure 3. *Schema of GA*

- Initialization

An individual is a single solution, and a population is a set of individuals. This step is to create individuals for our base population. First, the population is created randomly. Then, Self-correction strategy is applied to all individuals.

- Selection (for recombination)

In this study, the roulette wheel selection chooses individuals to be transferred to the next generation. It is a well-known approach for the selection step. For an individual with a better fitness value, the probability of selecting it will also be higher [18].

- Crossover

Crossover, or recombination, is fusing two parents to create new offspring. This process exchanges some parts of the parents' strings to create a new string that may inherit some good traits from both parents. This work uses the uniform crossover. Each bit is chosen from either parent with a similar probability [19].

- Mutation

The bit-flip mutation is chosen. Several positions are randomly selected, and their values are flipped. This means that if the gene's value is 1 in a selected position, it will become 0; if the gene value is 0, it will become 1.

- Survival

The best individuals from the parents and children will be sorted according to their fitness values. Those with good fitness values will be selected to pass on to the next generation [19].

3.2. Hybrid Genetic Algorithm (HGA)

The process of probing new areas of the search space is called exploration. Otherwise, concentrating on existing spaces to find the optimum is called exploitation. The two processes play an essential role in problem-solving by search. It is trendy that when the diversity of the population rises, GAs move into exploration; when the diversity of the population reduces, GAs move into exploitation [20], [21].

The HGA is shown in Figure 4.

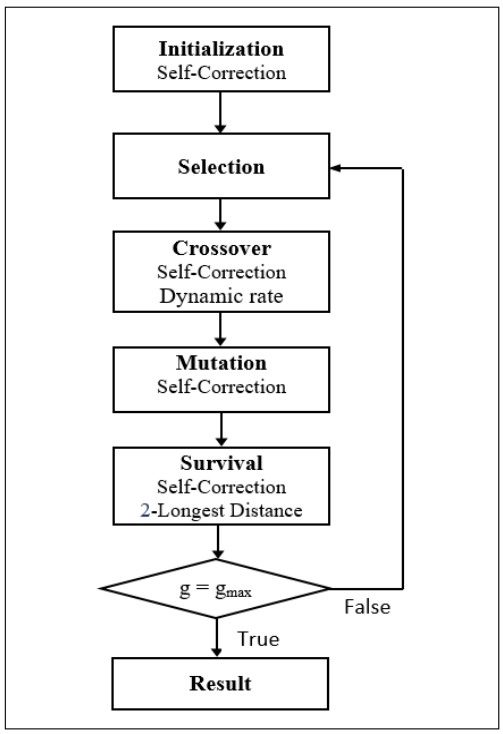

Figure 4. *Schema of HGA*

Retaining a balance between exploration and exploitation is essential to finding the globally optimal solution. A diverse population is one way to achieve this balance [20]. The Jaccard distance is used to measure dissimilarity between sets.

The stages of Initialization, Selection, Survival, and Self-correcting strategy are the same as those in the SGA algorithm.

- Crossover

We use the dynamic crossover rate. For parent p_1, p_2, the crossover rate is computed as follows:

$$p_m = \max\left(0.25, \frac{\alpha(p_1,p_2)}{\alpha_{max}}^{e^{-1}} \times 0.95\right) \quad (2)$$

where p_1 and p_2 are parent, and $\alpha(p_1,p_2)$ is $d_J = 1 - J(p_1,p_2)$.

- Survival

The best individuals from the parents and children will be sorted according to their fitness values.

The individuals with the better fitness value are selected. The selection must avoid converging on local optima. Therefore, the solutions chosen for the next generation must differ by at least 5%.

Besides, to increase the diversity of the population and avoid falling into local optimization, we also use the 2-Longest Distance strategy that was proposed and demonstrated to be effective [22].

4. Experiments and Evaluations

This section evaluates the effectiveness of our HGA. We compare the GA and the HGA. The two algorithms are compared by the quality of the found solution and the running time.

The test set is from the SteinLib Testdata Library (https://steinlib.zib.de/steinlib.php). In this work, we choose the significance value alpha is 0.05.

- Quality of solutions

The distances determined by GA and HGA are shown in Table 1.

The deviation of the distance between heuristic algorithms (GA and HGA) and the optimal distance is calculated as follows:

$$dev = \frac{d-dopt}{dopt} \times 100\% \quad (3)$$

Table 1. *Distances determined by GA and HGA*

ORD	Test Name	Distance of GA	Distance of HGA	Shortest Distance
1	c01	85	85	85
2	c02	144	144	144
3	c03	760.53	754.46	754
4	c04	1081.69	1080.23	1079
5	c05	1581.58	1579	1579
6	c06	55	55	55
7	c07	102.56	102.08	102
8	c08	514.11	511.54	509
9	c09	718.69	709.62	707
10	c10	1097.5	1094.08	1093
11	c11	32	32	32
12	c12	46	46	46
13	c13	262.61	259.46	258
14	c14	327.08	325.38	323
15	c15	557.94	557.23	556
16	c16	11	11	11
17	c17	18.03	18	18
18	c18	117.17	116.15	113
19	c19	155.08	155.46	146
20	c20	274.39	274.46	267
21	d01	106	106	106
22	d02	220	220	220
23	d03	1576.69	1569.31	1565
24	d04	1957.72	1938	1935
25	d05	3266.92	3255.54	3250
26	d06	67	67	67
27	d07	103	103	103
28	d08	1084.47	1079.69	1072
29	d09	1462.89	1457.23	1448
30	d10	2120.67	2116.77	2110

Shapiro–Wilk is used to determine the distribution of deviations of GA and HGA algorithms. The p-value of GA's deviation is 0.002, and the p-value of HGA's deviation is 2.871e-09. This means that these distributions do not come from normal distributions.

For comparing the deviations, hypothesis H_0 is that the deviations of the two algorithms are equal. The Wilcoxon test is used, and the p-value is 0.003. Thus, the hypothesis H0 is rejected. In other words, the deviations of the two algorithms are unequal. Therefore, their means are compared. The deviation of the HGA algorithm is only equal to 65% of that of GA.

- The running time

Table 2 shows the running times of these two algorithms. Shapiro–Wilk is used to determine the distribution of running times for GA and HGA algorithms. The p-value of GA's running time is 0.002, and the p-value of HGA's running time is 0.002. This means that these distributions are not normal.

Table 2. *Running time of GA and HGA*

ORD	Test Name	GA	HGA
1	c01	1857.12	3861.61
2	c02	2169.2	4171.42
3	c03	4990.13	7410.84
4	c04	6177.08	8736.18
5	c05	8894.48	12431.18
6	c06	1897.15	4288.42
7	c07	2026.18	4527.61
8	c08	5029.36	7963.45
9	c09	6288.11	9481.3
10	c10	9000.29	12544.92
11	c11	1903.59	4759.11
12	c12	2081.35	5169.29
13	c13	5093.86	8741.28
14	c14	5932.04	10140.37
15	c15	8889.02	13218.46
16	c16	3656.27	13041.55
17	c17	3690.18	12998.47
18	c18	6453.35	16844.25
19	c19	6842.04	17791.47
20	c20	10269.41	21465.76
21	d01	2455.99	5411.55
22	d02	2911.61	6208.13
23	d03	9194.47	13747.32
24	d04	11447.03	17598.52
25	d05	18485.44	24187.25
26	d06	2656.34	6125.22
27	d07	2587.16	6382.66
28	d08	10057.51	14413.54
29	d09	12696.56	17093.97
30	d10	17971.66	24352.67

For comparing the deviations, hypothesis H0 is that the two algorithms' running times are equal. The Wilcoxon test is used, and the p-value is 0.001. Thus, hypothesis H0 is rejected. In other words, the two algorithms' running times are unequal. Their means are compared. The HGA algorithm's running time equals 153.83% of the GA algorithm's running time.

The deviation of the HGA algorithm is only 65% of that of GA. The HGA algorithm's running time is 153.83% of the GA algorithm's running time.

5. Conclusions

In this study, we proposed a new HGA to solve SPG, a foundation problem for optimizations in WSNs [23]. We use the binary string representation for a set of chosen edges. The 2-Longest Distance strategy is proposed to increase the diversity of the population and avoid falling into local optimization. Besides, we use the dynamic crossover rate and the solutions chosen for the next generation to increase the diversity of the population. The proposed algorithm is better than GA. The deviation of the HGA algorithm is only 65% of that of GA, and the HGA algorithm's running time is 153,83% of the GA algorithm's running time.

Acknowledgment: This research is funded by the Ministry of Education and Training under project number B2023.DNA.13.

REFERENCES

[1] D. M. Rehfeldt, "*Faster algorithms for Steiner tree and related problems: From theory to practice*", Technische Universität Berlin, 2021.

[2] H. Gong, L. Fu, X. Fu, L. Zhao, K. Wang, and X. Wang, "Distributed Multicast Tree Construction in Wireless Sensor Networks", *IEEE Trans. Inf. Theory*, vol. 63, no. 1, pp. 280–296, Jan. 2017, doi: 10.1109/TIT.2016.2623317.

[3] B. Y. Wu and K.-M. Chao, *Spanning Trees and Optimization Problems*. Chapman and Hall/CRC, 2004.

[4] E. N. Gilbert and H. O. Pollak, "Steiner Minimal Trees", *SIAM J. Appl. Math.*, vol. 16, no. 1, pp. 1–29, Jan. 1968, doi: 10.1137/0116001.

[5] C. Gröpl, S. Hougardy, T. Nierhoff, and H. J. Prömel, "Approximation Algorithms for the Steiner Tree Problem in Graphs", *Combinatorial Optimization*, vol. 11, pp. 235–279, 2001, doi: 10.1007/978-1-4613-0255-1_7.

[6] A. Z. Zelikovsky, "An 11/6-approximation algorithm for the network steiner problem", *Algorithmica*, vol. 9, no. 5, pp. 463–470, May 1993, doi: 10.1007/BF01187035.

[7] M. Karpinski and A. Zelikovsky, "New Approximation Algorithms for the Steiner Tree Problems", *J. Comb. Optim.*, vol. 1997, pp. 47–65, 1997, doi: https://doi.org/10.1023/A:1009758919736.

[8] L. Bahiense, F. Barahona, and O. Porto, "Solving Steiner Tree Problems in Graphs with Lagrangian Relaxation", *J. Comb. Optim.*, vol. 103, no. 3, pp. 239–248, 2003, doi: 10.1023/A.

[9] C.-Y. Chen, "An Efficient Approximation Algorithm for the Steiner Tree Problem", in *Proceedings of the 2019 2nd International Conference on Information Science and Systems*, 2019, pp. 179–184, doi: 10.1145/3322645.3322649.

[10] A. Kapsalis, V. J. Rayward-Smith, and G. D. Smith, "Solving the Graphical Steiner Tree Problem Using Genetic Algorithms", *J. Oper. Res. Soc.*, vol. 44, no. 4, p. 397, Apr. 1993, doi: 10.2307/2584417.

[11] J. Hesser, R. Männer, and O. Stucky, “On steiner trees and genetic algorithms”, *Lecture Notes in Computer Science*, vol. 565, pp. 509–525. 1991; doi: 10.1007/3-540-55027-5_30.

[12] N. V. Huy and N. D. Nghia, “Solving graphical Steiner tree problem using parallel genetic algorithm”, in *2008 IEEE International Conference on Research, Innovation and Vision for the Future in Computing and Communication Technologies*, 2008, pp. 29–35, doi: 10.1109/RIVF.2008.4586329.

[13] Z. Chen, W. Hou, and Y. Dong, “The Minimum Steiner Tree Problem Based on Genetic Algorithm”, *DEStech Trans. Comput. Sci. Eng.*, 2018, doi: 10.12783/dtcse/mso2018/20491.

[14] Q. Zhang, S. Yang, M. Liu, J. Liu, and L. Jiang, “A New Crossover Mechanism for Genetic Algorithms for Steiner Tree Optimization”, *IEEE Trans. Cybern.*, vol. 52, no. 5, pp. 3147–3158, 2022, doi: 10.1109/TCYB.2020.3005047.

[15] H. Esbensen, “Computing near-optimal solutions to the steiner problem in a graph using a genetic algorithm”, *Networks*, vol. 26, no. 4, pp. 173–185, 1995, doi: 10.1002/net.3230260403.

[16] J. B. Kruskal, “On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem”, *Proc. Am. Math. Soc.*, vol. 7, no. 1, p. 48, 1956, doi: 10.2307/2033241.

[17] R. E. Tarjan, “Efficiency of a Good But Not Linear Set Union Algorithm”, *J. ACM*, vol. 22, no. 2, pp. 215–225, 1975, doi: 10.1145/321879.321884.

[18] T. Chun-Wei and C. Ming-Chao, *Handbook of Metaheuristic Algorithms*. Elsevier, 2023.

[19] R. Leardi, “Genetic Algorithms”, in *Comprehensive Chemometrics*, Elsevier, 2009, pp. 631–653.

[20] M. Črepinšek, S.-H. Liu, and M. Mernik, “Exploration and exploitation in evolutionary algorithms”, *ACM Comput. Surv.*, vol. 45, no. 3, pp. 1–33, 2013, doi: 10.1145/2480741.2480752.

[21] D. T. Dang, H. B. Truong, and N. T. Nguyen, “Hybrid Genetic Algorithms to Determine 2-Optimality Consensus for a Collective of Ordered Partitions”, in *ecture Notes in Computer Science, vol 14162. Springer, Cham.*, 2023, pp. 3–15.

[22] D. T. Dang, N. T. Nguyen, and D. Hwang, “Hybrid genetic algorithms for the determination of DNA motifs to satisfy postulate 2-Optimality”, *Appl. Intell.*, vol. 53, no. 8, pp. 8644–8653, 2023, doi: 10.1007/s10489-022-03491-7.

[23] N. T. Hanh, H. T. T. Binh, V. Q. Truong, N. P. Tan, and H. C. Phap, "Node placement optimization under Q-Coverage and Q-Connectivityconstraints in wireless sensor networks”, *Journal of Network and Computer Applications*, vol. 212, 2023, doi: 10.1016/j.jnca.2022.1035.