Improving the Heuristic Algorithms to Solve a Steiner-minimal-tree Problem in Large Size Sparse Graphs

Tran Viet Chuong[1], Phan Tan Quoc[2], Ha Hai Nam[3]

[1] Information Technology & Communication Center, Information & Communication Department of Ca Mau Province, Vietnam
[2] Faculty of Information Technology, Saigon University, Vietnam
[3] Vietnam National Institute of Software and Digital Content Industry (NISCI), Ministry of Information and Communications

Correspondence: Tran Viet Chuong, chuongcm74@gmail.com
Communication: received 5 January 2022, revised 23 January 2022, accepted 28 February 2022
DOI: 10.32913/mic-ict-research-vn.v2022.n1.1029

***Abstract*: Steiner Minimal Tree (SMT) is a combinatorial optimization problem that has many important applications in science and engineering; this is an NP-hard class problem. In recent decades, there have been a series of scientific papers published for solving the SMT problem using the approaches from exact solutions (such as dynamic programming, branch and bound) and approximate solutions (such as heuristic algorithm, metaheuristic algorithm). This paper proposes an improvement for two heuristic algorithms, PD-Steiner and SPT-Steiner, to solve a SMT problem in large size sparse graphs with edge weights not exceeding 10, and validates this proposal on large-size sparse graphs up to 100000 vertices. These experimental results are useful information for further research on the SMT problem.**

Keywords: *Steiner minimal tree, sparse graph, heuristic algorithm, metaheuristic algorithm.*

I. INTRODUCTION

A. Definitions

This section contains definitions and properties pertaining to the SMT problem.

Definition 1: Steiner tree [3]

Given $G = (V(G), E(G))$ is an interconnected undirected single graph with non-negative edge weights; where $V(G)$ is a set of n vertices, $E(G)$ is a set of m edges, $w(e)$ is the weight of edge e, $e \in E(G)$. A Steiner tree of L is a tree T that connects all vertices in a subset L of $V(G)$.

L is called a terminal set, its vertices are called terminal vertices, and vertices belonging to T but not L, are called Steiner vertices. Unlike other spanning tree problems, the Steiner tree only needs to traverse all of the vertices in the terminal set L and possibly some more vertices in the set $V(G)$.

Definition 2: The cost of Steiner tree [3]

Given that $T = (V(T), E(T))$ is a Steiner tree of graph G, the cost of tree T, denoted $C(T)$, is the total weight of the tree's edges, or $C(T) = \sum_{e \in E(T))} w(e)$.

Definition 3: Steiner Minimal Tree [3]

Given the graph G described above, the problem of establishing a Steiner tree at the lowest possible cost is called the Steiner Minimal Trees problem – *SMT*; or, more succinctly, Steiner Trees problem.

SMT is a combinatorial optimization problem in graph theory. In general, SMT has been proven to belong to the NP-hard problem [3, 13]. There are two special cases for *SMT* problem that can be solved in polynomial time: When $L = V(G)$ and $|L| = 2$ ($|L|$ denotes the number of vertices in the set L): When $L = V$, *SMT* problem can be solved by the smallest spanning tree algorithms; such as Prim [30], Kruskal [30]; when $|L| = 2$, *SMT* problem can be solved by algorithms of finding shortest paths between two vertices, such as Dijkstra [30].

For brevity, in this article, the *graph* is understood as a single, undirected, interconnected graph, with non-negative weights.

Example 1:

Given a graph G with 9 vertices and 10 edges as shown in Figure 1 and *terminal* set $L = \{2, 8, 9\}$.

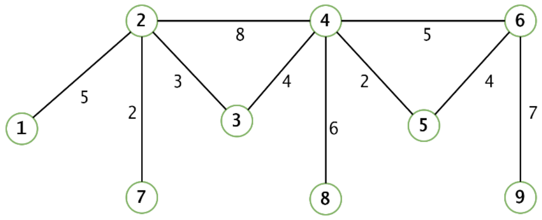

Figure 1. Illustration of a weighted interconnected undirected graph G.

The Steiner Minimal Tree founding corresponding to the *terminal* set L on graph G is T with $V(T) = \{2, 3, 4, 6, 8, 9\}$ and $E(T) = \{(2,3), (3,4), (4,6), (4,8), (6,9)\}$ as illustrated in Figure 2; tree T has the Steiner vertices set $\{3, 4, 6\}$ and the cost of 25.

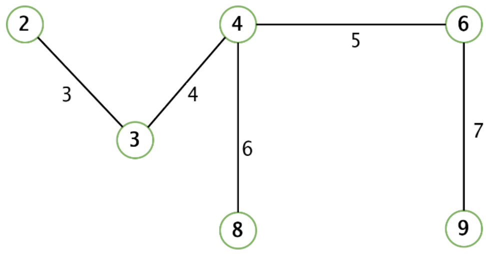

Figure 2. Steiner Minimal Tree corresponding to terminal set L of graph G.

B. Application of SMT problem

SMT problems can be found in important applications in several scientific and engineering fields such as the communication network design problems [2, 6, 11, 33, 41], VLSI design problems (Very Large Scale Integrated) [10], and problems related to network systems with the lowest cost [8, 10, 12, 18, 20, 23], etc.

C. Some forms of Steiner Minimal Tree problems

Steiner Minimal Tree problems are currently being studied in the following forms:

The first one is the Steiner Tree problem with Euclidean distance [13, 17, 38]. In the OXY coordinate plane, given a graph $G = (V(G), E(G))$ and a vertex set $Y \subseteq V$. We need to find trees passing through all the vertices in set Y and have a minimum total length, allowing us to add several sub-points (Steiner points) taken from the vertices of V. Euclide distance between two points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ is defined as the length of the line connecting the two vertices P_1, P_2.

The second form is the Steiner Tree problem with rectilinear distance [45]. In the OXY coordinate plane, given a graph $G = (V(G), E(G))$ and a vertex set $Y \subseteq V$ We need to find trees passing through all the vertices in the set Y and have a minimum total length, allowing us to add a number of sub-points (Steiner points) taken from the vertices of V. $d(P_1, P_2) = |x_1 - x_2| + |y_1 - y_2|$ is the rectilinear distance between two points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$.

The third form is the Steiner Tree problem, with the distance between vertices defined as the weights of edges (random distance values) [4, 24], which has been chosen as the scope of this paper's research on the Steiner Minimal Tree problem.

D. Some SMT studies and problems that need to be solved

Due to its scientific nature and wide applicability, SMT problems have attracted the attention of many scientists in the world for decades [27, 35, 44]. Some of them are Martin Zachariasen [21], Tobias Polzin [37], Pieter Oloff De Wet [26], Xinhui Wang [40], Jon William Van Laarhoven [17], and Zhiliu Zhang [45].

Currently, the publications on SMT problems can be divided into the following approaches: graph reduction algorithms, algorithms to find the exact solution, algorithms to find approximate solutions, heuristic algorithms, and metaheuristic algorithms.

***The first approach*: graph reduction algorithms**

There are some works on graph reduction for Steiner Tree problems regarding the techniques to minimize the graph size such as the works of Jeffrey H. Kingston and Nicholas Paul Sheppard [16], Thorsten Koch and Alexander Martin [36], C. C. Ribeiro and M.C. Souza [5], etc.

The general concept of graph reduction algorithms is aimed at two goals. The first one is to increase the number of vertices in the terminal set; the second one is to remove vertices of the graph that are certainly not on the Steiner Minimal Tree that has been sought.

The quality of algorithms solving the SMT problems depends on the size of the coefficient $n - |L|$. Therefore, the purpose of graph reduction algorithms is to minimize the coefficient $n - |L|$.

The graph reduction algorithms are considered an important data preprocessing step to improve the quality of solving SMT problems. This step has become more and more necessary for algorithms to find the exact solutions, such as dynamic programming or branch and bound.

***The second approach*: algorithms to find the exact solution**

Some studies to find the exact solution to an SMT problem are known as the dynamic programming algorithm of Dreyfus and Wagner [32], an algorithm based on the Lagrange relaxation of Beasley [15], branch and bound algorithms of Koch and Martin [36], and Xinhui Wang [40, 25].

Although this approach can help find the exact solution, it can only solve small-scale problems. As a result, their utility is limited. Solving the exact SMT problem is really a challenge in combinatorial optimization theory [22, 38].

This approach is an important basis for evaluating the solution quality of other approximate algorithms when solving SMT problems.

***The third approach*: α - approximation algorithms.**

The α - approximation algorithm aims to find an approximated solution with α ratio to the optimal solution [3].

The advantage of this approximation algorithm is that it is mathematically guaranteed in the sense mentioned above. Meanwhile, its drawback is that the approximate ratio found in practice is often much worse than the quality of solutions found by other approximate algorithms based on experimentation. The MST-Steiner algorithm of Bang Ye Wu and Kun-Mao Chao has the ratio of 2 [3], while the Zelikovsky-Steiner algorithm has the ratio of 11/6 [3]; The best ratio found for the SMT problem is currently 1.39 [7, 29, 30, 34].

***The fourth approach*: heuristic algorithms**

The heuristic algorithm is the specific experience of finding a solution to a particular optimization problem. The heuristic algorithm often finds an acceptable solution in the time allowed, but is uncertain whether it is the exact solution. Heuristic algorithms are also unlikely to be effective on all types of data for a particular problem.

The typical heuristic algorithms for SMT problems are SPH [5], Heu [36], distance network heuristic of Kou, and Markowsky and Berman [19].

***The fifth approach*: metaheuristic algorithms**

The metaheuristic algorithm uses several heuristics combined with auxiliary techniques to explore the search space. It belongs to the class of optimal search algorithms [42, 43].

There have been a lot of works using metaheuristic algorithms to solve SMT problems such as the Variable neighbor search algorithm, the Hill climbing search algorithm, the Genetic algorithms [9], the Tabu search algorithm [5, 39], the Parallel genetic algorithm (PGA) [1, 24], and the Bees algorithm.

Having considered various types of approaches, the authors of this paper would like to propose an improvement for the heuristic algorithms to solve the SMT problem in the case of large sparse graphs, giving an acceptable quality of the solution and a faster running time than other known heuristic algorithms. When applied to real-world problems, this proposed method is more meaningful.

E. Experimental data system

Definition 4: Sparse graphs

According to the conventional definition (at the URL: https://www.baeldung.com/cs/graphs-sparse-vs-dense), a simple graph with n vertices is a sparse graph with no more than $n(n-1)/4$ edges.

To experiment with related algorithms, most of the works use 78 datasets which are sparse graphs in the standard experimental data system for the Steiner tree problem (at the URL: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/steininfo.html [14]) which contain the following information about graph groups:

Steinb graphs have 50 to 100 vertices, and 63 to 200 edges; steinb contains a total of 18 graphs. Steinc graphs have 500 vertices, and 625 to 12500 edges; steinc contains a total of 20 graphs. Steind graphs have 1000 vertices, and 1250 to 25000 edges; steind contains a total of 20 graphs. The steine graphs have 2500 vertices, and 3125 to 62500 edges; steine graph contains a total of 20 graphs.

TABLE I
INFORMATION ON GROUPS OF GRAPHS

Groups of graphs	n	m	Number of tests
steinb.txt	50..100	63..200	18
steinc.txt	500	625..12500	20
steind.txt	1000	1250..25000	20
steine.txt	2500	3125..62500	20

Because the problem belongs to the *NP-hard* class, the application of the *SMT* problem should be considered from either the perspective of design when priority is given to the quality of the solution, or the perspective of implementation when run-time is referred. The purpose of this paper is to improve the heuristic algorithms to solve the *SMT* problem in the case of large sparse graphs with acceptable solution quality and shorter running time when compared to currently available heuristic algorithms. This proposed method is more meaningful when considering the perspective of the implementation when applying the Steiner tree problem to reality. Experiments with the improved algorithms were conducted on a data system with 80 graphs ranging in size from n to 100000 vertices.

II. HEURISTIC ALGORITHMS FOR SOLVING AN SMT PROBLEM

This paper proposes an improvement for 2 heuristic algorithms PD-Steiner and SPT-Steiner [4] based on the idea combination of finding the shortest path and the smallest spanning tree of the graph to solve SMT problems in large size sparse graphs with an edge weight of less than 10. Because two heuristic algorithms SPT-Steiner, PD-Steiner [4], and other currently known heuristics do not solve this problem, the proposed ones are called i-PD-Steiner and i-SPT-Steiner. The authors of this paper conducted experiments and compared the quality of the i-PD-Steiner algorithm and the i-SPT-Steiner algorithm with MST-Steiner algorithm [3]. These algorithms have input data as graph $G = (V, E, w)$, *terminal* vertices set $L \subseteq V$; and the output is the cost of the Steiner tree T.

A. MST-Steiner algorithm

Algorithm 1: MST-Steiner algorithm of Bang Ye Wu and Kun-Mao Chao has the ratio of 2 [3]

Inputs: Graph $G = (V, E, w)$ and a terminal set $L \subset V$
Outputs: Steiner tree T
begin
  Construct the metric closure G_L on the terminal set L. G_L is the complete graph in which each edge is weighted by the shortest path between the nodes in G.;
  Find a minimum spanning tree T_L on G_L.;
  T ← ∅.;
  for *each edge* $e = (u, v) \in E(T_L)$ **do**
    Find a shortest path P from u to v on G.;
    if P *contains less than two vertices in* T **then**
      Add P to T;
    else
      Let p_i and p_j be the first and the last vertices already in T;
      Add sub-paths from u to p_i and from p_j to v to T;
    end
  end
  return T;
end

MST-Steiner algorithm has the complexity $O(n|L|^2)$ [3].

B. i-SPT-Steiner algorithm

Definition 5: The shortest path tree [3]

Given a graph G, Shortest Path Tree (SPT) with the origin at vertex s is the spanning tree of G with a set of edges that is on the shortest paths starting from vertex s to the remaining vertices of G.

The shortest path tree originating at the vertex s of graph G can be found by applying Dial algorithm [28] (a variant of Dijkstra algorithm is recommended for small edge weights) to find the shortest path from vertex s to all remaining vertices.

In the experimental data system of this paper, the maximum edge weight is 10. The complexity of Dial algorithm is $O(m + n * C)$ [28] where C is the maximum weighted value of the graph.

Algorithm 2: i-SPT-Steiner (improved-Shortest Path Tree-Steiner)

Inputs: Graph $G = (V, E, w)$ and a terminal set $L \subset V$
Outputs: Steiner tree T
begin
  Find the shortest path trees starting at the vertices of the set V: SPT_1, SPT_2, …, $SPT_{|v|}$, each SPT_i only need to find the shortest path tree containing all vertices $v \in L$ (In this step, use Dial algorithm [28] to find the shortest paths instead of using the Dijkstra algorithm as in SPT-Steiner heuristic [4]).;
  For each tree $T = SPT_i$, traversal pendants $u \in V(T)$, if $u \notin L$ and u is a pendant vertex, delete the edge containing u from $E(T)$, delete vertex u in $V(T)$ and update the degree of the vertex adjacent to the vertex u in T. Repeat this step until T no longer changes (this step is called deleting redundant edges - the edge contains a pendant vertex u); after this step, the $SPT_{i'}$ is obtained, corresponding to SPT_i.;
  In the process of deleting redundant edges, the queue is used to contain vertices. These vertices are not in set L, and the priority is the vertex with small degree. This technique of using the queues during this phase minimizes the time to remove redundant edges. ;
  Find a $SPT_{i'}$ with the lowest total weight.;
  return T;
end

The complexity for i-SPT-Steiner algorithm

Since Dial algorithm uses a queue data structure with time complexity of $O(m + n * C)$ [28], step 1 has the complexity of $O(n * (m + n * C))$, step 2 has the complexity of $O(n^2)$, and step 3 has the complexity of $O(n)$. As a result, the i-SPT-Steiner algorithm has time complexity of $O(n * (m + n * C))$.

Comparing with the SPT-Steiner algorithm [4], each shortest path tree must traverse through all the vertices of the set V, so the SPT-Steiner algorithm has a much longer running time. However, these have been improved in the i-SPT-Steiner algorithm.

In the i-SPT-Steiner algorithm, it is only required to find the shortest path tree that contains all the vertices in the set L. This is due to the fact that the remaining vertices that have not been traversed are definitely redundant and will be removed when the deleting redundant edges step (in step 2) is performed. This will help to reduce the running time of the program (about 2 - 3 times faster with our experimental data). Furthermore, a priority queue data structure is used to reduce the complexity of the step of deleting redundant edges.

Example 2: Illustration of step-by-step implementation of the i-SPT-Steiner algorithm

Given a graph G with 10 vertices and 15 edges as Figure 3; where set $L = \{1, 5, 6\}$.

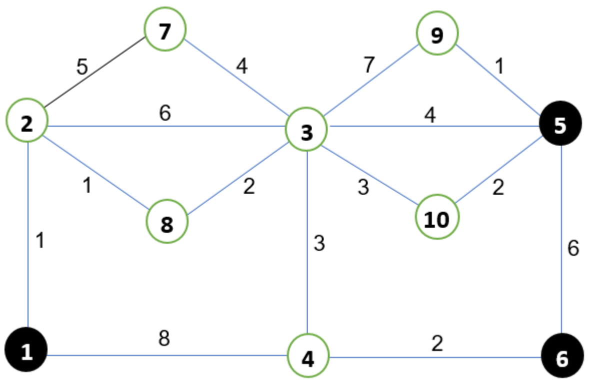

Figure 3. Weighted interconnected undirected graph G

According to the definition 4 and the i-SPT-Steiner algorithm, we construct the shortest-path trees originating at vertices **1, 2, 3, 4, 5, 6, 7, 8, 9,** and **10**. After deleting the edges, we have Steiner trees with values of 13, 13, 13, 13, 14, 15, 19, 13, 14, and 15 respectively.

We consider Figure 3. Firstly, the shortest path tree rooted at vertex **1** need to be found; as shown in Figure 4. Next, the excess edges **(2, 7), (5, 9), (3, 10)** are removed, and a Steiner tree with a cost of 13 is obtained, as shown in Figure 5.

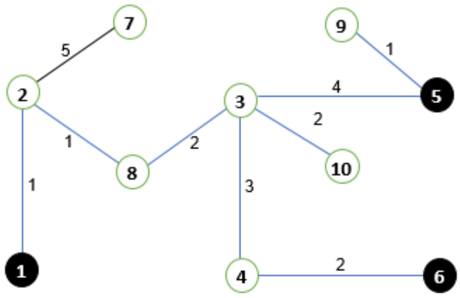

Figure 4. The shortest path tree rooted at vertex **1**

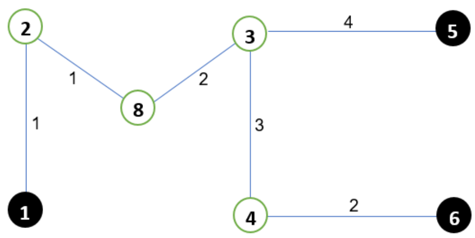

Figure 5. Steiner tree after redundant edges have been deleted

According to the above i-SPT-Steiner algorithm, the found Steiner tree costs 13, as shown in Figure 5.

Algorithm 3: i-PD-Steiner (improved-Prim + Dijkstra-Steiner)

---

1 **Inputs**: Graph $G = (V, E, w)$ and a terminal set $L \subset V$
2 **Outputs**: Steiner tree T
3 **begin**
4   Select a vertex $u \in L$; set $V(T) = \{u\}$;
5   **for** *each vertex* $v \in L$ **do**
6   Find the shortest path tree from vertex v;
7   (In this step, use Dial algorithm [28] to find the shortest paths instead of using the Dijkstra algorithm as in PD-Steiner heuristic [4]);
8   **end**
9   **while** *T does not contain all vertices in the set L* **do**
10   From each vertex $v \in L$ and v is not already in T, select the shortest path P from vertex v to vertex $z \in T$, with z as the vertex of the shortest path of all paths starting from vertex v to vertices of $V(T)$;
11   Add vertices and edges on path P to tree T and ignore the vertices and edges already in T;
12   **end**
13   **return** T;
14 **end**

C. i-PD-Steiner algorithm

The complexity for i-PD-Steiner algorithm

Line 1 has the complexity of $O(1)$;

Line 3 Due to using the Dial algorithm to find the shortest path tree, the complexity is $O(m+n*C)$. As a result, the loop in **line 2** has complexity of $O(|L|*(m+n*C))$.

Line 6 has complexity of $O(|L|*|T|)$, so the loop in **line 5** has complexity of $O(|L|^2 * |T|)$.

Altogether, the i-PD-Steiner algorithm has time complexity: $O(|L| * max(m + n * C, |L| * |T|))$.

Example 3: Illustration of step-by-step implementation of the i-PD-Steiner algorithm

The consideration of Figure 3 makes an illustration of the step-by-step implementation if i-PD-Steine algorithm is applied. Firstly, choose $V(T) = \{\mathbf{1}\}$, the shortest path from vertex **5** to vertex **1** is shown in Figure 6 (vertex **5** is chosen because it's a vertex in L, not in $V(T)$ which has the shortest path to a vertex in T). The next step is to find the shortest paths from vertex **6** to vertices **1, 2, 3, 5,** and **8**. The ones found have values of **9, 8, 5, 9, and 7** respectively. After that, the shortest path from vertex **6** to vertex **3** is selected. The found Steiner tree is shown in Figure 7 at a cost of 13.

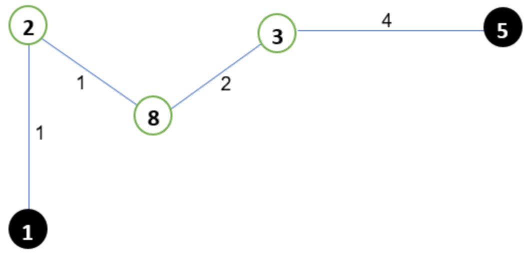

Figure 6. The shortest path from vertex **5** to vertex **1**

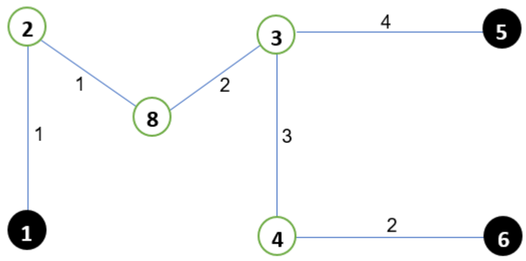

Figure 7. The shortest path from vertex **6** to vertex **3**

And according to the above i-PD-Steiner algorithm, the found Steiner tree costs 13, such as in Figure 7.

III. EXPERIMENT AND EVALUATION

A. Experimental data

Since using the heuristic algorithms to approach SMT problems has more advantages than using the metaheuristic algorithms for the large size graphs, at least in terms of timing, 80 datasets are recommended. Of them, 20 test sets are sparse graphs with 10000 vertices that are randomly generated and named *steinf1.txt, steinf2.txt, ..., steinf20.txt*; 20 test sets are sparse graphs with 20000 vertices that are randomly generated and named *steing1.txt, steing2.txt,. . . , steing20.txt*; 20 test sets are sparse graphs with 50000 vertices randomly generated and named *steinh1.txt, steinh2.txt,. . . , steinh20.txt* and 20 test sets are sparse graphs with 100000 vertices that are randomly generated and named *steini1.txt, steini2.txt,. . . , steini20.txt*.

These graphs are generated first from a random spanning tree that traverses all n vertices of the graphs; then, the edges are randomly generated until the graph has m edges; Edge weights are also randomly generated and have a maximum value of 10.

B. Experimental environment

MST-Steiner, i-SPT-Steiner, i-PD-Steiner algorithms are installed in C++17 using the Code::Blocks 17.12 environment and GNU GCC ver 9.3.0 compiler. They are also tested on a virtual server running Ubuntu 20.04.1 LTS (Focal Fossa), 64-bit, Intel (R) Xeon (R) Platinum 8160 @ 2.8 GHz CPU, 16 cores, 33MB Cache, and 64GB RAM.

Compile command that has been used for the experiment:

```
g++ -std=c++17 –O2 -o MST_STEINER MST_STEINER.cpp
g++ -std=c++17 –O2 -o SPT_STEINER SPT_STEINER.cpp
g++ -std=c++17 –O2 -o PD_STEINER PD_STEINER.cpp
```

In particular, most of the data structures have been changed so as to become more dynamic and optimal data structures. As a result, the experiment can be tested with large sparse graphs with up to 100000 vertices.

C. Experimental results and evaluation

The experimental results of the algorithms are recorded in Table II, Table III, Table IV, and Table V. These tables follow the hereafter patterns and labels. The first column (Test) is the name of the datasets in the experimental data system; the number of vertices (n), edges (m) and vertices in the *terminal* set ($|L|$) of each graph; The subsequent columns record the value of the Steiner tree cost for each algorithm.

TABLE II
THE EXPERIMENTAL RESULT OF THE ALGORITHMS ON STEINF GRAPH GROUP

Test	n	m	L	MST-Steiner	i-SPT-Steiner	i-PD-Steiner
steinf1.txt	10000	93750	10	58	49	51
steinf2.txt	10000	93750	20	97	95	94
steinf3.txt	10000	93750	834	2119	2551	2027
steinf4.txt	10000	93750	1250	2801	3532	2691
steinf5.txt	10000	93750	2500	4957	6567	4826
steinf6.txt	10000	125000	10	40	40	39
steinf7.txt	10000	125000	20	71	67	66
steinf8.txt	10000	125000	834	1961	2399	1838
steinf9.txt	10000	125000	1250	2576	3303	2444
steinf10.txt	10000	125000	2500	4282	5823	4124
steinf11.txt	10000	156250	10	35	35	35
steinf12.txt	10000	156250	20	77	74	70
steinf13.txt	10000	156250	834	1727	2199	1627
steinf14.txt	10000	156250	1250	2335	3049	2243
steinf15.txt	10000	156250	2500	3933	5398	3797
steinf16.txt	10000	187500	10	43	38	38
steinf17.txt	10000	187500	20	75	69	66
steinf18.txt	10000	187500	834	1597	2050	1513
steinf19.txt	10000	187500	1250	2244	2910	2153
steinf20.txt	10000	187500	2500	3764	5122	3624

Evaluating 20 datasets on *steinf* group displayed in Table II leads to certain comparative values in terms of quality. i-PD-Steiner algorithm gives *better* solution quality, *equivalent, less than* MST-Steiner algorithm with the percentages of 95.0%, 5.0%, and 0.0%; i-SPT-Steiner algorithm gives *better* solution quality, *equivalent, less than* MST-Steiner algorithm with the percentages of 30.0%, 10.0%, and

60.0%; and i-PD-Steiner algorithm gives *better* solution quality, *equivalent, less than* i-SPT-Steiner algorithm with the percentages of 85.0%, 10.0% and 5.0%.

The comparison of algorithms on steinf datasets are illustrated by the chart as shown in Figure 8.

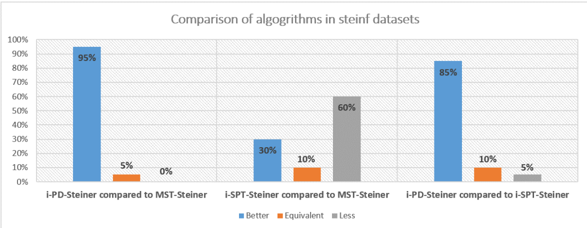

Figure 8. The comparison of algorithms on *steinf* datasets

TABLE III
THE EXPERIMENTAL RESULT OF THE ALGORITHMS ON STEING GRAPH GROUP

Test	n	m	L	MST-Steiner	i-SPT-Steiner	i-PD-Steiner
steing1.txt	20000	215000	15	79	75	75
steing2.txt	20000	215000	25	110	102	105
steing3.txt	20000	215000	950	2426	3081	2348
steing4.txt	20000	215000	1750	4039	5085	3838
steing5.txt	20000	215000	3780	7318	9689	7099
steing6.txt	20000	275000	15	71	66	64
steing7.txt	20000	275000	25	112	111	100
steing8.txt	20000	275000	950	2245	2830	2129
steing9.txt	20000	275000	1750	3643	4709	3489
steing10.txt	20000	275000	3780	6511	8773	6319
steing11.txt	20000	385000	15	65	60	59
steing12.txt	20000	385000	25	98	94	90
steing13.txt	20000	385000	950	2041	2577	1919
steing14.txt	20000	385000	1750	3211	4253	3081
steing15.txt	20000	385000	3780	5788	7981	5641
steing16.txt	20000	447500	15	61	55	54
steing17.txt	20000	447500	25	95	93	87
steing18.txt	20000	447500	950	1954	2485	1845
steing19.txt	20000	447500	1750	3079	4063	2970
steing20.txt	20000	447500	3780	5534	7647	5380

Evaluating 20 datasets on *steing* group displayed in Table III leads to certain comparative values in terms of quality. i-PD-Steiner algorithm gives *better* solution quality, *equivalent, less than* MST-Steiner algorithm with the percentages of 100.0%, 0.0%, and 0.0%; i-SPT-Steiner algorithm gives *better* solution quality, *equivalent, less than* MST-Steiner algorithm with the percentages of 40.0%, 0.0%, and 60.0%; and i-PD-Steiner algorithm gives *better* solution quality, *equivalent, less than* i-SPT-Steiner algorithm with the percentages of 90.0%, 5.0% and 5.0%.

The comparison of algorithms on steing datasets are illustrated by the chart as shown in Figure 9.

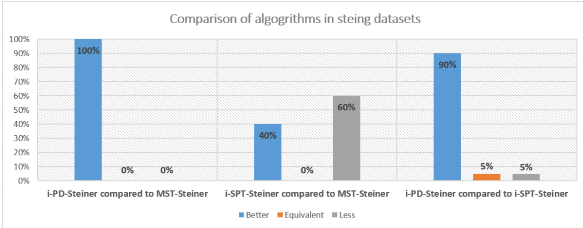

Figure 9. The comparison of algorithms on *steing* datasets

TABLE IV
THE EXPERIMENTAL RESULT OF THE ALGORITHMS ON STEINH GRAPH GROUP

Test	n	m	L	MST-Steiner	i-SPT-Steiner	i-PD-Steiner
steinh1.txt	50000	425000	15	76	71	71
steinh2.txt	50000	425000	25	129	116	118
steinh3.txt	50000	425000	950	2707	3296	2587
steinh4.txt	50000	425000	1750	4344	5466	4125
steinh5.txt	50000	425000	3780	7947	10348	7563
steinh6.txt	50000	475000	15	71	70	66
steinh7.txt	50000	475000	25	99	105	96
steinh8.txt	50000	475000	950	2531	3129	2402
steinh9.txt	50000	475000	1750	4198	5299	3951
steinh10.txt	50000	475000	3780	7718	10099	7336
steinh11.txt	50000	528000	15	80	72	74
steinh12.txt	50000	528000	25	112	110	106
steinh13.txt	50000	528000	950	2498	3082	2378
steinh14.txt	50000	528000	1750	4026	5132	3784
steinh15.txt	50000	528000	3780	7275	9702	6939
steinh16.txt	50000	587500	15	70	67	63
steinh17.txt	50000	587500	25	123	109	109
steinh18.txt	50000	587500	950	2408	2962	2262
steinh19.txt	50000	587500	1750	3886	4987	3643
steinh20.txt	50000	587500	3780	7127	9489	6797

Evaluating 20 datasets on *steinh* group displayed in Table IV leads to certain comparative values in terms of quality. i-PD-Steiner algorithm gives *better* solution quality, *equivalent, less than* MST-Steiner algorithm with the percentages of 100.0%, 0.0%, and 0.0%; i-SPT-Steiner algorithm gives *better* solution quality, *equivalent, less than* MST-Steiner algorithm with the percentages of 35.0%, 0.0%, and 65.0%; and i-PD-Steiner algorithm gives *better* solution quality, *equivalent, less than* i-SPT-Steiner algorithm with the percentages of 80.0%, 10.0% and 10.0%.

The comparison of the algorithms on steinh datasets are illustrated by the chart as shown in Figure 10.

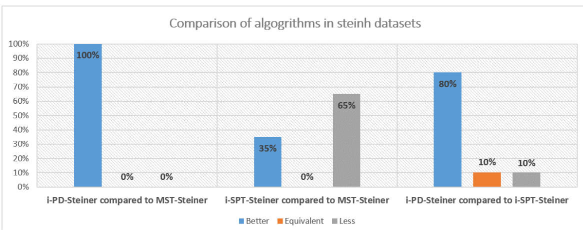

Figure 10. The comparison of algorithms on *steinh* datasets

TABLE V
THE EXPERIMENTAL RESULT OF THE ALGORITHMS ON STEINI GRAPH GROUP

Test	n	m	L	MST-Steiner	i-SPT-Steiner	i-PD-Steiner
steini1.txt	100000	125000	25	188	190	180
steini2.txt	100000	125000	45	369	364	358
steini3.txt	100000	125000	1250	6390	6677	6363
steini4.txt	100000	125000	2450	12209	12810	12178
steini5.txt	100000	125000	4500	21754	22868	21664
steini6.txt	100000	200000	25	148	144	142
steini7.txt	100000	200000	45	264	273	253
steini8.txt	100000	200000	1250	4978	5510	4902
steini9.txt	100000	200000	2450	9066	10376	8969
steini10.txt	100000	200000	4500	15126	17456	14879
steini11.txt	100000	500000	25	111	110	104
steini12.txt	100000	500000	45	184	191	173
steini13.txt	100000	500000	1250	3133	3849	2974
steini14.txt	100000	500000	2450	5487	6889	5207
steini15.txt	100000	500000	4500	8951	11629	8532
steini16.txt	100000	2500000	25	66	72	66
steini17.txt	100000	2500000	45	120	128	111
steini18.txt	100000	2500000	1250	2031	2556	1951
steini19.txt	100000	2500000	2450	3366	4529	3268
steini20.txt	100000	2500000	4500	5167	7591	5118

Evaluating 20 datasets on *steini* group displayed in Table V leads to certain comparative values in terms of quality. i-PD-Steiner algorithm gives *better* solution quality, *equivalent, less than* MST-Steiner algorithm with the percentages of 95.0%, 5.0%, and 0.0%; i-SPT-Steiner algorithm gives *better* solution quality, *equivalent, less than* MST-Steiner algorithm with the percentages of 15.0%, 0.0%, and 85.0%; and i-PD-Steiner algorithm gives *better* solution quality, *equivalent, less than* i-SPT-Steiner algorithm with the percentages of 100.0%, 0.0% and 0.0%.

The comparison of the algorithms on steini datasets are illustrated by the chart as shown in Figure 11.

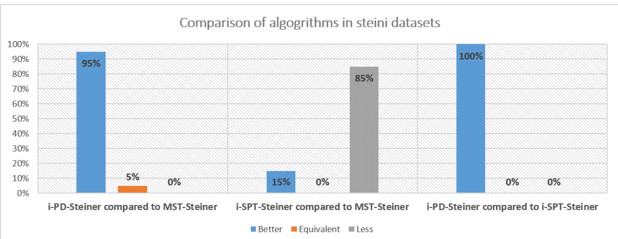

Figure 11. The comparison of algorithms on *steini* datasets

D. Evaluation of experimental results

Evaluating 80 datasets leads to certain comparative values in terms of quality. i-PD-Steiner algorithm gives *better* solution quality, *equivalent*, *less than* MST-Steiner algorithm with the percentages of 97.5%, 2.5%, and 0.0%; i-SPT-Steiner algorithm gives *better* solution quality, *equivalent*, *less than* MST-Steiner algorithm with the percentages of 30.0%, 2.5%, and 67.5%; and i-PD-Steiner algorithm gives *better* solution quality, *equivalent*, *less than* i-SPT-Steiner algorithm with the percentages of 88.75%, 6.25% and 5.0%. We noted the average run time of three algorithms i-PD-Steiner, MST-Steiner and i-SPT-Steiner on datasets as shown in Table VI.

TABLE VI
AVERAGE RUNNING TIME OF ALGORITHMS (UNIT IN SECONDS)

Groups of graphs	i-PD-Steiner	MST-Steiner	i-SPT-Steiner
steinf.txt	83.758	137.538	928.68
steing.txt	280.56	421.402	2374.775
steinh.txt	416.057	766.585	30416.253
steini.txt	1243.415	1561.921	93943.235

In other words, i-PD-Steiner algorithm has a much faster runtime than the MST-Steiner and i-SPT-Steiner algorithms. The runtime depends on not only the time complexity of the algorithm but also the experimental environment. Besides, the information about the runtime of the algorithms displayed above also provides reliable reference information about these algorithms (The queue data structure has been chosen to install the above algorithms).

The comparison of i-PD-Steiner, i-SPT-Steiner, and MST-Steiner algorithms on a total of 80 datasets are illustrated by the chart as shown in Figure 12.

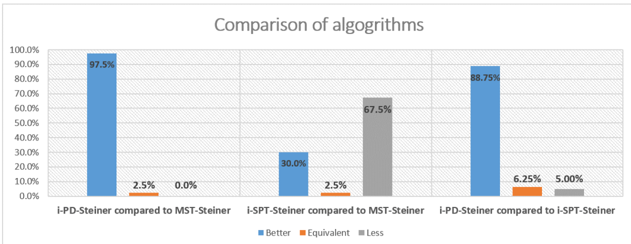

Figure 12. The comparison of the algorithms on 80 datasets

IV. CONCLUSION

The authors of this paper proposed the improvement to the heuristic algorithms SPT-Steiner and PD-Steiner [4] to solve the SMT problem in large size sparse graphs with edge weight not exceeding 10. Experiments were set up and evaluation was made in details on 80 datasets of sparse graphs with large sizes up to 100000 vertices. i-PD-Steiner algorithm gives *better* quality solution or *equivalent* MST-Steiner algorithm on 100.0% datasets. i-SPT-Steiner algorithm gives *better* quality solution or *equivalent* MST-Steiner algorithm on 32.5% datasets. i-PD-Steiner algorithm gives *better* quality solution or *equivalent* i-SPT-Steiner algorithm above 95.0% of the data. The runtime of the i-PD-Steiner algorithm is much shorter than the one of the i-SPT-Steiner algorithm. The two heuristic algorithms i-SPT-Steiner and i-PD-Steiner as well as the experimental results in this paper are useful information for further studies on the SMT problem, especially for solving the SMT problems in the case of large sparse graphs.

References

[1] Ankit Anand, Shruti, Kunwar Ambarish Singh, ”An efficient approach for Steiner tree problem by genetic algorithm”, *International Journal of Computer Science and Engineering (SSRG-IJCSE)*, vol.2, 2015, pp.233-237.

[2] Arie M.C.A. Koster, Xavier Munoz (Eds), ”Graphs and algorithms in communication networks”, *Springer*, 2010, pp.1-177.

[3] Bang Ye Wu, Kun-Mao Chao, ”Spanning trees and optimization problems”, *Chapman&Hall/CRC*, 2004, pp.13-139.

[4] Tran Viet Chuong, Phan Tan Quoc, Ha Hai Nam, ”Proposing heuristic algorithms to solve the Steiner Minimal Tree problem”, *Proceedings of the 10th National Conference on Fundamental and Applied Information Technology (FAIR)*; The University of Danang, August 17-18, 2017, ISBN: 978-604-913-614-6, Pages 138-147.

[5] C. C. Ribeiro, M.C. Souza, ”Tabu Search for the Steiner problem in graphs”, *Networks*, 36, 2000, pp.138-146.

[6] Chin Lung Lu, Chuan Yi Tang, Richard Chia-Tung Lee, ”The full Steiner tree problem”, *Elsevier*, 2003, pp.55-67.

[7] Chi-Yeh Chen, ”An efficient approximation algorithm for the Steiner tree problem”, National Cheng Kung University, Taiwan, 2018.

[8] Davide Bilò (University of Sassari), ”New algorithms for Steiner tree reoptimization”, *ICALP*, 2018.

[9] Daniel Markus Rehfeldt, ”Generic Approach to Solving the Steiner Tree Problem and Variants”, Fachbereich Mathematik der Technischen University at Berlin, 2015.

[10] Ding-Zhu Du, J. M. Smith, J.H. Rubinstein, ”Advances in Steiner trees”, 2000, pp.1-322.

[11] Fichte Johannes K., Hecher Markus, and Schidler André, ”Solving the Steiner Tree Problem with few Terminals”, University of Potsdam, Germany, 2020.

[12] Ivana Ljubic, ”Solving Steiner Trees – Recent Advances”, *Challenges and Perspectives*, 2020.

[13] Jack Holby, ”Variations on the Euclidean Steiner Tree Problem and Algorithms”, St. Lawrence University, 2017.

[14] J.E.Beasley, OR-Library: URL http://people.brunel.ac.uk/ mastjjb/jeb/orlib/steininfo.html

[15] J. E. Beasley, ”An SST-Based algorithm for the Steiner problem in graphs”, *Networks*, 19, 1989, pp.1-16.

[16] Jeffrey H. Kingston, Nicholas Paul Sheppard, ”On reductions for the Steiner problem in graphs”, Basser Department of Computer Science the University of Sydney, Australia, 2006, pp.1-10.

[17] Jon William Van Laarhoven, ”Exact and heuristic algorithms for the euclidean Steiner tree problem”, University of Iowa, 2010, (Doctoral thesis).

[18] Job Kwakernaak, ”Pipeline network optimization using Steiner nodes”, Wageningen University and Research Centre, The Netherlands, 2020.

[19] L. Kou, G. Markowsky, L. Berman, ”A Fast Algorithm for Steiner Trees”, *acta informatica*, Vol.15, pp.141-145, 1981.

[20] M. Hauptmann, M. Karpinski (Eds), ”A compendium on Steiner tree problems”, 2015, pp.1-36.

[21] Martin Zachariasen, ”Algorithms for Plane Steiner Tree Problems”, University of Copenhagen, 1998 (PH.D. Thesis).

[22] Marcello Caleffi, Ian F. Akyildiz, Luigi Paura, ”On the solution of the Steiner tree np-hard problem via physarum bionetwork”, *IEEE*, 2015, pp.1092-1106.

[23] Matjaz Kovse, ”Vertex decomposition of steiner wiener index and Steiner betweenness centrality”, University of Wroc law, Poland, 2016.

[24] Nguyen Viet Huy, Nguyen Duc Nghia, ”Solving graphical Steiner tree problem using parallel genetic algorithm”, *RIVF*, 2008.

[25] Ondra Suchý, ”Exact algorithms for Steiner tree”, Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic, 2014.

[26] Pieter Oloff De Wet, ”Geometric steiner minimal trees”, university of South Africa, 2008 (Thesis).

[27] Prosenjit Bose, Anthony D'Angelo, Stephane Durocher, ”On the Restricted 1-Steiner Tree Problem”, *Funded in part by the Natural Sciences and Engineering Research Council of Canada*, 2020.

[28] Prof. James Orlin, 15.082J Network Optimization MIT Course, Fall 2010.

[29] Radek Hušek, Dušan Knop, Tomáš Masařík, ”Approximation Algorithms for Steiner Tree Based on Star Contractions: A Unified View”, *International Symposium on Parameterized and Exact Computation (IPEC 2020)*, ACM, 2020.

[30] Reyan Ahmed and et al, ”Multi-level Steiner trees”, *ACM J. Exp. Algor.*, 1(1), 2018.

[31] Robert Sedgewick, Kevin Wayne, ”Algorithms”, Fourth edition, Addsion-Wesley, pp.518-700, 2011.

[32] S. E. Dreyfus, R. A. Wagner, ”The Steiner Problem in Graphs”, *Networks*, Vol.1, pp.195-207, 1971.

[33] Siavash Vahdati Daneshmand, ”Algorithmic Approaches to the Steiner Problem in Networks”, Mannheim, 2003.

[34] Thomas Pajor, Eduardo Uchoa, Renato F. Werneck, ”A robust and scalable algorithm for the Steiner problem in graphs”, *Springer*, 2018.

[35] Thomas Bosman, ”A solution merging heuristic for the Steiner problem in graphs using tree decompositions”, VU University Amsterdam, The Netherlands, 2015, pp.1-12, 2015.

[36] Thorsten Koch, Alexander Martin, ”Solving Steiner tree problems in graphs to optimality”, Germany, 1996, pp.1-31.

[37] Tobias Polzin, ”Algorithms for the Steiner Problem in Networks”, Universit¨at des Saarlandes, 2003 (Thesis).

[38] Vu Dinh Hoa, ”Steiner Tree Problem”, *Institute of Mathematics*, Vietnam Academy of Science and Technology, http://math.ac.vn.

[39] Wassim Jaziri (Edited). ”Local Search Techniques: Focus on Tabu Search”. In-teh, Vienna, Austria, 2008, pp.1-201.

[40] Xinhui Wang, ”Exact algorithms for the Steiner tree problem”, *doctoral thesis*, ISSN 1381-3617, 2008.

[41] Xiuzhen Cheng, Ding-Zhu Du, ”Steiner trees in industry”, *Kluwer Academic Publishers*, vol.5, pp.193-216, 2004.

[42] Xin-She Yang, ”Engineering optimization”, *WILEY*, 2010, pp.21-137.

[43] Xin-She Yang, ”Nature-inspired metaheuristic algorithms”, *LUNIVER Press*, 2010, pp.53–62.

[44] Yahui Sun, ”Solving the Steiner Tree Problem in Graphs using Physarum-inspired Algorithms”, 2019.

[45] Zhiliu Zhang, ”Rectilinear Steiner Tree Construction”, Lincoln, Nebraska, USA, 2016 (THESIS).

Chuong Tran-Viet received a Master of Science degree in Computer Science in 2005 from Hanoi National University of Education.
He is currently working at the Center for Information Technology and Communications of Ca Mau province, Department of Information and Communications of Ca Mau province, Viet Nam.
Research Area: Evolutionary and Optimization Algorithms.
Email: chuongtv@camau.gov.vn

Nam Ha-Hai received a Master of Informatics degree in 2004 from Hanoi University of Science and Technology and Ph.D in 2008 in UK.
He is currently working at: Vietnam National Institute of Software and Digital Content Industry, Ministry of Information and Communication, Viet Nam.
Research Area: Distributed systems and optimization.
Email: namhh@ptit.edu.vn

Quoc Phan-Tan received Master degree of computer science in 2002 from University of Science, Vietnam National University Ho Chi Minh City and received Doctor degree of computer science in 2015 from Hanoi University of Science and Technology (HUST).
He is currently a lecturer of Information Technology Faculty of Sai Gon University, Ho Chi Minh City, Viet Nam.
His research interests are metaheuristic algorithms.
Email: quocpt@sgu.edu.vn