G. Grigoras and P. Lorenz (Eds.)

doi:10.3233/FAIA230914

A Novel Heuristics for the Strong Generalized Minimum Label Spanning Tree Problem

Yinyan LONG[a], Xiaoxia LI[b], Zhuming LONG[b], Fuying WU[b], Hao LONG[b1]
[a] *School of Computer Engineering and Science, ShangHai University, ShangHai, 200444, China*
[b] *School of Software, Jiangxi Normal University, Nanchang, 330022, China*

Abstract: The strong generalized minimum label spanning tree problem (SGMLSTP) is to search the minimum label spanning tree (MLST) from an Edge-labeled graph (ELG), in which each edge is associated with one or more labels. SGMLSTP is commonly existed in reality and proven NP-hard. In recent years, researchers have proposed some algorithms; however, high computational costs are still severe obstacle, especially for large size graphs. In this paper, we propose a novel heuristics to solve SGMLSTP. We decompose the problem into two sub-problems, one is to search a connected subgraph with minimum labels from the original graph, the other is to search a spanning tree from the subgraph. As the latter sub-problem is solved, we focus on the former sub-problem and propose a community-based zigzag piloting algorithm: Firstly a label graph is derived from the original edge-labeled graph; then the label graph is partitioned and some label community (or community combinations) is chosen to form an initial solution; finally, the zigzag piloting process is applied to refine the initial solution. Label partition finds the initial solution quickly, the zigzag piloting process improves solution refinement. Experimental results on typical benchmark datasets show better effectiveness and performance of our algorithm than that of the state-of-the-art algorithms.

Keywords: edge-labeled graph; minimum label spanning tree; label community; zigzag piloting

1. Introduction

An edge-labeled graph (ELG) is one with each edge associating some qualitative characterization(s), instead of a quantitative measurement. ELG applications can be found in many fields in reality [1], for example, in the scene of transportation, each road might be assigned one or more labels representing different models; in a television network, labels might correspond to different channels; in a computer network, labels might be assigned to different kind of services, etc. Among these applications, the fewer the chosen labels, the smaller the economic cost, and the less the complexity.

The minimum label spanning tree problem (MLSTP) is to find a spanning tree with the minimum number of labels (MLST) from an edge-labeled graph, it was firstly

[1] Corresponding Author: Hao LONG, Ph.D, Professor, born in 1970, his research interests include artificial intelligence, graph theory, complex networks, scheduling and optimization. e-mail: hhlong2010@hotmail.com

addressed by Chang and Leu [2], in which each edge has only one label; Chen et al. [3] proposed a generalized version of MLSTP, namely GMLSTP, where each edge has more labels but each edge of the extracted tree only choose one of its labels. Cerrone et al [17] introduced the strong GMLSTP problem (SGMLSTP), in which each edge of the extracted tree applies all its labels.

In the last decades, researchers found it is unlikely to solve MLSTP, GMLSTP, or SGMLSTP in polynomial time, therefore, they mainly adopted evolutionary algorithms or heuristics to handle these problems. However, high computational cost and low effectiveness are still obstacles, especially for large scale graphs. In this paper, we propose a novel approach to solve SGMLSTP. Our contributions include: (1)we rewrite the mathematical model of GMLSTP; (2) we decompose GMLSTP into two sub-problems, one is to search a sub-graph containing MLSTs from the original graph, another is to search a spanning tree from the sub-graph; as the latter sub-problem is solved, we propose a community-based zigzag piloting algorithm to solve the former one; (3) we perform experiments on typical benchmark datasets to evaluate the proposed algorithm and the compared algorithms.

The remainder of this paper is organized as follows. Section 2 reviews the related work. The new formulations of the problem and some new terms are introduced in Section 3. Section 4 describes the proposed algorithm. The experimental results are demonstrated in Section 5. Finally, the conclusions and our future work are reported in Section 6.

2. Related Work

2.1 Algorithms for the SGMLSTP and Its Variants

SGMLSTP is a generalized form of MLSTP and GMLSTP. In the last decade, some researchers have achieved some progress in solving these problems. Generally, these efforts can be roughly classified into two classes: evolutionary and heuristic algorithms.

MLSTP has been proven NP-complete [2], the authors also presented an A*-based exact algorithm and an approximation approach based on MVCA (the maximum vertex covering algorithm), which is commonly applied in construction or rebuilding phases of evolutionary methods. Xiong et al. [4] and Cerrone et al. [5] adopted the genetic algorithm and the modified genetic algorithm to solve MLSTP; Zahra et al. [6] adopted VNS (Variable neighborhood search) technique, Consoli et al.[7] combined VNS and GRAS (Greedy randomized adaptive search) techniques to search for the optimal solutions for MLSTP. Consoli et al. [8] adopted the concepts of complementary space as well as auto-adjusting parameters over a VNS framework, which can acquire better solutions. Silva et al. [9] proposed a compact binary integer programming model to solve GMLSTP, in [10] the authors introduced a mixed-integer linear program (MIP) formulation for MLSTP and applied the capacity of exploration of a new local search method based on MIP to find results. Other evolutionary algorithms included the Simulated Annealing and Reactive Tabu Search [11], the Ant Colony Optimization [12], firefly algorithm [13], multi-objective optimization [14], cross-entropy method [15], etc.

Heuristics show better performance in finding MLST [1], Chwatal and Raidl [16] abstracted MLSTP and GMLSTP with the mixed integer programming, They proposed several cuts to strengthen the models and implemented branch-and-cut or branch-and-cut-and-price algorithms to search feasible MLST solutions; Cerrone et al. [17] applied carousel greedy to solve MLSTP; In [18] the authors introduced a single-commodity flow mathematical formulation that perfectly represents MLST and SGMLST, and proposed three greedy algorithms, namely MMVCA (multi-label MVCA), MMVCA extended with CG(Constructive Greedy), and CG with the Pilot Method, to search promising solutions.

In most cases, the evolution algorithms can find suboptimal solutions for NP-hard problems, their main disadvantage is the high computational cost, especially for large-size graphs; heuristic algorithms usually require less computation time, but their effectiveness heavily depends on the heuristic knowledge.

2.2 Community Detection

Communities are the mid-level components of graphs with dense internal connections and sparse external connections, which usually consist of network members with higher similarity and correspond to special behaviors or functionalities. Because community detection is an important issue in network analysis, In the last decade, numerous algorithms were proposed to serve scientific or social needs, these algorithms can be loosely divided into four categories: optimization, clustering, model-based methods and heuristics methods [19,20].

Optimization methods define objective functions to evaluate network partitions, such as the modularity function, the Surprise function, the topological potential function, the spectral graph cutting function, et al. Many optimization techniques, such as the genetic algorithm, immune network or simulated annealing, can be employed to search the optimum graph divisions. The clustering techniques define some similarity measurement and classify nodes or edges of higher similarity into the same community, the similarity function can be defined on the eigenvalue spectrum of graph matrices, the number of common neighbors, the number of polygons, clustering coefficients, betweenness, density, etc. Model-based methods use probabilistic, diffusion, or dynamics models to simulate the community detection problem, these models include the stochastic block model, label propagation, random walks, Potts models, etc. The heuristic methods comprise the largest family of community detection algorithms, which usually apply a divisive or expanding strategy to search communities, some measurement or objective function is applied to terminate the division or expanding process. In our previous work, a fast and effective expanding heuristic algorithm, namely ICDA [21] was put forward to support weighted network division.

Exhaustive search methods are of lower efficiency and performance for SMLSTP. To avoid such disadvantage, we utilize the associations among labels to construct the initial solution and apply some piloting strategies to refine it.

3. Notations and Mathematical Model

Let $\mathrm{G} =< V, E, L >$ be an undirected and unweighted graph, in which V is the node set, E is the edge set and L is the label set. $|\mathrm{V}| = \mathrm{n}, |\mathrm{E}| = \mathrm{m}$ and $|\mathrm{L}|$ is the number of nodes, edges and labels, respectively. Every node is identified with a natural number $\mathrm{i}\ (1 \leq \mathrm{i} \leq \mathrm{n})$; symbol $\mathrm{e_{ij}} \in \mathrm{E}$ represents the edge between nodes $\mathrm{v_i}$, $\mathrm{v_j}$, its weight is $\mathrm{w(e_{ij})}$; usually it is associated with a label set $l(\mathrm{e_{ij}})(l(\mathrm{e_{ij}}) \subseteq \mathrm{L})$ (or say $l(\mathrm{e_{ij}})$ is the label function). For given edge subset $S(S \subseteq \mathrm{E})$, $V(S) =$

$\{v_i | e_{ij} \in S \wedge e_{ij} \in E, 1 \leq i,j \leq n, i \neq j\}$ is the corresponding node set, the labeled sub-graph they are made up of is denoted as $< V(S), S, L(V(S), S) >$ $(L(V(S), S) = \bigcup_{e_{ij} \in S} l(e_{ij}))$, $\varpi(S)$ is the number of its disjoint components of the graph $< V, S, L(V(S), S) >$.

SGMLST is to find a spanning tree of G with the minimum labels, it is to say, The goal is to find a tree $T =< V, E', L' >$, $E' \subseteq E(|E'| = n - 1)$, $L' = \bigcup_{e_{ij} \in E'} l(e_{ij}) \subseteq L$, and $|L'|$ is minimized.

Definition 1 (Denoted Edges) given label set $\alpha = \{l_1, l_2, \dots, l_t\}$, the denoted edges of the label set α are denoted as $D(\alpha) = \{e_{ij} | l(e_{ij}) \subseteq \alpha, e_{ij} \in E\}$.

According to this new concept, we can rewrite the SGMLST problem as two sub-problems: one is to search a minimum label set with its denoted edges containing a spanning tree of the original graph; another is to search MST from the sub-graph. They are presented as follows:

$$\text{Min } \sum_{l_i \in L} y_i \tag{1}$$

$$\text{s.t. } \varpi\left(D\left(\bigcup_{y_i=1} \{l_i\}\right)\right) = 1 \tag{2}$$

$$\varpi\left(D\left(\bigcup_{y_i=1} \{l_i\} - l\right)\right) > 1 (\forall l \in \bigcup_{y_i=1} \{l_i\}) \tag{3}$$

$$y_i = [0,1] \tag{4}$$

$$\sum_{e_{ij} \in D\left(\bigcup_{y_i=1} \{l_i\}\right)} x_{ij} = n - 1 \tag{5}$$

$$\text{s.t. } |V(\bigcup_{x_{ij}=1} \{e_{ij}\})| = n \tag{6}$$

$$\varpi\left(\bigcup_{x_{ij}=1} \{e_{ij}\}\right) = 1 \tag{7}$$

$$x_{ij} = [0,1] \tag{8}$$

Formula (1)-(4) constitute the former sub-problem, formula (1) presents the objective that the selected labels should be minimized; constraints (2) and (3) ensure the denoted edges of the chosen label set $(\bigcup_{y_i=1} \{l_i\})$ connect all vertexes of the original graph; constraint (4) gives the corresponding values of labels.

Formula (5)-(6) form the equations of the latter sub-problem, formula (5) presents the objective of the spanning tree with the least number of labels; constraints (6) and (7) request the selected edges is a spanning tree of the original graph; constraint (8) gives the corresponding values of edges.

Because searching MST from a graph is already solved, in this paper we focus on the former sub-problem. Firstly we introduce some concepts helpful for such efforts.

Definition 2 (Label Graph) Given ELG $G =< V, E, L >$, the corresponding label graph is $LG =< L, E_L >$, $e^L_{xy} \in E_L$ is the relationship between labels l_x and l_y if they coexist in some edge of G, $E_L = \{e^L_{xy} | l_x \in l(e_{ij}) \wedge l_y \in l(e_{ij}), e_{ij} \in E\}$, $w(e^L_{xy}) = \sum_{e^L_{xy}} 1/|l(e_{ij})|$.

The label graph consists of labels from the original graph as vertexes, their coexistences as weighted connections. Given labels l_a and l_b, if they coexist on some edge e_{ij} in the original graph, l_a and l_b are transformed as two vertexes of the label graph and they are connected with part of the edge weight as $1/|l(e_{ij})|$, the total weight of the edge between them is the summary probability of their co-existence, which equals to $\sum_{e^L_{xy}} 1/|l(e_{ij})|$.

Definition 3 (Label Community) when a given weight label graph $LG =< L, E_L >$ is partitioned, its disjoint label communities $\{C_1, \ldots, C_K\}$ ensure labels with higher connections are clustered together and labels with sparse connections belong to different components, which can be measured with the modularity $Q = \frac{1}{2W}\sum_{x,y}[w_{xy} - \frac{(w_x w_y)}{2W})]\,\delta(C_x, C_y)$ (where $W = \sum_{x,y} w(e^L_{x,y})$, and δ is the Kronecker function, if the two labels l_x and l_y belong to the same disjoint community $(C(l_x) == C(l_y))$, $\delta\big(C(l_x), C(l_y)\big) = 1$; otherwise it equals to 0.

Definition 5(Preferable index) for label communities $C_k (1 \le k \le K)$ of the label graph $LG =< L, E_L >$, its preferable index is denoted as $\mathrm{pi}(C_k) = [|C_k| + \varpi\big(D(C_k)\big)]$.

Labels of the same community have close associations. Given a label community, the less its preferable index is, the more likely its denoted edges belong to an optimum solution of the former sub-problem. For the former sub-problem, we choose someone community (or combinations of communities) with smaller preferable indexes to form the original solutions.

Definition 6(Label additional index) Given a label set $\alpha = \{l_1, l_2, \ldots, l_t\} \sqsubseteq L$ of labeled graph $G =< V, E, L >$, $\varpi\big(D(\alpha)\big) > 1$. The additional index of label $l(l \in L¥\alpha)$is $A_\alpha(l) = \varpi\big(D(\alpha)\big) - \varpi\big(D(\alpha \cup \{l\})\big)$.

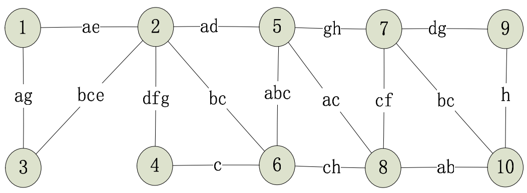
Any label with a positive additional index is an additional one, the larger it is, the more likely it should be appended into the current chosen labels.

Definition 7(Label Reduced index) Given a label set $\alpha = \{l_1, l_2, \ldots, l_t\} (\alpha \sqsubseteq L)$ of labled graph $G =< V, E, L >$, $\varpi\big(D(\alpha)\big) = 1$. The reduced index of label $l(l \in \alpha)$ is denoted as:

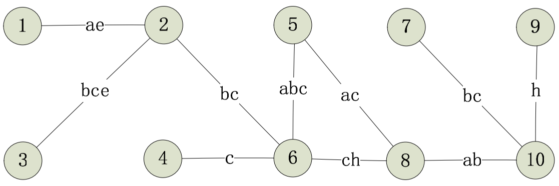
$R_\alpha(l) = D(\alpha) - D(\alpha - \{l\})$.

Any label with smaller reduced index can be swapped out to refine the current solution set.

When some label community (or community combinations) is chosen as an initial solution, the denoted edges of these labels are likely not a connected subgraph of the original graph, we can add external labels in addition: we sort external labels according to their additional indexes, each time the label with the largest additional index is added until the denoted edges of all chosen labels constitute a connected subgraph of the original graph. To optimize the current solution, we can also reduce labels with smaller reduced indexes. The expanding and reducing sub-process can appear in any combination, consist of the whole zigzag piloting process. Figure 1 illustrates an edge-labeled example graph and the optimum solution to its former sub-problem. Figure 2 shows the corresponding label graph with communities being extracted out and their denoted edges.

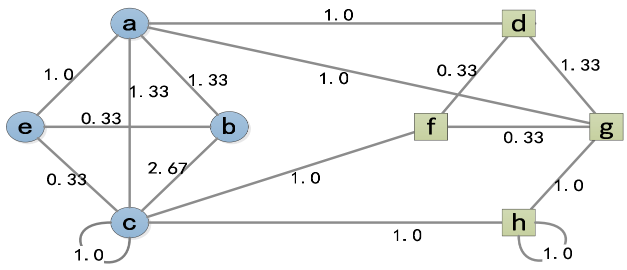


(a)Example edge-labeled graph

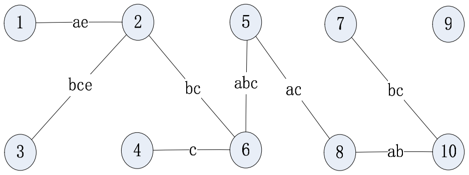


(b) the optimal solution to the former sub-problem

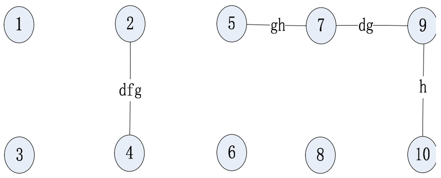
Figure 1 Example labeled graph and the optimal solution to the former sub-problem



(a) Label graph and label communities



(b)Denoted edges of the left label community



(c)Denoted edges of the right label community

Figure 2 Label graph and label communities, denoted edges of different communities

Figure 1(a) presents an edge-labeled example graph with 10 nodes and 16 edges, in which labels $\{a, b, c, e, h\}$ (Fig. 1(b)) is the optimal solution of the former sub-problem, from which a minimum label spanning tree can be acquired with the existing algorithm for MST. Figure 2(a) demonstrates the label graph of the example edge-labeled graph, in which label communities are drawn with different shapes: the left one contains labels $\{a, b, c, e\}$ and the right one includes labels $\{d, f, g, h\}$; the denoted edges of the left and the right label community are illustrated in Figure 2(b) and Figure 2(c). The left label community has the preferable index of 6 (the label community has 4 members, their denoted edges constitute 2 connected components); the denoted edges of the right label community have the preferable index of 10 (the label community has 4 members, and their denoted edges constitute 6 components). We choose the left label community to form an initial solution for the former sub-problem, because its denoted edges don't contain any spanning tree of the original graph, label {h} is additionally added to form a feasible solution. As none of the left label community can be removed, hence labels {a,b,c,e}∪{h} form a optimum solution for the former sub-problem of the example edge-labeled graph.

4. The Proposed Algorithm

In this section, we introduce the proposed algorithm for the former sub-problem of the SGMLSTP. This algorithm is based on the label communities and the zigzag piloting strategy, it mainly consists of three phases: (1) derive the label graph; (2) partition the label graph into communities and choose label communities (or combination of communities) with smaller preferable indexes as initial solutions for the former sub-problem, (3) adopt the zigzag piloting strategy to append additional labels and remove redundant labels. Algorithm 1,2,3 describe the details of the three phases.

Algorithm 1 label graph extraction

```
Input: G =< $V, E, L$ >
Output: LG =< $L, E_L$ > //label graph
(1)$W[|L|][|L|] \leftarrow 0$;            //label co-occurrence
(2)$E_L \leftarrow \emptyset$;              //connection between labels;
(3)For each edge $e_{ij}$   // label co-occurrence calculating
(4)    For two label $l_x, l_y \in l(e_{ij})$:
(5)        $W[l_x][l_y] \leftarrow W[l_x][l_y] + 1/|l(e_{ij})|$;
(6)For i, j=1..|L|                  //label graph creation
(7)        if $W[i][j]>0$;
(8)            $E_L \leftarrow E_L \cup \{(i, j, W[i][j])$
```

Algorithm 2 community detection and initial solution construction

```
Input: LG =< $L, E_L$ >, Comm, k
Output: initSol                          //initial solutions
(1) initSol← Ø
(2) Comm←ICDA($E_L$)                //label graph partition
(3)for each item Comm[i]:           //calculate preferable indexes for each label community
(4)    calculate pr(Comm[i])=$|\mathrm{Comm[i]}| + \varpi(D(\mathrm{Comm[i]}))$
(5)Comm←sort(Comm,pr)
(6) initSol←combination(Comm,p) //initial solutions by combinating label communities
```

Algorithm 3 the zigzag piloting strategy

```
Input: G =< $V, E, L$ >, initSol
Output: Optsol                       //optimum solution
(1) Optsol← Ø
(2)for each item in initSol:         //zigzag piloting process for initial solutions
(3)    if $\varpi(D(\mathrm{Comb[i]}))>1$:
```

```
(4)        for $l_x \in L¥\,Comb[i]$:   // calculate appending indexes for Nonselected labels
(5)            calculate $A_{Comb[i]}(l_x)$
(6)        $L¥\,Comb[i] \leftarrow sort(L¥\,Comb[i], A_{Comb[i]}(l_x))$ //sort Nonselected labels
(7)      $Comb[i] \leftarrow Comb[i] \cup first(L¥\,Comb[i])$   //append additional labels
(8)  if $\varpi(D(Comb[i])) = 1$:
(9)      for $l_x \in Comb[i]$:
(10)             if $\varpi(D(Comb[i] \cup l_x)) = 1$:
(11)                 $Comb[i] \leftarrow Comb[i] - l_x$              //remove redundant labels
(12)Optsol$\leftarrow \min_i |Comb[i]|$     //the optimum solution is the Comb item with the least labels
```

We assume the total number of labels is $|L|$ and the average label number for edges is L_a, the number of label communities is $|C|$. In algorithm 1, Steps (3)-(5) search all labeled edges, calculate label co-occurrence and search related edges for labels, the computation cost is $O(m * L_a^2)$; Steps (6)-(8) create the label graph according to labels co-occurrence, the complexity is $O(|L|^2)$, Hence algorithm 1 has the complexity as $O(mL_a^2 + |L|^2)$.

In algorithm 2, Step (2) calls ICDA algorithm to partition the label graph and acquire label communities with the cost of $O(k * |L|)$ (k is a constant related to the average degree of the label graph), step (4) calculates the preferable index for a label community, in which the cost to search the denoted edges is $O(L_a * m)$ and the cost to calculate the components of the denoted edges is $O(m + n)$, the complexity to calculate the preferable indexes for all label communities (step (3) and (4)) is $O(|C| * (L_a * m + m + n)) \approx O(|C| * |L| * m)$; step (5) sorts all label communities according to their preferable indexes, the cost is $O(|C|^2)$; the next step chooses p label communities at the forefront and combine them to acquire some initial solutions, the cost is $O(p)$; algorithm 2 has the cost of $O(m|C||L| + |C|^2 + p)$

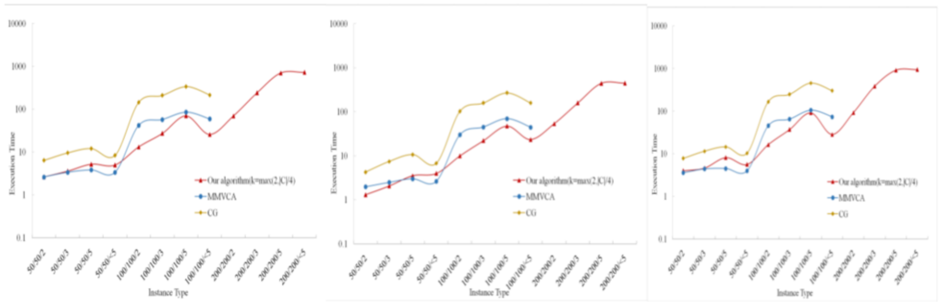
In algorithm 3, steps (2)-(11) apply the zigzag piloting on each initial solution, describe the zigzag piloting process, in which steps (3)-(7) accomplish the expanding sub-process, step (4) and (5) calculate the appending indexes for non-selected labels with the cost of $O(|L| * (L_a * m + m + n))$, step (6) sorts the non-selected labels with cost $O(|L|^2)$, step (7) appends the first non-selected label into the initial solution with the cost $O(1)$, the cost of the expanding sub-process is $O(|L| * (L_a * m + m + n) + |L|^2)$. Steps (8)-(11) execute the reduction sub-process with the cost of $O(|L| * (L_a * m + m + n))$. The summary cost of the zigzag piloting process (from step(2) to step(11)) is $O([|L| * (L_a * m + m + n) + |L|^2 + |L| * (L_a * m + m + n)]) \approx O(m|L|^2)$. The cost of step (12) is $O(p)$. The summary cost of algorithm 3 is $O(pm|L|^2)$.

By combining the cost of three algorithms, the complexity of the proposed algorithm is $O(mL_a^2 + |L|^2 + m|C||L| + |C|^2 + p + |L|^2m) \approx O(|L|^2m)$. When the cost to solve the later sub-problem is also included (the complexity is $O(n\log n)$), the total complexity is $O(|L|^2m + n\log n)$.

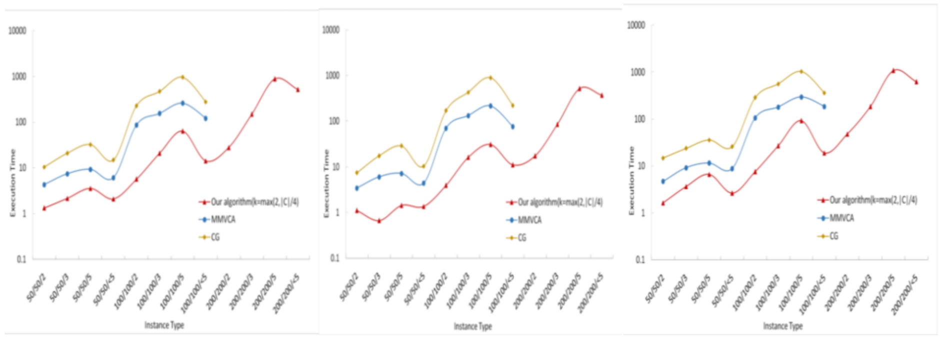
5. Experimental Results

In this section, we choose two state-of-the-art algorithms, carousel greedy [17] and MMVCA [18] to compare with the proposed algorithm. All the algorithms are implemented on the Python platform and experiments are executed on a Windows10 desktop with Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz and 16G RAM.

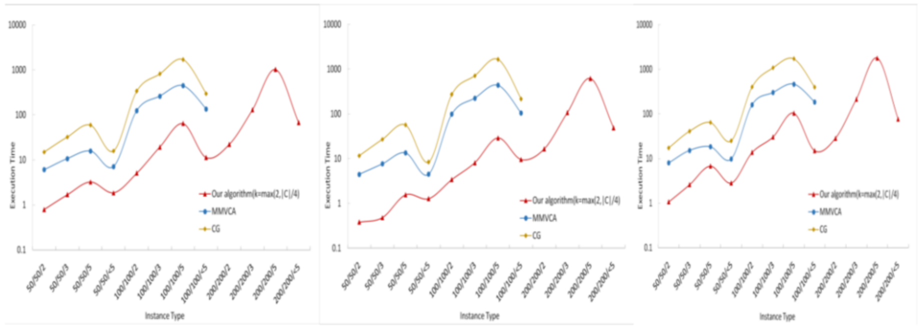
The benchmark instances are generated as Cerrone et al. [18] described. The instances are randomly generated labeled graphs with the node number |V| as 50, 100, or 200; the label number |L| equals to the node number and the edge number |E| is equals to $h * \mathrm{n}(\mathrm{n}-1)/2$, where h is a density measure equals 0.2, 0.5, or 0.8; the label number for each edge is set as 2, 3, 5, or a random integer less than 5. For each combination of parameters, ten different instances will be generated and the total number of labeled graphs is 360. We take the best, the worst and the average execution time and the minimum labels of ten instances of each combination of parameters for the algorithms' comparison, to avoid endless execution, the execution time limit is set as 7,200 seconds (2 hours). Figure 3 and 4 illustrate the experimental results of the compared algorithms.



(a) Average/Best/Worst execution time on different instances (density=0.2)



(b) Average/Best/Worst on different instances (density=0.5)

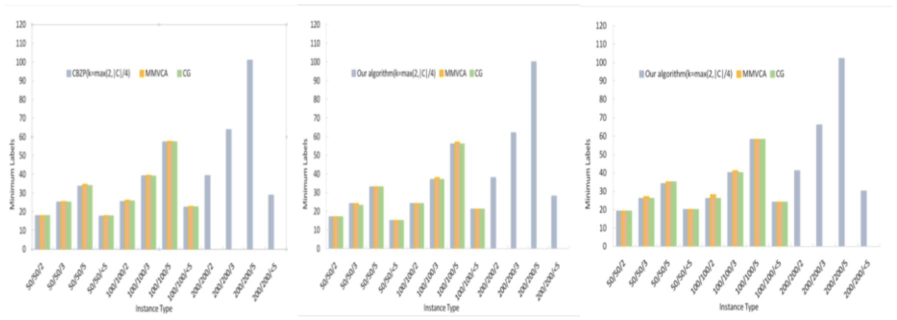


(c) Average/Best/Worst on different instances (density=0.8)

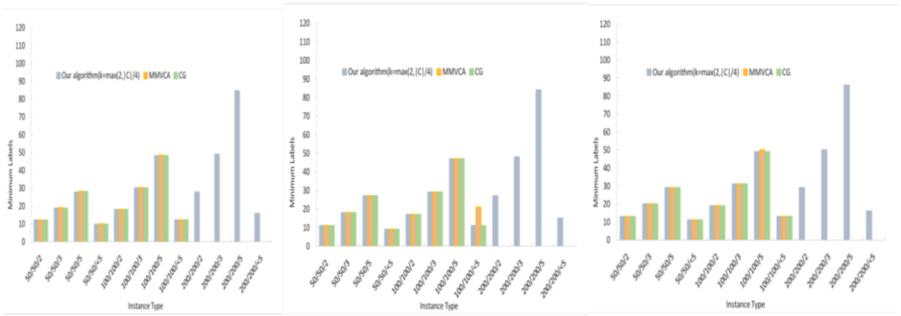
Figure 3 the execution time of the compared algorithms on different instance

Figure 3(a) show the average, the best and the worst execution time on 10 instances when the density is set as 0.2; figure 3(b) show the average, the best and the worst execution time on 10 instances when the density is set as 0.5; figure 3(c) show the average, the best and the worst execution time on 10 instances when the density is set as 0.8. In almost every case, our proposed algorithm requires the least execution time, when the graph size is expanded as 200 nodes and 200 labels, CG and MMVCA can't accomplish searching of MLST in the execution limit.

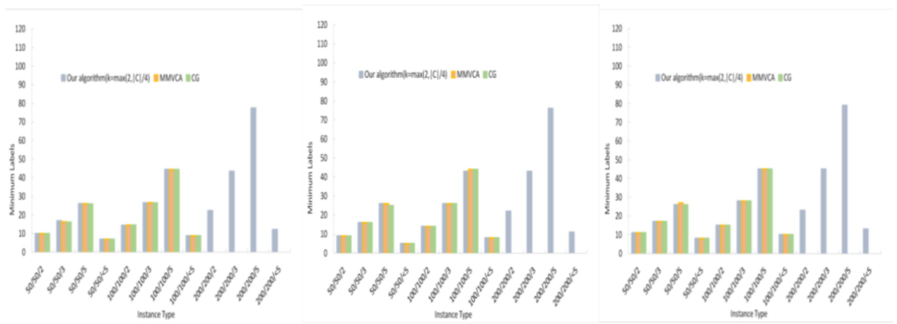
Figure 4 show the average, the best and the worst minimum labels on 10 instances when the density is set as 0.2, 0.5, 0.8, respectively. We can see in different situations, our proposed algorithm can acquire MLST solutions not worse off than that of CG and MMVCA, conversely, on a small part of instances; our algorithm can acquire better solutions.



(a) Average/Best/Worst minimum label numbers on different instances (density=0.2)



(b) Average/Best/Worst minimum label numbers on different instances(density=0.5)



(c) Average/Best/Worst minimum label numbers on different instances (density=0.8)

Figure 4 the minimum labels of the compared algorithms on different instance

6. Conclusions

In this paper, we decompose SGMLSTP into two sub-problem and focus on the former one, we develop a community-based zigzag piloting heuristics to solve it: labels that always appear together have close associations, we transform the labels and their relationships into a label graph and partition it into disjoint communities, combination of label communities are chosen as an initial solution for the former sub-problem, and the zigzag piloting strategy is adopted to refine the initial solution. We compare the heuristics with the state-of-the-art algorithms on some benchmark datasets. Experimental results show our heuristics outperforms current algorithms.

Acknowledgements

This work is funded by National Natural Science Foundation of China (61862034), the Natural Science Foundation of Jiangxi Province (20202BABL202013).

Reference

[1]Granata D,Cerulli R,Scutellà MG, Raiconi A, et al. Maximum flow problems and an np-complete variant on edge-labeled graphs. In Hand book of combinatorial optimization. Springer, New York, 2013.pp 1913–1948.
[2]Chang R S, Leu S J. The minimum labeling spanning trees. Inf. Process. Lett., 1997, 63:277–282.
[3]Chen Y, Cornick N, Hall AO, Shajpal R, Silberholz J, Yahav I, Golden BL. Comparison of heuristics for solving the gmlst problem. In Telecommunications modeling, policy, and technology. Springer, Boston, 2008. pp 191–217.
[4]Xiong Y, Golden B L, Wasil E A. (2005b). A one-parameter genetic algorithm for the minimum labeling spanning tree problem. IEEE Trans. Evolutionary Computation, 2005, 9:55–60.
[5]Cerrone C, Cerulli R, Gaudioso M. OMEGA one multi ethnic genetic approach. Optimization Letters, 2015, 10(2):1-16.
[6]Zahra, Naji-Azimi, et al. Variable neighborhood search for the cost constrained minimum label spanning tree and label constrained minimum spanning tree problems. Computers & Operations Research, 2010, 37(11):1952-1964.
[7]Consoli S , Darby-Dowman K, Mladenovi N, et al. Greedy randomized adaptive search and variable neighborhood search for the minimum labelling spanning tree problem. Eur J Oper Res. 2009, 196(2):440–449.
[8]Consoli S, Mladenovic N, JA Moreno Pérez. Solving the minimum labelling spanning tree problem by intelligent optimization. Appl. Soft Comput., 2015,28,440–452.
[9]Silva T , Gueye S , Michelon P , et al. A polyhedral approach to the generalized minimum labeling spanning tree problem. EURO Journal on Computational Optimization, 2019, 7(1):44-77.
[10]Silva T , Queiroga E , Ochi L S , et al. A hybrid meta-heuristic for the minimum labeling spanning tree problem. European Journal of Operational Research, 2019, 274(1):22-34.
[11]Cerulli R , Fink A , Gentili M , et al. Meta-heuristics comparison for the minimum labelling spanning tree problem. In The Next Wave in Computing, Optimization, and Decision Technologies. 2005. pp.93–106).
[12]Chwatal, A M, Raidl, G. R.. Solving the minimum label spanning treeproblem by ant colony optimization. In GEM, 2010. pp 91–97. CSREA Press.
[13]Lin MG, Liu, FJ ,Zhao HH ,Chen JZ . A Novel Binary Firefly Algorithm for the Minimum Labeling Spanning Tree Problem. CMES-Computer Modeling in Engineering & Sciences. 2020, 125(1):197-214.
[14]Lai XS ,Zhou YR, He J ,Zhang J. Performance Analysis of Evolutionary Algorithms for the Minimum Label Spanning Tree Problem. IEEE Transactions on Evolutionary Computation. 2014,18(6): 860-872.
[15]Vaisman R. Finding minimum label spanning trees using cross-entropy method. NETWORKS, 2021, doi:10.1002/net.22057
[16]Chwatal A M, Raidl G R. Solving the Minimum Label Spanning Tree Problem by Mathematical Programming Techniques. Advances in Operations Research, 2011, 2011:1-38.

[17]Cerrone C, Cerulli R ,Golden B. Carousel greedy: A generalized greedy algorithm with applications in optimization. Computers & Operations Research, 2017,85:97-112.

[18]Cerrone C , D'Ambrosio C , Raiconi A . Heuristics for the strong generalized minimum label spanning tree problem. Networks, 2019.74(2):

[19]Hao L. Overlapping community detection with least replicas in complex networks. Information Sciences, 2018:S0020025518302524.

[20]Long H, Liu X W. A UNIFIED COMMUNITY DETECTION ALGORITHM IN LARGE-SCALE COMPLEX NETWORKS. Advances in Complex Systems, 2019, 22(03):7821-7826.

[21]Moscato V, G Sperlì. A survey about community detection over On-line Social and Heterogeneous Information Networks. Knowledge-Based Systems, 2021, 224(1):107112.