IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 25, 2019, accepted March 19, 2019, date of publication April 3, 2019, date of current version April 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2908011

A Multi-Stage Metaheuristic Algorithm for Shortest Simple Path Problem With Must-Pass Nodes

ZHOUXING SU, JUNCHEN ZHANG, AND ZHIPENG LÜ
SMART, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
Corresponding author: Zhipeng Lü (zhipeng.lv@hust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61370183 and Grant 71320107001.

ABSTRACT The shortest simple path problem with must-pass nodes (SSPP-MPN) aims to find a minimum-cost simple path in a directed graph, where some specified nodes must be visited. We call these specified nodes as must-pass nodes. The SSPP-MPN has been proven to be NP-hard when the number of specified nodes is more than one, and it is at least as difficult as the traveling salesmen problem (TSP), a well-known NP-hard problem. In this paper, we propose a multi-stage metaheuristic algorithm based on multiple strategies such as k-opt move, candidate path search, conflicting nodes promotion, and connectivity relaxation for solving the SSPP-MPN. The main idea of the proposed algorithm is to transform the problem into classical TSP by relaxing the simple path constraint and try to repair the obtained solutions in order to meet the demands of the original problem. The computational results tested on three sets of totally 863 instances and comparisons with reference algorithms show the efficacy of the proposed algorithm in terms of both solution quality and computational efficiency.

INDEX TERMS Constrained shortest path, must-pass nodes, multi-stage metaheuristic algorithm, routing problem, traveling salesman problem.

I. INTRODUCTION

The shortest simple path problem with must-pass nodes arises in many industrial applications. For example, many services provided by communication companies and Internet service providers are related to routing problems in the optical network. There are some nodes with special properties, such as relays, which need to be accessed in a lightpath in order to meet the demand for resource, time, capacity, and other constraints. These requirements can be empirically transformed to the must-pass constraint. In road or railway networks, issues such as supply chain management, sightseeing planning, and shopping routing [1] etc., can also be reduced to the shortest simple path problem with must-pass nodes (SSPP-MPN). Similar to the classical routing problems, each edge between a pair of nodes is associated with a weight. Meanwhile, there are some nodes that must be included in the path, which distinguishes the SSPP-MPN from the classical shortest path problem. Thus, the objective of the SSPP-MPN is to find a minimum cost simple path that satisfies the must-pass constraint.

The associate editor coordinating the review of this manuscript and approving it for publication was Huaqing Li.

Due to the significance of SSPP-MPN in terms of both theory and practice, many researchers have studied this problem in the literature during the last decades. SSPP-MPN was first introduced by Saksena and Kumar [2] and the authors designed an algorithm based on optimality principle. However, Dreyfus [3] pointed out that Saksena and Kumar's algorithm is incorrect because some preconditions are proven to be wrong. Dreyfus indicated that if the simple path constraint is neglected (that is to say, there may exist loops), the SSPP-MPN can be converted into TSP. Specifically, the shortest distance between each pair of must-pass nodes (including the source and destination nodes) in the SSPP-MPN can be considered as the distance of each pair of nodes in the TSP. Then, the algorithms such as [4]–[6] for solving the well-studied TSP can be directly used to solve the SSPP-MPN. Ibaraki proposed a dynamic programming algorithm and a branch-and-bound algorithm to solve the SSPP-MPN [7], where it was pointed out that branch-and-bound algorithm is more effective than dynamic programming. Vardhan et al. [8] proposed a heuristic algorithm to solve the SSPP-MPN. They divided the search process into two stages. The first one is to determine all the candidate paths between two adjacent must-pass nodes by a network

flow algorithm and the order of must-pass nodes is given by random arrangement or applying a depth-first search algorithm from source to destination. The second stage uses backtracking algorithm to select paths which connect adjacent must-pass node pairs and satisfy the simple path constraint. Gomes *et al.* [9] proposed a heuristic algorithm for finding the shortest path with must-pass nodes with a protection path. The algorithm finds two node-disjoint paths from source to destination simultaneously, and the first path must satisfy the requirements of must-pass nodes. Their algorithm improves the Saksena and Kumar's algorithm, using the k-shortest path algorithm to search for multiple candidate paths between must-pass nodes, but the algorithm does not guarantee that a feasible solution can always be obtained. Martins and Gomes *et al.* [10]–[12] proposed two heuristics for the SSPP-MPN which are Path with Specified Nodes (PSN) and Path with Specified Nodes using Trap Avoidance (PSNTA). The authors pointed out that PSN is more effective than PSNTA for large scale instances.

Andrade [13], [14] proposed several mathematical formulations for the SSPP-MPN. First, he proposed a $Q2$ model based on spanning tree polygons, followed by an analysis of the compact model $Q3$ based on the duality principle. Another commodity flow based formulation called $Q4$ is also presented, and it is able to produce better lower bound than $Q2$ and $Q3$. The author tests these formulations on a set of randomly generated instances and instances transformed from TSPLIB. Experiments demonstrate that $Q3$ outperforms $Q2$ in most cases.

In this paper, we propose a multi-stage metaheuristic (MSM) algorithm based on a classical TSP algorithm for solving the SSPP-MPN. The proposed MSM algorithm integrates several distinguishing features, such as using the LKH algorithm with k-opt moves to determine the order of must-pass nodes, a method to generate candidate paths in order to minimize the number of conflicting nodes, a conflicting nodes promotion strategy to enforce repeatedly visited nodes to be virtual must-pass nodes and a connectivity relaxation technique for repairing feasibility. Our MSM algorithm is tested on three sets of totally 863 benchmark instances in the literature and shows its efficacy in terms of both solution quality and computational efficiency.

The remainder of the paper is organized as follows. The problem definition and mathematical formulation of the problem are presented in Section II. The proposed multi-stage metaheuristic algorithm is presented in Section III. Experimental results and the analysis are described in Section IV, before concluding the paper in Section V.

II. PROBLEM DESCRIPTION AND MATHEMATICAL FORMULATION

A. PROBLEM DESCRIPTION

The SSPP-MPN can be described as follows: Let $G = (V, E)$ be a directed graph, $V = \{1, 2, \ldots, n\}$ be the set of nodes and $E = \{(i, j)|1 \leq i \leq n, 1 \leq j \leq n\}$ be the set of edges, where each edge is associated with a positive weight D_{ij}. Given a set of must-pass nodes $V_m \subset V$, and the source node s and the destination node t, the problem aims to find a simple path $P = \{(s, v_1), (v_1, v_2), \ldots, (v_k, t)|V' = \cup_{i=1}^{k}\{v_k\}, V_m \subset V' \subseteq V\}$ of minimum cost in graph G from the source node s to the destination node t, where each node in V_m must be visited.

If the set of must-pass nodes contains all the other nodes except for the source and destination nodes, i.e., $|V_m| = n-2$, then all the nodes should be visited, and the problem is equivalent to a TSP by merging the destination node t and the source node s into a single node. As a classical NP-hard problem, TSP aims to find a minimum cost Hamiltonian circuit [15]. On the other hand, if the set of must-pass nodes is empty, namely $|V_m| = 0$, SSPP-MPN becomes the shortest path problem from s to t. However, if the number of must-pass nodes is more than 1, the SSPP-MPN is at least as hard as TSP with $|V_m| + 1$ dimensions.

B. MATHEMATICAL FORMULATION

In this section, we show three integer programming mathematical formulations for the SSPP-MPN. Inspired by the TSP formulation [16], the mathematical formulations of the SSPP-MPN are closely related to that of TSP [14].

Before introducing these models, we define some common constraints for different formulations as below. Let x_{ij} be a boolean decision variable which denotes that edge (i, j) exists in the solution if $x_{ij} = 1$, otherwise it is not visited.

$$\min \sum_{i \in V} \sum_{j \in V, j \neq i} D_{ij} \cdot x_{ij} \tag{1}$$

$$\textit{subject to} \sum_{j \in V-\{i\}} x_{ij} - \sum_{j \in V-\{i\}} x_{ji} = \begin{cases} -1 & i = t \\ 0 & \forall i \subset N - \{s, t\} \\ 1 & i = s \end{cases} \tag{2}$$

$$\sum_{j \in V-\{i\}} x_{ij} = 1 \quad \forall i \in V_m \tag{3}$$

$$\sum_{j \in V-\{i\}} x_{ij} \leq 1 \quad \forall i \in V - V_m \tag{4}$$

This is the basic model for the SSPP-MPN. Objective (1) aims to minimize the total distance of the path. Constraint (2) ensures the path connectivity that the in-degree and out-degree must be consistent at each node except for s and t, i.e., there will always be a pair of incoming edge and outgoing edge for each visited node. Constraint (3) guarantees that each must-pass node should be visited exactly once. Constraint (4) is the simple path constraint which makes sure that each node will never be visited more than once.

However, this basic model is not complete. If we only apply constraints (2)–(4), the solutions may contain some cycles called sub-tours which break the path connectivity. In order to make this basic model complete, constraints for eliminating sub-tours should be added. Here, we present three different ways to conduct the sub-tour elimination.

1) CONVENTIONAL FORMULATION (CF)

We add constraint (5) to the basic model to obtain a complete formulation of the SSPP-MPN. The idea comes from conventional model of TSP in [17]. Constraint (5) indicates that there must be at least one absent edge in each sub-tour, denoted as S, which can be either a cycle or a path that does not include all the must-pass nodes. Note that the number of constraint (5) is exponential.

$$\sum_{(i,j)\in S} x_{ij} \leq |S| - 1 \quad \forall S \subset V, 2 \leq |S| < n \tag{5}$$

2) SEQUENTIAL FORMULATION (SF)

We introduce a continuous decision variable y_i, representing the sequence of visit to node i. Constraint (6) restricts that if edge (i, j) is in the solution, node i should be visited before j, i.e., $y_i < y_j$. If sub-tour exists and edge (i, j) is contained in the cycle, then $y_j < y_i$ should also be true since there is a path from j to i in the cycle, which means that at least one of $y_i < y_j$ and $y_j < y_i$ must be violated. Thus, no sub-tour could appear in the solution without breaking constraint (6).

$$y_i - y_j + nx_{ij} \leq n - 1 \quad \forall i,\ j \in V - \{s\}, i \neq j \tag{6}$$

3) COMMODITY FLOW BASED FORMULATION (CFF)

Intuitively, we can consider the SSPP-MPN as a problem of sending some units flow from source node s to sink node t. By restricting that every node on the path consumes one unit of the flow, there will be no sub-tours if there are non-negative amount of flow along the path, since the source which produces the initial flow is not on the sub-tours. We introduce a continuous decision variable z_{ij}, representing the flow on edge (i, j). Constraint (7) restricts that the flow only goes through the edges in the path, constraint (8) guarantees that only $n - 1$ units flow come out from source s, and constraint (9) requires that every node in the solution except s consumes one unit of flow.

$$z_{ij} \leq (n-1)x_{ij} \quad \forall i,\ j \in V, i \neq j \tag{7}$$

$$\sum_{j\in V-\{s\}} z_{sj} = n - 1 \tag{8}$$

$$\sum_{i\in V-\{j\}} z_{ij} - \sum_{k\in V-\{j\}} z_{jk} = \sum_{i\in V-\{j\}} x_{ij} \quad \forall j \in V - \{s\} \tag{9}$$

Compared with the CF model, the SF and CFF formulations are more compact, because they only have polynomial number of constraints.

III. MULTI-STAGE METAHEURISTIC ALGORITHM

We propose a multi-stage metaheuristic (MSM) algorithm that integrates several strategies based on a mutli-stage search framework, such as a TSP solver to determine the order of must-pass nodes and other sophisticated strategies to eliminate loops. Our algorithm consists of three main stages. The first stage is to determine the order of the must-pass nodes by employing a TSP solver. The second stage is to minimize the total number of conflicting nodes, i.e., the repeatedly visited nodes, as well as the distance of the path between adjacent must-pass nodes. The last stage aims to obtain a valid solution by promoting conflicting nodes to must-pass ones. MSM algorithm terminates at any stage if a valid solution is obtained. Besides, there is a post-processing procedure to search for a feasible solution in case that all of the three above mentioned main stages fail.

Algorithm 1 The Main Framework of the MSM Algorithm

Input: Graph G, must-pass node set V_m, source node s, target node t

Output: Path with least conflicting nodes and minimum cost P^*

1: $P^* \leftarrow \varnothing$
2: **repeat**
3: $G' \leftarrow$ TransformProblem(G, V_m, s, t) // Section III-A
4: $T \leftarrow$ SolveTSP(G') // Section III-B
5: $P \leftarrow$ SearchCandidatePath(G, T) // Section III-C
6: **if** ConflictNodes(P) $= \varnothing$ **then**
7: **return** P
8: **end if**
9: **if** P is better than P^* **then**
10: $P^* \leftarrow P$
11: **end if**
12: $V_m \leftarrow$ PromoteNodes(ConflictNodes(P), V_m) // Section III-D
13: **until** Stop condition is met
14: $P^* \leftarrow$ RelaxConnectivity(G, V_m, s, t, P^*) // Section III-E
15: **return** P^*;

The pseudocode of our MSM algorithm is given in Algorithm 1. First of all, the original graph G is converted to a new graph G' by just considering the must-pass nodes (including s and t) and the shortest paths between each pair of must-pass nodes (including s and t) in G as its edges (line 3). Then, a TSP solver is launched to find a Hamiltonian cycle P with the minimum cost in G' (line 4). Obviously, a simple path in G' may be transformed back to a path with loop in G. If the conflicting number of the path P is equal to zero, then a feasible solution is obtained, where the conflicting number of the path P is equal to the number of nodes visited for multiple times in the original graph G. Otherwise, a candidate path search procedure is used to optimize the conflicting node number of the path P (line 5). If it is unsuccessful, a node promoting procedure is employed to virtually treat the multi-visited nodes as must-pass nodes (line 12). The above procedures are repeated until a feasible solution is obtained or the stop condition is met (lines 2–13). If the best path is still infeasible, a repairing procedure is invoked as double insurance (line 14).

The main challenge of the SSPP-MPN lies in determining the order of the must-pass nodes. After determining the order of these specified nodes, path selection between adjacent

must-pass nodes is employed to obtain a shortest path without repeatedly visited nodes. These two stages interact with each other. Adjusting the order of the must-pass nodes can optimize the path distance. However, the simple path constraint may be violated with the new order. Since this problem is closely related to the TSP, we may use the TSP solver to tackle this subproblem. In [3], Dreyfus pointed out that the SSPP-MPN without simple path constraints can be transformed into TSP, where must-pass nodes are connected via their shortest path between them.

A. PROBLEM TRANSFORMATION AND TOUR RESTORATION

Since our MSM algorithm employs a TSP solver to tackle the subproblem of the SSPP-MPN, in this section we show how to convert SSPP-MPN into TSP and restore the TSP solution to the path in the original graph.

We denote $G' = (V', E')$ as graph of TSP, the node set $V' = V_m \cup \{s, t\}$, the edge set $E' = \{(i, j)|i, j \in V'\}$, and denote D'_{ij} as the weight of edge (i, j) in E'. D'_{ij} is calculated as the shortest path distance between nodes i and j in G where nodes in V' are skipped for simple path constraint. If two nodes in G' are not connected, the weight of the edge between them is infinite.

Algorithm 2 Problem Transformation

Input: Graph G, must-pass node set V_m, source node s, target node t
Output: Transformed graph G'

```
1: for each vertex v in V_m ∪ {s, t} do
2:     cost(v), path(v) ← ShortestPath(G, v, V_m)
3: end for
4: for each vertex v in V_m ∪ {t} do
5:     cost(v, s) ← cost(v, t)
6: end for
7: for node pair (u, v) in G' do
8:     cost'(u, v) ← cost(u, v)
9: end for
10: return G'
```

The pseudocode of the transformation algorithm is given in Algorithm 2. Procedure *ShortestPath* calculates the shortest distance and paths for each node to other nodes in V' by a shortest path algorithm [18]. Note that the shortest path cannot visit other must-pass nodes. Otherwise, the simple path constraint might be violated. Thus, a set of nodes to be excluded is passed to the *ShortestPath* procedure (line 2). Variables *cost* and *path* are two matrices to respectively store the shortest distance and path for each pair of nodes in V'. Moreover, $cost(v)$ and $path(v)$ represent vectors of costs and paths whose source node is v, respectively. Since solution of TSP is a tour, after converting G into a TSP instance G', the starting node s and ending node t in G are merged into a single node (it can also be considered as adding an edge (t, s) in the solution). Therefore, the cost of other nodes to sink t is set to source s (lines 4-6). After that, all of the normal nodes and the corresponding edges are dropped, and only the edges between the must-pass nodes are copied to the transformed graph G' (lines 7–9).

When a tour T on graph G' is obtained by solving TSP, we need to restore the tour to the original path on graph G in order to obtain a feasible solution of the SSPP-MPN. Since the shortest distance and the path of each pair of nodes in V_m on graph G are given in advance, we only need to restore it by querying matrix *path*.

B. TSP OPTIMIZATION PROCEDURE

The TSP optimization procedure consists of two stages. The first one relaxes the simple path constraint and directly solves TSP on graph G'. The second stage aims to minimize the total number of conflicting nodes on the path, as well as optimizing the path cost as the second objective.

1) DETERMINING THE ORDER OF MUST-PASS NODES

In the first stage, the MSM algorithm directly solves the TSP by a classical solver on graph G' without considering the number of conflicting nodes in order to determine the sequence of the must-pass nodes. Since the main challenge of the problem is the order of must-pass nodes, our first objective is to find a good order of the must-pass nodes by relaxing the simple path constraints. Then, the conflicting nodes are eliminated in the second stage.

If there is no conflicting node in the restored solution after the TSP optimization, then a feasible solution is obtained. Otherwise, the second stage for minimizing the number of conflicting nodes is required. In fact, the effectiveness of MSM heavily relies on the performance of the TSP solver. Only if the sequence of must-pass nodes is of high quality, subsequent candidate path search and conflicting nodes promotion strategy are meaningful. Therefore, we employ an efficient TSP solver called LKH proposed in [4] which has been widely used in solving TSP and other related problems.

2) MINIMIZING THE NUMBER OF CONFLICTING NODES

After the order of the must-pass nodes is determined, there may exist conflicting nodes on the path. The reason might be that the order of the must-pass nodes is inappropriate because the simple path constraints are neglected.

In order to obtain a more reasonable sequence of the must-pass nodes, we further perform k-opt moves [19] with an additional condition, where the simple path constraints are not relaxed. As long as there are conflicting nodes on the path, it is an illegal solution. Therefore, there are two objectives in this stage, namely the number of conflicting nodes on the path and path weight. For example, assume that there are two paths T_1 and T_2, the weights of the two paths are w_1 and w_2, and numbers of conflicting nodes on the paths are c_1 and c_2. Since the main objective is to reduce the number of conflicting nodes, if condition $c_1 < c_2 \vee (c_1 = c_2 \wedge w_1 < w_2)$ is satisfied, we say that path T_1 is better than path T_2. It is obvious that the optimal solutions of this transformed problem are

the optima of the original problem. However, the optimality and feasibility are not guaranteed by this procedure. This is resulted from the heuristic nature of the LKH algorithm.

After this procedure, if there is no conflicting node on the path, then MSM algorithm obtains an optimal or near-optimal solution. Otherwise, it indicates that the SSPP-MPN cannot be solved by the LKH algorithm. It is necessary to search for candidate paths and use other advanced strategies to eliminate loops.

C. CANDIDATE PATH SEARCH

The TSP optimization procedure aims to optimize the order of the must-pass nodes as well as the number of conflicting nodes. Since paths between the must-pass nodes are connected via the shortest path in the stage of TSP optimization, valid solution of the SSPP-MPN may not be obtained by only changing the order of the must-pass nodes. Given a sequence of must-pass nodes, there may exist repeatedly visited nodes on the path connected by the shortest path, i.e., the set of conflicting nodes $C \neq \varnothing$. In this case, we need to search candidate paths instead of the shortest one. The candidate path search (CPS) strategy tries to eliminate loops on the path obtained by the TSP optimization procedure, such as the k-shortest path algorithm [20] to reduce the number of conflicting nodes.

In our algorithm, for each pair of must-pass nodes, we consider its shortest path and the shortest path without visiting the conflicting node. The purpose is to obtain a trade-off between solution quality and computational efficiency, for considering more candidate paths will be time-consuming and sacrifice the solution quality. If the candidate path search strategy cannot eliminate the conflicting nodes, a conflicting nodes promotion strategy which will be introduced later in Section III-D will promote these conflicting nodes to must-pass nodes.

Specifically, the set of conflicting nodes is denoted as $C = \{c_1, c_2, \ldots, c_k\}$. For each conflicting node $c_i \in C$, there are usually four (sometimes three) must-pass nodes involved because the conflicting node c_i is the intersection of two pairs of must-pass nodes. The solution of the SSPP-MPN can be represented as $P(c_i) = \{\ldots, m_1, \ldots, c_i, \ldots, m_2, \ldots, m_3, \ldots, c_i, \ldots, m_4, \ldots\}$, where only the conflicting node and involved must-pass nodes are shown. In order to eliminate the conflicting node c_i, the following candidate path search strategy is used:

- Step 1. Segment (m_1, m_2) keeps using the shortest path, while segment (m_3, m_4) is replaced by the shortest path without visiting conflicting node c_i. Note that the new candidate path should not overlap with segments other than these two segments.
- Step 2. Segments (m_1, m_2) and (m_3, m_4) are sequentially replaced by the shortest path without visiting the conflicting node c_i, where simple path constraint must be satisfied.

By swapping (m_1, m_2) and (m_3, m_4), we could get other two steps for candidate path generation. We perform the procedure defined in Algorithm 3 to eliminate the conflicting nodes as well as optimizing the path distance. From all the combinations of the candidate paths, we choose a feasible solution with the lowest cost (lines 4–7). If all the above candidate paths cannot eliminate the conflicting node c_i, we record c_i and will promote it to a must-pass node in the following stage.

Algorithm 3 Candidate Paths Search

Input: Graph G, tour in transformed graph T
Output: Path P

```
1: P ← RestoreTour(G, T)
2: for each vertex v in ConflictNodes(P) do
3:    M ← LocateOneInvolvedSegment(G, v)
4:    search candidate paths for M
5:    if better candidate paths were found then
6:       update paths for segments M
7:    end if
8: end for
9: return P
```

D. CONFLICTING NODES PROMOTION

It is possible that a valid solution can still not be obtained after the previous TSP procedure and candidate path search strategies. In this case, some conflicting nodes are repeatedly visited, implying that it is very likely that these conflicting nodes should exist in the optimal solution. Based on this assumption, we propose a conflicting nodes promotion (CNP) strategy. Specifically, we promote these conflicting nodes to virtually become must-pass nodes, which enforces each of them to be visited just once. This procedure is equivalent to adding the conflicting nodes to the set of must-pass nodes in the original graph, then the previous two procedures are relaunched.

The conflicting nodes promotion strategy is mainly used to increase the chance for our MSM algorithm to obtain a feasible solution. Especially, there could be significant improvement on dense graphs. In the extreme case, a feasible solution can be always obtained if all the nodes on a complete graph are promoted to must-pass nodes because the problem is equivalent to a TSP problem. If a conflicting node is incorrectly promoted, a sub-optimal path may be obtained. However, the aforementioned TSP optimization and candidate path search procedures attempt to minimize the number of conflicting nodes as far as possible. As a result, the conflicting nodes promotion strategy can promote very limited number of nodes to must-pass nodes to improve the search effectiveness.

E. CONNECTIVITY RELAXATION

If each normal node is promoted or there exists a pair of must-pass nodes that no path can be found between them, it can be inferred that the CPS and CNP procedures will have to stop even if there is no feasible solution found. In order

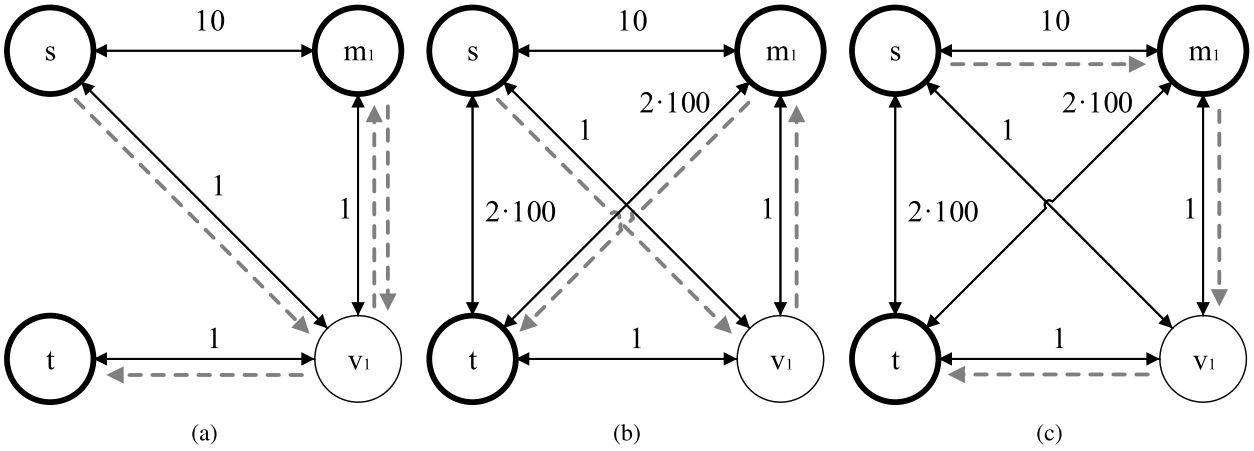

FIGURE 1. Graph transformation and neighborhood move example. (a) The infeasible path with a conflicting node; (b) the transformed complete graph and the mapped path; (c) the resulting path of a swap neighborhood move.

to improve the robustness of the algorithm, an additional landscape smoothing technique is applied after each preceding procedures failed. This landscape smoothing technique makes it possible to pass through insurmountable barrier (hard constraints) in the solution space. It is achieved by adding a virtual edge between each pair of nodes which are not connected with edges directly in the original topology. The penalties for passing through the virtual edges are very large and they correlate to the distance between the node pairs. In detail, the length of a virtual edge is equal to the length of the shortest path between its end points multiplying the upper bound of the optimal path length. After connecting the node pairs, the original graph is transformed into a new complete graph. Then, we map the path with conflicting nodes on the original graph to a feasible path with huge penalty on the new graph. This is done by bypassing the repeatedly visited nodes through the newly added virtual edges. However, it just changes the type of infeasibility instead of fixing one so far. So a randomized local search procedure utilizing the insertion, deletion, ejection, move, swap, and mirror neighborhood structures proposed in [21] is executed to search for a better path. Note that the deletion neighborhood cannot remove the original must-pass nodes from the current path. In fact, the initial solution of the randomized local search can be any path found in preceding procedures, but the MSM starts the search from the path with least conflicting nodes.

Fig. 1 illustrates the graph transformation and a simple example of a neighborhood move. Assume that the path represented with dashed arrow in Fig. 1(a) is the path with least conflicting nodes found before the failure of CPS and CNP technique. The transformation is demonstrated in Fig. 1(b). On one hand, virtual edges $s \leftrightarrow t$ and $m_1 \leftrightarrow t$ are added and their weights are set according to the lengths of the shortest paths between the end points. On the other hand, the original path $s \rightarrow v_1 \rightarrow m_1 \rightarrow v_1 \rightarrow t$ is mapped to $s \rightarrow v_1 \rightarrow m_1 \rightarrow t$ where the second visit to v_1 is bypassed and it goes from m_1 to t directly. Fig. 1(c) shows a possible neighborhood move that repairs the solution by swapping the sequence of visit to m_1 and v_1, which transforms the path into $s \rightarrow m_1 \rightarrow v_1 \rightarrow t$, thereby eliminating the virtual edges and making it a feasible path.

In sum, the key components of the proposed MSM algorithm are illustrated in Fig. 2. First, it relaxes the simple path constraint and transforms the SSPP-MPN into TSP and the transformed problem is solved by utilizing an effective algorithm for TSP. Then, the tour for the original problem is restored and candidate paths will be evaluated if conflicting nodes exist. Next, if there are still conflicting nodes, the MSM algorithm will promote them as must-pass nodes and restart from the problem transformation. Finally, an additional connectivity relaxation procedure is applied in case that all the aforementioned strategies fail.

IV. COMPUTATIONAL RESULTS AND ANALYSIS

A. BENCHMARK INSTANCES AND EXPERIMENTAL PROTOCOL

The benchmark instances in our test consist of three sets. The first set consists of 240 instances generated from 12 networks by randomly selecting must-pass nodes as done in [12]. The second set contains 367 instances provided by Andrade [14]. The last set contains 256 large instances generated by ourselves. For the 12 networks, five of them are obtained from SNDLib [22], other five ones were generated with the Doar-Leslie model [9], [23] and the rest two networks were CORONET CONUS and TeliaSonera [12], respectively. The number of must-pass nodes of the 240 instances is the same as in [9], [12]. For Andrade's datasets, there are 340 random instances whose number of nodes are 20, 40, 80, 200, and 300 (denoted by A20, A40, A80, A200, A300, respectively), and there are 27 instances from TSPLIB [24] (denoted by Aatsp). For our own dataset, the number of nodes are generally larger than the aforementioned datasets. They can be further divided into three categories, which are crafted sparse graph, random dense graph with few must-pass nodes and random

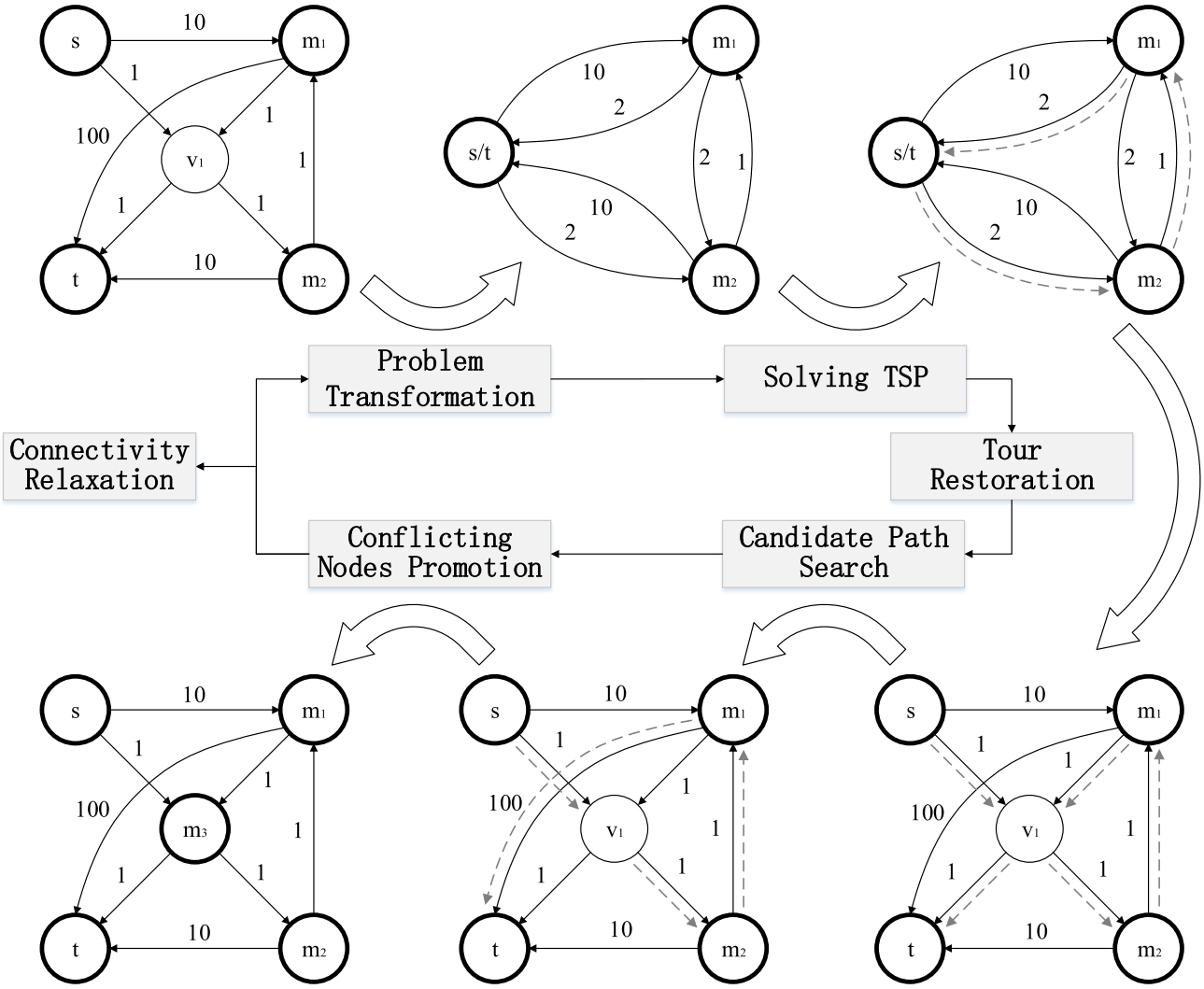

FIGURE 2. The key components of the proposed MSM algorithm.

dense graph with many must-pass nodes, respectively. In the crafted category, there are 16 instances where the topology or the distribution of the must-pass nodes are elaborately designed. There are some *clique* instances which are composed of a lot of small complete sub-graphs but are sparse graphs in general, some *grid* instances which are generated by adding some random edges into grid frameworks, some *hop* instances whose weights on edges are equal and some *detour* instances where the must-pass nodes are far from the shortest path between the source and the destination. In the random categories, there are 150 instances with 50 to 150 must-pass nodes and 90 instances with 50 to 1200 must-pass nodes. The scales of the self-generated instances are explicit in their names. The number following the character *v* gives the vertex number, the one coming after the character *e* presents the edge number, and the number following the character *d* means the number of the must-pass nodes. The proposed MSM algorithm is programmed in C++ and tested on Windows Server 2012, with Intel Xeon E5-2609 2.5 GHz CPU and 32 GB memory. The MIP models are solved by Gurobi 7.5 [25].

B. COMPUTATIONAL RESULTS

To assess the performance of our MSM algorithm, we perform five independent runs on each instance and record the average results. The results are compared with the PSN algorithm proposed by Martins and Gomes *et al.* [12] and the *Q*4 model proposed by Andrade [14]. Tables 1 and 2 show the average results over all the instances for each dataset. The first row gives the number of must-pass nodes in both tables.

In Table 1, Column PSN Gap presents the average optimality gap obtained by Martins's algorithm in percentage. Column MSM Gap reports the average gap to the optimality obtained by MSM algorithm in percentage. Column MSM Time gives the average running time in seconds taken by the

TABLE 1. Computational results on the 240 instances in [12].

Dataset	$\|V_m\| = 2$			$\|V_m\| = 4$			$\|V_m\| = 8$			$\|V_m\| = 10$			$\|V_m\| = 20$		
	PSN Gap	MSM Gap	MSM CPU	PSN Gap	MSM Gap	MSM CPU	PSN Gap	MSM Gap	MSM CPU	PSN Gap	MSM Gap	MSM CPU	PSN Gap	MSM Gap	MSM CPU
newyork	0.11	0	0.03	1.95	0	0.03	-	-	-	-	-	-	-	-	-
norway	0.09	0	0.03	2.27	0.80	0.04	-	-	-	-	-	-	-	-	-
india35	0.24	0	0.03	1.80	1.20	0.03	3.59	2.70	0.03	3.80	0.95	0.03	-	-	-
pioro40	0.24	0	0.04	2.63	0	0.04	3.61	3.50	0.04	4.88	0	0.03	-	-	-
germany50	0.58	3.10	0.04	3.46	1.30	0.03	5.87	1.03	0.04	6.13	3.40	0.04	-	-	-
N500G0	0.26	4.20	0.06	2.46	8.90	0.06	5.60	4.30	0.06	6.48	0.20	0.09	9.82	2.40	0.13
N500G1	0.32	5.60	0.05	1.89	1.20	0.06	5.09	4.50	0.06	6.11	0.98	0.06	9.29	0.46	0.15
N500G2	0.30	4.30	0.05	2.55	2.10	0.06	5.55	3.00	0.06	6.39	5.60	0.07	9.04	4.10	0.13
N500G3	0.28	1.50	0.05	2.51	2.50	0.05	5.07	4.10	0.07	5.92	0.73	0.08	9.78	1.10	0.09
N500G4	0.21	2.00	0.06	2.12	2.50	0.06	4.33	1.60	0.06	4.94	3.90	0.08	9.32	7.20	0.16
CORONET	0.62	2.30	0.05	1.93	4.50	0.06	2.28	9.00	0.04	2.72	1.40	0.05	-	-	-
TeliaSonera	0.21	0	0.03	0.27	0	0.06	0.08	0	0.03	-	-	-	-	-	-

TABLE 2. Computational results on the 367 instances in [14].

Dataset	$\|V_m\| = 0.25\|V\|$			$\|V_m\| = 0.5\|V\|$			$\|V_m\| = 0.75\|V\|$			$\|V_m\| = \|V\| - 2$		
	$Q4$ CPU	MSM CPU	MSM Gap	$Q4$ CPU	MSM CPU	MSM Gap	$Q4$ CPU	MSM CPU	MSM Gap	$Q4$ CPU	MSM CPU	MSM Gap
A20	0.08	0.04	1.8	0.09	0.04	0.19	0.11	0.04	0.06	0.21	0.05	0.07
A40	0.32	0.05	0.4	0.46	0.05	0.30	0.49	0.06	0.12	0.57	0.06	0.00
A80	1.33	0.06	0.4	2.18	0.10	0.08	3.04	0.15	0.02	3.42	0.12	0.00
A200	38.69	0.33	0.7	36.09	1.52	2.70	39.27	3.86	0.42	68.29	0.58	0.00
A300	-	-	-	-	-	-	-	-	-	594.20	0.31	0.00
Aatsp	-	-	-	-	-	-	-	-	-	50.45	0.29	0.01

TABLE 3. Computational results on the 16 self-generated instances (crafted sparse graphs).

Instances	Optima	CF CPU	SF CPU	FF CPU	$Q3$ CPU	$Q4$ CPU	MSM CPU	MSM Best	MSM Avg.
c.v500e4396d50	152	-	-	10.11	-	9.08	0.17	152	152
c.v500e5990d199	4669	1.65	4.40	8.50	1.61	8.82	20.80	4669	4669
c.v1000e5995d199	9967	5.17	17.74	29.11	8.03	31.09	71.45	9967	9976
c.v1000e18281d200	874	57.53	128.80	236.25	2112.59	361.77	82.73	874	890
c.v1000e18293d100	449	103.02	238.25	358.16	1319.63	362.90	1.62	450	450
c.v2000e37052d400	1777	-	-	-	-	-	3561.00	1777	1777
c.v200e990d20	597	1.29	1.28	2.63	1.22	0.96	0.06	597	597
c.v200e990d40	899	1.64	2.29	1.64	2.23	1.38	0.08	899	899
c.v300e995d40	1305	1.22	2.42	4.04	1.04	4.51	0.27	1305	1305
c.v500e2986d100	1959	2.10	6.66	11.96	7.52	13.51	2.67	1968	1968
c.v1000e3995d100	2209	7.04	18.59	39.37	13.23	30.82	0.31	2209	2209
c.v2000e39832d100	745	32.02	75.90	330.91	75.01	190.90	2.23	745	745
c.v300e844d20	375	0.40	1.56	2.07	0.43	1.81	0.09	375	375
c.v500e2000d22	447	1.24	3.03	6.66	1.25	6.08	0.11	447	447
c.v2000e27632d85	2640	31.29	81.08	162.82	61.71	144.72	2.42	2640	2640
c.v2000e39832d100	2370	37.20	100.01	520.42	544.10	359.71	4.98	2370	2380

proposed algorithm to converge. As we can see from Table 1, our algorithm is able to find the optimal solutions easily on the instances with few must-pass nodes. Although the MSM algorithm reaches sub-optimal solutions sometimes, the gap between the objective of the sub-optimal solution and the optimal objective is smaller than that of the PSN algorithm for 37 out of 48 datasets. Especially, when the number of must-pass nodes increases, the MSM has smaller gaps than PSN. Besides, the proposed MSM algorithm is able to produce feasible solution on all the instances, while Martins's algorithm failed to solve some instances in the germany50 and CORONET datasets [12]. These experiments demonstrate that our algorithm is competitive compared with the PSN algorithm.

In Table 2, Column $Q4$ Time shows the average running time in seconds for the $Q4$ model to reach the optimal solution proposed by Andrade on each dataset. Column MSM Time presents the average running time of our algorithm in seconds. Column MSM Gap reports the average optimality gap of our MSM algorithm in percentage. Since the MIP solvers always find the optimal solutions, i.e., the gap is 0, the optimality gaps of the $Q4$ model are not reported in Table 2. Table 2 discloses that, the MSM can reach the optimal solutions in most cases, and the time consumption is relatively low compared with the $Q4$ model, especially on instances with large number of must-pass nodes.

In addition to the classic instances which have been tested by other researchers, we have conducted experiments on our own instances and the computational results are listed in Tables 3 to 5.

In detail, Table 3 reports a detailed computational results obtained by the four different MIP models and the MSM

TABLE 4. Computational results on the 150 self-generated instances (random dense graphs with few must-pass nodes).

	Obj.		CPU			Obj.		CPU			Obj.		CPU	
Instance	Q4	MSM	Q4	MSM	Instance	Q4	MSM	Q4	MSM	Instance	Q4	MSM	Q4	MSM
v1000e519000d118	132	132	2682	2.06	v1351e736295d90	116	116	420	3.65	v1682e622340d89	127	127	1229	3.05
v1006e682068d101	106	106	379	2.56	v1353e989043d141	-	147	3600	4.71	v1684e2509160d116	-	119	3600	13.06
v1009e941397d128	-	130	3600	3.56	v1356e1768224d134	-	136	3600	8.12	v1691e581704d54	93	93	1189	2.91
v1012e486772d50	73	73	124	1.67	v1364e261888d95	325	157	3600	1.51	v1692e993204d91	117	117	2667	4.27
v1025e496100d78	102	102	249	1.91	v1371e1207851d149	-	155	3600	6.13	v1698e2008734d110	-	115	3600	8.83
v1041e716208d84	97	97	2351	2.85	v1378e366548d99	1952	152	3600	1.96	v1700e1013200d96	129	129	1857	4.55
v1056e671616d149	-	154	3600	3.20	v1385e623250d81	118	118	3009	3.10	v1702e2765750d81	1273	84	3600	10.53
v1057e835030d77	86	86	2031	3.11	v1388e875828d77	97	97	3232	3.98	v1705e1962455d66	80	80	3112	7.48
v1059e1103478d112	-	113	3600	4.48	v1394e588268d106	136	136	3511	3.09	v1706e2651124d123	-	126	3600	11.25
v1076e1084608d106	-	108	3600	4.68	v1398e1076460d99	114	114	1252	4.83	v1714e1083248d124	-	145	3600	5.20
v1082e925110d146	-	148	3600	4.17	v1402e536966d136	172	172	1656	2.93	v1720e1217760d102	-	127	3600	5.36
v1089e760122d69	89	89	2769	2.91	v1415e1705075d128	316	131	3600	7.95	v1727e1290069d50	73	73	2982	4.75
v1094e286628d119	157	157	455	1.92	v1419e865590d112	-	132	3600	4.09	v1739e1396417d96	298	111	3600	6.22
v1097e310451d52	83	83	381	1.30	v1420e218680d52	105	105	89	1.26	v1747e1484950d76	95	95	1316	6.30
v1102e469452d71	97	97	1599	2.02	v1421e1071434d127	-	132	3600	4.97	v1752e1575048d123	-	132	3600	7.49
v1106e873740d68	84	84	1162	3.38	v1436e1970192d64	68	68	1581	8.21	v1771e464002d68	115	115	406	2.38
v1107e1033938d56	62	62	901	3.65	v1445e1458005d87	1483	99	3600	6.34	v1772e1461900d134	-	145	3600	8.11
v1113e609924d143	150	150	1532	2.78	v1460e677440d91	120	120	3348	3.16	v1773e1698534d90	1632	103	3600	7.69
v1115e564190d119	-	135	3600	2.88	v1476e1564560d92	1702	102	3600	6.31	v1775e2909225d98	1599	100	3600	11.62
v1120e694400d52	71	71	246	2.70	v1483e2077683d116	273	119	3600	8.60	v1785e2732835d114	-	115	3600	11.54
v1127e645771d110	120	120	1599	2.95	v1484e1638336d129	-	132	3600	7.35	v1792e1320704d53	951	78	3600	5.06
v1141e791854d143	-	148	3600	3.68	v1488e1092192d134	-	145	3600	5.20	v1799e3121265d61	170	66	3600	11.51
v1145e1111795d72	77	77	1201	6.53	v1494e182268d90	183	183	319	1.06	v1801e3128337d143	-	144	3600	13.95
v1155e684915d124	132	132	1065	3.05	v1497e1925142d101	1720	105	3600	8.45	v1805e1783340d114	-	132	3600	8.73
v1159e1089460d112	114	114	684	4.38	v1503e1576647d57	200	73	3600	6.19	v1838e1474076d89	1623	111	3600	6.33
v1168e1261440d122	-	124	3600	5.38	v1513e842741d131	-	155	3600	4.06	v1843e2545183d62	1117	75	3600	9.47
v1170e1334970d73	78	78	731	5.30	v1524e944880d92	1568	114	3600	4.93	v1851e199908d100	214	214	484	1.24
v1171e789254d84	95	95	1946	3.09	v1528e191000d68	137	137	370	1.04	v1857e1047348d124	337	153	3600	5.43
v1177e1293523d104	-	108	3600	4.98	v1534e1093742d107	127	127	2170	4.81	v1857e1704726d82	286	100	3600	6.94
v1182e554358d140	152	152	1383	2.66	v1544e1926912d106	-	109	3600	8.05	v1862e2651488d99	105	105	3454	11.24
v1195e182835d81	143	143	355	1.01	v1549e563836d101	-	143	3600	2.69	v1865e2642705d112	-	116	3600	11.56
v1201e598098d56	82	82	525	2.23	v1550e1767000d82	89	89	2162	6.90	v1868e1763392d128	-	135	3600	8.73
v1205e653110d113	-	129	3600	2.77	v1553e1619779d75	1235	85	3600	6.35	v1869e1704528d137	-	148	3600	8.14
v1211e773829d128	-	131	3600	3.29	v1557e326970d112	-	179	3600	1.99	v1888e1463200d103	-	126	3600	6.55
v1217e690039d110	301	123	3600	2.85	v1558e1799490d90	96	96	3427	7.75	v1890e2407860d55	1110	67	3600	10.03
v1222e498576d89	120	120	250	2.21	v1568e1401792d86	262	101	3600	5.91	v1904e2753184d83	267	90	3600	11.16
v1232e636944d98	117	117	2595	2.87	v1572e1829808d145	-	148	3600	8.72	v1907e226933d73	157	157	377	1.18
v1240e715480d101	115	115	733	3.03	v1580e2483760d147	-	148	3600	11.66	v1911e2973516d85	1452	90	3700	12.17
v1260e1224720d129	-	132	3600	5.30	v1581e954924d131	-	153	3600	4.89	v1913e2490726d54	958	74	3600	10.06
v1271e640584d131	322	143	3600	3.32	v1588e2051696d117	309	122	3600	8.86	v1914e803880d118	-	161	3600	4.24
v1279e860767d69	90	90	893	3.41	v1593e1932309d88	1382	94	3600	7.65	v1915e1941810d107	-	125	3600	8.89
v1290e526320d140	156	156	1343	2.72	v1594e1476044d139	-	143	3600	6.77	v1918e3277862d94	1487	98	3600	13.42
v1299e970353d61	80	80	1542	3.70	v1595e1290355d101	1733	117	3600	5.45	v1928e3399064d69	220	78	3600	12.67
v1302e1683486d96	1487	99	3600	6.68	v1621e239908d142	245	245	864	2.34	v1948e3268744d102	-	105	3600	13.50
v1307e262707d91	143	143	138	1.39	v1621e685683d71	107	107	583	3.04	v1974e3231438d137	362	138	3600	15.40
v1327e1388042d132	-	134	3600	6.50	v1633e364159d105	170	170	767	1.91	v1975e1988825d143	-	153	3600	10.07
v1332e1189476d88	276	96	3600	6.04	v1638e869778d58	86	86	551	4.24	v1977e2202378d143	-	150	3600	11.09
v1337e1472037d119	-	121	3600	6.38	v1648e1835872d77	210	89	3600	9.70	v1980e1787940d109	-	128	3600	8.30
v1341e1027206d72	84	84	363	4.24	v1660e1679920d57	73	73	2376	7.37	v1994e1886324d146	-	159	3600	9.59
v1342e308660d136	194	194	977	2.36	v1671e2018568d141	-	143	3600	11.49	v1995e3227910d55	1134	68	3600	13.17

algorithm on the 16 crafted instances.[1] Column Optima reports the optimal objective values to the instances obtained by the MIP models under unlimited timeout. Column CF CPU, SF CPU, FF CPU, $Q3$ CPU, and $Q4$ CPU presents the CPU time taken by conventional model, sequential model, commodity flow based model, Andrade's $Q3$, and $Q4$ model, respectively. Column MSM best and MSM Average give the best and average objective values collected from the five independent runs of the MSM algorithm. From Table 3 we can observe that the MSM algorithm is able to reach the optima on 14 out of 16 instances, and it always finds optimal solutions on 11 ones. For the instances where the MSM algorithm only got the sub-optima, the gaps are relatively narrow (less than 0.5%). Besides, the computational time taken by the MSM algorithm is significantly shorter than solving the MIP models on all the instances except the instances with clique sub-structure. Moreover, all MIP models failed to find the optima on c.v2000e37052d400 within one hour time limit, and there are three models which fail to solve the c.v500e4396d50, while the MSM is able to handle these hard cases.

For the instances based on random dense graphs, only the $Q4$ model is included in the comparison because it outperforms other three models for almost all the instances. The benchmark results are given in Tables 4 and 5. Column Obj represents the best objective value found within the one-hour time limit by each approach. Column CPU reports the CPU time consumed for finding the solution with the best objective value. If the objective value is not given and the CPU time is 3600 seconds, it means that no feasible solution is found. According to Table 4, it can be clearly seen that the MSM algorithm obtained high quality solutions on all the instances within 20 seconds stably, while the $Q4$ model only

[1]The PSN algorithm proposed in [12] is not included in this comparison since it is inconvenient for the authors to provide their solvers.

TABLE 5. Computational results on the 90 self-generated instances (random dense graphs with many must-pass nodes).

	Obj.		CPU			Obj.		CPU			Obj.		CPU	
Instance	Q4	MSM	Q4	MSM	Instance	Q4	MSM	Q4	MSM	Instance	Q4	MSM	Q4	MSM
v1000e999000d391	-	392	3600	6.09	v1319e1361208d489	-	490	3600	11.19	v1626e305688d533	-	578	3600	6.47
v1003e628881d128	131	131	2283	2.47	v1344e1178688d782	-	783	3600	16.97	v1640e2376360d294	-	295	3600	15.15
v1018e968118d148	337	149	3600	3.75	v1360e864960d87	1582	112	3600	3.49	v1655e885425d494	-	496	3600	10.73
v1022e405734d388	-	389	3600	3.68	v1364e377828d818	-	819	3600	11.80	v1661e1533103d906	-	907	3600	27.06
v1043e1001280d68	74	74	2046	3.31	v1366e1098264d441	-	442	3600	9.26	v1665e1591740d74	-	86	3600	6.20
v1046e595174d180	-	182	3600	2.63	v1373e1646227d482	-	483	3600	12.87	v1673e2148132d678	-	679	3600	23.67
v1049e978717d181	-	182	3600	4.01	v1375e338250d407	-	427	3600	5.57	v1679e1109819d374	-	375	3600	9.99
v1053e414882d463	-	464	3600	4.58	v1379e373709d390	-	407	3600	4.98	v1682e862866d252	-	264	3600	6.57
v1061e809543d130	-	133	3600	3.12	v1401e1882944d91	1358	94	3600	7.04	v1718e1195728d1172	-	1173	3600	37.86
v1067e294492d257	-	274	3600	2.36	v1433e1675177d791	-	792	3600	20.81	v1738e2151644d197	-	198	3600	12.40
v1068e1014600d147	-	149	3600	3.97	v1443e1193361d914	-	915	3600	21.78	v1746e1636002d279	-	280	3600	11.54
v1089e675180d454	-	455	3600	6.08	v1452e1043988d907	-	908	3600	20.32	v1764e2141496d1076	-	1077	3600	40.43
v1090e654000d569	-	570	3600	8.08	v1464e2070096d456	-	457	3600	14.96	v1768e2738632d892	-	893	3600	38.61
v1109e947086d261	-	262	3600	5.16	v1469e1332383d253	-	254	3600	7.74	v1774e431082d1075	-	1078	3600	23.18
v1118e1175018d225	-	226	3600	5.32	v1483e809718d717	-	718	3600	13.72	v1789e2461664d804	-	805	3600	33.17
v1129e534017d362	-	364	3600	4.27	v1486e291256d611	-	633	3600	8.38	v1791e3100221d1068	-	1069	3600	48.06
v1137e143262d105	173	173	338	0.87	v1488e2194800d361	-	362	3600	13.75	v1795e445160d716	-	725	3600	17.16
v1148e337512d127	167	167	1502	1.93	v1491e254961d405	-	470	3600	4.81	v1812e2139972d261	-	262	3600	13.32
v1150e716450d645	-	646	3600	10.20	v1493e1378039d283	-	285	3600	8.40	v1813e683501d526	-	530	3600	10.05
v1175e853050d262	-	263	3600	4.84	v1494e182268d541	-	647	3600	63.09	v1858e2285340d118	295	122	3600	10.55
v1182e247038d343	-	372	3600	3.15	v1499e475183d869	-	870	3600	14.78	v1859e3152864d687	-	688	3600	33.68
v1191e1313673d326	-	327	3600	7.48	v1505e1413195d364	-	365	3600	9.91	v1861e2811971d1004	-	1005	3600	41.81
v1194e1179672d119	-	123	3600	4.67	v1510e1485840d190	-	191	3600	7.44	v1863e1017198d147	-	173	3600	5.79
v1208e1234576d458	-	459	3600	9.37	v1519e428358d76	1529	120	3600	2.18	v1888e3073664d294	-	295	3600	20.29
v1213e844248d462	-	463	3600	7.61	v1543e1414931d331	-	332	3600	9.74	v1920e1628160d281	-	282	3600	11.34
v1223e163882d222	315	315	1084	4.65	v1558e190076d475	-	592	3600	38.98	v1926e2808108d1150	-	1151	3600	48.92
v1229e854155d216	-	219	3600	4.58	v1560e273000d729	-	760	3600	11.73	v1944e324648d150	-	251	3600	3.35
v1243e488499d61	220	93	3600	1.96	v1574e206194d526	-	642	3600	6.20	v1965e2161500d914	-	915	3600	35.40
v1293e321957d610	-	612	3600	7.04	v1579e2003751d683	-	684	3600	21.81	v1993e360733d230	-	341	3600	3.72
v1300e1201200d141	-	145	3600	7.68	v1616e1971520d680	-	681	3600	21.36	v1999e1733133d800	-	801	3600	27.93

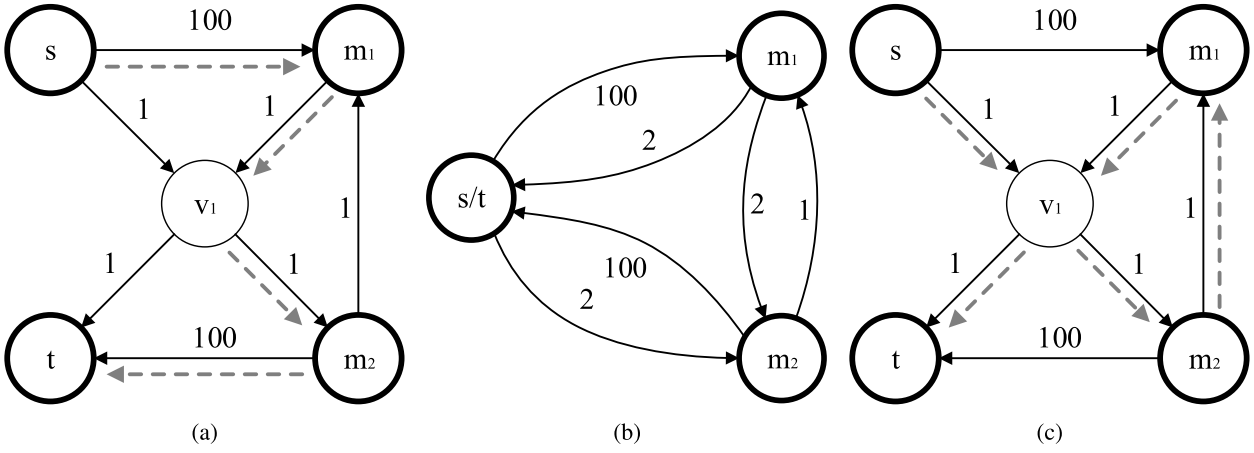

FIGURE 3. A special case where CPS fails to find feasible solutions. (a) The optimal solution; (b) the transformed problem; (c) the infeasible restored solution.

found feasible solutions on 99 instances. For the ones whose feasible solutions were found by the $Q4$ model, the MSM algorithm got better results on 37 out of them, and obtained the same results on 62 out of them. From Table 5 we can observe that, the computational time increases as the number of must-pass nodes grows for both the $Q4$ model and our MSM algorithm. However, the MSM algorithm was still able to find the optimal or sub-optimal solutions on each instance within 70 seconds. As for the $Q4$ model, the feasible solutions were found only on 11 instances within the time limit, and no better solution is found compared to the MSM algorithm. These results show the advantage of the MSM algorithm on large random dense graphs, and it implies that the MSM algorithm is highly scalable and it is able to tackle the instances with large size in a reasonable time. In sum, the MSM reaches a good balance between solution quality and computational efficiency.

C. ANALYSIS AND DISCUSSION

Due to the heuristic nature of the candidate path search (CPS) and conflicting nodes promotion (CNP) strategies, they might fail to find the optimal or even feasible solutions on certain topology structures. We will illustrate the imperfection of each strategy via three handcrafted instances, and then evaluate the different combinations of the strategies on the benchmark instances.

Fig. 3 shows a special case where the CPS fails to find a feasible solution. In the directed graph, s and t stand for the source node and the target node, respectively, and m_i represents the must-pass nodes and v_i are the normal nodes.

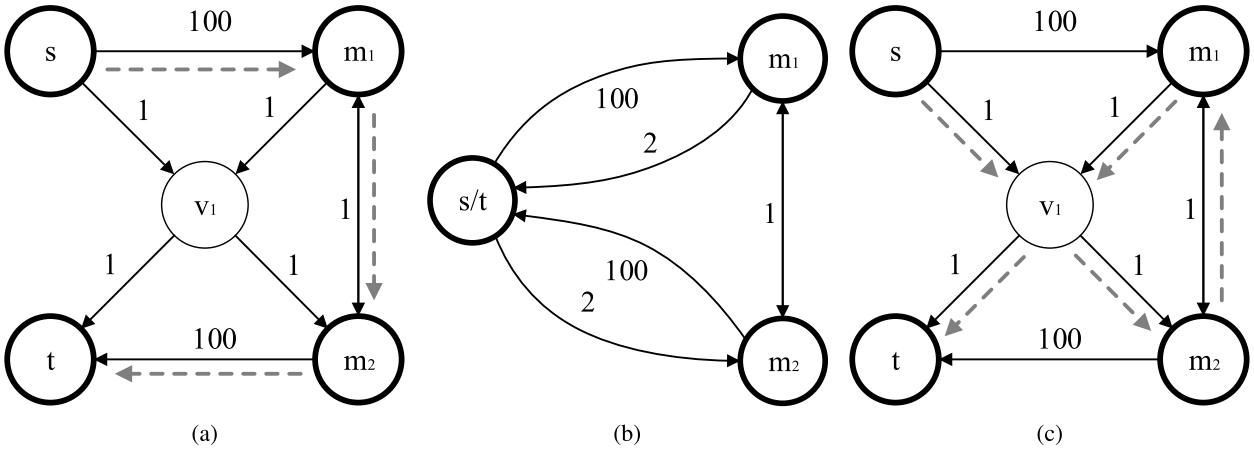

FIGURE 4. A special case where CNP fails to find optimal solutions. (a) The optimal solution; (b) the transformed problem; (c) the sub-optimal restored solution.

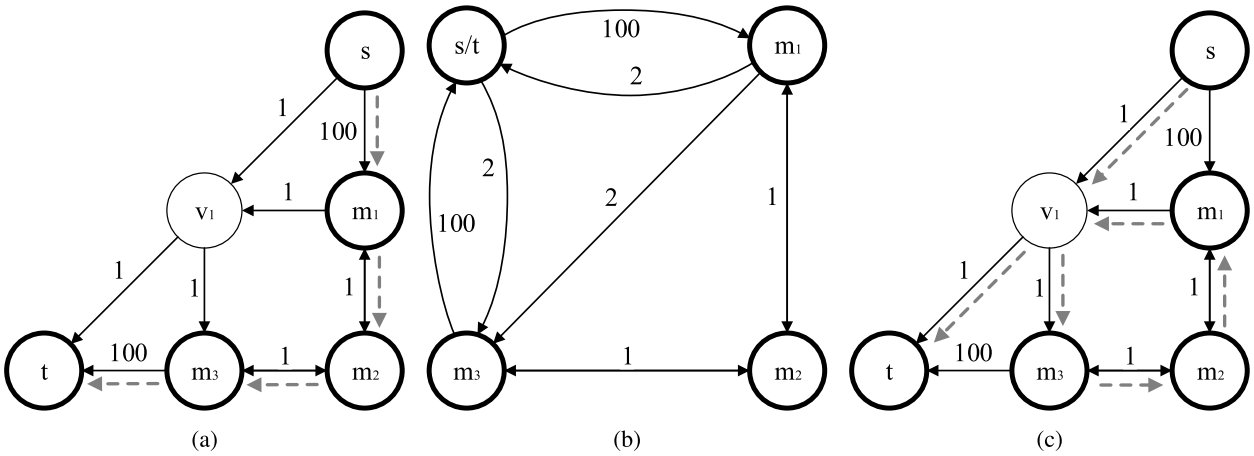

FIGURE 5. A special case where CNP fails to find feasible solutions. (a) The optimal solution; (b) the transformed problem; (c) the infeasible restored solution.

Fig.s 4 and 5 follow the same notations. The dashed arrow in Fig. 3(a) presents the optimal and the only feasible path $s \to m_1 \to v_1 \to m_2 \to t$ whose length is 202. The graph will be transformed into Fig. 3(b) and a TSP is solved on the transformed graph as Section III-A described. However, the optimal solution of the TSP restores to an infeasible path $s \to v_1 \to m_2 \to m_1 \to v_1 \to t$ which visits v_1 twice given in Fig. 3(c). Unfortunately, there is no candidate paths from s to m_2 and from m_1 to t at all, which means that it is unable to solve this case only with the CPS technique under current framework. However, if the normal node v_1 is promoted using the CNP technique, the conflict will be avoided easily.

Fig. 4 shows a special case where the CNP fails to find the optimal solution. It is almost the same as Fig. 3, except an additional edge goes from m_1 to m_2. The optimal solution is represented by the dashed arrow in Fig. 4(a) which is $s \to m_1 \to m_2 \to t$ and whose length is 201. The graph will be transformed into Fig. 4(b) and the optimal solution of the TSP on the transformed graph restores to an infeasible path $s \to v_1 \to m_2 \to m_1 \to v_1 \to t$ which visits v_1 twice given in Fig. 4(c). Unfortunately, promoting the conflicting node v_1 will lead to a sub-optimal solution $s \to m_1 \to v_1 \to m_2 \to t$ whose length is 202. Moreover, the CNP fails to find feasible solutions in another special case illustrated in Fig. 5. The optimum is $s \to m_1 \to m_2 \to m_3 \to t$ whose length is 202 and it is highlighted with dash arrow in Fig. 5(a). The resulting TSP problem of the transformation is given in Fig. 5(b), and an infeasible path $s \to v_1 \to m_3 \to m_2 \to m_1 \to v_1 \to t$ will be retrieved after solving it as we can see from Fig. 5(c). According to the conflicting nodes promotion, the repeatedly visited normal node v_1 will become a must-pass node, and this produces an unsolvable problem. It can be inferred that it will be harder to find out the optimal solution only with the CNP technique under the current framework.

To further evaluate the effectiveness of the important components of the proposed MSM, we have performed experiments to compare our MSM algorithm with its simplified versions without the conflicting nodes promotion (CNP), the candidate path search (CPS) or the connectivity relaxation (CR) strategies, respectively. Specifically, four variants of the MSM algorithm are considered in this experiment. The first one disables the CR technique, the second one disables

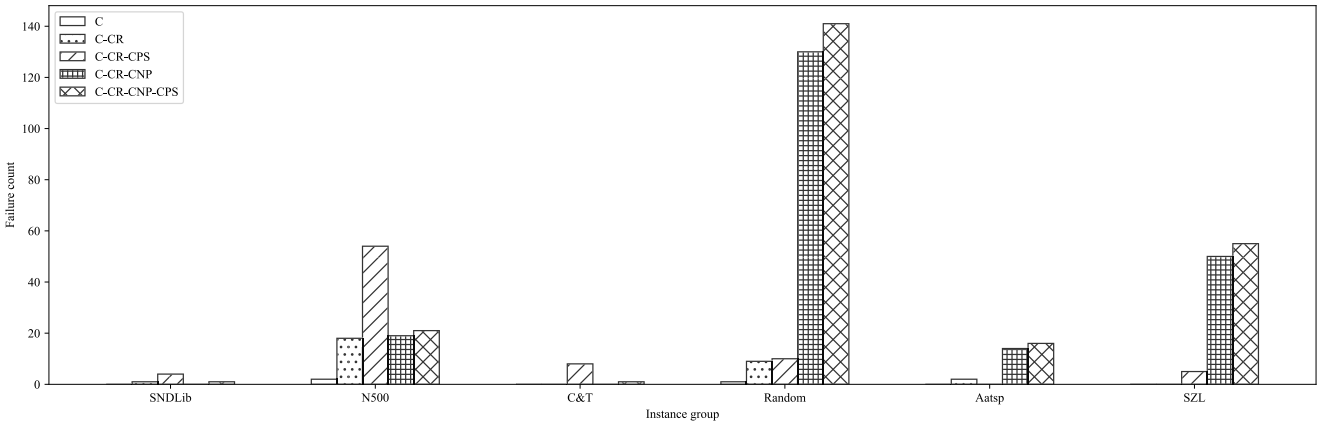

FIGURE 6. Failure count comparison among different strategies.

the CR and CPS techniques, the third one disables the CR and CNP techniques, and none of the three techniques is enabled in the last one. Since the CR technique is a post-process outside the main loop, we did not enumerate each combination with or without the CR technique. Unlike solving the TSP on complete graphs, it is hard to find a simple path visiting specific must-pass nodes on sparse graphs because there could be some critical nodes which are on the shortest paths to multiple must-pass node pairs. We have executed five independent runs on each instance with each version following the experimental protocol described in Section IV-A, and count the number of runs where the algorithms fail to find a feasible solution. In order to make the illustration clearer, we regroup the datasets into six groups, denoted by SNDLib (newyork, norway, and india35, etc.), N500 (N500G0 to N500G4), C&T (CORONET and TeliaSonera), Random (A20 to A300), Aatsp, and SZL (our own), respectively.

The summarized results of this experiment are presented in Fig. 6. The y-axis gives the number of failures of obtaining a feasible solution, and the x-axis gives the instance groups. The bars marked with C represent the statistics of the complete version of the MSM, while -CR, -CNP and -CPS means that the connectivity relaxation, the conflicting nodes promotion, and the candidate path search are disabled in the corresponding version of the algorithm, respectively.

Fig. 6 discloses that the complete version of the proposed MSM outperforms the other four versions on each dataset. Moreover, the version only without the CR post-process generally got better results than the other three incomplete versions, except that there are insignificant disadvantages on SNDLib and Aatsp groups, which implies that the connectivity relaxation technique is essential to make the performance of the MSM algorithm more stable. In detail, the version without the CPS technique, i.e., only enabling the CNP technique, always obtains feasible solutions on Aatsp group, but fails to find any feasible solution on most instances in SNDLib, N500, and C&T groups. The performance of the configuration without the CNP technique, i.e., only enabling the CPS technique, is similar to but slightly better than the one with neither the CNP nor the CPS techniques, and always found feasible solutions on SNDLib and C&T groups. Moreover, the unsuccessful rate increases dramatically in the Random group when the conflicting nodes promotion technique is disabled. These phenomena indicate that each of the CNP and the CPS techniques works well on certain topology structures, but not robust enough in general cases. Thus, we need to integrate both of them into the MSM algorithm to overcome their drawbacks and make full use of their advantages. Although the MSM fails sometime too, the probability is very low, as there are only 3 failures out of $863 \times 5 = 4315$ runs. The reason might lie in the heuristic nature of the LKH algorithm and the conflicting node promotion strategy. Therefore, simply restarting the algorithm until a feasible solution is found or running multiple parallel solvers might guarantee 100% successful rate. This experiment justifies the importance of the conflicting nodes promotion and candidate path search strategies.

V. CONCLUSION

In this paper, we have studied the shortest simple path problem with must-pass nodes by converting it into TSP and proposed a multi-stage metaheuristic algorithm that uses multi-strategies, such as k-opt move based TSP solver to determine the order of must-pass nodes, candidate path search and conflicting nodes promotion for eliminating conflicting nodes, and connectivity relaxation to ensure a valid solution. In addition, we have presented three mathematical programming formulations of the SSPP-MPN. Computational results tested on three sets of 863 instances show the effectiveness and efficiency of the proposed algorithm. Finally, additional analysis indicates the importance of the components in the proposed MSM algorithm. As we can observe from the analysis, there are still rooms for improvement for the MSM algorithm. In future, the techniques to fix incorrectly promoted nodes will be studied in order to improve the robustness of the proposed algorithm. The rapid assessment or incremental evaluation strategy of TSP solver to reduce the running time and improve efficiency will also be studied in future work.

REFERENCES

[1] I. Pardines and V. Lopez, "Shop&Go: TSP heuristics for an optimal shopping with smartphones," *Sci. China Inf. Sci.*, vol. 56, no. 11, pp. 1–12, 2013.
[2] J. P. Saksena and S. Kumar, "The routing problem with 'K' specified nodes," *Oper. Res.*, vol. 14, no. 5, pp. 909–913, 1966.
[3] S. E. Dreyfus, "An appraisal of some shortest-path algorithms," *Oper. Res.*, vol. 17, no. 3, pp. 395–412, 1969.
[4] K. Helsgaun, "An effective implementation of the Lin–Kernighan traveling salesman heuristic," *Eur. J. Oper. Res.*, vol. 126, pp. 106–130, Oct. 2000.
[5] W. Rao, X. Wang, C. Jin, and F. Liu, "On the universal strategy for improving a certain type of construction heuristic for the traveling salesman problem," *Scientia Sinica Informationis*, vol. 45, no. 8, p. 1060, 2015.
[6] M. Hou and D. Liu, "A novel method for solving the multiple traveling salesmen problem with multiple depots," *Chin. Sci. Bull.*, vol. 57, no. 15, pp. 1886–1892, 2012.
[7] T. Ibaraki, "Algorithms for obtaining shortest paths visiting specified nodes," *SIAM Rev.*, vol. 15, no. 2, pp. 309–317, 1973.
[8] H. Vardhan *et al.*, "Finding a simple path with multiple must-include nodes," in *Proc. IEEE Int. Symp. Modeling, Anal. Simulation Comput. Telecommun. Syst.*, Sep. 2009, pp. 1–3.
[9] T. Gomes, S. Marques, L. Martins, M. Pascoal, and D. Tipper, "Protected shortest path visiting specified nodes," in *Proc. 7th Int. Workshop Reliable Netw. Design Modeling (RNDM)*, Oct. 2015, pp. 120–127.
[10] T. Gomes, L. Martins, S. Ferreira, M. Pascoal, and D. Tipper, "Algorithms for determining a node-disjoint path pair visiting specified nodes," *Opt. Switching Netw.*, vol. 23, pp. 189–204, Jan. 2017.
[11] L. Martins, T. Gomes, and D. Tipper, "An efficient heuristic for calculating a protected path with specified nodes," in *Proc. Int. Workshop Resilient Netw. Design Modeling*, Sep. 2016, pp. 150–157.
[12] L. Martins, T. Gomes, and D. Tipper, "Efficient heuristics for determining node-disjoint path pairs visiting specified nodes," *Networks*, vol. 70, no. 4, pp. 292–307, 2017.
[13] R. C. de Andrade, "Elementary shortest-paths visiting a given set of nodes," in *Proc. Simpósio Brasileiro de Pesquisa Operacional*, Sep. 2013, pp. 2378–2388.
[14] R. C. de Andrade, "New formulations for the elementary shortest-path problem visiting a given set of nodes," *Eur. J. Oper. Res.*, vol. 254, no. 3, pp. 755–768, 2016.
[15] M. DeLeon, "A study of sufficient conditions for hamiltonian cycles," *Rose-Hulman Undergraduate Math. J.*, vol. 1, no. 1, p. 6, 2000.
[16] D. L. Miller and J. F. Pekny, "Exact solution of large asymmetric traveling salesman problems," *Science*, vol. 251, no. 4995, pp. 754–761, 1991.
[17] A. J. Orman and H. P. Williams, "A survey of different integer programming formulations of the travelling salesman problem," in *Optimisation, Econometric and Financial Analysis*, E. J. Kontoghiorghes and C. Gatu, Eds. Berlin, Germany: Springer, 2007, pp. 91–104.
[18] D. B. Johnson, "A note on dijkstra's shortest path algorithm," *J. ACM*, vol. 20, pp. 385–388, Jul. 1973.
[19] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Oper. Res.*, vol. 21, no. 2, pp. 498–516, 1973.
[20] J. Y. Yen, "Finding the K shortest loopless paths in a network," *Manage. Sci.*, vol. 17, pp. 712–716, Jul. 1971.
[21] T. Benoist, F. Gardi, A. Jeanjean, and B. Estellon, "Randomized local search for real-life inventory routing," *Transp. Sci.*, vol. 45, no. 3, pp. 381–398, 2011.
[22] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0—Survivable network design library," *Networks*, vol. 55, pp. 276–286, May 2010.
[23] M. Doar and I. Leslie, "How bad is naive multicast routing?" in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, vol. 1, Mar./Apr. 1993, pp. 82–89.
[24] G. Reinelt, "TSPLIB—A traveling salesman problem library," *ORSA J. Comput.*, vol. 3, no. 4, pp. 376–384, 1991.
[25] *Gurobi Optimizer Reference Manual Version 7.5*, Gurobi Optimization LLC, Houston, TX, USA, 2017.

ZHOUXING SU received the B.E. degree in computer science and technology from the Huazhong University of Science and Technology, China, in 2014, where he is currently pursuing the Ph.D. degree in computer software and theory with the Laboratory of Smart Computing and Optimization, School of Computer Science and Technology. His researches focus on using metaheuristics and mathematical programming to solve real-world applications such as personnel rostering, inventory routing, and facility location problem.

JUNCHEN ZHANG received the B.S. degree in applied physics from Southeast University, China, in 2013, and the master's degree in computer software and theory from the Huazhong University of Science and Technology, China, in 2018. He is currently a Software Engineer with Intellifusion Inc., China. His research interests include heuristic algorithm, similarity search, parallel and distributed computing, and program optimization.

ZHIPENG LÜ received the B.S. degree in applied mathematics from Jilin University, China, in 2001, and the Ph.D. degree in computer software and theory from the Huazhong University of Science and Technology, China, in 2007. He was a Postdoctoral Research Fellow with LERIA, Department of Computer Science, University of Angers, France, from 2007 to 2011. He is currently a Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, and also the Director of the Laboratory of Smart Computing and Optimization. His areas of research interests include artificial intelligence, computational intelligence, operations research and adaptive metaheuristics for solving large-scale real-world and theoretical combinatorial optimization, and constrained satisfaction problems.