Efficient Heuristic Algorithms for Single-Vehicle Task Planning With Precedence Constraints

Bai, Xiaoshan; Cao, Ming; Yan, Weisheng; Ge, Shuzhi Sam

Published in:
IEEE Transactions on Cybernetics

DOI:
10.1109/TCYB.2020.2974832

Publication date:
2021

Citation for published version (APA):
Bai, X., Cao, M., Yan, W., & Ge, S. S. (2021). Efficient Heuristic Algorithms for Single-Vehicle Task Planning With Precedence Constraints. *IEEE Transactions on Cybernetics*, *51*(12), 6274 - 6283. https://doi.org/10.1109/TCYB.2020.2974832

Efficient Heuristic Algorithms for Single-Vehicle Task Planning With Precedence Constraints

Xiaoshan Bai, Ming Cao, *Senior Member, IEEE*, Weisheng Yan, Shuzhi Sam Ge, *Fellow, IEEE*, and Xiaoyu Zhang

***Abstract*—This article investigates the task planning problem where one vehicle needs to visit a set of target locations while respecting the precedence constraints that specify the sequence orders to visit the targets. The objective is to minimize the vehicle's total travel distance to visit all the targets while satisfying all the precedence constraints. We show that the optimization problem is NP-hard, and consequently, to measure the proximity of a suboptimal solution from the optimal, a lower bound on the optimal solution is constructed based on the graph theory. Then, inspired by the existing topological sorting techniques, a new topological sorting strategy is proposed; in addition, facilitated by the sorting, we propose several heuristic algorithms to solve the task planning problem. The numerical experiments show that the designed algorithms can quickly lead to satisfying solutions and have better performance in comparison with popular genetic algorithms.**

***Index Terms*—Heuristic algorithms, lower bound, precedence constraints, task planning, topological sorting.**

I. Introduction

Task assignment, in the context of logistic systems, is to assign to a fleet of vehicles a set of tasks distributed at different target locations in a bounded area [1]–[6] or assign to a single vehicle efficient sequences to visit a set of target locations [7] while minimizing the total travel distance [8] or time [9]. The task assignment problem is a variant of the NP-hard vehicle routing problem (VRP) or the traveling salesman problem (TSP) [10], [11], implying that unaffordable computational time might be required to calculate the optimal solution as the number of target locations grows [12]. So the existing research works usually either test their algorithms on some benchmarks or compare the results with those existing solutions of known performances [13], [14]. Leading methods and the latest advances on the heuristics for solving TSP were summarized in [15]. Held and Karp [16] pointed out that the weight of a minimum-weight 1-tree/1-arborescence can be used as a lower bound for the optimal solution of the symmetric/asymmetric TSP. Caseau and Laburthe [17] presented a set of techniques to make constraint programming a chosen technique for solving small TSPs. They indicated that two lower bounds, namely, the weight of an undirected minimal spanning tree and the weight of a minimal spanning arborescence, can be used for pruning the search tree. The Lin–Kernighan heuristic (LKH) [18] starts with a randomly generated TSP tour and utilizes the generalized 2-opt exchange of links (2-opt move) to improve the tour. The LKH is effective for solving the symmetric TSP. However, many design and implementation decisions need to be determined for constructing an algorithm based on the LKH, and most of the decisions have a great influence on the efficiency of the algorithm. A vacancy chain scheduling was designed by Dahl *et al.* [19] to formalize robot interactions for the multirobot task assignment. A game-theoretic approach was designed by Belhaiza [20] for a multiple-criterion VRP with multiple time windows where a hybrid neighborhood search heuristic is applied. Using membership functions and fuzzy rules, a fuzzy route planning algorithm was developed to plan routes for ground vehicle operations in urban areas [21]. A marginal-return-based constructive heuristic was designed in [22] to solve the sensor–weapon–target assignment problem, where the interdependencies between weapons and sensors are considered.

For logistic scheduling, some customers/targets can have priority over the others due to their importance or urgency to be served. In such cases, the precedence constraints on the ordering of the visiting sequence of the targets have to be respected, and consequently the assignment of one target is directly affected by those other targets which need to be visited earlier as specified by the associated precedence constraints. For some instances of the VRP with time windows [23], precedence constraints on visiting the customers are in the form of the time windows to visit specific customers over the planning horizon. For the TSP with precedence constraints requiring a given subset of targets to be visited in some prescribed linear order [24], a polynomial-time algorithm was proposed, guaranteeing quantifiable performances. The

Manuscript received December 12, 2019; accepted February 14, 2020. Date of publication March 4, 2020; date of current version December 22, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61603094 and Grant 61633002. This article was recommended by Associate Editor J. Chen. *(Corresponding author: Xiaoshan Bai.)*

Xiaoshan Bai is with the College of Mechatronics and Control Engineering, Shenzhen University, Shenzhen 518060, China, and also with the School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: baixiaoshan.nwpu@gmail.com).

Ming Cao is with the Faculty of Science and Engineering, University of Groningen, 9747 Groningen, The Netherlands (e-mail: m.cao@rug.nl).

Weisheng Yan is with the School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: wsyan@nwpu.edu.cn).

Shuzhi Sam Ge is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576 (e-mail: samge@nus.edu.sg).

Xiaoyu Zhang is with the College of Artificial Intelligence, Nankai University, Tianjin 300350, China (e-mail: zhangxiaoyu@nankai.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2020.2974832.

Digital Object Identifier 10.1109/TCYB.2020.2974832

TSP with precedence constraints investigated in [24] can be transformed into the standard TSP by treating each subset of targets with the linear visiting constraints as one single target. In [25], an efficient GA integrated with a topological sorting technique (TST) was designed to solve the TSP with precedence constraints. The GA uses a new crossover operator, namely, moon crossover mimicking the changes of the moon, to adjust the priorities for sequencing the target locations while the TST is used to guarantee the feasibility of the planned path. Later on, a new GA based on topological sorting was developed to solve precedence-constrained sequencing problems [26]. The crossover operator used in [26] needs only one parent of chromosomes to undergo the crossover evolution and each chromosome represents a feasible solution to the problem. For the precedence-constrained TSP, Kubo and Kasugai [27] presented a branch-and-bound algorithm that incorporates lower bounds computed from the Lagrangean relaxation. In [28], several families of inequalities were derived to formulate the precedence-constrained asymmetric TSP (PCATSP). Oberlin *et al.* [29] formulated the PCATSP as a split dual model, which can be solved by using standard TSP solvers and linear program solvers.

In our previous work [30], several clustering-based algorithms have been proposed for a fleet of vehicles to efficiently visit a set of target locations in a time-invariant drift field while trying to minimize the vehicles' total travel time. As a follow up, a co-evolutionary multipopulation genetic algorithm was proposed for multiple vehicles to deliver products to a set of target locations in a time-varying drift field [31], and an auction-based algorithm was designed for task assignment of multiple vehicles in a drift field with obstacles [32]. In [33], we investigated the dynamic task assignment for multiple vehicles to visit a set of target locations where some target locations are initially known and the other target locations are dynamically generated during the vehicles' movement. In addition, we have investigated the task assignment for heterogeneous vehicles with precedence constraints [34]. Motivated by the discussed research works, this article investigates the precedence-constrained task planning problem (PCTPP) for which one vehicle needs to visit a set of target locations subject to precedence constraints for visiting the targets while trying to minimize the vehicle's total travel distance. We solve the problem by inserting the target locations iteratively into the vehicle's path taking into account the precedence constraints. Two critical questions arise: 1) which target should be inserted in each iteration and 2) where should it be inserted such that no precedence constraint is violated. Inspired by the existing TST [25], [26], we first propose to sort the precedence constraints backward, which enables us to further design several heuristic task planning algorithms to put the target locations in sequence respecting the precedence constraints. Our main contributions are as follows.

1) Using tools from the graph theory, we construct a lower bound for the optimal solution, which can be used to approximately measure the performance of a task planning algorithm.

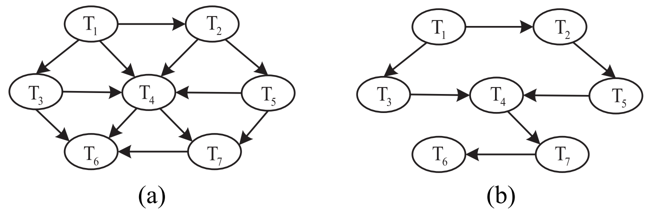

Fig. 1. Digraph $\mathcal{G}^p = (V^p, E^p)$ shows precedence constraints on visiting several target locations (a) digraph in [25] and (b) transitive reduction of (a).

2) The proposed TST enables the vehicle to visit all the target locations while satisfying every precedence constraint.
3) The designed heuristic algorithms have better performances compared with competing genetic algorithms.

The remainder of this article is organized as follows. In Section II, the mathematical formulation of the PCTPP is given. Section III analyzes the problem, and Section IV discusses several task planning algorithms. The simulation results are shown in Section V and the conclusion of this article is made in Section VI.

II. Mathematical Formulation

A. Problem Setup

Consider that one vehicle needs to visit n target locations subject to precedence constraints described by a digraph specifying which target locations need to be visited before other target locations. More specifically, a target location is represented by a vertex in the digraph, and there is a directed edge/path from one vertex to another if and only if the former needs to be visited before the latter. A feasible solution to the problem is a path for the vehicle to visit all target locations while respecting every precedence constraint. Obviously, the problem has feasible solutions only if the digraph does not have direct cycles, that is, the digraph is acyclic.

We assume that the vehicle is not required to return to its initial location. The objective is to minimize the vehicle's total travel distance to visit all target locations while satisfying every precedence constraint.

B. Formulation as Optimization Problem

Let $\mathcal{T} = \{1, \ldots, n\}$ be the set of vertices representing n target locations, and 0 denote the index of the depot where the vehicle is initially located. For $\forall i, j \in \mathcal{I}$, where $\mathcal{I} = \{0\} \cup \mathcal{T}$, let $d = (d(i,j))_{\forall i,j \in \mathcal{I}}$ be the distance matrix where $d(i,j)$ denotes the distance between i and j, and the binary variable $p_{ij} = 1$ if one requires vertex i to be visited before vertex j, and $p_{ij} = 0$ if there is no such requirement. As an example shown in Fig. 1(a), we use a digraph $\mathcal{G}^p = (V^p, E^p)$, consisting of a subset of vertices in $\mathcal{T}$ and a set of directed edges E^p, to show the precedence constraints among the vertices. Note that $p_{ij} = 1$ if and only if there is at least one directed path from i to j in $\mathcal{G}^p$. The binary decision variable $\sigma_{ij}, i, j \in \mathcal{I}$, is used, which equals one if and only if it is planned that the vehicle travels directly from location i to j, and $\sigma_{ij} = 0$ otherwise.

Inspired by the two-commodity network flow model [35], we use the network flow model to formulate the problem.

Here, each target location is assumed to have one unit demand of a commodity and the vehicle starts serving the targets from its initial location with n units of commodities. The variable c_i is used to denote the quantity of the commodity when the vehicle leaves vertex i, and correspondingly it holds $c_0 = n$. Then, the problem is to minimize the vehicle's total travel distance to visit all target locations

$$f = \sum_{i \in \mathcal{I}, j \in \mathcal{T}} d(i,j)\sigma_{ij} \tag{1}$$

subject to

$$\sum_{i \in \mathcal{I}} \sigma_{ij} = 1 \quad \forall j \in \mathcal{T} \tag{2}$$

$$\big(c_i - c_j\big)\sigma_{ij} = \sigma_{ij} \quad \forall i \in \mathcal{I} \quad \forall j \in \mathcal{T} \tag{3}$$

$$\sum_{j \in \mathcal{T}} \sigma_{ij} \leq 1 \quad \forall i \in \mathcal{I} \tag{4}$$

$$\big(c_j - c_i\big)p_{ij} \leq 0 \quad \forall i \in \mathcal{I} \quad \forall j \in \mathcal{T} \tag{5}$$

$$\sum_{i,j \in \mathcal{S}} \sigma_{ij} \leq |\mathcal{S}| - 1 \quad \forall \mathcal{S} \subseteq \mathcal{T} \text{ with } |\mathcal{S}| \geq 2. \tag{6}$$

Constraint (2) ensures that each target is visited once and only once; (3) means that the vehicle's commodity quantity at location j decreases by 1 compared with that at location i if j is visited directly after i; (4) ensures that the vehicle's initial location and each target is departed at most once; (5) ensures that the vehicle's commodity quantity at location i is larger than that at j if i is a predecessor of j; and (6) guarantees that no subtour exists when visiting the target locations.

After formulating the task planning problem as a constrained minimization problem, we present in the following section the analysis of the optimization problem.

III. Problem Analysis

The graph $\mathcal{G}^p = (V^p, E^p)$ that specifies the precedence constraints on visiting the target locations has a transitive reduction whenever the following two conditions hold at the same time: 1) one vertex i has multiple directed paths to another vertex j and 2) i has one edge directly pointing at j. An example is shown in Fig. 1(a), where target vertex T_1 has three independent directed paths to T_4 as $T_1 \rightarrow T_2 \rightarrow T_4$, $T_1 \rightarrow T_3 \rightarrow T_4$, and $T_1 \rightarrow T_4$. Here, we call the precedence constraint corresponding to the edge from T_1 to T_4 in Fig. 1(a) the *immediate precedence constraint*. Since T_1 needs to be visited before T_2 and T_2 needs to be visited before T_4, the immediate precedence constraint from T_1 to T_4 is redundant. As a result, the digraph shown in Fig. 1(a) can be simplified to Fig. 1(b) by deleting some redundant precedence constraints.

If the digraph $\mathcal{G}^p$ is empty, the PCTPP reduces to the TSP which is an NP-hard problem [12]. However, when precedence constraints exist, it is not clear whether the PCTPP is still NP-hard. On the one hand, the solution space for planning reduces when there are some precedence constraints specifying which target locations need to be visited before other target locations, which can lead to faster optimization processes if the solution space is somehow structured. On the other hand, sorting the precedence constraints to generate a feasible solution for visiting all the target locations requires more computational operations in solving the optimization problem. As a result, it is necessary to investigate the computational complexity for optimally solving the investigated problem.

Remark 1: The precedence-constrained TSP (PTSP) has been shown to be NP-hard by Charikar *et al.* [36]. As a result, the PCTPP is NP-hard as PTSP is a special case of the PCTPP.

A. Lower Bound on the Optimal Solution

Due to the NP-hardness of the PCTPP, computing for an optimal solution to the problem can be time consuming. As a result, one natural idea is to develop heuristic algorithms to look for suboptimal solutions. However, evaluating the quality of one suboptimal solution in terms of its comparison with the optimal is another issue to be solved. Thus, we will construct a lower bound of (1) to measure the proximity of a suboptimal solution from the optimal. To construct a lower bound of (1) rigorously, the definition of the arborescence of a digraph from graph theory is first introduced.

Definition 1: An arborescence is a digraph with a single root, where exactly one directed path starts from the root to any other vertex [37].

In this section, a lower bound on the minimum travel distance for the vehicle to visit all target locations while respecting every precedence constraint is obtained by calculating a min-cost arborescence (MCA) of a weighted digraph $\mathcal{G}^d$ introduced later. The sum of the edge weights of an MCA is the minimum among all arborescences of $\mathcal{G}^d$, and Edmonds' algorithm [38] can be used to achieve an MCA within polynomial computational time.

Now consider the undirected graph $\mathcal{G} = (V, E, D)$ consisting of the $n+1$ vertices in $\mathcal{I}$, a set of weighted undirected edges E, and a distance matrix D that contains the weight of each edge in E which is the distance between the two vertices associated with the edge. The digraph $\mathcal{G}^p$ and the weighted $\mathcal{G}$ are the inputs to the problem (1). We integrate $\mathcal{G}(V, E, D)$ and $\mathcal{G}^p = (V^p, E^p)$ that specifies the precedence constraints on visiting the target vertices, and then obtain the weighted directed graph $\mathcal{G}^d = (V, E^d, D^d)$ consisting of the $n+1$ vertices in V, a directed edge set E^d where an edge orienting from vertex i to j exists if vertex i can be visited before vertex j as shown in $\mathcal{G}^p$, and a distance matrix D^d that contains the weight of each edge in E^d based on D. We give an example of how to formulate the digraph $\mathcal{G}^d$ based on the digraph $\mathcal{G}^p$ shown in Fig. 1(b) and the corresponding weighted undirected graph $\mathcal{G}$. For each vertex i in $\mathcal{I}$, a vertex set S_i is used to keep the indices of the vertices before which i can be visited. S_i is calculated in a backward manner. First, $S_i = \mathcal{I}\backslash\{0,\ i\}$ for every vertex $i \notin V^p$. Then, for Fig. 1(b), S_6 is first obtained as $S_6 = \cup_{i \in \mathcal{I}\backslash\{V^p\}} i$. Afterward, $S_i = \cup_{p'_{ij}=1}(\{j\} \cup S_j)$, where $p'_{ij} = 1$ if the vertex i has one edge directly pointing at j in the simplified digraph $\mathcal{G}^p$ as T_1 and T_2 in Fig. 1(b). Thus, $S_7 = \{6\} \cup S_6$. Iteratively, S_i can be obtained for every $i \in V^p$. Then, an edge (i, j) exists in E^d connecting vertex i and every $j \in S_i$, and the corresponding $D^d(i, j)$ stores the distance between vertices i and j; for the other cases, $D^d(i, j) = \infty$. We use f_a as the sum of every edge weight of an MCA of the weighted directed graph $\mathcal{G}^d$, and f_o is the optimal value for the objective function in (1). Now, we investigate the property of the optimal solution.

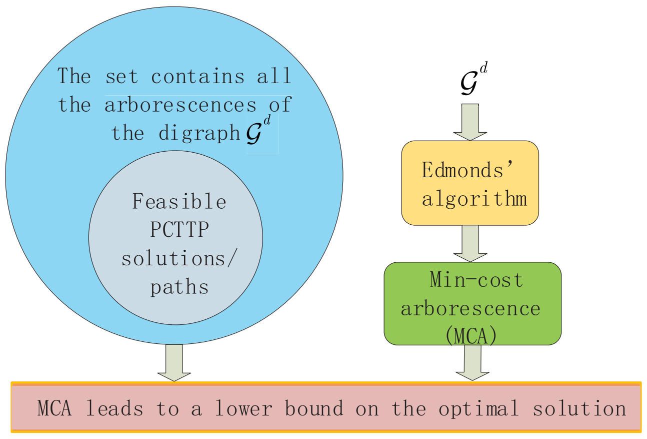
The set contains all the arborescences of the digraph $\mathcal{G}^d$
Feasible PCTTP solutions/ paths
$\mathcal{G}^d$
Edmonds' algorithm
Min-cost arborescence (MCA)
MCA leads to a lower bound on the optimal solution

Fig. 2. Relationship between the MCA and a lower bound on the optimal solution to the PCTPP.

Proposition 1: It holds that $f_a \le f_o$.

Proof: According to Definition 1, an optimal path for the vehicle starting from its initial location to visit all target locations while satisfying every precedence constraint on visiting them is in fact an arborescence of the digraph $\mathcal{G}^d$. Since f_a is the sum of every edge weight of an MCA of $\mathcal{G}^d$, it holds that $f_a \le f_o$. Fig. 2 shows the relationship between the MCA and a lower bound on the optimal solution to the PCTPP. ■

Having performed the theoretical analysis, in the next section, we construct heuristic algorithms.

IV. Task Planning Algorithms

Inspired by [25] and [26], the PCTPP can be solved by iteratively inserting a "viable" target vertex into a path until all the targets are inserted, where a TST is needed for determining which targets are viable to be inserted into the current vehicle path at each iteration and an inserting method (IM) is required to choose the proper position for inserting each target. We will give the definition of *viable target vertex* later. In this section, we first present two TSTs, and then we propose several task planning algorithms facilitated by the sorting.

A. Topological Sorting Techniques

Through topological sorting, one can obtain all the feasible paths in a directed graph [39]. During sorting, the vehicle's path can be built either forward or backward. Initially, we let $\mathcal{G}^{p'} = \mathcal{G}^p$.

1) Forward Topological Sorting Technique: In [25] and [26], the precedence-constrained TSP is solved by iteratively employing a TST which can sort the target vertices in $\mathcal{G}^{p'}$ that do not have any predecessor in each iteration. The target vertices without any predecessor are called the *viable target vertices* for the forward topological sorting. For forward topological sorting, once a target vertex is inserted, we update $\mathcal{G}^{p'}$ by deleting the vertex denoting the target vertex and the corresponding edges leaving the vertex in the current $\mathcal{G}^{p'}$. We give an example of how to generate a feasible path from the representation scheme by considering the precedence constraints shown in the simplified digraph Fig. 1(b) assuming $n = 7$. In the digraph, the first target vertex sorted to be inserted into the vehicle's path is T_1, since it is the only target vertex in the digraph without any predecessor. Then, T_1 is stored in the path, and at the same time T_1 and the edges (T_1, T_2), (T_1, T_3) coming from T_1 are removed from $\mathcal{G}^{p'}$. In the next iteration, T_2 and T_3 are viable target vertices of the resulting $\mathcal{G}^{p'}$, which can then be inserted in the vehicle path after T_1. The process continues until all the target vertices in the digraph $\mathcal{G}^p$ are inserted into the path.

2) Backward Topological Sorting Technique: Similar to the forward topological sorting just discussed, one can also construct a backward topological sorting which constructs the vehicle path backward by iteratively inserting one target vertex in $\mathcal{G}^{p'}$ without any successor into the feasible position on the current path in each iteration. The target vertices without any successor are called *viable targets* for the backward topological sorting. For backward topological sorting, once inserting a target vertex, the vertex denoting the target vertex and the precedence constraints corresponding to the edges pointing at the vertex in the current $\mathcal{G}^{p'}$ are deleted to update $\mathcal{G}^{p'}$. We also give an example of how to generate a feasible vehicle path from the representation scheme by considering Fig. 1(b) assuming $n = 7$. In the digraph, the first target vertex to be inserted into the vehicle's path is T_6, since it is the only target vertex without any successor. Then, T_6 is stored in the path, and at the same time T_6 and the edge (T_7, T_6) pointing at T_6 are removed from $\mathcal{G}^{p'}$. In the resulting $\mathcal{G}^{p'}$, T_7 is viable to be inserted into the path before T_6 as it has no successor in $\mathcal{G}^{p'}$ after deleting T_6. One feasible path will be generated by continuing the procedure.

B. Forward Task Planning Algorithms

Let $\mathcal{R}_t$ contain the ordered target vertices already inserted into the path after iteration t, and the vertex set $\mathcal{T}^A_{\mathcal{R}_t}$ contain the indices of those target vertices that have not been inserted and have no predecessor in $\mathcal{G}^{p'}$ after iteration t. Let (i, j) denote an edge from i to j in the simplified $\mathcal{G}^p$, and $\mathcal{T}^j_{\mathcal{R}_t} = \{i \in \mathcal{R}_t : (i, j) \in E^p\}$ for each $j \in \mathcal{T}^A_{\mathcal{R}_t}$. We use the set $\mathcal{T}^j_a$ to contain the ordered locations in $\mathcal{R}_t$ after which target vertex j can be inserted while satisfying all the precedence constraints on visiting j. It is straightforward to check that j can only be inserted into $\mathcal{R}_t$ after all the target vertices in $\mathcal{T}^j_{\mathcal{R}_t}$. If $\mathcal{T}^j_{\mathcal{R}_t} = \emptyset$, j can be inserted at any position of $\mathcal{R}_t$. For the forward task planning algorithms, $\mathcal{R}_t$ is initially set as $\{0\}$ where the vehicle is initially located.

1) Forward Nearest Inserting Algorithm: The first heuristic algorithm is the forward nearest inserting algorithm (FNIA) where the target vertex $j^\star \in \mathcal{T}^A_{\mathcal{R}_t}$ to be inserted and its inserting position $(q^\star + 1)$ in iteration $(t + 1)$ satisfy

$$(q^\star, j^\star) = \underset{q \in \mathcal{T}^j_a,\ j \in \mathcal{T}^A_{\mathcal{R}_t}}{\operatorname{argmin}} \ d(\mathcal{R}_t(q), j) \tag{7}$$

where $\mathcal{T}^j_a = \{p, \ldots, |\mathcal{R}_t|\}$; $p = \max_{i \in \mathcal{T}^j_{\mathcal{R}_t}} \operatorname{find}(\mathcal{R}_t, i)$ is the farthest position to the end of $\mathcal{R}_t$ after which target vertex j can be inserted; and $\mathcal{R}_t(q)$ is the qth ordered target vertex on the route $\mathcal{R}_t$. The operator $\operatorname{find}(\mathcal{R}_t, i)$ finds the location in $\mathcal{R}_t$ where the target vertex i is inserted. Afterward, the path $\mathcal{R}_t$

is updated to

$$\mathcal{R}_{t+1} = \begin{cases} \{\mathcal{R}_t, j^\star\}, & \text{if } q^\star = |\mathcal{R}_t| \\ \{\mathcal{R}_t(1:q^\star), j^\star, \mathcal{R}_t(q^\star+1:|\mathcal{R}_t|)\}, & \text{otherwise} \end{cases} \tag{8}$$

where $|\mathcal{R}_t|$ is the size of $\mathcal{R}_t$ and $\mathcal{R}_t(1 : q^\star)$ contains the ordered target vertices located between the first and the $q^\star$th locations of $\mathcal{R}_t$.

After inserting $j^\star$, we delete $j^\star$ and all the precedence constraints initiating from $j^\star$ to update $\mathcal{G}^{p'}$. Then, the forward TST is used to update $\mathcal{T}^A_{\mathcal{R}_{t+1}}$ which contains the viable target vertices after iteration $t + 1$. The inserting procedure continues according to (7) and (8) until all the vertices in $\mathcal{T}$ are inserted into the vehicle's path.

An example of how FNIA works is shown as follows. Assume that the current vehicle path for visiting the targets under the precedence constraints shown in Fig. 1(b) is $\mathcal{R}_t = \{T_1, T_2, T_5\}$. Then, the viable target set is $\mathcal{T}^A_{\mathcal{R}_t} = \{T_3\}$ as T_3 is the only target vertex without any predecessor after deleting the target vertices already in $\mathcal{R}_t$ and the corresponding edges in $\mathcal{G}^{p'}$. As T_1 is the only target vertex that should be visited before T_3, and T_3 can be inserted at any place after T_1. Assume that $\mathcal{R}_t = \{T_1, T_3, T_2, T_5\}$ after inserting T_3. Then, the next viable target vertex T_4 can only be inserted after T_5 as T_5 has the precedence constraint over T_4, where $p = 4$ according to (7). One feasible path is $\mathcal{R}_t = \{T_1, T_3, T_2, T_5, T_4, T_7, T_6\}$ after operating the inserting procedure iteratively.

2) Forward Minimum Marginal-Cost Algorithm: The other forward task planning algorithm is the forward minimum marginal-cost algorithm (FMMA), which finds the target vertex $j^\star \in \mathcal{T}^A_{\mathcal{R}_t}$ to be inserted and its inserting position $q^\star$ in $\mathcal{R}_t$ in iteration $(t + 1)$ by

$$\left(q^\star, j^\star\right) = \underset{p+1 \le q \le |\mathcal{R}_t|+1,\ j \in \mathcal{T}^A_{\mathcal{R}_t}}{\operatorname{argmin}} \left\{d\left(\mathcal{R}_t \oplus_q j\right) - d(\mathcal{R}_t)\right\} \tag{9}$$

where $p = \max_{i \in \mathcal{T}^j_{\mathcal{R}_t}} \text{find}(\mathcal{R}_t, i)$ is the farthest position to the end of $\mathcal{R}_t$ after which target vertex j can be inserted; and the operation $\mathcal{R}_t \oplus_q j$ inserts j at the qth position of $\mathcal{R}_t$. Target vertex j is inserted to the end of $\mathcal{R}_t$ if $q = |\mathcal{R}_t| + 1$, and $d(\mathcal{R}_t)$ denotes the total travel distance for the vehicle to visit all the targets in $\mathcal{R}_t$. Then, path $\mathcal{R}_t$ is updated to

$$\mathcal{R}_{t+1} = \mathcal{R}_t \oplus_{q^\star} j^\star. \tag{10}$$

C. Backward Task Planning Algorithms

For constructing the backward task planning algorithms, let $\mathcal{T}^A_{\mathcal{R}_t}$ contain the indices of those target vertices in $\mathcal{G}$ that have not been inserted and have no successor in $\mathcal{G}^{p'}$ after iteration t. Let $\mathcal{T}^A_{\emptyset}$ contain the indices of those target vertices in $\mathcal{G}$ that have no successor in $\mathcal{G}^p$, which is the initialization of $\mathcal{T}^A_{\mathcal{R}_0}$. Let $\mathcal{T}^j_{\mathcal{R}_t} = \{i \in \mathcal{R}_t : (j, i) \in E^p\}$ for each $j \in \mathcal{T}^A_{\mathcal{R}_t}$, and $\mathcal{T}^j_a$ contain the ordered locations in $\mathcal{R}_t$ before which j can be inserted while satisfying every precedence constraint from j. It is straightforward that j can only be inserted in $\mathcal{R}_t$ before all the target vertices in $\mathcal{T}^j_{\mathcal{R}_t}$. If $\mathcal{T}^j_{\mathcal{R}_t} = \emptyset$, j can be inserted at any position of $\mathcal{R}_t$. For backward task planning algorithms, it should be noted that $\mathcal{R}_t$ is initially empty, which differs from the forward task planning algorithms. The difference leads to the importance to insert the first target vertex into $\mathcal{R}_t$ properly.

TABLE I
TST AND IM USED TO CONSTRUCT EACH HEURISTIC

TST \ IM	Nearest inserting	Minimum marginal-cost inserting
Forward	FNIA	FMMA
Backward	BNIA	BMMA

1) Backward Nearest Inserting Algorithm: The third heuristic algorithm is the backward nearest inserting algorithm (BNIA) where the route $\mathcal{R}_t, t = 0$, is initialized to contain the target

$$j^\star = \underset{k \in \mathcal{T}^A_{\mathcal{R}_{1j}},\ j \in \mathcal{T}^A_{\emptyset}}{\operatorname{argmin}} d(k, j) \tag{11}$$

where $\mathcal{T}^A_{\mathcal{R}_{1j}}$ is the viable target set in iteration $t = 1$ if the first viable target vertex to be inserted is $\mathcal{R}_0 = \{j\}$.

Then, the target vertex $j^\star \in \mathcal{T}^A_{\mathcal{R}_t}$ to be inserted and its inserting position $q^\star$ in iteration $t + 1$ are

$$\left(q^\star, j^\star\right) = \underset{q \in \mathcal{T}^j_a,\ j \in \mathcal{T}^A_{\mathcal{R}_t}}{\operatorname{argmin}} d(\mathcal{R}_t(q), j) \tag{12}$$

where $\mathcal{T}^j_a = \{1, \ldots, p\}$ and $p = \min_{i \in \mathcal{T}^j_{\mathcal{R}_t}} \text{find}(\mathcal{R}_t, i)$ is the farthest position to the start of $\mathcal{R}_t$ before which target vertex j can be inserted. Afterward, $\mathcal{R}_t$ is updated to

$$\mathcal{R}_{t+1} = \begin{cases} \{j^\star, \mathcal{R}_t\}, & \text{if } q^\star = 1 \\ \{\mathcal{R}_t(1:q^\star-1), j^\star, \mathcal{R}_t(q^\star:|\mathcal{R}_t|)\}, & \text{otherwise.} \end{cases} \tag{13}$$

After inserting $j^\star$, we delete $j^\star$ and all the precedence constraints directly pointing at $j^\star$ in $\mathcal{G}^{p'}$. Then, the backward TST is used to update $\mathcal{T}^A_{\mathcal{R}_{t+1}}$ which contains the viable target vertices after iteration $t + 1$. Finally, the inserting procedure continues according to (12) and (13) until all the vertices in $\mathcal{T}$ are inserted into the vehicle's path.

2) Backward Minimum Marginal-Cost Algorithm: The other backward task planning algorithm is the backward minimum marginal-cost algorithm (BMMA) where the initial route $\mathcal{R}_0$ is initialized as (11). Then, in iteration $t+1$ BMMA determines the target vertex $j^\star \in \mathcal{T}^A_{\mathcal{R}_t}$ to be inserted into $\mathcal{R}_t$ and its inserting position $q^\star$ in $\mathcal{R}_t$ by

$$\left(q^\star, j^\star\right) = \underset{1 \le q \le p-1,\ j \in \mathcal{T}^A_{\mathcal{R}_t}}{\operatorname{argmin}} \left\{d\left(\mathcal{R}_t \oplus_q j\right) - d(\mathcal{R}_t)\right\} \tag{14}$$

where $p = \min_{i \in \mathcal{T}^j_{\mathcal{R}_t}} \text{find}(\mathcal{R}_t, i)$. Then, the path $\mathcal{R}_t$ is updated according to (10). Fig. 3 shows the process of the proposed heuristics, and Table I presents the mechanisms for constructing the heuristics.

D. Computational Complexity

We discuss the computational complexity for running FNIA, FMMA, BNIA, and BMMA in this section. The four algorithms iteratively insert a target vertex to the vehicle's path whose length is $|\mathcal{R}_t| = t$ after the tth iteration of the inserting

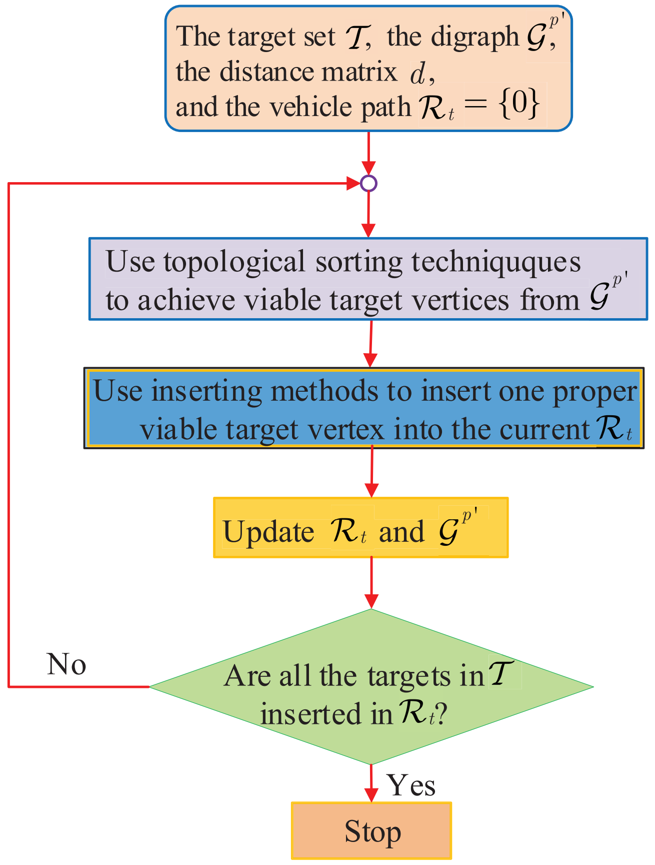
The target set $\mathcal{T}$, the digraph $\mathcal{G}^{p'}$, the distance matrix d, and the vehicle path $\mathcal{R}_t = \{0\}$
Use topological sorting techniquqes to achieve viable target vertices from $\mathcal{G}^{p'}$
Use inserting methods to insert one proper viable target vertex into the current $\mathcal{R}_t$
Update $\mathcal{R}_t$ and $\mathcal{G}^{p'}$
Are all the targets in $\mathcal{T}$ inserted in $\mathcal{R}_t$?
No
Yes
Stop

Fig. 3. Process of the proposed heuristics.

operation. The computational complexity of FNIA is determined by (7) where finding p requires at most $|\mathcal{T}^j_{\mathcal{R}_t}||\mathcal{R}_t|$ basic operations in the $(t+1)$th iteration of the assignment. Thus, to find $q^\star$ and $j^\star$ in (7), at most $|\mathcal{T}^j_a||\mathcal{T}^A_{\mathcal{R}_t}||\mathcal{T}^j_{\mathcal{R}_t}||\mathcal{R}_t|$ basic operations are needed in the $(t+1)$th iteration, where $|\mathcal{T}^j_a| \le |\mathcal{R}_t|$, $|\mathcal{T}^A_{\mathcal{R}_t}| \le n - |\mathcal{R}_t|$, and $|\mathcal{T}^j_{\mathcal{R}_t}| \le |\mathcal{R}_t|$. As a consequence, at most $t^3(n-t)$ basic operations are required in the $(t+1)$th iteration. Taking the sum for t to change from 1 to n, we obtain the computational complexity of FNIA $\sum_{t=1}^n t^3(n-t)$, resulting in $O(n^5)$. Here, a function $f(x)$ is $O(g(x))$ if constants c and x' exist such that $f(x) \le cg(x), \forall x \ge x'$. Similar to FNIA, the computational complexity of FMMA is determined by (9) where at most $2|\mathcal{R}_t||\mathcal{T}^A_{\mathcal{R}_t}||\mathcal{T}^j_{\mathcal{R}_t}||\mathcal{R}_t|$ basic operations are required in the $(t+1)$th iteration. Thus, the computational complexity of FMMA is also $O(n^5)$.

The analysis of the computational complexity of the backward task planning algorithms BMMA and BNIA is similar to those of FMMA and FNIA. The extra operations required by BMMA and BNIA are to achieve the first target to be inserted into the vehicle's path $\mathcal{R}_t$ when $t = 1$ as shown in (11). When $t = 1$, one of the maximum n target vertices is chosen to be inserted into $\mathcal{R}_t$ and for each candidate target vertex at most $n-1$ basic operations are required to calculate the best target $j^\star$ in (11). Thus, the computational time for BMMA is $n(n-1) + \sum_{t=2}^n 2t^3(n-t)$ while that for BNIA is $n(n-1) + \sum_{t=2}^n t^3(n-t)$. Then, the computational complexity of BMMA is $O(n^5)$, and the computational complexity of BNIA is $O(n^5)$.

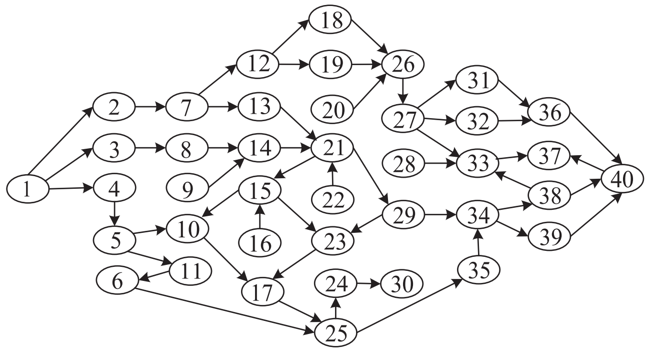

Fig. 4. Precedence-constrained digraph $\mathcal{G}^p$ contains 40 target vertices.

V. Simulations

For the precedence-constrained planning problems, extensive simulations are carried out to test the proposed algorithms compared with the GAs [25], [26]. We name the compared algorithms [25], [26] respectively, GA02 and GA11 to distinguish them, and set the genetic parameters for the two algorithms as follows according to [25] and [26]. For GA02, 500 is the maximum generation number; 150 is the population size; 0.5 is the crossover rate; and 0.2 is the mutation rate [25]. For GA11, 2000 is the maximum generation number; 20 is the population size; 0.5 is the crossover rate; and 0.05 is the mutation rate [26]. We implement the comparing experiments on an Intel Core i5-4590 CPU 3.30 GHz with 8-GB RAM, and compile the algorithms by MATLAB under Windows 7. For each algorithm, the solution quality is quantified by the ratio

$$r = \frac{f}{f_a} \tag{15}$$

where f is the objective value resulting from (1) and f_a is the sum of every edge weight of an MCA of the weighted directed target-vehicle graph $\mathcal{G}^d$. Since $f_a \le f_o$ as shown in Proposition 1 where f_o is the vehicle's minimum travel distance, the value r closer to 1 implies a better quality of the solution.

We first test the algorithms on the task planning problem where 40 target locations are subject to the precedence constraints shown in Fig. 4 which is simplified from [25, Fig. 11]. Ten instances of the initial positions of the targets and the vehicle are randomly distributed in a square area with edge length 10^3m. We perform 20 trials of the GAs for each instance to eliminate their randomness. The ratio r of the proposed algorithms and the average r of the GAs on each instance, and the average computational time are shown in Tables II and III, respectively. First, GA11 betters than GA02 since its r values of every instance shown in Table II are smaller than that of GA02. Second, GA11 is better than BNIA and FNIA as most of its r are smaller than that of BNIA and FNIA, and so does BMMA to BNIA. Furthermore, it is difficult to compare the performance of BNIA and FNIA as the ratio r of BNIA on almost half of the instances is smaller than that of FNIA. Finally, FMMA is the best algorithm among all the algorithms as it achieves the smallest r for most of the instances in Table II.

TABLE II
RATIO r OF THE ALGORITHMS (A) FOR THE PCTPP WITH 40 TARGET LOCATIONS UNDER DIFFERENT INSTANCES (I)

I \ A	GA02	GA11	BNIA	FNIA	BMMA	FMMA
1	2.6761	2.4081	2.5437	2.9080	2.7453	2.3119
2	3.2416	2.9354	3.5064	3.0226	3.3357	2.6937
3	2.9583	2.6377	3.1190	3.0937	2.7908	2.4118
4	3.1600	2.9423	3.3459	3.5400	3.2492	2.7504
5	3.4219	3.0991	3.1706	3.6018	2.7523	3.1078
6	3.8582	3.2057	3.2941	3.5416	3.4363	3.3479
7	2.9699	2.6514	3.2417	2.5874	2.9346	2.7137
8	2.8946	2.5995	3.1214	2.7540	2.5047	2.5036
9	3.6385	3.2609	3.3994	3.3647	3.3037	2.8736
10	3.4780	3.0181	3.3529	3.5016	3.2527	2.8095

TABLE III
CORRESPONDING COMPUTATIONAL TIME (s) FOR THE ALGORITHMS (A) TO OBTAIN THE SOLUTION TO THE PCTPP WITH 40 TARGET LOCATIONS UNDER DIFFERENT INSTANCES (I)

I \ A	GA02	GA11	BNIA	FNIA	BMMA	FMMA
1	65.132	26.402	0.055	0.067	0.108	0.097
2	65.174	25.570	0.067	0.055	0.054	0.056
3	65.247	25.448	0.068	0.050	0.049	0.055
4	64.051	26.467	0.066	0.050	0.046	0.055
5	64.005	26.305	0.067	0.051	0.049	0.054
6	63.895	25.980	0.067	0.050	0.051	0.053
7	64.851	25.945	0.068	0.051	0.049	0.053
8	64.761	25.906	0.068	0.051	0.050	0.051
9	64.849	26.031	0.067	0.051	0.048	0.055
10	64.508	25.885	0.069	0.050	0.051	0.054

TABLE IV
UPPER TRIANGULAR PART OF THE DIM, WHERE $\mathrm{DIM}(i, j) = 1$ IF THERE IS A SIGNIFICANT PERFORMANCE DIFFERENCE (5% LEVEL) BETWEEN ALGORITHM (A) i AND ALGORITHM j FOR SOLVING THE PCTTP WITH 40 TARGET LOCATIONS AND $\mathrm{DIM}(i, j) = 0$ OTHERWISE

A \ A	GA02	GA11	BNIA	FNIA	BMMA	FMMA
GA02	-	1	0	0	0	1
GA11	-	-	1	1	0	1
BNIA	-	-	-	0	1	1
FNIA	-	-	-	-	0	1
BMMA	-	-	-	-	-	1
FMMA	-	-	-	-	-	-

TABLE V
UPPER TRIANGULAR PART OF THE QIM, WHERE $\mathrm{QIM}(i, j) = 1$ IF ALGORITHM (A) i PERFORMS WORSE COMPARED WITH ALGORITHM j FOR SOLVING THE PCTTP WITH 40 TARGET LOCATIONS AND $\mathrm{QIM}(i, j) = 0$ OTHERWISE

A \ A	GA02	GA11	BNIA	FNIA	BMMA	FMMA
GA02	-	1	0	1	1	1
GA11	-	-	0	0	0	1
BNIA	-	-	-	1	1	1
FNIA	-	-	-	-	1	1
BMMA	-	-	-	-	-	1
FMMA	-	-	-	-	-	-

To further evaluate the solution quality r for the instances, we carry out the Wilcoxon signed-rank test in a two-tail test with the 5% significance level to compare the performance of each pair of the algorithms. The Wilcoxon signed-rank test uses two steps to check which algorithms have significantly better performance over which other algorithms: 1) first check whether any two of the algorithms have significant performance difference between each other and 2) compare the performances between the two algorithms. For solving the PCTTP with 40 target locations, Table IV shows the upper triangular part of the difference index matrix (DIM) resulting from the Wilcoxon signed-rank test, where $\mathrm{DIM}(i, j) = 1$ if there is a significant performance difference (5% level) between algorithms i and j and $\mathrm{DIM}(i, j) = 0$ otherwise. Table V shows the upper triangular part of the quality index matrix (QIM) due to its symmetry, where $\mathrm{QIM}(i, j) = 1$ if algorithm i performs worse compared with algorithm j and $\mathrm{QIM}(i, j) = 0$ otherwise. Table IV shows that:

1) GA02 has a significant performance difference compared with GA11 and FMMA;
2) GA11 has a significant performance difference compared with BNIA, FNIA, and FMMA;
3) BNIA has a significant performance difference compared with BMMA and FMMA;
4) FNIA has a significant performance difference compared with FMMA;
5) BMMA has a significant performance difference compared with FMMA.

Table V shows that GA02 performs worse than GA11 and FMMA; GA11 performs better than BNIA and FNIA, but worse than FMMA; BNIA performs worse than BMMA and FMMA; FNIA is worse than FMMA, and so as BMMA to FMMA. It is clear that the ratio r of the ten instances differ significantly between the algorithms (r from small to large corresponds to FMMA $\rightarrow$ GA11 $\rightarrow$ GA02; FMMA $\rightarrow$ BMMA $\rightarrow$ BNIA; GA11 $\rightarrow$ BNIA and GA11 $\rightarrow$ FNIA) while there is no significant difference between GA02, BNIA, and FNIA. This implies that the algorithms have an increasingly better performance as GA02 − GA11 − FMMA; BNIA − BMMA − FMMA; BNIA − GA11; and FNIA − GA11, which agrees with the previous performance analysis for the algorithms. The better performance of FMMA over FNIA and respectively, BMMA over BNIA show that the marginal-cost-based target inserting strategy (9) is more efficient than (7). The reason lies partly in the fact that the minimum marginal-cost algorithms FMMA and BMMA achieve the sequences for visiting the target locations after calculating the incurred travel distance at every possible position on the vehicle's path; in contrast, FNIA and BNIA are myopic in the sense that they connect the target location to be inserted to the nearest feasible target location already in the vehicle's route. The algorithms BMMA and FMMA both use the minimum marginal-cost strategy integrating one topological sorting technology to construct the vehicle route. For FMMA, the first target vertex to be inserted is based on the position of the depot where the vehicle is initially located while for BMMA there is no fixed position taken as a reference to insert the first target. According to (9) and (14), the target vertex to be inserted in each iteration and the corresponding inserting place are affected by the targets already on the vehicle route. As a result, FMMA generally performs better than BMMA. Another encouraging observation is the smaller computational time of the proposed algorithms compared with those of the GAs as

TABLE VI
RATIO r OF THE ALGORITHMS (A) FOR THE PCTPP WITH 120 TARGET LOCATIONS UNDER DIFFERENT INSTANCES (I)

I \ A	GA02	GA11	BNIA	FNIA	BMMA	FMMA
1	7.1649	6.4352	2.3694	2.1289	1.9676	2.1816
2	6.8112	6.0086	2.3690	2.2344	2.0426	1.7751
3	6.7092	6.1408	2.2823	2.1282	1.7761	1.6990
4	7.2989	6.3534	2.3335	2.1735	2.0934	1.7743
5	7.0276	6.2104	2.2075	2.4045	1.8039	1.8240
6	6.5691	5.9119	2.1725	2.1687	2.1508	1.6071
7	7.0084	6.3090	2.3123	2.1848	1.8477	1.9724
8	6.8299	5.9554	2.5437	2.4327	2.2522	1.6809
9	6.7315	6.0670	2.2190	2.3470	2.0092	1.7668
10	7.4726	6.4898	2.3001	2.2418	2.0086	1.7249

TABLE VII
CORRESPONDING COMPUTATIONAL TIME (s) FOR THE ALGORITHMS (A) TO SOLVE THE PCTPP WITH 120 TARGET LOCATIONS UNDER DIFFERENT INSTANCES (I)

I \ A	GA02	GA11	BNIA	FNIA	BMMA	FMMA
1	264.688	92.080	0.447	0.298	0.856	1.045
2	262.247	90.465	0.413	0.254	0.917	1.076
3	259.837	90.826	0.415	0.325	0.739	0.988
4	255.616	93.153	0.437	0.334	0.931	1.177
5	254.956	91.157	0.433	0.333	0.816	0.935
6	255.771	93.190	0.421	0.330	0.808	0.832
7	254.860	90.653	0.379	0.310	1.036	0.793
8	264.596	92.780	0.437	0.328	0.883	1.009
9	261.572	90.072	0.409	0.325	0.844	0.822
10	260.058	90.002	0.420	0.316	0.869	0.813

shown in Table III. Small computational time not only can alleviate the burden on equipping expensive computers but also enables the vehicle to quickly respond to the environmental changes such as the request to visit newly generated target locations.

Then, we test the algorithms on the problem with 120 target locations where every target location has only one precedence requiring it to be visited either before or after another target location as in the dial-a-ride problem [40]. For the simulation, ten instances of the targets' locations and the vehicle's initial position are randomly distributed in a square area with edge length 10^3 m. For each instance, we perform 20 trials of the GAs. The ratio r of the proposed algorithms and the average r of the GAs on each instance, and the average computational time are shown in Tables VI and VII. First, Table VI shows that the proposed algorithms BNIA, FNIA, BMMA, and FMMA perform better than the GAs since all the ratio r of the proposed algorithms are smaller than those of GA02 and GA11. Second, in Table VI, the ratio r of BNIA, FNIA, BMMA, and FMMA is in general around twice the optimal for all the instances, which again verifies the satisfying performance of the proposed algorithms. Furthermore, the forward algorithm FNIA (FMMA) is better than the backward algorithm BNIA (BMMA) as the latter generally has a larger r for most instances shown in Table VI. This is due to the direction on constructing the vehicle route as analyzed in the previous test scenario in which $n = 40$. Finally, FMMA is in general the best among the algorithms as it obtains the smallest r for most of the instances in Table VI. The Wilcoxon

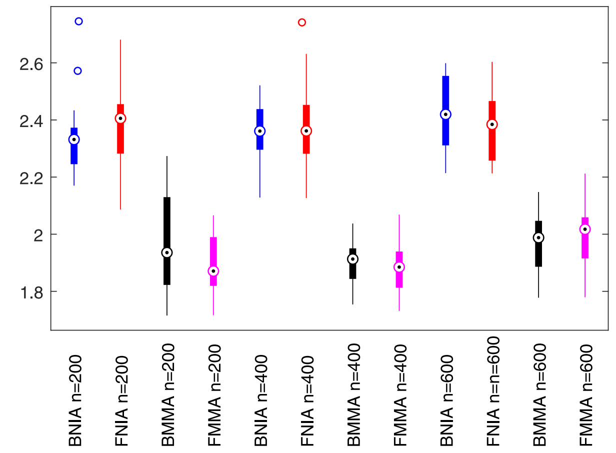

Fig. 5. Box plots for the solution quality r of the proposed algorithms with the number of target locations $n \in \{200, 400, 600\}$.

signed-rank test is also carried out in a two-tail test with the 5% significance level for each pair of the algorithms. It is clear that the ratio r of the ten instances differ significantly between the algorithms (r from left to right corresponds to FMMA $\rightarrow$ BMMA $\rightarrow$ FNIA $\rightarrow$ BNIA $\rightarrow$ GA11 $\rightarrow$ GA02). This implies that the algorithms have an increasingly better performance as GA02 – GA11 – BNIA – FNIA – BMMA – FMMA. The computational time of the proposed algorithms is still far smaller compared with those of the GAs according to Table VII. Comparing the computational times shown in Tables III and VII, one can conclude that the proposed algorithms are more scalable for the task planning problem compared with GA02 and GA13.

To further verify the performance of the proposed algorithms, we increase the number of the target locations to $n \in \{200, 400, 600, 800, 1000, 1200\}$ wherein each scenario every target location has only one precedence requiring it to be visited either before or after another target location. For each scenario, 20 instances of the targets' locations and the vehicle's initial position are randomly distributed in a square area with edge length 10^3 m. The box plots of the ratio r of the proposed algorithms and the average computational time are shown in Figs. 5–7, respectively. First, the box plots denoting the performance of BNIA and FNIA shown in Figs. 5 and 6 are comparatively taller than those of BMMA and FMMA with FMMA generally having the lowest box plots, which shows the better performance of BMMA and FMMA as those illustrated in Table VI. Second, with the increasing target number n, the box plots for the solution quality r of FMMA do not vary much, and are shorter in comparison with the other algorithms, suggesting that FMMA is more robust than the other algorithms. Third, with the increasing target number n, the box plots for the solution quality r of BMMA and FMMA are generally below 2, which implies that BMMA and FMMA can achieve near-optimal solutions for the challenging task planning problem. The average computational time of the algorithms shown in Fig. 7 increases as increasing the number of the target locations, where BMMA and BNIA require more time to calculate the solutions. However, it is still fast for the FMMA and FNIA to obtain the solutions for the challenging problem when the number of target locations is 1200.

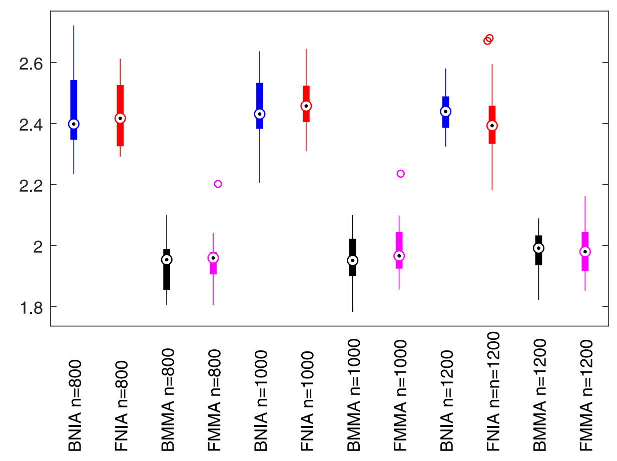

Fig. 6. Box plots for the solution quality r of the proposed algorithms with the number of target location $n \in \{800, 1000, 1200\}$.

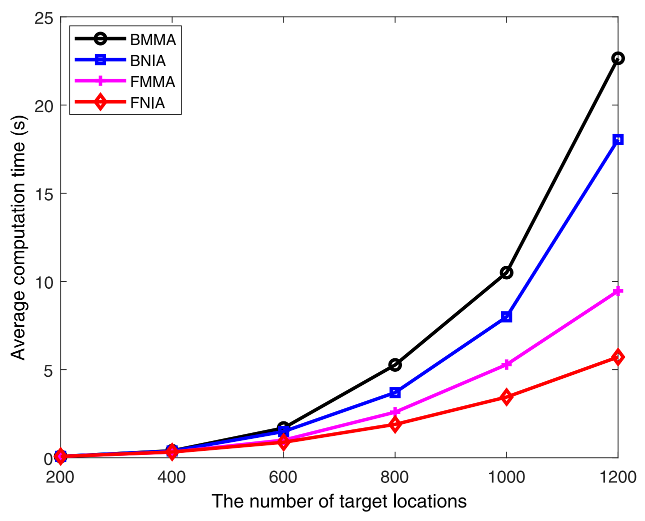

Fig. 7. Corresponding average computational time (s) for the proposed algorithms to achieve the solution to the PCTPP with different numbers of target locations.

VI. Conclusion

In this article, we have investigated the PCTPP, in which one vehicle needs to efficiently visit a set of target locations while satisfying the precedence constraints on visiting the targets. The problem has been shown to be NP-hard and a lower bound on the optimal solution has been found. Inspired by the existing TST, we have proposed a new topological sorting strategy, ensuring the generated solution to be feasible. Integrating these TSTs, four heuristic task planning algorithms have been designed. The simulation results have shown that the designed algorithms can achieve satisfying solutions to the PCTPP quickly in comparison with the existing competing algorithms. The proposed algorithms will be extended for the multivehicle task assignment with precedence constraints. Another research direction is to investigate the task planning problem with strict precedence constraints where some subsets of vertices must be visited in some prescribed order.

References

[1] R. J. Duro, M. Graña, and J. de Lope, "On the potential contributions of hybrid intelligent approaches to multicomponent robotic system development," *Inf. Sci.*, vol. 180, no. 14, pp. 2635–2648, 2010.

[2] W. Zhao, Q. Meng, and P. W. Chung, "A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario," *IEEE Trans. Cybern.*, vol. 46, no. 4, pp. 902–915, Apr. 2016.

[3] J. Turner, Q. Meng, G. Schaefer, A. Whitbrook, and A. Soltoggio, "Distributed task rescheduling with time constraints for the optimization of total task allocations in a multirobot system," *IEEE Trans. Cybern.*, vol. 48, no. 9, pp. 2583–2597, Sep. 2018.

[4] A. T. Hafez and M. A. Kamel, "Cooperative task assignment and trajectory planning of unmanned systems via HFLC and PSO," *Unmanned Syst.*, vol. 7, no. 02, pp. 65–81, 2019.

[5] D. Zhu, Y. Qu, and S. X. Yang, "Multi-AUV SOM task allocation algorithm considering initial orientation and ocean current environment," *Front. Inf. Technol. Electron. Eng.*, vol. 20, no. 3, pp. 330–341, 2019.

[6] X. Ge, Q.-L. Han, X.-M. Zhang, L. Ding, and F. Yang, "Distributed event-triggered estimation over sensor networks: A survey," *IEEE Trans. Cybern.*, vol. 50, no. 3, pp. 1306–1320, Mar. 2020.

[7] M. M. Flood, "The traveling-salesman problem," *Oper. Res.*, vol. 4, no. 1, pp. 61–75, 1956.

[8] J. Yu, S.-J. Chung, and P. G. Voulgaris, "Target assignment in robotic networks: Distance optimality guarantees and hierarchical strategies," *IEEE Trans. Autom. Control*, vol. 60, no. 2, pp. 327–341, Feb. 2015.

[9] S. L. Smith and F. Bullo, "Monotonic target assignment for robotic networks," *IEEE Trans. Autom. Control*, vol. 54, no. 9, pp. 2042–2057, Sep. 2009.

[10] P. Toth and D. Vigo, *Vehicle Routing: Problems, Methods, and Applications*. Philadelphia, PA, USA: Soc. Ind. Appl. Math., 2014.

[11] M. Burger, Z. Su, and B. De Schutter, "A node current-based 2-index formulation for the fixed-destination multi-depot travelling salesman problem," *Eur. J. Oper. Res.*, vol. 265, no. 2, pp. 463–477, 2018.

[12] E. L. Lawler *et al.*, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, vol. 3. New York, NY, USA: Wiley, 1985.

[13] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Comput. Oper. Res.*, vol. 31, no. 12, pp. 1985–2002, 2004.

[14] C. Ma, W. Liang, M. Zheng, and B. Yang, "Relay node placement in wireless sensor networks with respect to delay and reliability requirements," *IEEE Syst. J.*, vol. 3, no. 3, pp. 2570–2581, Sep. 2019.

[15] V. Raman and N. S. Gill, "Review of different heuristic algorithms for solving travelling salesman problem," *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 5, pp. 423–425, 2017.

[16] M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees," *Oper. Res.*, vol. 18, no. 6, pp. 1138–1162, 1970.

[17] Y. Caseau and F. Laburthe, "Solving small TSPS with constraints," in *Proc. Int. Conf. Logic Program. (ICLP)*, vol. 97, 1997, pp. 316–330.

[18] 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.

[19] T. S. Dahl, M. Matarić, and G. S. Sukhatme, "Multi-robot task allocation through vacancy chain scheduling," *Robot. Auton. Syst.*, vol. 57, no. 6, pp. 674–687, 2009.

[20] S. Belhaiza, "A game theoretic approach for the real-life multiple-criterion vehicle routing problem with multiple time windows," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1251–1262, Jun. 2018.

[21] P. J. Durst, C. T. Goodin, C. L. Bethel, D. T. Anderson, D. W. Carruth, and H. Lim, "A perception-based fuzzy route planing algorithm for autonomous unmanned ground vehicles," *Unmanned Syst.*, vol. 6, no. 04, pp. 251–266, 2018.

[22] B. Xin, Y. Wang, and J. Chen, "An efficient marginal-return-based constructive heuristic to solve the sensor–weapon–target assignment problem," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 12, pp. 2536–2547, Dec. 2019.

[23] J. Michallet, C. Prins, L. Amodeo, F. Yalaoui, and G. Vitry, "Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services," *Comput. Oper. Res.*, vol. 41, pp. 196–207, Jan. 2014.

[24] H.-J. Böckenhauer, T. Mömke, and M. Steinová, "Improved approximations for TSP with simple precedence constraints," *J. Discrete Algorithms*, vol. 21, pp. 32–40, Jul. 2013.

[25] C. Moon, J. Kim, G. Choi, and Y. Seo, "An efficient genetic algorithm for the traveling salesman problem with precedence constraints," *Eur. J. Oper. Res.*, vol. 140, no. 3, pp. 606–617, 2002.

[26] Y. Yun and C. Moon, "Genetic algorithm approach for precedence-constrained sequencing problems," *J. Intell. Manuf.*, vol. 22, no. 3, pp. 379–388, 2011.

[27] M. Kubo and H. Kasugai, "The precedence constrained traveling salesman problem," *J. Oper. Res. Soc. Jpn.*, vol. 34, no. 2, pp. 152–172, 1991.

[28] E. Balas, M. Fischetti, and W. R. Pulleyblank, "The precedence-constrained asymmetric traveling salesman polytope," *Math. Program.*, vol. 68, nos. 1–3, pp. 241–265, 1995.

[29] P. Oberlin, S. Rathinam, and S. Darbha, "Combinatorial motion planning for a dubins vehicle with precedence constraints," in *Proc. Dyn. Syst. Control Conf.*, 2009, pp. 715–722.

[30] X. Bai, W. Yan, and M. Cao, "Clustering-based algorithms for multivehicle task assignment in a time-invariant drift field," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2166–2173, Oct. 2017.

[31] X. Bai, W. Yan, S. S. Ge, and M. Cao, "An integrated multi-population genetic algorithm for multi-vehicle task assignment in a drift field," *Inf. Sci.*, vol. 453, pp. 227–238, Jul. 2018.

[32] X. Bai, M. Cao, W. Yan, and D. Xue, "Distributed multi-vehicle task assignment in a time-invariant drift field with obstacles," *IET Control Theory Appl.*, vol. 13, no. 17, pp. 2886–2893, Nov. 2019.

[33] X. Bai, M. Cao, and W. Yan, "Event-and time-triggered dynamic task assignments for multiple vehicles," *Auton. Robots*, to be published.

[34] X. Bai, M. Cao, W. Yan, and S. S. Ge, "Efficient routing for precedence-constrained package delivery for heterogeneous vehicles," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 1, pp. 248–260, Jan. 2020.

[35] A. J. D. Lambert, "Exact methods in optimum disassembly sequence search for problems subject to sequence dependent costs," *Omega*, vol. 34, no. 6, pp. 538–549, 2006.

[36] M. Charikar, R. Motwani, P. Raghavan, and C. Silverstein, "Constrained TSP and low-power computing," in *Proc. Workshop Algorithms Data Struct.*, 1997, pp. 104–115.

[37] S. G. Williamson, *Combinatorics for Computer Science*. North Chelmsford, MA, USA: Courier Corp., 1985.

[38] J. Edmonds, "Optimum branchings," *J. Res. Nat. Bureau Stand. B*, vol. 71, no. 4, pp. 233–240, 1967.

[39] M. A. Weiss, *Data Structures and Algorithm Analysis in C*. Redwood City, CA, USA: Benjamin-Cummings Publ., 1993.

[40] Y. Molenbruch, K. Braekers, and A. Caris, "Operational effects of service level variations for the dial-a-ride problem," *Central Eur. J. Oper. Res.*, vol. 25, no. 1, pp. 71–90, 2017.