mathematics

MDPI

Article

The Field Technician Scheduling Problem with Experience-Dependent Service Times

Vincent F. Yu [1], Yueh-Sheng Lin [1], Panca Jodiawan [1], Shih-Wei Lin [2,3,4,*] and Yu-Chi Lai [5]

[1] Department of Industrial Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan; vincent@mail.ntust.edu.tw (V.F.Y.); d10401001@mail.ntust.edu.tw (Y.-S.L.); pancajodiawan@gmail.com (P.J.)
[2] Department of Information Management, Chang Gung University, Taoyuan 333, Taiwan
[3] Department of Emergency Medicine, Keelung Chang Gung Memorial Hospital, Keelung 205, Taiwan
[4] Department of Industrial Engineering and Management, Ming Chi University of Technology, New Taipei 243, Taiwan
[5] Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan; cheeryuchi@gmail.com
* Correspondence: swlin@mail.cgu.edu.tw

Abstract: This research studies the Field Technician Scheduling Problem with Experience-Dependent Service Times (FTSP–EDST), involving three main features: matching maintenance tasks with available technicians, sequencing the tasks, and considering the experience-dependent service times. Given a limited number of technicians, the objective is to maximize the collected profit for servicing tasks. This study formulates the problem as a mixed-integer linear programming model and proposes a Modified Iterated Local Search (MILS) to solve the benchmark problem instances of various sizes. A set of FTSP–EDST instances is generated based on existing publicly accessible data, and MILS is utilized to solve these newly generated instances. Computational results confirm the effectiveness of MILS in solving FTSP–EDST.

Keywords: technician routing and scheduling; experience-dependent service time; Iterated Local Search; profit maximization

MSC: 90B06; 90C11; 90C59

check for updates

Citation: Yu, V.F.; Lin, Y.-S.; Jodiawan, P.; Lin, S.-W.; Lai, Y.-C. The Field Technician Scheduling Problem with Experience-Dependent Service Times. *Mathematics* **2023**, *11*, 4562. https://doi.org/10.3390/math11214562

Academic Editor: Paolo Mercorelli

Received: 6 October 2023
Revised: 30 October 2023
Accepted: 3 November 2023
Published: 6 November 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Power plant maintenance service providers face many challenges, one of which is that their customers are geographically dispersed. Due to limited resources, the providers must develop an effective plan to fulfill their customer needs. Dutot et al. [1] present a real-life-based problem from the telecommunication services of France Telecom. A number of technicians are grouped into teams to serve tasks within a given planning period. Technicians own a set of skills where each skill has a unique proficiency level. Each task requires a set of skills with certain proficiency levels. Consequently, not all tasks can be performed by any technician. Moreover, a routing task based on the well-known vehicle routing problem [2] is integrated. Considering the wide range of real-life applications, the resulting problem called the Technician Routing and Scheduling Problem (TSRP) is intriguingly complex.

There is no guarantee that all tasks can be performed in a given planning period due to resource constraints. In order to handle this issue, we follow Damm et al. [3] who propose an objective of maximizing the total priority of tasks performed and name it the Field Technician Scheduling Problem (FTSP). However, a limitation by Damm et al. [3] is that all technicians who are capable of performing a task have the same service time regardless of their proficiency levels, resulting in either overestimating or underestimating the total

time required to perform tasks assigned to technicians. The issue may further lead to the exploitation of technicians. In fact, each technician has a certain proficiency level in each skill domain, in turn leading to a different efficiency in handling a task [4]. A technician's efficiency is commonly associated with the time required to perform tasks, i.e., the higher the proficiency level, the shorter the service time required. Thus, we extend the problem addressed in Damm et al. [3] by explicitly considering the impact of proficiency levels in skill domains owned by available technicians toward service times to perform tasks. We hereafter name this the Field Technician Scheduling Problem with Experience-Dependent Service Times (FTSP–EDST).

We first formulate a mixed-integer linear programming model (compact formulation) for FTSP–EDST and develop a solution method based on a metaheuristic. We use the compact formulation in a MILP solver, which is commonly incapable of solving large-scale problems, yet real-world problems generally vary in size. Both the literature and our experimental works further described in Section 5.4 confirm the effectiveness of a metaheuristic-based solution method compared to commercial exact solvers commonly utilized for solving related problems [3,5–7]. To sum up, our contributions are as follows:

1. We investigate a new variant of FTSP called the Field Technician Scheduling Problem with Experience-Dependent Service Time (FTSP–EDST). The problem is inspired by the fact that each technician may require a different amount of service time, depending on his/her skill levels.
2. We formulate a mixed-integer linear programming model for FTSP–EDST. The model is extended from the model for FTSP developed by Damm et al. [3].
3. A scalable solution approach based on the Iterated Local Search (ILS) is developed to provide high-quality solutions for FTSP–EDST. Several modifications and adjustments are implemented to tackle underlying problem characteristics, resulting in the development of the Modified Iterated Local Search (MILS) metaheuristic.

The rest of this paper runs as follows. Section 2 provides a non-exhaustive review of TSRP and its closely related problem, TOPTW. Section 3 formally describes FTSP–EDST and its mathematical formulation. Section 4 explains the development of the metaheuristic-based solution method in detail. Section 5 presents a series of computational results. Finally, Section 6 offers the conclusion of our work.

2. Literature Review

2.1. Technician Scheduling and Routing Problem

Dutot et al. [1] propose a technician and task scheduling problem (TTSP) based on a real-life problem in a large telecommunication company. Each technician possesses a set of skill domains, and each task has a certain priority and requires a set of skills at certain levels. Outsourcing tasks is possible as long as it is still within the total budget given. The technicians are grouped into teams, and the aim of TTSP is to produce an assignment of technician teams to service tasks with a minimized weighted makespan of all priorities. Kovacs et al. [8] extend TTSP by considering the routing aspect for servicing tasks, calling it technician scheduling and routing problem (TSRP). Its objective is to design routes for vehicles with a given number of technicians with different levels of skills and a number of tasks requiring different levels of skills and time windows to be served. Matching technicians to tasks adds complexity to the design of cost-efficient routes. Besides the variant with team building (i.e., allowing one or more technicians in one vehicle due to a specific condition in which most of the tasks cannot be conducted by only one technician), another variant without team building is also addressed. Similar to TTSP, an outsourcing option is available. However, the outsourcing cost is integrated as a part of an objective function instead of being taken into account as a constraint.

The high practicability of TSRP has driven investigations into TSRP variants. Zamorano and Stolletz [9] address an extension of TSRP with team building by considering a wider planning horizon—i.e., one week. It aims to produce weekly schedules that minimize total operational costs involving traveling costs, waiting time of customers, and technician

overtime. Each task may have multiple time windows, depending on the flexibility of the task. As the plan expands over multiple days, the composition of technicians in every team may differ every day.

Chen et al. [4] also extend the TSRP with multiple planning periods by explicitly considering the experience-based learning of technicians. In particular, a technician gains a certain experience associated with the task serviced, leading to a reduced service time for servicing a task requiring the same skills. Instead of integrating the planning into a wider planning horizon as conducted by Zamorano and Stolletz [9], a rolling horizon approach is utilized—i.e., solving TSRP in every period considered—due to the assumption in which demands become known at the beginning of a day.

Mathlouthi et al. [10] give another variant arising from the repair and maintenance services of electronic transaction equipment. Each technician departs from their origin, such as home, to visit customers, in turn forming a variant of TSRP without team building. Each customer requires one or more spare parts, and some customers may need special parts. Each technician is allowed to visit a depot consisting of spare parts and special parts required by customers, thus increasing the number of customers that can be served. The objective of the problem proposed is to maximize the total gains resulting from servicing customers.

Pourjavad and Almehdawe [11] extend TSRP without team building by considering multiple days and overnight, i.e., a technician returns to the depot after two or more days. This consideration arises due to the wide range of working areas. The objective of the problem is to minimize total operational costs consisting of traveling costs, technicians' costs, and overnight costs. Analyses are conducted to show which scenarios, like with or without overnight, result in a better objective.

Due to increasing concern regarding the negative environmental impact caused by the transportation sector, Villegas et al. [5] address a variant of TSRP without team building by considering a mixed fleet between conventional vehicles and electric vehicles (EVs). To realistically model the problem, the driving style of each technician is taken into account, leading to different energy consumptions of EVs. Moreover, recharging stations with multiple technologies are provided for EVs to recharge, and a non-linear function for recharging is used to approximate realistic recharging times.

Exact algorithms and metaheuristics are the most common solution approaches to be developed for tackling TSRP. Regarding the field of exact algorithms, Branch-and-price, developed by Barnhart et al. [12], is the most popular algorithm [9,13]. The main limitation of the exact algorithm is the problem size to be solved. Zamorano and Stolletz [9] propose a branch-and-price that can solve TSRP, encompassing multiple planning periods, 25 tasks, and 5 days, while Mathlouthi et al. [13] develop a similar type of exact algorithm that can provide solutions for up to 45 tasks.

Metaheuristics are well known due to their capability to provide feasible solutions in reasonable times. Local-search-based heuristics that generate a new solution by applying local search neighborhood moves have been developed, e.g., record-to-record algorithm [4] and Tabu Search [10]. Kovacs et al. [8] propose another type of metaheuristic, Adaptive Large Neighborhood Search, which is based on a principle of destroying and repairing a solution over iterations.

Pekel [14] develops Improved Particle Swarm Optimization by integrating local search operators into the original Particle Swarm Optimization and shows the superiority of such integration. Pekel [15] modifies a basic Genetic Algorithm in a similar way, naming it Improved Genetic Algorithm, and shows that it is superior compared to the method proposed in Pekel [14]. Villegas et al. [5] give a two-phase parallel matheuristic by iteratively solving the well-known set-covering problem [16]. It produces a new solution after building a pool of routes by implementing a greedy randomized adaptive search procedure in a parallel fashion.

2.2. Team Orienteering Problem

A closely related problem to TSRP is the Team Orienteering Problem (TOP). Here, a route maximizes total scores collected from the visited places given a limited amount of time. The application can be found in the areas of sport, home fuel delivery, and technician routing [17].

Tang and Miller-Hooks [18] pioneer the implementation of TOP in the field of technician routing by identifying a similar condition, i.e., each technician needs to maximize the number of tasks to be finished given specific working hours. However, they assume that all technicians have the appropriate skills to perform all tasks. Damm et al. [3] offer a variant of TSRP called FTSP. Unlike Tang and Miller-Hooks [18], Damm et al. [3] explicitly consider skill domains owned by each technician and time windows for performing tasks into the problem. The objective of Damm et al. [3] is to maximize the total priorities gained by servicing tasks. In a similar fashion, time windows are incorporated in TOP by Vansteenwegen et al. [17], forming a new variant called Team Orienteering Problem with Time Windows. While FTSP explicitly considers the skill domains of technicians, TOPTW has no such assumptions. Consequently, TOPTW is a special case of FTSP where all technicians are able to perform all tasks.

TOPTW has shown interesting computational complexity, drawing attention from the research community to develop various metaheuristics that provide high-quality solutions to the problem. Vansteenwegen et al. [17] present the first metaheuristic based on Iterated Local Search to provide solutions in a significantly short time. Since then, various metaheuristics and matheuristics have been developed to improve the solution quality. On the one hand, well-known metaheuristics employing a single solution and utilizing local search neighborhood moves to produce new solutions involve Simulated Annealing [19], Variable Neighborhood Search [20], and Iterated Local Search [17,21]. On the other hand, population-based metaheuristics commonly developed for TOPTW are ant colony [22], artificial bee colony algorithm [6,23], and discrete particle swarm optimization [7]. Hu and Lim [24] propose a matheuristic containing various local-search-based moves, a Simulated Annealing acceptance criterion, and an integer programming model for combining routes.

3. Problem Description and Formulation

We define FTSP–EDST on a directed graph $G = (V, A)$, where $V = \{0, \ldots, N\}$ denotes the set of nodes comprising the depot and the tasks, and $A = \{(i,j) | i \in V, j \in V, i \neq j\}$ represents the set of arcs connecting nodes in V. In addition, we define $V' = \{1, \ldots, N\}$ as the set of tasks to be serviced. A set of technicians T is available at the depot to serve tasks. The traveling time from node i to node j is d_{ij}.

We define S as the set of skill domains and L as the set of proficiency levels. On the one hand, each technician $t \in T$ possesses a subset of skills with certain proficiency levels defined in a binary matrix $\left[p^t_{ls}\right]_{|L|\times|S|}$. If technician t has a proficiency level l for skill domain s, then $p^t_{ls} = 1$. This also means that $p^t_{l's}$ is 1 for every $l' < l$. On the other hand, each task $j \in V\prime$ requires a subset of skills with certain proficiency levels defined in an integer vector $\left[q^j_s\right]_{|S|}$. Technician $t \in T$ is able to perform task $j \in V'$ if $p^t_{ls} = 1$, where $l = q^i_s$ for all $s \in S$. To simplify the mathematical model, we rewrite the information contained in matrix $\left[p^t_{ls}\right]_{|L|\times|S|}$ and vector $\left[q^j_s\right]_{|S|}$ into a binary parameter λ_{jt}, where $j \in V'$ and $t \in T$ contain the information for whether technician t is capable of servicing task j.

Each task $j \in V'$ has a profit ρ_j and time windows $\left[e_j, l_j\right]$. We define (e_0, l_0) as the depot opening and closing times. The service time of task j depends on the skills possessed by the technician who performs the task. We define a service time matrix $[z_{ll'}]_{|L|\times|L|}$, where $z_{ll'}$ denotes the service time of a technician who has a skill domain with proficiency level l to perform a task requiring a skill domain with proficiency level $l\prime$. In order to calculate the service time r_{tj} of technician t for serving task j, we sum up the service times from all skill domains required by the task based on matrix $[z_{ll'}]_{|L|\times|L|}$. A technician's route starts

from the depot, visits a sequence of tasks that can be performed based on λ_{jt}, and returns to the depot.

The decision variables are described as follows: x_{ijt} denotes a binary variable that tracks whether technician t traverses from node i to node j, y_{jt} denotes a binary variable tracking the assignment of task j to technician t, and τ_i represents the service start time at node i. The objective of FTSP–EDST is to create routes for available technicians that maximize profit collected from finishing tasks.

Figure 1 illustrates a solution for FTSP–ES. Two technicians are presented with 10 tasks. There are 4 tasks that can only be performed by the first technician (T1, T3, T4, and T10), 3 tasks that can only be performed by the second technician (T2, T5, and T6), and 3 tasks that can be performed by either one of two technicians (T7, T8, and T9). Among all tasks, two tasks remain unserved, i.e., T1 and T6.

$$\textbf{Max } \sum_{i\in V',\, t\in T} \rho_i y_{it} \tag{1}$$

Subject to

$$\sum_{t\in T|\ \lambda_{it}=1} y_{it} \le 1 \qquad i \in V' \tag{2}$$

$$\sum_{j\in V|i\neq j,\ \lambda_{jt}=1} x_{ijt} = \sum_{j\in V|i\neq j,\ \lambda_{jt}=1} x_{jit} \qquad t \in T,\ i \in V',\ \lambda_{it} = 1 \tag{3}$$

$$\sum_{j\in V|i\neq j,\lambda_{jt}=1} x_{ijt} = y_{it} \qquad t \in T,\ i \in V',\ \lambda_{it} = 1 \tag{4}$$

$$\sum_{i\in V'|\lambda_{it}=1} x_{0it} \le 1 \qquad t \in T \tag{5}$$

$$\tau_i + r_{ti} + d_{ij} \le \tau_j + M\left(1 - x_{ijt}\right) \qquad t \in T,\ i \in V, j \in V',\ i \neq j,\ \lambda_{jt} = 1,\ \lambda_{it} = 1 \tag{6}$$

$$e_i \sum_{t\in T}\ y_{it} \le \tau_i \le l_i \sum_{t\in T} y_{it} \qquad i \in V' \tag{7}$$

$$\tau_i + r_{ti} + d_{i0} \le l_0 \qquad t \in T,\ i \in V',\ \lambda_{it} = 1 \tag{8}$$

$$y_{it} = 0 \qquad t \in T,\ i \in V',\ \lambda_{it} = 0 \tag{9}$$

$$x_{ijt} \in \{0,1\} \qquad \forall i, j \in V, t \in T \tag{10}$$

$$y_{it} \in \{0,1\} \qquad \forall i \in V',\ t \in T \tag{11}$$

$$\tau_i \in R^+ \qquad \forall i \in V \tag{12}$$

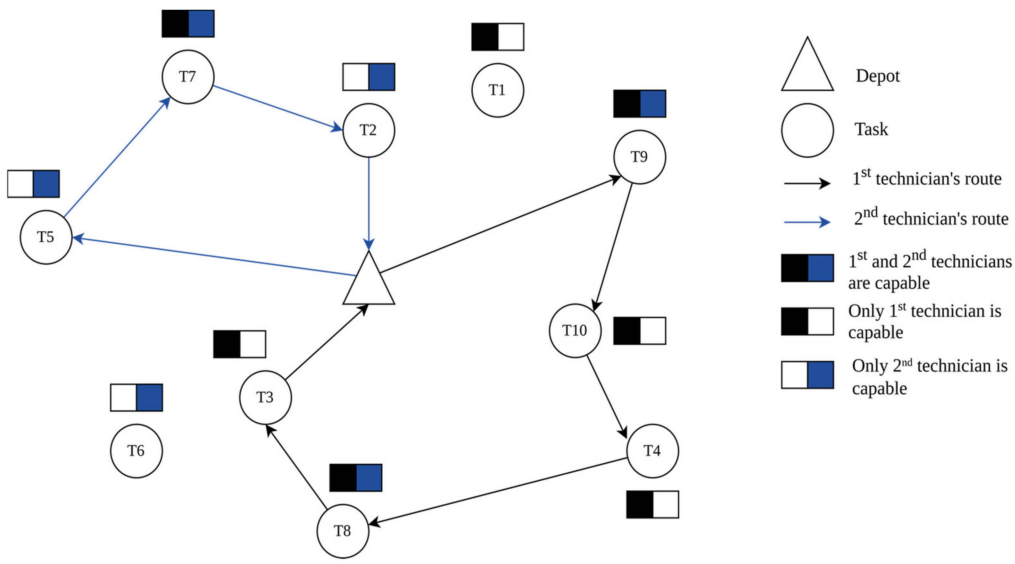

Figure 1. A solution illustration for FTSP–EDST.

Objective function (1) aims to maximize the total profits collected by all employed technicians. Constraint (2) ensures that a task can only be performed by a capable technician. Constraint (3) guarantees flow conservation. Constraint (4) connects the flow variables with the assignment variables. Constraint (5) ensures that each technician only performs at most one trip. Constraint (6) tracks the service start time at visited tasks. Constraint (7) ensures that each task is serviced in its time window. Constraint (8) guarantees that each technician returns to the depot at the upper-bound time windows of the depot at the latest. Constraint (9) ensures that an assignment of a task to a technician cannot be

performed if the technician is incapable. Constraints (10)–(12) define the natural ranges of decision variables.

4. Modified Iterated Local Search

Iterated Local Search has been widely used for providing solutions to various optimization problems, like logistics and transportation [25], production scheduling [26], nurse rostering [27], and school or university timetabling problems [28]. ILS works on a single solution by iteratively applying a set of local search operators, i.e., operators creating a neighborhood solution by changing the structure of a given solution. In order to escape from the local optima, ILS commonly implements a perturbation procedure after executing local search operators by randomly changing the structure of a given solution [29].

Our algorithm is inspired by the works of Vansteenwegen et al. [17] and Labadie et al. [30] for solving TOPTW. In particular, Vansteenwegen et al. [17] propose the first ILS with a mechanism for inserting unvisited locations. Several locations currently visited in the solution are removed, and the insertion mechanism is implemented as an effort to improve the profit as much as possible. Labadie et al. [30] implement local search operators existing in the literature for reducing the total traveling time before applying their proposed insertion mechanism to improve the chance of obtaining a solution with higher profit. These two mechanisms have lately been widely adopted for basic components of various heuristics proposed for solving TOPTW [21,24].

TOPTW is a special case of FTSP–EDST as mentioned in Section 2.2. Thus, we develop a Modified Iterated Local Search by including local search operators and insertion mechanisms with three modifications. First, we implement a straightforward constant time feasibility check procedure regarding compatibility between technicians and tasks and an efficient time windows feasibility check procedure adopted from Nagata et al. [31]. The first checking procedure can be realized by storing the information in a matrix of binary parameters indicating the compatibility between technicians and tasks. Second, we implement the acceptance criterion of the Simulated Annealing heuristic as a complementary mechanism to the perturbation procedure of Iterated Local Search (ILS). Third, we integrate noise into the insertion mechanism described in Section 4.3 to help the algorithm escape from the trap of local optima.

4.1. Generating an Initial Solution

The solution representation we develop for FTSP–EDST consists of two types of information, technicians' routes and auxiliary information, which enable an efficient time windows feasibility check with $\mathcal{O}(1)$ time complexity for several classical operators, which we describe in Section 4.3. We refer to Nagata et al. [31] for the detailed description of the required auxiliary information and the efficient time windows feasibility check procedure.

The Initial solution serves as the input for our MILS. The procedure iteratively inserts a task resulting in the least change in traveling time into a set of technicians and terminates when no task is feasible to be inserted into the solution.

4.2. Modified Iterated Local Search for Solving FTSP–EDST

Our proposed MILS described in Algorithm 1 requires five parameters: η_{max}, β, γ, α, and η_{loop}. After generating an initial solution, the temperature T required for the Simulated Annealing acceptance criterion is calculated in Line 2. The calculation shows that T is initially set into a value such that a solution, which is $1 + \beta$ worse than the initial solution, is accepted with a probability of γ. We set the best solution so far as S^* and the neighborhood solution S^w to be the same as the incumbent solution S.

Two loops are nested in our proposed MILS. The first loop described in Lines 5–19 aims to generate new neighborhood solutions in an attempt to find solutions of better quality. At the same time, T is gradually reduced by a multiplier α, and a termination criterion is checked in the second loop, i.e., the maximum number of consecutive non-improving first loops η_{max}.

The first loop starts with the implementation of a local search procedure to S^w (Section 4.3). If the updated S^w has an objective (i.e., total collected profit) higher than S, then S is replaced by S^w. Furthermore, S^* is replaced by S^w if S^w produces a better objective than the current S^*. However, S^w is not immediately rejected if the resulting objective is lower than that of S. The Simulated Annealing acceptance criterion is implemented. In particular, S^w has a probability of $e^{\frac{obj(S^w)-obj(S)}{T}}$ to be accepted.

The perturbation procedure *shake*( ) described in Line 18 is utilized to randomly change the structure of S^w. The procedure takes i_s as an input and removes i_s consecutive tasks visited in every technician's route. The starting position of a task to be removed is randomly determined, and the position returns to the beginning of the processed route if no task can be removed further at the current position. The value of i_s is adjusted in either Line 11 into its default value or Line 19. Furthermore, *adjustValue*( ) increases the value of i_s by 1 every 2 iterations of the first loop and adjusts the value into 1 when it is greater than the number of tasks visited in the longest route. Finally, the first loop's routine iterates up to η_{loop}.

Algorithm 1 Pseudocode for MILS

Input: $\eta_{max}, \beta, \gamma, \alpha, \eta_{loop}$
Output: S^*

```
1   S ← InitialSol( )
2   T ← −β(obj(S^w)) / ln(γ)
3   i_n ← 0, i_s ← 1, S^w ← S, S* ← S
4   while i_n < η_max do
5       for i_loop = 1 to η_loop
6           S^w ← localSearch( )
7           if obj(S^w) > obj(S) then
8               S ← S^w
9               if obj(S^w) > obj(S*) then
10                  S* ← S^w
11                  i_n ← 0, i_s ← 1
12          else
13              r ← U(0,1)
14              if (r < e^((obj(S^w)−obj(S))/T)) then
15                  S ← S^w
16              else
17                  S^w ← S
18          S^w ← shake(i_s)
19          i_s ← adjustValue( )
20      T ← αT, i_n ← i_n + 1
21  return S*
```

4.3. Neighborhood Moves

The local search in MILS contains two phases: (1) traveling time reduction and (2) profit improvement. The first phase has three neighborhood moves described as follows:

1. Relocate: inserting a task into a new position.
2. Exchange: swapping the positions of two tasks.
3. 2-opt: removing and replacing two arcs in a route with re-ordering the tasks visited in between the arcs.

Two variants are proposed for both relocate and exchange moves: intra-route (only involving one route) and inter-route (involving two routes) operators, while 2-opt only

works as an intra-route operator. The implementation sequence for these operators is (1) relocate-inter, (2) exchange-inter, (3) relocate-intra, (4) exchange-intra, and (5) 2-opt.

The second phase consists of two neighborhood operators: (1) insertion and (2) replace. First, the insertion operator iteratively inserts a task selected based on a particular criterion, i.e., the highest ratio of squared profit value and the traveling time change resulting from the insertion of the task [17]. Second, the replace operator selects and replaces a visited task with an unvisited task in an attempt to further increase the total profit collected. In addition, we add noise to the measurement metric used by the insertion and replacement operators, i.e., we multiply the measured value with a value randomly generated from $U(0.9, 1.1)$.

The $\mathcal{O}(1)$ time windows feasibility checking procedure is implemented on first-phase inter-route operators and second-phase operators. Next, the classical time windows feasibility checking procedure with a non-constant time complexity (depending on a route's length) is utilized for first-phase intra-route operators.

Each operator in both phases iteratively checks a feasible combination and utilizes the first improvement strategy, i.e., the operator terminates its search once a neighborhood solution with a lower traveling time is found. Algorithm 2 summarizes the local search of our MILS, and Algorithm 3 describes the second phase of the local search. The improvement described in Line 5 depends on which neighborhood move is executed: (1) total traveling time reduction for neighborhood moves that belong to the first phase ($N_i, i = 1, \ldots, 5$) and (2) increased total profit for the second phase (N_6).

Algorithm 2 *localSearch*( )

Input: S^w and the set of neighborhood moves $N_i, i = 1, \ldots, 6$
Output: S^w

```
1   i_ls ← 1
2   while i_ls ≤ 6 do
3       improved ← false
4       S' ← N_i(S^w)
5       if improvement is found then
6           S^w ← S'
7           i_ls ← 1
8       else
9           i_ls ← i_ls + 1
10  Return S^w
```

Algorithm 3 Pseudocode for second phase of local search, N_6

Input: S^w, Insertion and Replace operators
Output: S^w

```
1   S' ← Replace(Insertion(S^w ))
2   if Obj(S') > Obj(S^w) then
3       S^w ← S'
4   return S^w
```

5. Computational Results

The proposed MILS is coded in C++ programming language, and MILP described in Section 3 is written in AMPL with CPLEX as the solver. Both MILS and MILP are executed in a computer with an intel Core i7-11700 2.5 GHz CPU. Since FTSP–EDST is a new problem, a set of new benchmark instances is generated and described in Section 5.1. We conduct parameter tuning for our proposed algorithm and present the results in Section 5.2. Section 5.3 gives the computational results related to TOPTW's existing benchmark instances. We compare the results obtained by our algorithm with the solution method

obtained by the literature. Section 5.4 finally presents the results of solving FTSP–EDST with both our algorithm and MILP.

5.1. Generating FTSP–EDST Instances

A set of new FTSP–EDST instances is generated based on publicly available instances provided by TSRP instances [8]. The TSRP instances are selected from the well-known Solomon VRPTW instances [32]. In particular, 12 instances are chosen: C101, C103, R101, R103, RC101, RC103, C201, C203, R201, R203, RC201, and RC203. Each instance consists of one depot and 100 tasks. Euclidean distance measures the travel time between two locations. Each VRPTW instance generates three TSRP instances with different numbers of skill domains and proficiency levels: 5×4 (in the form $|S| \times |L|$), 6×6, and 7×4 matrices. We use the "no-team" version in which each task only requires one technician.

From each TSRP instance, we generate one FTSP–EDST instance. First, we limit the number of technicians by selecting a certain number of available technicians in TSRP instances and turn the demand information provided in a TSRP instance into the profit obtained for serving tasks. Second, we generate the experience-dependent service time by utilizing Tables 1 and 2, depending on how many proficiency levels exist in an instance. Both Tables 1 and 2 contain a set of multipliers that depend on two factors: the proficiency level required to finish a task (R_i) and the proficiency level possessed by a technician (A_i). Given a TSRP instance, the steps for generating a processing time for performing a task based on skill proficiency possessed by a technician are described below.

1. Determine μ the maximum number of skill domains required among all tasks.
2. Divide the actual service time of each task by μ.
3. Generate a new service time for each task depending on the proficiency levels of skill domains possessed by each technician. We use Tables 1 and 2 for this purpose. For example, given an instance with four proficiency levels and a task requiring two skill domains, one of them requires a proficiency level of R_1, and another requires R_2. A technician possesses these skill domains, both with proficiency levels of A_3. Let the value obtained from step 2 equal 5 min. Based on the given information and Table 1, the technician requires $0.8(5) + 1.0(5) = 9$ min to finish the task. Similar mechanisms apply for an instance with six proficiency levels.

Table 1. Experience-dependent service times per skill domain for four proficiency levels.

	R_1	R_2	R_3	R_4
A_1	1.2	-	-	-
A_2	1.0	1.2	-	-
A_3	0.8	1.0	1.2	-
A_4	0.6	0.8	1.0	1.2

Table 2. Experience-dependent service times per skill domain for six proficiency levels.

	R_1	R_2	R_3	R_4	R_5	R_6
A_1	1.5	-	-	-	-	-
A_2	1.25	1.5	-	-	-	-
A_3	1	1.25	1.5	-	-	-
A_4	0.9	1	1.25	1.5	-	-
A_5	0.8	0.9	1	1.25	1.5	-
A_6	0.7	0.8	0.9	1	1.25	1.5

We create three groups of FTSP–EDST instances based on problem size: small, medium, and large. The numbers of tasks (technicians) involved in small, medium, and large instances are 50 (3), 75 (4), and 100 (5), respectively. Each group has 36 instances, and thus we have 108 newly generated FTSP–EDST instances in total. The generated instances can be downloaded at http://swlin.cgu.edu.tw/data/FTSP.zip (accessed on 5 October 2023).

5.2. Parameter Tuning

This section presents a detailed description of parameter tuning for our MILS. We randomly select five TOPTW instances originally developed by Cordeau et al. [33]. For each instance, we use four combinations of the number of vehicles, i.e., 1 to 4. In total, we utilize 20 TOPTW instances.

One Factor at A Time (OFAT) is utilized to obtain a good combination of MILS parameters. Table 3 summarizes candidates for its parameters. The mechanism iteratively selects one parameter to be tuned while fixing the values of remaining parameters. We use average best solutions over selected instances as the measurement metric for selecting the best value for each tested parameter. However, we also pay attention to the increase in computational time while using the aforementioned metric.

Table 3. Candidates for tuning MILS parameters.

Parameter	Description	Values
η_{max}	Maximum non-improving iterations	100, **125**, 150, 175
(β, γ)	Multipliers for initial temperature	(**0.5**, **0.5**), (0.25, 0.5), (0.25, 0.25), (0.5, 0.25)
α	Multiplier for temperature reduction	**0.95**, 0.90, 0.85, 0.80
η_{loop}	Number of inner-loop iterations	30, 40, **50**, 60

Bold values indicate the selected parameter values after performing OFAT.

Figure 2 presents the impact of changing parameters' values toward both average best solutions (Avg.) and average computational time (CPU) in seconds (s). Based on Figure 2a, the best candidate in terms of solution quality is 175. However, we select 125 because it provides good solution quality with less computational effort. Based on Figure 2d, we also use a similar argument for selecting the value of η_{loop}. Finally, the selected combination of MILS parameters $\left(\eta_{max}, \beta, \gamma, \alpha, \eta_{loop}\right)$ is $(125, 0.5, 0.5, 0.95, 50)$.

5.3. MILS Performance on Solving TOPTW Benchmark Instances

MILS is utilized to provide a solution to TOPTW benchmark instances. We evaluate the performance of our MILS by following four types of TOPTW benchmark instances. The four types of TOPTW benchmark instances, which include Solomon 100 (29 instances), Solomon 200 (27 instances), Cordeau 1–10 (10 instances), and Cordeau 11–20 (10 instances), are provided by Vansteenwegen et al. [17]. We execute our algorithm five times and compare the results with solutions provided by metaheuristics found in the literature: ILS [17], VNS [20], SSA [19], GVNS [30], I3CH [24], ABC [23], SAILS [21], and HABC [6].

Table 4 summarizes the comparative results. The performance of each algorithm is evaluated by two metrics: the gap between the Best Known Solutions (BKSs) and solutions obtained by an algorithm Gap (%) compiled from the previously mentioned metaheuristics and the proposed MILS and computational time CPU (s). Gap (%) is calculated using Equation (13), where S represents the average solution quality provided by an algorithm. Table 4 presents the average gap (Avg) and the maximum gap (Max) from solving a group of instances.

$$\text{Gap}(\%) = \frac{\text{BKS} - S}{\text{BKS}} \times 100\% \tag{13}$$

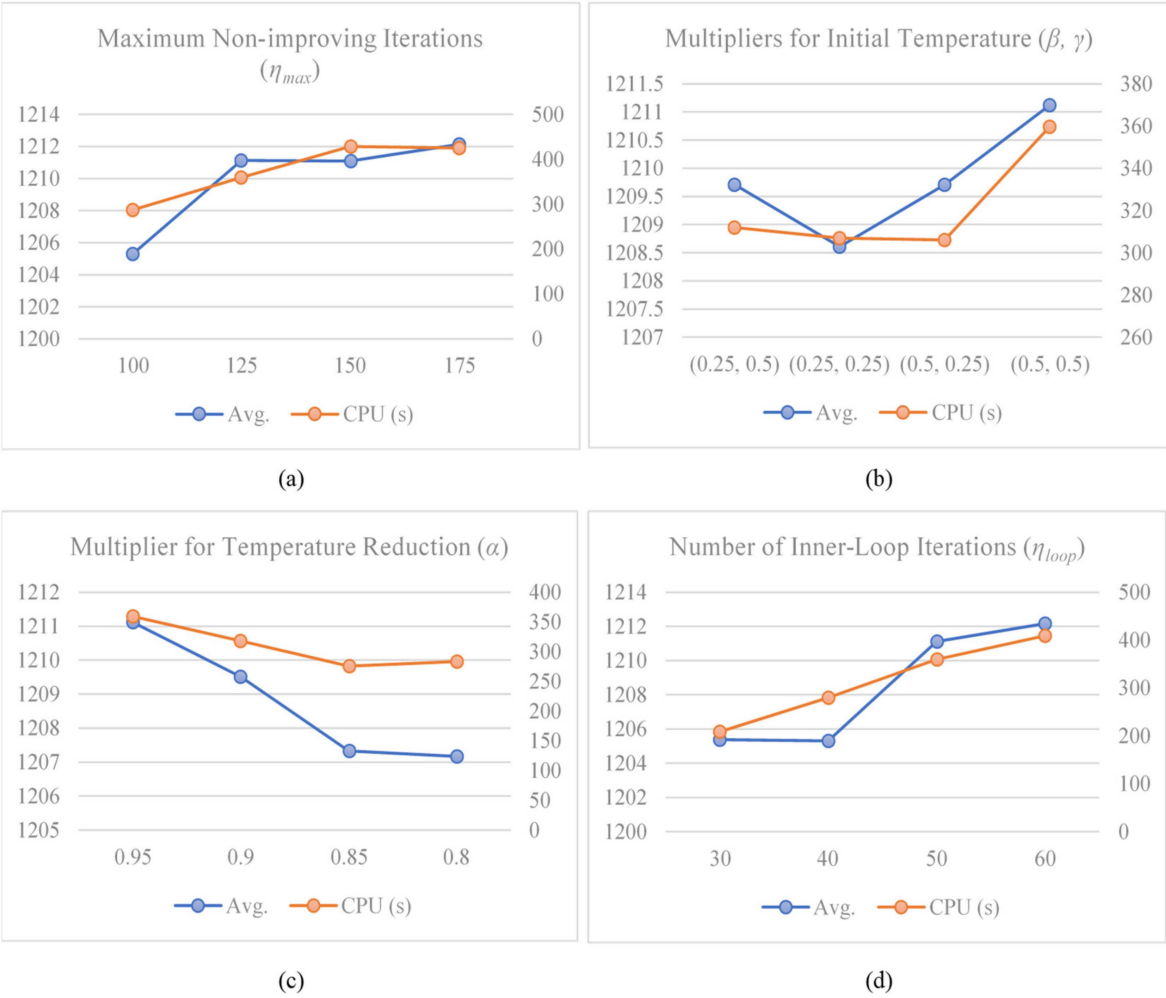

Figure 2. Summary of OFAT experimental results.

Our algorithm obtains a reasonably good solution quality. Avg ranges from 0 to 1.97%, while Max ranges from 0 to 6.44%. Our algorithm outperforms the results of the aforementioned methods in terms of solution quality for several groups of instances: Solomon 200 #vehicle 1 (Avg and Max), Cordeau 11–20 #vehicle 1 (Avg), Solomon 200 #vehicle 2 (Avg and Max), Cordeau 1–10 #vehicle 2 (Avg and Max), Cordeau 11–20 #vehicle 2 (Avg and Max), and Cordeau 11–20 #vehicle 3 (Max). Two other methods that show their superiority are SAILS [21], which performs well in the group of Solomon 100, and I3CH [24], which obtains high-quality solutions in all instances with a larger number of vehicles (#vehicle = 3 and 4).

Vansteenwegen et al. [17] provide solutions to TOPTW benchmark instances under a significantly short computational time with worse solution quality as an obvious trade-off. The average computational time of our algorithm is comparable with other methods in the group of Solomon 100, Cordeau 1–10, and Cordeau 11–20, while it takes more computational time for the group of Solomon 200. The experimental results on TOPTW benchmark instances show that our MILS is competitive with other methods, and thus we further utilize the algorithm for solving FTSP–EDST.

5.4. MILS Performance on Solving FTSP–EDST Instances

Table 5 presents the results of solving three groups of FTSP–EDST instances. In order to assess the performance of MILS, we execute MILP, as described in Section 3, and compare the obtained results with results provided by MILS. The results are dissected into two main columns, solution quality "objective value" and computational time in seconds "CPU (s)"

with sub-columns representing the group of instances. Regarding "objective value", we run our MILS five times and record the best solution (MILS*) and the average solution (MILS). Regarding MILP, we set 7200 s as the time budget for CPLEX. An additional set of columns representing gaps between solutions (best and average) obtained by MILS and CPLEX is added to the table to highlight the performance of our MILS.

CPLEX can provide feasible solutions to all instances. In particular, CPLEX can obtain 28, 17, and 14 optimal solutions for small, medium, and large instances, respectively, while the remaining solutions are feasible yet not proven optimal because of time limitations. The average computational times for small, medium, and large instances are 1898.40 s, 4093.29 s, and 4817.60 s, respectively. The bigger the problem size is, the longer time CPLEX will require to provide a solution, and the capability for providing high-quality solutions also decreases. On the other hand, MILS obtains all feasible solutions for small, medium, and large instances with average computational times of 8.09 s, 31.08 s, and 71.27 s, respectively.

The gaps in Table 5 are obtained by comparing the results obtained by MILS and those of CPLEX. The zero gap indicates that the solution obtained by both CPLEX and MILS are of the same quality, while the positive gap shows that the MILS provides a better solution compared to that of CPLEX. The average gaps in Table 5, ranging from 2.86% to 10.46%, clearly show the superiority of MILS over CPLEX for each tested dataset. In particular, our MILS provides 8, 17, and 21 better solutions compared to that of CPLEX for small, medium, and large datasets, respectively. The average solution qualities of MILS* and MILS show that our MILS provides reasonably robust solutions to each FTSP–EDST dataset. In conclusion, our MILS performs well in solving the newly generated FTSP–EDST instances of different problem sizes.

To show the convergence trend of the proposed approach, two large benchmark problems, C103_6x6 and RC103_7x4, are solved, and the obtained objective function values and iteration numbers are shown in Figure 3. As can be seen, the improvement of the solution decreases as the iteration number increases, and after a certain number of iterations, the solution does not improve. Thus, a larger number of iterations may not improve the quality of the solution.

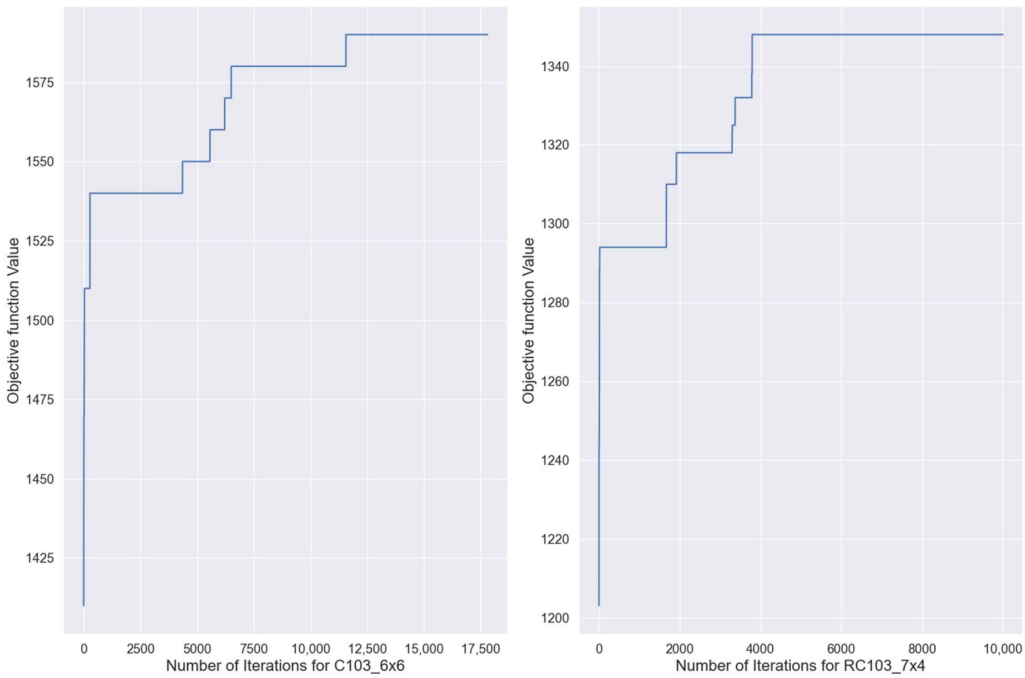

Figure 3. Evolution of best solutions obtained for C103_6x6 and RC103_7x4.

Table 4. Summary of comparative results on TOPTW benchmark instances.

	Set	ILS		VNS		SSA		GVNS		I3CH		ABC		SAILS		HABC		MILS	
		Gap (%)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)
#vehicle = 1																			
Solomon 100	Avg	1.94	0.24	0.07	85.39	0.05	22.29	2.46	29.08	0.69	26.71	0.46	4.42	**0.00**	4.10	0.62	23.23	0.07	1.63
	Max	9.43	0.50	0.50	124.34	0.67	48.74	7.52	367.80	11.41	33.90	4.04	20.30	**0.00**	30.60	8.31	30.26	3.36	2.70
Solomon 200	Avg	2.99	1.67	1.01	857.80	0.97	45.53	2.99	75.53	1.46	132.14	1.44	21.35	0.94	182.03	0.70	59.33	**0.29**	74.58
	Max	5.80	2.80	2.62	1253.45	3.59	69.86	7.28	974.00	4.75	236.89	4.05	40.60	3.13	360.60	3.06	80.67	**1.16**	219.58
Cordeau 1–10	Avg	4.72	1.75	1.08	822.07	0.97	107.57	1.61	12.37	1.05	109.01	1.34	78.54	0.91	75.69	**0.57**	122.33	0.60	15.25
	Max	9.26	4.60	3.11	1772.44	2.69	269.46	4.98	32.70	4.38	222.22	3.30	247.40	4.24	208.20	**2.35**	280.47	3.40	35.41
Cordeau 11–20	Avg	9.49	1.98	3.30	1045.93	3.64	160.70	4.19	24.22	4.21	130.23	2.93	103.67	2.20	118.34	3.11	153.94	1.96	22.49
	Max	17.06	5.30	7.08	2441.90	8.61	528.85	9.64	60.90	11.18	248.20	8.60	346.90	**5.64**	156.80	7.86	356.44	6.44	46.83
#vehicle = 2																			
Solomon 100	Avg	1.96	0.89	1.08	68.78	0.16	35.16	1.75	73.88	0.50	69.31	0.44	13.00	**0.10**	57.43	0.29	38.20	0.29	6.33
	Max	5.44	1.50	3.63	100.00	1.12	67.56	4.83	616.40	3.30	149.22	3.57	28.20	**0.63**	212.50	1.53	63.66	1.91	10.27
Solomon 200	Avg	3.11	2.60	1.02	813.69	0.97	80.45	2.17	19.79	0.49	463.80	1.17	30.65	1.41	254.50	0.81	88.44	**0.38**	561.10
	Max	5.72	5.10	2.56	1421.47	3.13	198.74	4.92	135.50	1.36	890.89	2.73	49.60	3.12	297.20	2.68	98.73	**1.10**	1608.51
Cordeau 1–10	Avg	6.16	4.76	3.83	524.83	2.39	171.33	1.73	39.09	1.05	247.05	2.54	149.69	2.04	186.93	0.77	213.66	**0.54**	82.95
	Max	10.76	10.30	6.04	1018.85	6.05	417.69	3.18	94.30	3.41	447.01	6.00	445.70	4.48	303.00	2.64	504.80	**1.57**	233.02
Cordeau 11–20	Avg	7.61	5.21	3.37	618.78	3.64	200.44	1.92	82.44	2.46	304.63	3.19	221.84	2.86	218.87	1.87	212.94	**1.64**	120.66
	Max	13.65	11.00	5.13	1227.58	9.51	543.00	4.70	209.60	7.03	541.62	10.10	708.80	6.31	323.20	6.20	496.85	**4.90**	242.52
#vehicle = 3																			
Solomon 100	Avg	2.41	1.46	1.23	68.87	0.45	48.39	1.88	91.09	0.21	135.86	0.64	22.55	**0.09**	86.42	0.49	50.61	0.32	17.36
	Max	4.78	3.10	2.80	93.85	2.51	95.74	3.34	604.00	1.42	303.12	2.14	39.20	**1.07**	163.00	2.23	75.54	1.63	26.72
Solomon 200	Avg	1.12	1.71	0.13	322.33	0.47	53.84	0.27	7.32	**0.01**	89.24	0.34	12.49	0.05	110.34	0.40	63.84	0.11	515.29
	Max	4.30	3.40	1.30	814.38	3.31	124.15	1.57	21.00	**0.29**	981.16	2.87	45.30	0.58	284.80	3.31	103.55	0.76	1810.85
Cordeau 1–10	Avg	6.57	9.24	3.62	473.20	2.33	201.92	1.13	85.90	**0.35**	423.99	2.76	228.93	1.25	236.97	1.53	248.32	1.10	196.73
	Max	10.27	27.90	6.65	923.50	6.10	598.02	2.19	198.00	**0.81**	731.06	7.30	658.40	2.51	350.10	3.40	574.75	2.39	403.28
Cordeau 11–20	Avg	9.12	9.69	3.28	517.48	3.74	254.50	1.73	150.73	**1.04**	496.96	4.05	247.64	1.95	250.94	1.78	262.06	1.67	351.02
	Max	12.63	28.20	5.56	1070.14	8.27	706.68	3.74	413.50	3.94	909.74	8.80	700.10	5.64	408.60	4.02	562.03	**3.14**	785.47
#vehicle = 4																			
Solomon 100	Avg	3.19	2.42	1.64	66.85	0.55	61.47	1.92	86.59	**0.14**	199.54	1.02	21.39	1.21	112.31	0.45	68.98	0.56	39.30
	Max	5.53	3.80	4.17	93.79	2.40	133.37	3.56	513.00	**0.71**	373.13	3.11	46.70	2.40	207.50	1.59	88.27	1.30	79.71
Solomon 200	Avg	**0.00**	1.02	**0.00**	141.22	**0.00**	0.12	**0.00**	0.53	**0.00**	0.17	**0.00**	9.29	**0.00**	20.73	**0.00**	31.73	**0.00**	392.92
	Max	**0.00**	2.10	**0.00**	294.59	**0.00**	0.67	0.06	3.30	**0.00**	0.23	**0.00**	44.50	**0.00**	52.90	**0.00**	44.39	**0.00**	1053.36
Cordeau 1–10	Avg	6.86	14.07	3.34	403.17	1.99	255.69	1.40	127.33	**0.13**	566.49	3.26	233.28	3.35	234.35	1.29	291.72	1.21	361.70
	Max	9.41	38.70	7.13	800.31	4.63	709.62	3.74	309.90	**1.08**	1044.37	5.20	725.70	5.59	371.40	3.55	637.46	2.75	836.53
Cordeau 11–20	Avg	8.19	13.74	3.21	408.01	3.67	282.94	2.52	232.64	**0.16**	728.64	3.56	267.58	3.69	269.44	1.78	289.05	1.97	779.97
	Max	12.15	35.90	6.34	862.42	7.81	755.39	6.37	632.00	**1.06**	1196.65	6.17	826.30	6.81	389.70	3.44	634.00	3.94	1543.99

Bold value indicates that the corresponding algorithm achieves the smallest gap compared to the BKS among all the algorithms in the comparison.

Table 5. Summary of MILS and CPLEX performances on FTSP–EDST instances.

Instance Name	Objective Value									CPU (s)						Best GAP (%)						AVG GAP (%)		
	Small			Medium			Large			Small		Medium		Large		Small		Medium		Large		Small	Medium	Large
	CPLEX	MILS *	MILS	CPLEX	MILS *	MILS	CPLEX	MILS *	MILS	CPLEX	MILS	CPLEX	MILS	CPLEX	MILS	CPLEX	MILS *	CPLEX	MILS *	CPLEX	MILS *	MILS	MILS	MILS
C101_5x4	850	850	850.0	1260	1270	1270.0	1600	2.9	5.5	7200.0	31.7	7200.0	75.9	2.9	5.5	0.00	0.00	0.79	0.00	2.50	0.00	0.00	0.47	0.50
C101_6x6	780	780	780.0	1090	1090	1090.0	1410	4.9	8.7	7200.0	23.7	7200.0	57.1	4.9	8.7	0.00	0.00	0.00	0.00	0.71	0.00	0.00	0.37	0.71
C101_7x4	820	820	818.0	1250	1250	1250.0	1570	5.4	8.1	1644.4	26.3	7200.0	57.0	5.4	8.1	0.00	0.00	0.00	0.00	1.27	0.00	0.24	0.16	0.38
C103_5x4	860	860	860.0	1250	1290	1290.0	1500	26.7	9.0	7200.0	37.9	7200.0	126.5	26.7	9.0	0.00	0.00	3.10	0.00	10.67	0.00	0.00	0.00	0.27
C103_6x6	800	850	848.0	960	1240	1240.0	1230	7200.0	36.4	7200.0	130.6	7200.0	262.9	7200.0	36.4	5.88	0.00	22.58	0.00	29.27	0.00	0.24	0.81	0.65
C103_7x4	840	840	840.0	1260	1330	1330.0	1520	2315.8	10.6	7200.0	71.5	7200.0	167.1	2315.8	10.6	0.00	0.00	5.26	0.00	11.84	0.00	0.00	0.30	0.39
C201_5x4	860	860	860.0	1300	1300	1300.0	1670	0.3	5.6	1.8	21.2	4.4	42.6	0.3	5.6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C201_6x6	890	890	890.0	1310	1310	1310.0	1680	1.0	7.2	4.4	24.1	18.2	38.5	1.0	7.2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C201_7x4	840	840	840.0	1350	1350	1350.0	1720	0.3	5.4	2.1	17.6	3.6	30.3	0.3	5.4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C203_5x4	860	860	860.0	1300	1300	1300.0	1670	1.5	11.7	26.1	28.0	245.9	53.5	1.5	11.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C203_6x6	890	890	890.0	1310	1310	1310.0	1680	27.6	9.1	72.6	31.4	226.2	75.9	27.6	9.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C203_7x4	840	840	840.0	1350	1350	1350.0	1720	1.3	7.6	9.8	22.6	108.5	52.0	1.3	7.6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
R101_5x4	409	409	409.0	662	662	662.0	905	0.3	1.6	1.2	5.5	30.6	18.2	0.3	1.6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.44
R101_6x6	344	344	344.0	582	582	582.0	794	0.3	1.4	2.2	6.7	54.2	19.3	0.3	1.4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.06
R101_7x4	409	409	409.0	633	633	633.0	855	0.3	1.5	1.9	6.5	15.0	16.9	0.3	1.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05
R103_5x4	512	583	582.4	649	866	866.0	817	7200.0	8.1	7200.0	33.5	7200.0	78.9	7200.0	8.1	12.18	0.00	25.06	0.00	46.27	0.00	0.10	0.37	0.98
R103_6x6	402	503	499.0	561	785	785.0	770	7200.0	6.2	7200.0	23.7	7200.0	59.1	7200.0	6.2	20.08	0.00	28.54	0.00	41.69	0.00	0.80	0.15	0.86
R103_7x4	499	585	584.4	622	891	891.0	829	7200.0	7.7	7200.0	34.1	7200.0	81.5	7200.0	7.7	14.70	0.00	30.19	0.00	38.24	0.00	0.10	0.63	1.23
R201_5x4	714	714	714.0	1069	1069	1069.0	1387	5.8	6.8	39.6	27.2	310.4	61.8	5.8	6.8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
R201_6x6	690	690	690.0	1078	1078	1078.0	1381	7.1	10.5	592.4	34.4	143.7	86.6	7.1	10.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
R201_7x4	725	725	725.0	1106	1106	1106.0	1391	2.8	7.2	149.5	32.5	222.1	61.1	2.8	7.2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
R203_5x4	714	714	714.0	1069	1069	1069.0	1387	8.5	9.8	83.5	33.9	6879.2	64.3	8.5	9.8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
R203_6x6	690	690	690.0	1070	1078	1078.0	1381	36.5	14.2	7200.0	57.6	6771.4	148.8	36.5	14.2	0.00	0.00	0.74	0.00	0.00	0.00	0.00	0.00	0.00
R203_7x4	725	725	725.0	1106	1106	1106.0	1391	9.0	7.9	214.3	39.8	7200.0	59.1	9.0	7.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
RC101_5x4	580	580	580.0	802	857	857.0	983	1180.3	2.3	7200.0	11.4	7200.0	19.1	1180.3	2.3	0.00	0.00	6.42	0.00	7.53	0.00	0.00	0.00	0.28
RC101_6x6	463	463	463.0	757	771	771.0	960	256.1	1.9	7200.0	6.8	7200.0	19.8	256.1	1.9	0.00	0.00	1.82	0.00	7.29	0.00	0.00	0.70	0.40
RC101_7x4	540	540	540.0	794	794	794.0	973	144.4	2.4	7200.0	9.7	7200.0	24.9	144.4	2.4	0.00	0.00	0.00	0.00	7.09	0.00	0.00	0.00	0.00
RC103_5x4	639	769	769.0	709	1048	1048.0	1012	7200.0	6.6	7200.0	28.1	7200.0	93.2	7200.0	6.6	16.91	0.00	32.35	0.00	34.49	0.00	0.00	0.00	0.71
RC103_6x6	479	633	632.6	707	961	961.0	812	7200.0	6.1	7200.0	23.3	7200.0	71.2	7200.0	6.1	24.33	0.00	26.43	0.00	56.28	0.00	0.06	0.19	0.17
RC103_7x4	675	728	728.0	746	1036	1036.0	830	7200.0	5.3	7200.0	26.1	7200.0	71.7	7200.0	5.3	7.28	0.00	27.99	0.00	62.53	0.00	0.00	0.52	0.53
RC201_5x4	959	959	959.0	1338	1358	1358.0	1605	5.8	7.1	7200.0	28.6	7200.0	60.3	5.8	7.1	0.00	0.00	1.47	0.00	1.87	0.00	0.00	0.00	0.00
RC201_6x6	963	978	978.0	1303	1367	1367.0	1622	7200.0	11.2	7200.0	34.4	7200.0	75.0	7200.0	11.2	1.53	0.00	4.68	0.00	2.03	0.00	0.00	0.00	0.00
RC201_7x4	948	948	948.0	1374	1374	1374.0	1643	30.1	7.6	6960.9	27.1	7200.0	52.3	30.1	7.6	0.00	0.00	0.00	0.00	0.61	0.00	0.00	0.00	0.00
RC203_5x4	959	959	959.0	1348	1358	1358.0	1604	23.5	11.8	7200.0	33.7	7200.0	68.8	23.5	11.8	0.00	0.00	0.74	0.00	1.93	0.00	0.00	0.00	0.00
RC203_6x6	978	978	978.0	1316	1367	1367.0	1487	6621.2	12.1	7200.0	56.5	7200.0	147.8	6621.2	12.1	0.00	0.00	3.73	0.00	11.30	0.00	0.00	0.00	0.00
RC203_7x4	948	948	948.0	1374	1374	1374.0	1633	23.0	9.0	751.7	30.8	7200.0	66.6	23.0	9.0	0.00	0.00	0.00	0.00	1.22	0.00	0.00	0.00	0.00
Average	724.8	743.2	742.9	1049.0	1110.6	1109.2	1317.3	1417.7	1415.0	1898.4	8.1	4093.3	31.1	4817.6	71.3	2.86	0.00	6.16	0.00	10.46	0.00	0.04	0.13	0.27

The Wilcoxon rank sum test is performed on the results obtained by CPLEX and the proposed MILS. Because CPLX runs only once and MILS runs five trials, the average objective function values obtained by MILS are used in Wilcoxon rank sum test. At a confidence level of $\alpha = 0.05$, Table 6 showed that MILS is significantly better than CPLEX in solving medium, large, and all problem sets.

Table 6. Wilcoxon rank sum test for CPLEX and MILS.

Problem Size	Statistical Value	*p*-Value
Small	708.0	0.3321
Medium	843.5	0.0124 *
Large	891.5	0.0038 *
All	7230.0	0.0004 *

* indicates that significant difference exists.

6. Conclusions and Future Research

This research investigates the Field Technician Scheduling Problem with Experience-Dependent Service Times in which not all tasks are served due to the limitation of available technicians and the service time of performing a task depends on the skill level possessed by a technician, i.e., a higher skill leads to a lower service time and vice versa. The experience-dependent service time may lead to higher accuracy during the execution phase and may reduce the possibility of both overtime and underutilization.

Our experimental results show that MILS performs well in solving FTSP–EDST and the team orienteering problem with time windows as its special case. Results from state-of-the-art methods are compiled and compared with solutions obtained by MILS. The average gap ranges from 0 to 1.97% while the maximum gap ranges from 0 to 6.44%. Moreover, MILS outperforms the considered methods in several groups of instances in terms of average gap and/or maximum gap. Regarding the FTSP–EDST instances, CPLEX results are used for the sake of comparison since the problem is new. Our experimental results confirm the superiority of MILS over CPLEX in terms of solution quality and computational time. In addition, solutions provided by MILS are robust because the average solution quality only differs slightly from the best solution quality.

Several future directions are still available for further investigations. First, algorithmic improvement can be investigated in order to reduce the computational complexity of MILS. Hybridizing MILS with exact methods, such as mathematical programming techniques, as an attempt to improve solution quality can also be tested. Other variants of FTSP–EDST, such as considering workload balance or uncertain factors, can be further defined and addressed to cope with various real-world requirements.

Author Contributions: Conceptualization, V.F.Y. and P.J.; methodology, V.F.Y., Y.-S.L. and P.J.; validation, Y.-S.L. and P.J.; formal analysis, Y.-S.L. and P.J.; investigation, V.F.Y. and S.-W.L.; data curation, Y.-S.L., P.J. and Y.-C.L.; writing—original draft preparation, Y.-S.L. and P.J.; writing—review and editing, V.F.Y., S.-W.L. and Y.-C.L.; visualization, Y.-S.L., P.J. and Y.-C.L.; supervision, V.F.Y. and S.-W.L.; project administration, V.F.Y.; funding acquisition, V.F.Y., S.-W.L. and Y.-C.L. All authors have read and agreed to the published version of the manuscript.

Funding: The work of the first author was partially supported by the Ministry of Science and Technology of the Republic of China (Taiwan) under Grant MOST 111-2410-H-011-020-MY3. The fourth author would like to acknowledge the support from the Ministry of Science and Technology of the Republic of China (Taiwan) under Grants MOST 109-2410-H-182-009-MY3 and NSTC 112-2410-H-182-002-MY3, and from the Chang Gung Memorial Hospital under Grant BMRPA19. The fifth author's work was partially funded by the Ministry of Science and Technology of the Republic of China (Taiwan) under Grants MOST 109-2221-E-011-129-MY3 and MOST 109-2221-E-011-128-MY3.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dutot, P.F.; Laugier, A.; Bustos, A.M. *Technicians and Interventions Scheduling for Telecommunications*; France Telecom R&D: Paris, France, 2006.
2. Toth, P.; Vigo, D. *Vehicle Routing: Problems, Methods, and Applications*, 2nd ed.; SIAM: Philadelphia, PA, USA, 2014; pp. 1–481.
3. Damm, R.B.; Resende, M.G.; Ronconi, D.P. A biased random key genetic algorithm for the field technician scheduling problem. *Comput. Oper. Res.* **2016**, *75*, 49–63. [CrossRef]
4. Chen, X.; Thomas, B.W.; Hewitt, M. The technician routing problem with experience-based service times. *Omega* **2016**, *61*, 49–61. [CrossRef]
5. Villegas, J.; Guéret, C.; Mendoza, J.E.; Montoya, A. *The Technician Routing and Scheduling Problem with Conventional and Electric Vehicle*; HAL: Lyon, France, 2018.
6. Yu, V.F.; Jewpanya, P.; Lin, S.W.; Redi, A.A.N.P. Team orienteering problem with time windows and time-dependent scores. *Comput. Ind. Eng.* **2019**, *127*, 213–224. [CrossRef]
7. Yu, V.F.; Redi, A.A.N.P.; Jewpanya, P.; Gunawan, A. Selective discrete particle swarm optimization for the team orienteering problem with time windows and partial scores. *Comput. Ind. Eng.* **2019**, *138*, 106084. [CrossRef]
8. Kovacs, A.A.; Parragh, S.N.; Doerner, K.F.; Hartl, R.F. Adaptive large neighborhood search for service technician routing and scheduling problems. *J. Sched.* **2012**, *15*, 579–600. [CrossRef]
9. Zamorano, E.; Stolletz, R. Branch-and-price approaches for the multiperiod technician routing and scheduling problem. *Eur. J. Oper. Res.* **2017**, *257*, 55–68. [CrossRef]
10. Mathlouthi, I.; Gendreau, M.; Potvin, J.Y. A metaheuristic based on Tabu search for solving a technician routing and scheduling problem. *Comput. Oper. Res.* **2021**, *125*, 105079. [CrossRef]
11. Pourjavad, E.; Almehdawe, E. Optimization of the technician routing and scheduling problem for a telecommunication industry. *Ann. Oper. Res.* **2022**, *315*, 371–395. [CrossRef]
12. Barnhart, C.; Johnson, E.L.; Nemhauser, G.L.; Savelsbergh, M.W.; Vance, P.H. Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.* **1998**, *46*, 316–329. [CrossRef]
13. Mathlouthi, I.; Gendreau, M.; Potvin, J.Y. Branch-and-price for a multi-attribute technician routing and scheduling problem. *Oper. Res. Forum* **2021**, *2*, 1–35. [CrossRef]
14. Pekel, E. Solving technician routing and scheduling problem using improved particle swarm optimization. *Soft Comput.* **2020**, *24*, 19007–19015. [CrossRef]
15. Pekel, E. A simple solution to technician routing and scheduling problem using improved genetic algorithm. *Soft Comput.* **2022**, *26*, 6739–6748. [CrossRef]
16. Caprara, A.; Toth, P.; Fischetti, M. Algorithms for the set covering problem. *Ann. Oper. Res.* **2000**, *98*, 353–371. [CrossRef]
17. Vansteenwegen, P.; Souffriau, W.; Berghe, G.V.; Van Oudheusden, D. Iterated local search for the team orienteering problem with time windows. *Comput. Oper. Res.* **2009**, *36*, 3281–3290. [CrossRef]
18. Tang, H.; Miller-Hooks, E. A tabu search heuristic for the team orienteering problem. *Comput. Oper. Res.* **2005**, *32*, 1379–1407. [CrossRef]
19. Lin, S.W.; Yu, V.F. A simulated annealing heuristic for the team orienteering problem with time windows. *Eur. J. Oper. Res.* **2012**, *217*, 94–107. [CrossRef]
20. Tricoire, F.; Romauch, M.; Doerner, K.F.; Hartl, R.F. Heuristics for the multi-period orienteering problem with multiple time windows. *Comput. Oper. Res.* **2010**, *37*, 351–367. [CrossRef]
21. Gunawan, A.; Lau, H.C.; Vansteenwegen, P.; Lu, K. Well-tuned algorithms for the team orienteering problem with time windows. *J. Oper. Res. Soc.* **2017**, *68*, 861–876. [CrossRef]
22. Montemanni, R.; Weyland, D.; Gambardella, L. An enhanced ant colony system for the team orienteering problem with time windows. In Proceedings of the 2011 International Symposium on Computer Science and Society, Kota Kinabalu, Malaysia, 16–17 July 2011; pp. 381–384.
23. Cura, T. An artificial bee colony algorithm approach for the team orienteering problem with time windows. *Comput. Ind. Eng.* **2014**, *74*, 270–290. [CrossRef]
24. Hu, Q.; Lim, A. An iterative three-component heuristic for the team orienteering problem with time windows. *Eur. J. Oper. Res.* **2014**, *232*, 276–286. [CrossRef]
25. Penna, P.H.V.; Subramanian, A.; Ochi, L.S. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *J. Heuristics* **2013**, *19*, 201–232. [CrossRef]
26. Dong, X.; Huang, H.; Chen, P. An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion. *Comput. Oper. Res.* **2009**, *36*, 1664–1669. [CrossRef]
27. Burke, E.K.; Curtois, T.; Van Draat, L.F.; van Ommeren, J.K.; Post, G. Progress control in iterated local search for nurse rostering. *J. Oper. Res. Soc.* **2011**, *62*, 360–367. [CrossRef]
28. Song, T.; Liu, S.; Tang, X.; Peng, X.; Chen, M. An iterated local search algorithm for the University Course Timetabling Problem. *Appl. Soft Comput.* **2018**, *68*, 597–608. [CrossRef]
29. Lourenço, H.R.; Martin, O.C.; Stützle, T. Iterated local search. In *The Handbook of Metaheuristics*; Glover, F., Kochenberger, G., Eds.; Kluwer Academic Publishers: Dordrecht, The Netherlands, 2003; pp. 321–353.

30. Labadie, N.; Mansini, R.; Melechovský, J.; Calvo, R.W. The team orienteering problem with time windows: An lp-based granular variable neighborhood search. *Eur. J. Oper. Res.* **2012**, *220*, 15–27. [CrossRef]
31. Nagata, Y.; Bräysy, O.; Dullaert, W. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Comput. Oper. Res.* **2010**, *37*, 724–737. [CrossRef]
32. Solomon, M.M. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* **1987**, *35*, 254–265. [CrossRef]
33. Cordeau, J.F.; Gendreau, M.; Laporte, G. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks* **1997**, *30*, 105–119. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.