EASR

Engineering and Applied Science Research

https://www.tci-thaijo.org/index.php/easr/index

Published by the Faculty of Engineering, Khon Kaen University, Thailand

A self-adaptive differential evolution for the technician routing and scheduling problem

Voravee Punyakum[1)], Krisanarach Nitisiri*[2)], Kanchana Sethanan[2)] and Dusit Singpommat[1)]

[1)]Faculty of Technical Education, Rajamangala University of Technology Krungthep, Bangkok 10120, Thailand
[2)]Research Unit on System Modelling for Industry, Department of Industrial Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen 40002, Thailand

Received 13 December 2022
Revised 16 April 2023
Accepted 21 April 2023

Abstract

This paper presents two approaches for solving the Technician Routing and Scheduling Problem (TRSP). The first approach is an integer programming method, and the second is a Self-Adaptive Differential Evolution (SADE) algorithm. The TRSP involves scheduling jobs for service teams who have different skill sets. These jobs have time constraints and can only be completed by technicians with the necessary skills. The goal of the TRSP is to minimize the operating cost, which includes travel, late service penalties, technician overtime, and subcontracting costs. To evaluate the effectiveness of the proposed SADE and integer programming approaches, we conducted small-scale numerical experiments. We also compared the performance of SADE to that of the conventional Differential Evolution (DE) algorithm on medium and large-scale problems. The results indicate that SADE produces significantly higher quality solutions compared to DE.

Keywords: Technician scheduling, Vehicle routing, Differential evolution

1. Introduction

The optimal use of the available manpower is one of the primary business problems for service organizations, due to limited manpower resources, a rising service portfolio and customized requirements requested by customers [1]. For decades, related routing, scheduling, and rostering problems have been investigated to assist businesses in making the most efficient use of their manpower [2]. Companies' services frequently include difficult jobs that necessitate the participation of several personnel with a variety of skills. Furthermore, services are frequently offered at customers' locations, necessitating technician travel. To satisfy the requirements for this type of problem, businesses organizes service teams to simplify the planning of job schedules and service teams routing to customer locations [3]. As a result, the service team loses a portion of its daily working hours to travel to different locations. Hence, a routing problem arises in addition to the job assignment and scheduling problem [4]. The Technician Routing and Scheduling Problem (TRSP) is an operational issue that involves planning the routes and schedules for mobile service technicians. The TRSP combines elements of both the Workforce Scheduling and Routing Problem (WSRP) and the Vehicle Routing Problem (VRP). Essentially, the TRSP is a generalized version of the WSRP that takes into account additional factors from the VRP [5]. Examples of TRSP can be found in the installation and maintenance sector [6-9].

To solve the TRSP, metaheuristic methods are widely utilized because of their ability to handle problems more intensively and with more robust methodologies. Metaheuristics have been developed in the last decade to find a near-optimal solution for larger, more practical problems [10]. For example, Kovacs et al. [11] researched service TRSPs in which each team's routes were determined in such a way that travel expenses were minimized while time windows and task skill criteria were met. The purpose is to minimize the total routing and outsourcing costs solved by Adaptive large neighborhood search (ALNS). Next year, Pillac et al. [12] utilized the Adaptive Large Neighborhood Search (ALNS) method to solve the Technician Routing and Scheduling Problem (TRSP). In their approach, the task compatibility constraint included the availability of necessary spare parts and tools. This allowed technicians to complete tasks independently, without the need to form teams based on specific skill sets. By using the ALNS approach, Pillac et al. were able to find solutions to the TRSP that were efficient and effective. Pinheiro et al. [13] investigated WSRP, taking into account the need for desired skills, labor availability, and the objective of lowering operating costs, such as transportation and staffing. A variable neighborhood search (VNS) strategy was used to find the solution. Later, Çakırgil et al. [2] looked at the WSRP via the lens of an electricity distribution firm case study. The formation of a team, technical skills, and technician multi-skills were all taken into account. The goal of this study was to consider two objectives: completing high-priority jobs as quickly as possible and minimizing operating costs, including travel and outsourcing expenses. These objectives were used to guide the process of routing and scheduling technicians in order to achieve the best possible outcomes. By focusing on these two objectives, the researchers aimed to optimize the efficiency and effectiveness of the technician routing and scheduling process. A multi-objective VNS strategy was utilized to find the solution. In the same year, Pekel [14] proposed an improved particle swarm optimization (IPSO) to solve multi-period TRSP. The IPSO is a hybrid between particle swarm optimization (PSO) and the neighborhood operator. The results reveal that when using the branch-and-cut algorithm, IPSO produces superior outcomes in an acceptable amount of time. Later, Punyakum et al. [15] proposed a

*Corresponding author.
Email address: krisni@kku.ac.th
doi: 10.14456/easr.2023.29

hybrid optimization approach called the Hybrid Differential Evolution and Particle Swarm Optimization (HDEPSO) for solving multi-visit and multi-period Workforce Scheduling and Routing Problems (WSRP). The HDEPSO combines elements of both differential evolution (DE) and particle swarm optimization (PSO). The researchers tested the HDEPSO against both DE and PSO and found that it produced significantly higher quality solutions. In the same year, Punyakum et al. [16] also proposed a hybrid optimization algorithm called the Hybrid Particle Swarm and Whale Optimization Algorithm (HPSWOA) for solving dynamic WSRP with multiple visits and periods. The HPSWOA is a combination of PSO and the whale optimization algorithm (WOA). The researchers compared the HPSWOA to both PSO and WOA and found that it resulted in superior solutions.

Differential Evolution (DE) is a powerful optimization algorithm that is known for its simplicity and efficiency as a global optimization method. DE is particularly effective at finding optimal solutions because it has strong convergence properties, which means that it can quickly find good solutions to problems. Additionally, DE can be used with parallel computing, which allows the algorithm to analyze a designed vector population individually and reduce computational resources. This makes DE a useful tool for solving complex optimization problems in an efficient manner [17], which can be improved further by using adaptive parameters [18] and hybridizing strategies [15, 16]. DE is a also highly customizable algorithm that can be customized to incorporate the problem-specific constraints of the TRSP, which involves complex interactions between workers, tasks, and equipment. DE can capture these interactions and provide a way to model and analyze the system. Furthermore, DE can be used to optimize the allocation of resources by minimizing costs or maximizing productivity. This is particularly useful in workforce scheduling and routing problems where efficiency is a key consideration. TRSP is never solved with adaptive DE. This study aims to address the gap in existing approaches by proposing two methods for solving the Technician Routing and Scheduling Problem (TRSP): a mixed integer linear programming (MILP) and a self-adaptive differential evolution (SADE). The MILP approach is used to solve small-scale TRSP problems and to validate the performance of the SADE method. The SADE approach is designed to be more efficient and effective at solving larger-scale TRSP problems. By using both the MILP and SADE approaches, this study aims to provide a comprehensive solution to the TRSP.

The structure of this paper is as follows: The mathematical model for the Technician Routing and Scheduling Problem (TRSP) is described in Section 2. The development of the Self-Adaptive Differential Evolution (SADE) algorithm for the TRSP is presented in Section 3. The computational results of using the SADE algorithm on the TRSP are discussed in Section 4. Finally, the last section includes the conclusions and suggestions for future research on the TRSP.

2. Mathematical model for TRSP

The assumptions for the TRSP are as follows: (i) only one service team is allowed to serve a customer at a time; (ii) service teams is constantly stocked with the tools and spare parts required for maintenance; (iii) technicians are assigned to service teams before beginning their work; (iv) there is only one size available for transport vehicles; (v) maintenance services cannot be divided between service teams; (vi) service teams return to the company once they have completed their assigned services. N is set of customers and company (depot), however company is not included in N'. The following list includes the notation used throughout the paper.

Indices

i, j	: Index of customers and company (depot)
k	: Index of service teams
r	: Index of technician skills

Parameters

N	: Set of customers and company (depot)
N'	: Set of customers
K	: Set of service teams
R	: Set of skill types
$Tc_{i,j}$	: Transport cost for service teams traveling from customer i to j (units)
TDc_i	: penalty costs for the customer's service being late (unit per minutes)
OSc	: Subcontracting cost of customers i (unit per minutes)
OTc_k	: Overtime of service team k (unit per minutes)
e	: Starting time
f	: Finish time
a_i	: Starting time of customer i
b_i	: Finish time of customer i
$t_{i,j}$	: Travel time from customer i to customer j (minutes)
p_i	: Service time of customer i (minutes)
$g_{k,r}$	: Proficiency of service team of skill r (1 = basic, 2 = medium, 3 = expert)
n	: Number of service teams
$u_{i,r}$	: Requirements of proficiency of service team of skill r for customer i
h_k	: Number of technicians of service team k
w_i	: Number of technicians required by customer i
M	: Positive large number
$TDmax$	: max delay time
$OTmax$	: max overtime

Decision Variables

$X_{i,j,k}$	= 1, if service team k leaves customer i for customer j; = 0, otherwise.
$Y_{i,k}$	= 1, if service team k services customer i; = 0, otherwise.

OS_i = 1, when customer i requests subcontracting;
= 0, otherwise.
TD_i : Amount of maintenance late time for customer i
OT_k : Overtime of service team k
$s_{i,k}$: Starting time to service customer i by service team k

Objective function:

$$min \sum_{i\in N}\sum_{j\in N}\sum_{k\in K} X_{i,j,k}\cdot Tc_{i,j} + \sum_{i\in N} TD_i \cdot TDc_i + \sum OT_k \cdot OTc_k + \sum_{i\in N} OS_i \cdot OSc \cdot p_i \quad (1)$$

Subject to:

$$\sum_{k\in K} Y_{i,k} + OS_i = 1 \qquad \forall\, i \in N' \quad (2)$$

$$\sum_{j\in N'} X_{i,j,k} = Y_{i,k} \qquad \forall\, i \in N', i \neq j \quad (3)$$

$$\sum_{j\in N} (X_{i,j,k} - X_{j,i,k}) = 0 \quad \forall\, i \in N, k \in K, i \neq j \quad (4)$$

$$(s_{i,k} + p_i + t_{i,j}) \cdot Y_{i,k} - M\left(1 - X_{i,j,k}\right) \leq s_{j,k} \qquad \forall\, i \in N, j \in N', k \in K, i \neq j \quad (5)$$

$$s_{i,k} \geq a_i \cdot Y_{i,k} \qquad \forall\, i \in N', k \in K \quad (6)$$

$$s_{i,k} - (b_i - p_i) \cdot Y_{i,k} \leq TD_i \qquad \forall\, i \in N', k \in K \quad (7)$$

$$TD_i \leq TDmax \qquad \forall\, i \in N' \quad (8)$$

$$s_{j,k} \geq (e + t_{i,j}) \cdot X_{i,j,k} \qquad \forall j \in N', k \in K, i = 1 \quad (9)$$

$$s_{i,k} - OT_{i,k} + (t_{i,j} + p_i) \cdot X_{i,j,k} \leq f \qquad \forall\, i \in N', k \in K, j = 1 \quad (10)$$

$$\sum_{i\in N'} OT_{i,k} \leq OTmax \quad k \in K_q \quad (11)$$

$$\sum_{j=N'}\sum_{k\in K} X_{i,j,k} \leq n \quad i = 1 \quad (12)$$

$$w_i \cdot Y_{i,k} \leq h_k \qquad \forall i \in N', k \in K \quad (13)$$

$$u_{i,r} \cdot Y_{i,k} \leq g_{k,r} \quad \forall i \in N', k \in K, r \in R \quad (14)$$

$$X_{i,j,k} = \{0,1\} \quad \forall i, j \in N, k \in K \quad (15)$$

$$Y_{i,k} = \{0,1\} \quad \forall i \in N, k \in K \quad (16)$$

$$OS_i = \{0,1\} \quad \forall i \in N \quad (17)$$

The objective of this problem is to minimize the operating cost of routing and scheduling technicians to service customers, which includes the travel cost of service teams, the penalty cost for delay services, the technician overtime cost, and the subcontracting cost. Constraint (2) ensures that each customer must be serviced either by a service team or by a subcontractor. Constraint (3) ensures that customer i is serviced by technician team k. Constraint (4) ensures that, after service completion, the team transport vehicles had to leave the customer. Constraint (5) ensures that the starting time for servicing customer j is the finish time of customer i plus the travel time of service team k. Constraint (6) ensures that a service team must start servicing a customer only when the customer is ready. Constraint (7) allows for the possibility of a delay in service if a service team cannot finish the service in time. Constraint (8) limits the amount of delay allowed for a service. Constraint (9) designates the start service time for a service team. Constraint (10) allows for service team overtime if they arrive at the depot late. Constraint (11) limits the amount of overtime allowed for each service team. Constraint (12) ensure that the number of service teams leaving company (i = 1) does not exceed the number of service groups available. Constraint (13) ensures that each service team has sufficient personnel to service a customer. Constraint (14) ensures that the proficiency of skill types of each service team is sufficient to service a customer. Constraints (15) to (17) are basic restrictions on the binary variables.

3. Metaheuristic approach for solving TRSP problem

The Self-Adaptive Differential Evolution (SADE) is a metaheuristic algorithm that was designed to improve upon the original Differential Evolution (DE) algorithm [17]. SADE includes a self-adaptive parameter control method that allows it to adapt to the specific characteristics of the problem it is being applied to. SADE consists of five main processes: initialization, mutation, recombination, selection, and self-adaptive control parameter. Initialization involves generating the initial solution, which serves as the starting point for the optimization process. Mutation involves making small changes to the initial solution in order to explore new regions of the search space. Recombination combines the mutated solutions with the original solution in order to create a new, potentially better solution. Selection involves comparing the new solution to the original solution and choosing the one that is better. Finally, the self-adaptive control parameter process adjusts the parameters of the algorithm in order to improve its performance. The TRSP problem is considered to be NP-hard and highly complicated problem [12]. Since the optimal solution cannot be obtained using an optimization program for the largescale problems due to the vast number of variables in the TRSP problem, The SADE was proposed to solve the problem as follows:

3.1 Initialization

The initial step of the proposed algorithm involves creating an initial population of vectors, each of which represents a potential solution to the Technician Routing and Scheduling Problem (TRSP). The size of the population is predetermined. Each vector consists of two parts: a customer vector and a service team vector. The dimension of the customer vector, which is produced at random, is equal to the total number of customers. The vector is then sorted in ascending order using the Rank Order Value method (ROV) to determine the sequence of customers that will be visited by each service team. The service team vector is also randomly generated and has a dimension equal to the number of service teams. This vector is used to prioritize which service team will service each customer. Once the initial population has been created, the algorithm proceeds to the next steps, which involve mutating, recombining, and selecting the best solutions in order to improve upon the initial population. Figure 1 illustrates an example of vector construction. In this example, there are five customers and two service teams. The customer vector is initially represented as: 0.52, 0.84, 0.65, 0.12, 0.26. The corresponding customer IDs are 1, 2, 3, 4, and 5, respectively. After sorting the customer vector in ascending order, we obtain the following: 0.12, 0.26, 0.52, 0.65, 0.84. Therefore, the customer IDs are now arranged in the following order: 4, 5, 1, 3, 2. The customers are assigned by rank of the customer ID to the service team by the close vector's value. If the first service team's skill and the number of service team members do not match the customer, the customer will be assigned to the next service team, and so on until all customers have been assigned to their service teams. The customers who cannot be serviced by the service teams will be outsourced to third-party service providers. Then, the fitness value (operating cost) will be calculated. The operating cost includes travel cost, penalty cost for late services, technician overtime cost and subcontracting cost.

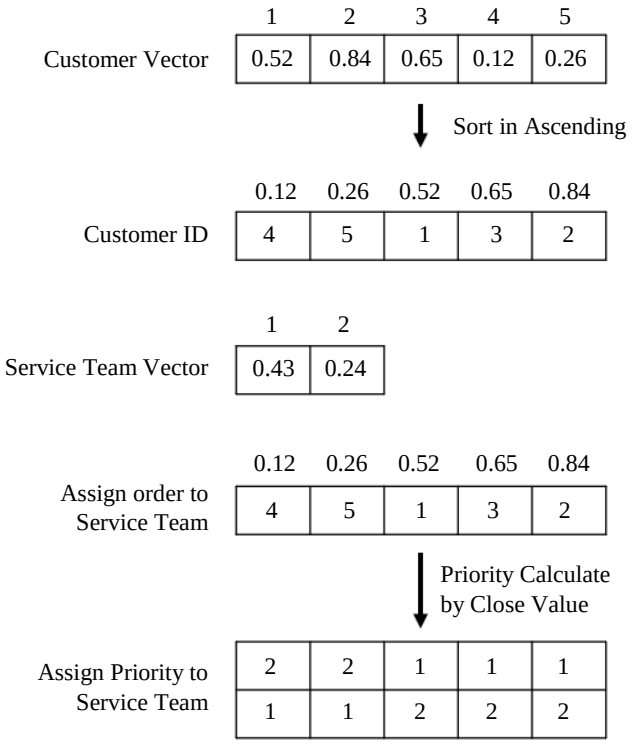

Figure 1 Example of vector construction

3.2 Mutation operation

In the Differential Evolution (DE) method, the vector number (NV) refers to the number of vectors employed during the iterative process. These NV vectors are randomly generated as the initial solutions. The second step in the DE process is the mutation operation. Eq. (18) is used to create a mutant vector by combining three vectors at random. The scaling factor F is an adaptive parameter, range from [0, 2]. The mutant vector is $V_{i,G+1}$, and the random vectors are $X_{r1,G}$, $X_{r2,G}$ and $X_{r3,G}$.

$$V_{i,G+1} = X_{r1,G} + F(X_{r2,G} + X_{r3,G}) \tag{18}$$

3.3 Recombination operation

The recombination process involves taking solutions from the previous iteration and combining them in some way to create new solutions for the current iteration. The trial vector ($U_{i,G+1}$) can be constructed using Eq. (19). The value of each position in the vector is determined based on a uniform random number ($Rand$) and the crossover rate (CR). If Rand is less than or equal to CR, the value at that position is replaced with the value from the mutant vector ($V_{ji,G+1}$). If Rand is greater than CR, the value at that position is taken from the target vector ($X_{ji,G}$). This process is repeated for each position in the vector.

$$U_{ji,G+1} = \begin{cases} V_{ji,G+1} & if\left(Rand_{j,G}[0,1] \le CR\right) \\ X_{ji,G} & otherwise \end{cases} \tag{19}$$

3.4 Selection operation

DE's selection process is similar to tournament selection in a genetic algorithm, which is a method for selection of individuals from a set of population. During each iteration of the process, the target vector ($X_{i,G}$) is compared to the trial vector ($U_{i,G+1}$). The vector that has the lowest objective value is chosen to move on to the next iteration ($X_{i,G+1}$). The selection formula is represented by Eq. (20).

$$X_{i,G+1} = \begin{cases} U_{i,G+1} & if(f(U_{i,G+1}) < f(X_{i,G})) \\ X_{i,G} & otherwise \end{cases} \tag{20}$$

3.5 Self-adaptive control parameter

By using adaptive parameters approach, the solution obtained from the DE algorithm can be improved. The SADE algorithm's adaptive part can adjust the scaling factor (F) and the crossover rate values (CR) throughout the mutation and recombination processes. Jia et al. [19] proposed a self-adaptive parameter control technique for the DE algorithm, which adjusts the values of $CR_{i,G}$ and $F_{i,G}$ based on individual fitness data. This study adopts this self-adaptive approach, which is shown in Eq. (21) and Eq. (22). The probabilities (τ) in these equations determine the adjustment of the factors $F_{i,G}$ and $CR_{i,G}$ during the mutation and recombination processes. f_{avg} and f_{min} represent the average fitness and the minimum fitness of the current population, respectively.

$$F_{i,G+1} = \begin{cases} 0.1 + \left(F_{i,G} - 0.1\right) \times \dfrac{f(X_{i,G+1}) - f_{min}}{f_{avg} - f_{min}} & if(Rand_{i1,G}[0,1] < \tau \; and \; f\left(X_{i,G+1}\right) < f_{avg}) \\ Rand_{i2,G}[0.1,1] & if(Rand_{i1,G}[0,1] < \tau \; and \; f\left(X_{i,G+1}\right) \ge f_{avg}) \\ F_{i,g} & othewise \end{cases} \tag{21}$$

$$CR_{i,G+1} = \begin{cases} CR_{i,G} \times \dfrac{f(X_{i,G+1} - f_{min})}{f_{avg} - f_{min}} & if\left(Rand_{i3,G}[0,1] < \tau \; and \; f\left(X_{i,G+1}\right) < f_{avg}\right) \\ Rand_{i4,G}[0,1] & if\left(Rand_{i3,G}[0,1] < \tau \; and \; f\left(X_{i,G+1}\right) \ge f_{avg}\right) \\ CR_{i,G} & othewise \end{cases} \tag{22}$$

To summarize the overall processes of the SADE algorithm, the algorithm begins by defining the initial solution sections for TRSP, such as the encoding and decoding vectors. A customer vector and a service team vector make up each vector. The objective is to minimize the operating cost. SADE is then employed with five processes: initialization (Section 3.1), mutation (Section 3.2), recombination (Section 3.3) and selection (Section 3.4) and self-adaptive feature (Section 3.5). The overall procedure for the SADE algorithm is illustrated in Figure 2.

Procedure: Self-adaptive differential evolution (SADE)
Input: data of TRSP, parameters of SADE (CR, F, τ, NV)
Output: Operating cost
Begin:
    Population Initialization;
    While exit condition not met **do**
        for n = 1 to NV
            Decoding and Evaluation;
            Mutation Operation;
            Recombination Operation;
            Selection operation;
            Self-adaptive control parameter;
        end
    end
end

Figure 2 Overall SADE algorithm procedure.

4. Computational experiments

The purpose of this study is to evaluate the efficiency and effectiveness of the SADE in TRSP. As part of the self-adaptive process for F and CR, DE parameters were set as self-adaptive control parameters with a probability of $\tau = 0.1$. The control parameters of traditional DE used for comparison were F=0.5, CR=0.9, population size = 100 and max iterations = 1500, as referenced from Jia et al. [19]. The proposed algorithms were implemented using Python and tested on a 2.38 GHz personal computer with 8 GB of RAM. The mathematical formulation was solved using Lingo program on the same setting. To provide an example, 10 instances were created (Table 1) with different average service times, numbers of customers, and numbers of service teams. The size of test problem is categorized as follows: Instance 1-4, 5-7, and 8-10 are small, medium, and large problem, respectively. The data used in Instance 11 is based on real case data. The penalty cost for each customer was determined by a range of values from 15 to 25. The service time for each customer was set at 90 minutes, and the time window for service was between 30 and 360 minutes. The overtime cost for the service team was also determined by a range of values from 6 to 10. The service teams began working at the start of the day and continued until 480 minutes had passed. The delay time must be less than 30 minutes. The maximum amount of overtime hours was limited to 120 minutes. The outsourcing service cost 20 units for each minute of service time.

Table 1 Test problem

Instance	No. of Customers	No. of Service teams	Avg. Service Time
1	10	4	60
2	14	4	83
3	17	4	80
4	20	4	80
5	25	7	76
6	30	7	75
7	35	7	75
8	45	12	74
9	55	12	72
10	65	12	78
11	63	10	73

Table 2 shows the distance between the depot and the customers. The service team is required to have 2 specific skills in order to fulfill the customer's needs. The operating costs for instance 1 are shown in Table 3.

The technician routing and scheduling problems were formulated as mixed integer programming models. The results for solving the small size problems using Lingo program are shown in Table 4.

Table 2 Distance matrix (km)

Customer	1	2	3	4	5	6	7	8	9	10	11
1	-	19	21	16	18	15	19	16	18	20	17
2	19	-	2	4	3	4	5	5	7	7	10
3	21	2	-	5	4	6	5	6	7	7	11
4	16	4	5	-	2	1	4	2	4	6	7
5	18	3	4	2	-	3	2	3	4	4	7
6	15	4	6	1	3	-	4	2	5	6	7
7	19	5	5	4	2	4	-	3	2	2	6
8	16	5	6	2	3	2	3	-	3	4	5
9	18	7	7	4	4	5	2	3	-	2	4
10	20	7	7	6	4	6	2	4	2	-	5
11	17	10	11	7	7	7	6	5	4	5	-

Table 3 Instance 1 results

Instance 1	Results
Operating cost	454
Travel cost	424
Penalty cost	30
Overtime cost	0
Subcontracting costs	0
Service teams used	Teams 2 and 3
Sequences of customers	-
Service team 2	1-10-2-4-3-5-7-1
Service team 3	1-11-8-9-6-1

Table 4 The operating costs of Lingo's solutions

Instance	Operating cost	Computational time [min]
1	454	2.29
2	1,000	4.47
3	1,112	52.42
4	3,561*	1,440

*Best solution found within limited computational time

The test instances 1 to 3 were successfully solved to find an optimal solution. The operating cost and computational time for each problem were generated and presented. For instance 1 through 4, the computational time increased significantly as the problem size increased, due to the complexity of the problem. In instance 4, the problem could not be solved in the limited time, the best operating cost found within 1440 minutes (24 hours) is 3,561. The TRSP problem was particularly difficult and complex for instances 5 through 11, as it involved a large number of variables. Additionally, the company only accepts computations that take less than 360 minutes (6 hours), which restricts the order in which each service team can visit a customer. The time used to obtain instances 5 to 11 by solving the mixed integer programming model (MILP) cannot be accepted. In order to satisfy the consumers requirement and to address both small and large problems, this study suggests that the metaheuristic (in this case, SADE) should be used to determines the order of customers that each service team should visit.

The performance of traditional DE and SADE was evaluated using 11 instances, the same as the MILP test. The results of DE and SADE for TRSP are displayed in Table 5. The computational experiments were conducted using a total of 10 runs to compare the best and average operating costs of each solution for each instance. For instances 1 and 2, DE and SADE found that the best operating cost was equal to the optimal solution. However, for instance 3, SADE produced the same result as the optimal solution, but DE did not. For instance 4, the optimal solution cannot be solved in the limited 1440 minutes and the best operating cost obtained from Lingo was used instead. For the rest of the test instances 4 to 11, the results show that SADE found the better solution compared to DE.

Table 5 Comparison of the best and average operating cost of each solution for each instance

Inst.	Solution by LINGO		Solution by DE				Solution by SADE			
	Best solution	**CPU time [min]**	**Avg.**	**Std.**	**Best solution**	**CPU time [sec]**	**Avg.**	**Std.**	**Best solution**	**CPU time [sec]**
1	**454**	2.29	458.6	9.34	**454**	4.31	454.8	1.67	**454**	3.00
2	**1,000**	4.47	1,015.9	12.53	**1,000**	17.38	1,004.5	7.35	**1,000**	6.15
3	**1,112**	52.42	1,155.3	35.61	1,117	36.10	1,144.7	18.59	**1,112**	45.91
4	3,561	1,440	3,828.6	223.40	3,510	47.11	3,583.0	191.11	**3,217**	55.21
5	-	-	2,332.1	190.79	2,003	50.74	2,049.8	72.92	**1,916**	28.18
6	-	-	4,033.3	288.06	3,631	57.92	3,148.8	201.07	**2,886**	86.41
7	-	-	9,220.5	609.37	7,678	54.30	8,212.2	530.90	**7,176**	118.42
8	-	-	8,879.9	471.38	8,029	234.40	7,486.4	414.14	**6,896**	72.55
9	-	-	16,838.0	767.41	15,343	298.93	15,135.1	732.80	**13,783**	304.16
10	-	-	32,112.6	672.07	31,222	279.13	30,186.8	706.74	**28,699**	330.15
11	-	-	34,099.80	637.60	32,943	296.48	33,003.87	726.01	**30,894**	260.55

The solutions obtained from SADE were compared with those from traditional DE using a *t*-test. Table 6 shows a *p*-value of less than 0.05, indicating that SADE is significantly different from traditional DE. The objective value of SADE was reduced by 7.16% compared to traditional DE. SADE outperformed traditional DE, providing better overall solutions and significantly reducing the company's operating costs, including the travel cost, penalty for late services cost, overtime cost, and subcontracting cost.

Table 6 Results of *t*-test

Sample	N	Mean	St. Dev	SE Mean	%Diff	*p*-value
DE	11	9,721	11,880	3,582	7.16	0.013
SADE	11	8,912	11,037	3,328		

5. Conclusions

One of the main business challenges of a service organization is making the best use of the available technicians. Additionally, services are frequently provided at customers' locations, requiring the technicians to travel to different locations. This research proposed a mixed integer programming formulation and a Self-Adaptive Differential Evolution (SADE) algorithm to identify an efficient route for sequencing customers to be serviced by each service teams an enhanced version of the Differential Evolution (DE) algorithm. It uses an adaptive auto-tuning method to adjust the parameters of the mutation and recombination operators during the optimization process. Numerical experiments were conducted to evaluate the efficiency and effectiveness of SADE in TRSP. According to the experimental results, SADE outperformed DE by about 7.16%. On average, SADE improved the results across all problem instances. These findings indicate that the proposed approach is effective and practical for use in the installation and maintenance sector.

In the future, it would be useful to extend the proposed SADE approach to include more constraints, such as multi-visit, multi-period, and team building in a real-world setting, as well as dynamic customers, a job splitting constraint, and multiple depots. In addition, further research could be conducted to develop approaches for addressing multiple objectives in the TRSP.

6. Acknowledgements

This work was supported by the Research Unit on System Modeling for Industry (Grant No. SMI.KKU 63003), Khon Kaen University, Thailand and Rajamangala University of Technology Krungthep, Thailand. The authors would also like to thank Mr. Ian Thomas for English language editing of the manuscript.

7. References

[1] Anoshkina Y, Meisel F. Technician teaming and routing with service-, cost-and fairness-objectives. Comput Ind Eng. 2019;135:868-80.

[2] Çakırgil S, Yücel E, Kuyzu G. An integrated solution approach for multi-objective, multi-skill workforce scheduling and routing problems. Comput Oper Res. 2020;118:104908.
[3] Castillo-Salazar JA, Landa-Silva D, Qu R. Workforce scheduling and routing problems: literature survey and computational study. Ann Oper Res. 2016;239:39-67.
[4] Pereira DL, Alves JC, de Oliveira Moreira MC. A multiperiod workforce scheduling and routing problem with dependent tasks. Comput Oper Res. 2020;118:104930.
[5] Zamorano E, Stolletz R. Branch-and-price approaches for the multiperiod technician routing and scheduling problem. Eur J Oper Res. 2017;257(1):55-68.
[6] Cordeau JF, Laporte G, Pasin F, Ropke S. Scheduling technicians and tasks in a telecommunications company. J Sched. 2010;13:393-409.
[7] Irawan CA, Ouelhadj D, Jones D, Stålhane M, Sperstad IB. Optimisation of maintenance routing and scheduling for offshore wind farms. Eur J Oper Res. 2017;256(1):76-89.
[8] Pillac V, Guéret C, Medaglia AL. A fast reoptimization approach for the dynamic technician routing and scheduling problem. In: Amodeo L, Talbi EG, Yalaoui F, editors. Recent Developments in Metaheuristics. Cham: Springer; 2018. p. 347-67.
[9] Mathlouthi I, Gendreau M, Potvin JY. A metaheuristic based on tabu search for solving a technician routing and scheduling problem. Comput Oper Res. 2021;125:105079.
[10] Sethanan K, Jamrus T. Hybrid differential evolution algorithm and genetic operator for multi-trip vehicle routing problem with backhauls and heterogeneous fleet in the beverage logistics industry. Comput Ind Eng. 2020;146:106571.
[11] Kovacs AA, Parragh SN, Doerner KF, Hartl RF. Adaptive large neighborhood search for service technician routing and scheduling problems. J Sched. 2012;15:579-600.
[12] Pillac V, Gueret C, Medaglia AL. A parallel matheuristic for the technician routing and scheduling problem. Optim Lett. 2013;7:1525-35.
[13] Pinheiro RL, Landa-Silva D, Atkin J. A variable neighbourhood search for the workforce scheduling and routing problem. In: Pillay N, Engelbrecht A, Abraham A, du Plessis M, Snášel V, Muda A, editors. Advances in Nature and Biologically Inspired Computing. Advances in Intelligent Systems and Computing. Cham: Springer; 2016. p. 247-59.
[14] Pekel E. Solving technician routing and scheduling problem using improved particle swarm optimization. Soft Comput. 2020;24:19007-15.
[15] Punyakum V, Sethanan K, Nitisiri K, Pitakaso R, Gen M. Hybrid differential evolution and particle swarm optimization for multi-visit and multi-period workforce scheduling and routing problems. Comput Electron Agric. 2022;197:106929.
[16] Punyakum V, Sethanan K, Nitisiri K, Pitakaso R. Hybrid particle swarm and whale optimization algorithm for multi-visit and multi-period dynamic workforce scheduling and routing problems. Mathematics. 2022;10(19):3663.
[17] Storn R, Price K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim. 1997;11:341-59.
[18] Chen CA, Chiang TC. Adaptive differential evolution: a visual comparison. 2015 IEEE Congress on Evolutionary Computation (CEC); 2015 May 25-28; Sendai, Japan. USA: IEEE; 2015. p. 401-8.
[19] Jia L, Gong W, Wu H. An improved self-adaptive control parameter of differential evolution for global optimization. In: Cai Z, Li Z, Kang Z, Liu Y, editors. Computational Intelligence and Intelligent Systems. Berlin: Springer; 2009. p. 215-24.