Computers and Operations Research 125 (2021) 105079

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/caor

A metaheuristic based on tabu search for solving a technician routing and scheduling problem

Ines Mathlouthi [a,c], Michel Gendreau [b,c], Jean-Yves Potvin [a,c,*]

[a] Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succ. Centre-Ville, H3C 3J7 Montréal, Québec, Canada
[b] Département de mathématiques et de génie industriel, Polytechnique Montréal, C.P. 6079, succ. Centre-Ville, H3C 3A7 Montréal, Québec, Canada
[c] Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, C.P. 6128, succ. Centre-Ville, H3C 3J7 Montréal, Québec, Canada

ARTICLE INFO

Article history:
Received 25 July 2020
Accepted 4 August 2020
Available online 20 August 2020

Keywords:
Technician routing and scheduling
Tabu search
Adaptive memory
Biased fitness

ABSTRACT

This paper addresses a technician routing and scheduling problem motivated by an application for the repair and maintenance of electronic transactions equipment. The problem exhibits many special features like multiple time windows for service, an inventory of spare parts carried by each technician and tasks that may require a special part to be performed. A problem-solving methodology based on tabu search, coupled with an adaptive memory, is proposed. The inclusion of solutions (local minima) in the adaptive memory takes into account both quality and diversity. Results are reported on test instances with up to 200 tasks. A comparison with a previously developed branch-and-price algorithm is also reported on instances of small size.

1. Introduction

This paper considers a Technician Routing and Scheduling Problem (TRSP) motivated by a real application for the repair or maintenance of electronic transactions equipment by a number of technicians. The problem is to assign a subset of tasks to each technician and to construct a route over each subset to optimize some objective, which may involve many criteria (like total traveled distance, overtime, etc.). The routes must also satisfy different types of constraints. First, there are compatibility constraints between tasks and technicians, since different skills are required to perform different tasks and a technician does not necessarily possess all those skills. Second, a task may also need a number of spare parts. If the technician does not have the required parts in his initial inventory when he starts his route, he can acquire them by visiting a depot at some point along the route. Typically, an infinite number of parts is assumed at the depot, although only a finite number can be carried by the technician. Third, a task may require a special part which is particularly expensive and can only be obtained by visiting the depot (i.e., a technician is not allowed to carry a special part from or to his home base location). Fourth, there are time bounds for the service of each task and for the return time of each technician at his home base location.

It is also assumed that not all tasks can be served by the technicians. Thus, a gain is associated with each task and the maximization of the total gain collected along the routes becomes part of the objective. The remaining tasks can then be considered for another day, with typically an increased gain (to avoid being left apart repeatedly). Hence, the gain is related to characteristics of the task or the customer who requests the service.

The remainder of the paper is organized as follows. In Section 2, we provide a review of various works on problems related to ours. In Section 3, we describe our problem. Then, after an introduction to recent ideas that have been integrated into tabu search to improve its performance, we detail our algorithm in Section 4. The test instances are introduced and are followed by computational results in Section 5. In particular, we provide a comparison with optimal solutions previously obtained with an exact method on instances of small sizes. Finally, a conclusion follows in Section 6.

2. Literature review

Most papers on the TRSP are based on real applications that exhibit specific features. To the best of our knowledge, the first work on this type of problem was reported in 1997 for a telecommunications application (Tsang and Voudouris, 1997). To solve this problem, the authors used different types of local search heuristics. Weigel and Cao (1999) then introduced a problem faced by a large retailer when providing on-site technical assistance. The authors solved the problem by developing a heuristic procedure to construct a route for each technician and by improving each route individually with Or-opt exchanges (Or, 1976). In Blakeley et al. (2003), the authors addressed a periodic maintenance problem for elevators and escalators. In this application, technician routes

* Corresponding author.

https://doi.org/10.1016/j.cor.2020.105079
0305-0548/

had to account for work regulations. A similar application was later addressed by Tang et al. (2007) with a tabu search heuristic. In 2007, the French Operations Research Society (ROADEF) initiated a challenge based on a problem encountered by France Telecom. This problem first involves a multi-period horizon. Also, teams of technicians must be created because each task needs multiple skills with different proficiency levels. For this challenge, Hashimoto et al. (2011) developed a greedy randomized adaptive search procedure (GRASP) while Cordeau et al. (2010) proposed and adaptive large neighborhood search (ALNS). Another multi-period TRSP for the maintenance of electric forklifts, where technicians are paired to form teams, is solved with a branch-and-price algorithm in Zamorano and Stolletz (2017).

A parallel matheuristic is proposed in Pillac et al. (2013) for a TRSP where tools and spare parts are taken into account. The matheuristic is made of a constructive heuristic, a parallel ALNS and a mathematical programming-based post-optimization procedure. In Mendoza et al. (2017) a technician routing problem with conventional and electric vehicles is introduced. Due to their relatively limited driving range, the electric vehicles need to visit one or more recharging stops along their route. The problem is also addressed with a parallel matheuristic, where a number of subproblems are first created and solved with GRASP. The routes of all local minima produced by GRASP are collected to create a repository of routes. Then, a set covering model is solved over these routes.

In Bostel et al. (2008), the authors introduced a problem faced by a water treatment and distribution company. Here, the technician routes must be planned over a period of one week for repair or maintenance. The tasks to be scheduled can either be known in advance or can occur dynamically. A memetic algorithm is used to solve the problem, which is first applied on the static tasks to produce initial tours for each day of the week. Then, the dynamic tasks are integrated into the current routes as they occur, still with the memetic algorithm. An exact approach based on column generation is also proposed in this work, but can only be applied to instances of small size. Pillac et al. (2012) also addressed a TRSP in which a fraction of the tasks occur dynamically. A parallel architecture is proposed to speed up the calculations. An initial solution is first created with the static tasks using a construction heuristic based on a regret measure (Potvin and Rousseau, 1993). This solution is then improved with ALNS (Pisinger and Ropke, 2007). The latter algorithm works by successively destroying (removing tasks) and repairing (reinserting tasks) the current solution to produce a new solution. For dynamic tasks, the part of the current solution already executed is fixed and the newly occurring tasks are incorporated into the solution by running the ALNS for a limited number of iterations. The same authors also proposed a fast reoptimization approach based on a parallel ALNS in Pillac et al. (2018).

In Mathlouthi et al. (2018), a mixed integer linear programming (MILP) model was proposed for our TRSP. Although the model is useful by providing a formal description of the problem, solving the model exactly with a commercial solver proved to be impractical, except for very small instances with no more than 15 tasks. In another work (Mathlouthi et al., 2017), a branch-and-price algorithm was developed and was able to solve exactly instances with up to 45 tasks. Given the limitations of exact approaches, we now propose a tabu search heuristic to address instances of larger sizes.

3. Problem definition

We start this section by describing the main entities, with their attributes, involved in our problem (for a formal MILP formulation, see Mathlouthi et al. (2018)). They are:

- Tasks
  - Parts: each task may require one or more spare parts of different types, as well as a special part, to be performed.
  - Gain: a gain is associated with each task that represents its importance (based on service priority, revenue, customer status, etc.).
  - Multiple time windows: a technician must begin his service within one of multiple time windows associated with task i, where a time window is defined by a lower bound e_i and an upper bound l_i. If the technician arrives before the lower bound, he can wait and start the service at the lower bound.
- Technicians
  - Home base: this is the starting and ending location of the technician's route.
  - Skills: each technician has one or more skills that allow him to serve certain tasks and not others.
  - Depot: each technician is a priori assigned to a single depot (among a number of possible depots) where he can replenish his inventory of parts.
  - Workday: each technician normally works between 9:00 AM and 5:00 PM, although overtime is allowed (at the expense of a penalty in the objective). Three breaks are taken during the day: two breaks of 15 min in the morning and afternoon and a mid-day break of 30 min. Each break must be taken within a specific time window. It should also be noted that the route of a technician cannot exceed a maximum traveled distance.
  - Inventory: each technician carries an initial inventory of spare parts when he leaves his home base location. If there are not enough spare parts to serve all tasks along a technician's planned route, or if one or more tasks require a special part, then a single visit to a preassigned depot is allowed along the route.
- Parts
  - Spare parts: there are different types of spare parts. Although the depot contains a virtually infinite number of spare parts, the technician can only carry a limited number of parts of each type.
  - Special parts: special parts are also available at the depot. A technician must visit the depot when one or more tasks along his route require a special part. No special part can be carried to or from the home base location.
- Depots

  There is a fixed number of depots with a virtually infinite number of parts.

The problem is to design a route over a subset of tasks for each of a fixed number of technicians while satisfying the hard constraints mentioned above, namely, the capacity (skills) of a technician and the required parts to perform a task, the multiple time windows for the service of a task and the maximum traveled distance of each route. Given that not all tasks can be served, the objective is to maximize the total gain collected, minus the total traveled distance and total overtime over all routes (or, equivalently, to minimize the sum of total traveled distance and total overtime minus the total gain collected). Each component in the objective has a weighting parameter to adjust its importance when evaluating a solution.

4. Problem-solving methodology

Our problem-solving methodology is based on tabu search (Glover, 1989) which has proven successful for a variety of hard

combinatorial problems, like vehicle routing (Gendreau et al., 1999; Gendreau et al., 1994), job shop scheduling (Kawaguchi and Fukuyama, 2016), quadratic assignment (Tabitha et al., 2009), technician routing and scheduling (Tang et al., 2007), and many others. Modern implementations involve the integration of adaptive memories (Rochat and Taillard, 1995) made of elite solutions or components of elite solutions in order to perform search intensification or search diversification. An example for a vehicle routing problem can be found in Taillard et al. (1997) where the routes of elite solutions are stored in memory and used to construct new starting solutions for the tabu search. Basically, a new solution is obtained by mixing routes from different elite solutions in memory. In our tabu search, we manage the adaptive memory as a true population of solutions with considerations for both quality and diversity. This approach has proven very successful in the hybrid genetic algorithm of Vidal et al. (2014) when solving different types of vehicle routing problems.

The proposed metaheuristic is shown in Algorithm 1, where s stands for the current solution and s^* for the best solution. It starts with a number of calls to the function Greedy(). The latter randomly applies one of two greedy construction heuristics to create an initial solution which is then improved with a local search descent. The resulting solutions are then filtered out to remove duplicates and are stored in adaptive memory. The best solution in adaptive memory is then used as the initial starting solution. The outer **while** loop is aimed at stopping the algorithm and returning the best solution s^* when the maximum allowed computation time has been consumed. Each iteration of the inner **while** loop involves the application of four consecutive tabu searches, each with a different neighborhood structure. The best solution found by a tabu search with a given neighborhood structure is returned and fed to the next one. Each tabu search also takes care to update s^* during its execution, if required. The stopping criterion for any given tabu search is based on a maximum number of iterations. Note also that every local minimum reached by a tabu search is stored in adaptive memory. The inner **while** loop is repeated until a complete pass through the four neighborhoods does not provide any improvement. At this point, the adaptive memory is filtered out to remove duplicates and is possibly reduced if the maximum memory size n_{max} is exceeded. Then, a new starting solution is created by combining routes of different solutions in adaptive memory. In the following, the main components of this search framework are described.

Algorithm 1

```
1: procedure SEARCH
2: for n_init iterations
3:         s_init ← Greedy ()
4:         s_init ← Local Search (s_init)
5:         Store s_init in adaptive memory
6:     s* ← select best solution in adaptive memory
7:     s ← s*
8: while Computation Time < T_max (or any other
   stopping criterion)
9:     while s is improved
10:        s ← Tabu (s, Inter-Route Move)
11:        s ← Tabu (s, Intra-Route Move)
12:        s ← Tabu (s, Swap)
13:        s ← Tabu (s, Swap-With-New)
14:    Update memory
15:    s ← Get solution from adaptive memory
return s*
```

4.1. Construction heuristics

The adaptive memory is initialized with n_{init} different solutions created with two randomized construction heuristics, each contributing to half of the solutions. These solutions are then improved with a local descent based on the same neighborhood structures than the ones used in the tabu search (see Section 4.2). In our experiments, n_{init} was set to ten times the number of technicians. The two construction heuristics are the following.

4.1.1. Sequential construction heuristic

In this heuristic, the routes of the technicians are constructed one by one. That is, as long as technicians are available, one of them is randomly selected and his route is initially created with a starting and ending location (home base location of the technician) and the three breaks. Then, a task is randomly selected among those that can be performed by the technician. The three best feasible insertions in the current route of the technician are considered and one of them is randomly selected. This is repeated until no new task can be added to the route. In case the current insertion requires a visit to the depot, the three best feasible insertions for the depot are kept by creating three different routes. The insertion procedure then follows with the three routes and the best route is selected at the end. By keeping a number of different routes, each associated with a different insertion place for the depot, the myopic behavior of the construction heuristic is alleviated.

4.1.2. Parallel construction heuristic

This heuristic constructs routes for all technicians in parallel. The route of each technician is first initialized with his starting and ending locations and the three breaks. Then, a task is randomly selected and the best feasible insertion in the route of each technician who can perform it is considered. The task is assigned to the best technician and inserted in his route. This is repeated until no more tasks can be feasibly inserted in the routes. During this insertion procedure, the depot is handled as in the sequential heuristic.

4.2. Tabu search

The tabu search improves the starting solutions obtained from the adaptive memory. Its solution space and neighborhood structures are now described.

4.2.1. Solution space

In the following, a solution is considered feasible (and is part of the solution space) even if the maximum traveled distance constraint is not satisfied. In such a case, the solution is immediately repaired. The repair procedure is applied in turn to each route that exceeds the maximum distance constraint. This procedure is very simple and removes, at each iteration, the task with the least impact on the objective value until the maximum distance constraint is satisfied again.

4.2.2. Neighborhoods

The tabu search exploits a sequence of four different neighborhood structures. More precisely:

- *Inter-Route Move*. A task is moved from one route to another and inserted at the best feasible insertion place.
- *Intra-Route Move*. A task is moved from its current position to another feasible position in the same route.
- *Swap*. Two tasks from two different routes are swapped. Each task is inserted at the best feasible insertion place in its new route.
- *Swap-With-New*. A task not currently in the solution is swapped with a task in the solution and is inserted at the best feasible insertion place.

After a move, each task not currently in the solution is considered for insertion in the routes that have been modified (in non increasing order of gain). During the exploration of each neighborhood, a first improvement strategy is implemented. Thus, the first feasible neighbor that provides an improvement over the current solution is chosen. Tabu restrictions are also imposed on each neighborhood. That is, tasks involved in recent moves are declared tabu for θ iterations (unless they can improve the best known solution), where θ is randomly selected in the interval $[\theta_{min}, \theta_{max}]$, with $\theta_{min} = 5$ and $\theta_{max} = 10$ in our experiments.

It should be noted that the algorithm loops through the four neighborhoods (in the order indicated above) as long as the solution improves. If a pass through the four neighborhoods does not provide any improvement, then a new initial solution is fetched from the adaptive memory to restart the search.

4.3. Adaptive memory

Here, we describe the procedures used to store and fetch solutions from the adaptive memory. Since every solution decomposes into a route per technician, is should be noted that the memory is implicitly divided into a number of smaller memories, where each memory contains the routes of a given technician (see Fig. 1). This partition is required during the fetching procedure because each technician has specific routes that start from and end at a different home base location and visit a particular preassigned depot. In the following, we first explain how the biased fitness of a solution in adaptive memory is computed, before describing the storing and fetching mechanisms.

4.3.1. Biased fitness

The quality of a solution in memory corresponds, on the one hand, to its objective value and, on the other hand, to its contribution to the diversity of solutions in that memory (Vidal et al., 2014). The diversity computation is based on the Hamming distance between two solutions s_1 and s_2, see Eq. (1). In this equation, $\Gamma(x)$ is an indicator function: it returns 1 if x is true, 0 otherwise; $T(i,s)$ returns the technician who performs task i in solution s, while $W(i,s)$ returns the index of the time window for the service of task i in solution s (since there are multiple time windows). When task i is not served by any technician, a dummy value is returned.

$$\delta^H(s_1, s_2) = \sum_{i \in s_1 \cup s_2} \Gamma(T(i,s_1) \neq T(i,s_2)) + \Gamma(W(i,s_1) \neq W(i,s_2)) \quad (1)$$

The diversity contribution of solution s is then computed as follows:

$$\Delta(s) = \frac{1}{n_c} \sum_{s\prime \in N_c} \delta^H(s, s\prime) \quad (2)$$

Thus, it is the average of the Hamming distances between s and the $n_c = |N_c|$ closest solutions in adaptive memory, where n_c is a parameter (set to 20% of the solutions in adaptive memory, as in Vidal et al., 2014). Then, the ranks $r_f(s)$ and $r_d(s)$ of each solution s in memory with regard to the objective value and diversity contribution, respectively, are computed (where rank 1 is best). With these ranks, the biased fitness BF of solution s is computed as follows, where n_m is the current number of solutions in memory:

$$BF(s) = (n_m - r_f(s) + 1) + \eta(n_m - r_d(s) + 1) \quad (3)$$

In this formula, the weight of the objective value component (first term in the summation) is implicitly set to 1 to guarantee a minimum contribution of this component to the biased fitness. Then, a more or less important bias towards diversity is added depending on the value of parameter $\eta \in [0, 1]$.

4.3.2. Storing and updating

Every local minimum visited by a tabu search, either based on the inter-route, intra-route, swap or swap-with-new neighborhood, is added to the adaptive memory. After one pass through the four neighborhoods (see Algorithm 1), the memory is updated as follows:

- duplicates (i.e., solutions with the same objective value) are removed;
- the biased fitness of each solution is computed;
- if the maximum memory size n_{max} is exceeded, the n_{max} solutions with the best biased fitness are kept. In our experiments, this parameter was set to one hundred times the number of technicians.

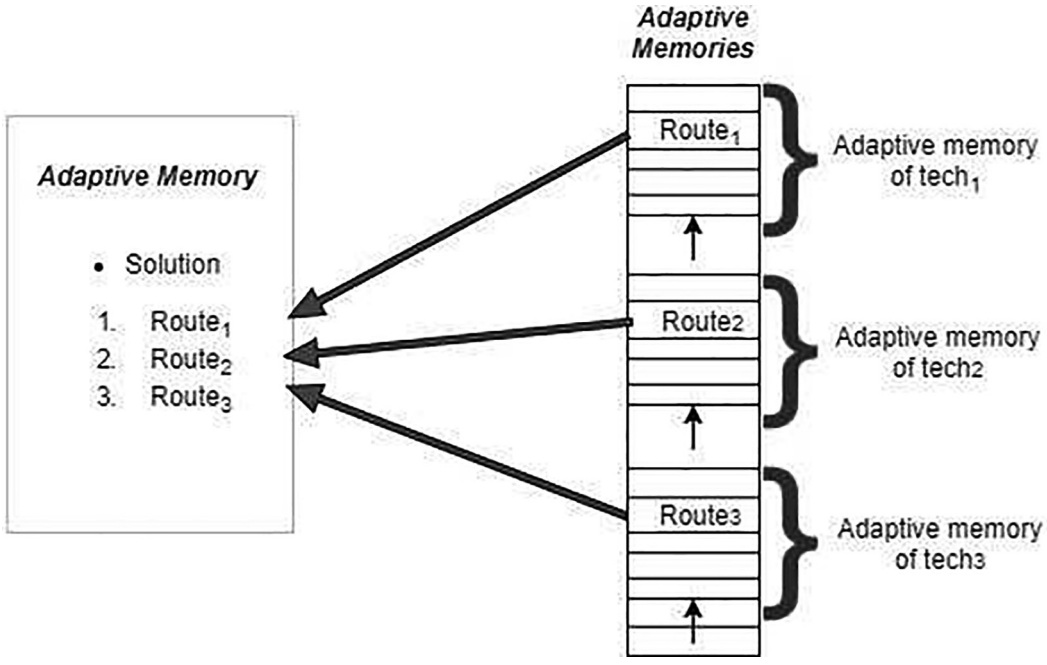

Fig. 1. Adaptive memories.

4.3.3. Fetching

As previously mentioned, each solution is decomposed into a route for each technician, where each route is stored in the memory of the corresponding technician. In this process, the route also inherits the biased fitness value of the whole solution.

The procedure to create a starting solution for the tabu search is the following. We proceed technician by technician. A first technician is considered and each route in the memory of this technician is assigned a probability of selection which is proportional to its biased fitness (roulette wheel). A route r is probabilistically chosen and included in the starting solution. Then, the memory of all other technicians is updated by removing any route with a task in common with route r. This process of (1) considering the next technician, (2) probabilistically selecting a route of that technician, (3) adding this route to the starting solution and (4) updating the memory of all remaining technicians, is repeated until all technicians are done (a complete starting solution is obtained) or until the remaining technicians have no valid routes in memory. In the latter case, the solution is completed by inserting additional tasks with the parallel construction heuristic of Section 4.1.

5. Computational experiments

In this section, the test instances for the experiments are first described. Then, we report the results obtained with four different variants of our metaheuristic. We also compare the solutions produced by our algorithm with the optimum on instances of small size.

5.1. Test instances

The test instances come from Mathlouthi et al. (2018), where the crow fly distance is assumed between each pair of locations and the speed of the vehicles is set at 50 km/h. The other characteristics are the following.

- *Service area*. The service area is a 40 km $\times$ 40 km or 50 km $\times$ 50 km squared area.
- *Depot Location*. There are 3 depots randomly located within the service area.
- *Task location*. Each task is randomly located within the service area.
- *Task service time*. The service time of each task is randomly chosen between 30 and 45 min.
- *Task gain*. The gain associated with a task is randomly chosen between 1 and 10.
- *Technician home base*. To get a good coverage of the tasks, the first two technicians are located at the opposite ends of the service area (along the diagonal). The other technicians are randomly located within the service area.
- *Technician skills*. With each technician is associated the percentage of tasks that he can perform: one third of the technicians can perform all tasks, one third of the technicians can perform 50% of the tasks and one third 25% of the tasks.
- *Parts*. The number of spare parts needed to perform a task is randomly chosen between 0 and 3. Then, each spare part is assigned a type, among 4 different types. Also, a special part is needed with a probability of 0.125.
- *Time windows*. Both narrow and wide time windows are considered. The latter are twice as wide as the former on average. The length of a narrow time window is randomly generated between 60 and 90 min. The lower bound of the first time window is chosen randomly between 9:00 AM and noon. The lower bounds of the remaining time windows are set between 2 and 3 h after the upper bound of the previous time window until a maximum of 3 time windows are obtained.
- *Maximum distance*. The maximum distance traveled by each technician during his workday is set to 125 km.

Different subsets of instances were obtained by varying the values of the following characteristics: size of the service area (either 40 km $\times$ 40 km or 50 km $\times$ 50 km), width of the time windows (either narrow or wide) and number of tasks (50, 100 or 200). Furthermore, tests were performed with 3 and 6 technicians for instances with 50 tasks, 6 and 12 technicians for instances with 100 tasks, and 12 and 24 technicians for instances with 200 tasks. Thus, we have $2 \times 2 \times 3 \times 2 = 24$ subsets of instances, with 10 instances in each subset. In the tables of results, the subsets are identified with the following fields: width of the time windows (either N for narrow or W for wide), size of the service area (either 40 or 50), number of tasks and number of technicians. For example N-40–50-3 is the subset of instances with narrow time windows, a 40 km $\times$ 40 km service area, 50 tasks and 3 technicians.

Finally, the weights in the objective function were set as follows: 500 for the gain, 5 for the distance (in kilometers) and 1 for overtime (in seconds). Overall, more emphasis is given to the total gain over the total distance and overtime. This setting allows a technician to do some overtime to perform a high gain task and is more challenging than the setting proposed in Mathlouthi et al. (2018).

5.2. Results

Here, we report the results of different experiments on a 3.07 GHz Intel Xeon X5675 processor, using the test instances introduced above. Our problem-solving methodology was run for T_{max} = 1 h on the instances of size 50 and 100, which was enough to allow convergence (see Fig. 2 for a typical evolution of the best solution on a particular instance of size 100, where the number of iterations refers to the outer while loop in step 8 of Algorithm 1). For the instances of size 200, a computation time of T_{max} = 2 h was required.

5.3. Parameter settings

The parameter settings were obtained with the irace package López-Ibáñez et al. (2016) based on one randomly chosen instance in each subset, for a total of 24 instances. The parameters are shown in Table 1 with their corresponding types and intervals, as well as the best values returned by irace. It should be noted that the memory size is the number of routes stored for one technician only. Thus, this number should be multiplied by the number of techicians to obtain the total memory size.

5.3.1. Impact of parameter η

In this section, we explore the impact of parameter η on the performance of our metaheuristic. This parameter controls the magnitude of the diversity contribution when a solution is evaluated in adaptive memory (biased fitness). When η = 0, only the objective value of a solution is considered. Conversely, when η = 1, the objective value and the diversity contribution have the same weight. We consider here six different values for η, including the best value 0.2, while keeping the other parameters at their best values found by irace. Table 2 reports, for each subset of instances, the average gap in percentage with the best known solution for each instance. These results show that including diversity has a positive impact, since the worst gaps are clearly obtained with η = 0. The best value $\eta = 0.2$ has an average gap of 1.6%, as indicated in the line Overall of Table 2.

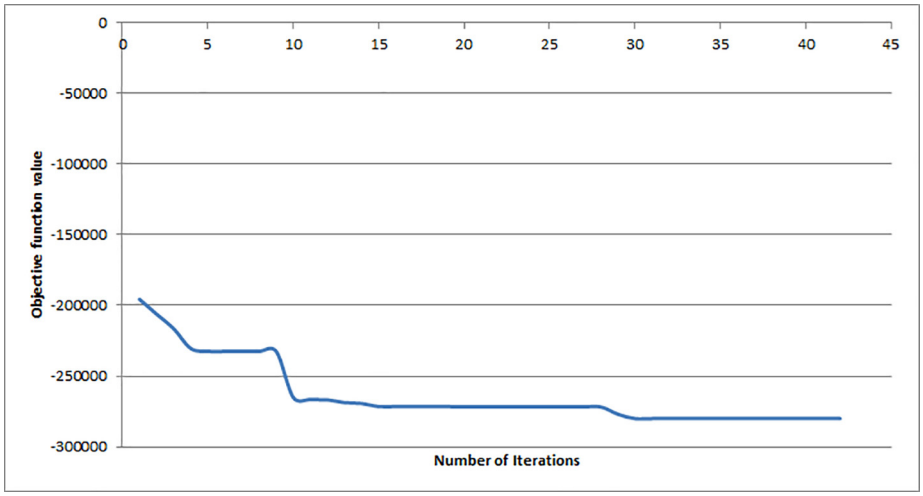

Fig. 2. Evolution of best solution.

Table 1
Parameter settings.

Parameter	Meaning	Type	Interval	IRACE
η	Population diversity	Real	[0, 1]	0.2
Θ_{min}	Lower bound of tabu tenure	Integer	[5, 10]	8
Θ_{max}	Upper bound of tabu tenure	Integer	[10, 20]	13
n_{init}	Initial memory size	Integer	[1, 10]	5
n_{max}	Maximum memory size	Integer	[10, 150]	110
n_{inter}	Number of Inter-Route moves	Integer	[10, 150]	104
n_{intra}	Number of Intra-Route moves	Integer	[10, 150]	131
n_{swap}	Number of Swap moves	Integer	[10, 150]	123
n_{swap-N}	Number of Swap-with-New moves	Integer	[10, 150]	123

Table 2
Impact of parameter η.

Name	Gap (%)					
	$\eta = 1$	$\eta = 0.8$	$\eta = 0.6$	$\eta = 0.4$	$\eta = 0.2$	$\eta = 0$
N-40-50-3	1.85	1.76	1.81	1.38	1.07	2.12
N-40-50-6	2.43	2.17	1.36	2.12	0.81	2.83
N-50-50-3	1.50	1.46	1.39	2.18	1.59	1.59
N-50-50-6	1.70	1.39	1.42	1.87	2.05	1.99
W-40-50-3	1.22	1.66	2.56	2.16	1.52	2.63
W-40-50-6	2.58	2.50	1.82	2.56	1.66	2.62
W-50-50-3	1.10	1.37	2.33	0.96	2.00	1.46
W-50-50-6	2.65	1.95	2.14	2.39	1.90	3.17
Avg.	1.88	1.78	1.85	1.95	1.58	2.30
N-40-100-6	1.80	1.37	1.05	1.01	0.90	1.63
N-40-100-12	1.76	1.74	1.90	1.86	1.97	2.03
N-50-100-6	1.56	1.73	1.34	1.80	2.49	2.27
N-50-100-12	1.46	1.85	0.97	1.27	1.91	1.63
W-40-100-6	1.46	1.35	1.68	0.94	1.08	1.25
W-40-100-12	1.61	2.20	1.63	2.12	1.57	1.83
W-50-100-6	1.57	2.55	3.64	1.73	1.31	2.99
W-50-100-12	1.36	1.28	1.21	1.27	1.57	1.32
Avg.	1.57	1.76	1.68	1.50	1.60	1.87
N-40-200-12	1.56	1.28	1.06	1.77	1.53	1.67
N-40-200-24	0.80	1.43	1.32	1.24	1.21	0.74
N-50-200-12	3.36	2.26	1.96	2.68	1.53	3.11
N-50-200-24	2.95	2.87	3.68	1.06	1.89	4.65
W-40-200-12	2.79	1.72	1.84	2.89	1.64	2.76
W-40-200-24	0.71	2.73	2.41	2.05	1.00	2.71
W-50–200-12	5.32	1.63	3.19	3.19	2.78	2.35
W-50-200-24	1.32	3.09	3.49	2.26	1.33	3.15
Avg.	2.35	2.13	2.37	2.14	1.61	2.64
Overall	1.93	1.89	1.97	1.86	1.60	2.27

5.3.2. Comparison with other variants

In this section, we compare our algorithm, called *TS-AM-I*, with four other variants:

TS-AM. In this variant with adaptive memory, the maximum traveled distance cannot be exceeded when exploring the neighborhood of the current solution. Thus, a neighbor solution is ignored if the maximum distance of one or more technician routes is exceeded.

TS-I. This variant is obtained by removing the adaptive memory (AM) from *TS-AM-I*. One of the two greedy construction heuristics is applied, through a call to Greedy(), when it is time to get a new starting solution.

TS. This variant is obtained by removing the adaptive memory (AM) from *TS-AM*. One of the two greedy construction heuristics is applied, through a call to Greedy(), when it is time to get a new starting solution.

Multi-Start. This variant is obtained by replacing each tabu search by a multi-start procedure in *TS-AM-I*. That is, a new starting solution is fetched from the adaptive memory as soon as a local optimum is reached. As a consequence, only improving moves are allowed.

Table 3 reports, for each subset of 10 instances and each variant, the number of times a variant found the best solution over the four variants (# Bst) and the average gap in percentage over the best solutions.

These results demonstrate the importance of considering solutions that exceed the maximum traveled distance, as well as including the adaptive memory with biased fitness in our problem-solving methodology. In fact, *TS-AM-I* found the best solution for 87% of the instances, as indicated in the line Overall of Table 3 (c.f., 8.7). A Wilcoxon signed-rank test between the gaps of *TS-AM-I* and those of its closest competitor *TS-AM* has also been performed using the null hypothesis *TS-AM-I* = *TS-AM* versus the alternative hypothesis *TS-AM-I* $<$ *TS-AM*. The test statistically demonstrated the superiority of *TS-AM-I* with a p-value of 0.0003, which is well below the standard significance level of 0.05.

5.3.3. Served tasks

Table 4 provides a picture of the average number (Ta) and fraction (Ta/All) of served tasks, as well as the average number of tasks served per technician (Ta/Te) for configurations with less or more technicians (Te). For instances with 50 tasks, 3 technicians can serve 42% of the tasks in the case of the smaller service area, with an average of 6.7 tasks in each route. This percentage decreases for

Table 3
Comparison of different methods.

Name	Multi-Start		TS		TS-I		TS-AM		TS-AM-I	
	# Bst	Gap (%)	# Bst	Gap (%)	# Bst	Gap (%)	# Bst	Gap (%)	# Bst	Gap (%)
N-40-50-3	0	31.9	0	22.3	0	15.2	0	10.0	10	0.0
N-40-50-6	0	43.1	0	15.2	0	10.2	2	3.1	8	0.3
N-50-50-3	0	37.3	0	22.0	0	18.8	0	19.4	10	0.0
N-50-50-6	0	51.6	0	19.2	1	14.7	0	4.8	9	0.1
W-40-50-3	0	30.5	0	12.5	0	23.1	3	12.7	7	0.8
W-40-50-6	0	41.0	0	9.3	0	5.6	3	3.3	7	0.6
W-50-50-3	0	40.2	0	24.0	0	28.0	0	18.9	10	0.0
W-50-50-6	0	51.4	0	16.4	0	12.8	0	4.8	10	0.0
Avg.	0	40.9	0	17.6	0.1	16.1	1.0	9.6	8.9	0.2
N-40-100-6	0	29.4	0	11.7	0	10.4	0	8.5	10	0.0
N-40-100-12	0	28.3	0	8.7	0	6.8	0	3.0	10	0.0
N-50-100-6	0	29.7	0	27.7	0	12.7	2	10.4	8	0.2
N-50-100-12	0	39.1	0	11.7	2	5.3	0	10.2	8	5.0
W-40-100-6	0	27.3	0	13.9	0	7.5	0	7.4	10	0.0
W-40-100-12	0	27.4	0	9.6	1	3.9	0	5.2	9	1.0
W-50-100-6	0	30.1	0	13.2	2	10.3	1	3.7	7	0.1
W-50-100-12	0	36.5	0	8.1	2	3.7	3	2.6	5	1.3
Avg.	0	31.0	0	13.1	0.9	7.6	0.8	6.4	8.4	0.1
N-40-200-12	0	13.8	0	7.9	0	8.0	2	8.2	8	0.4
N-40-200-24	0	21.3	0	8.0	0	1.0	0	1.2	10	0.0
N-50-200-12	0	23.7	0	8.6	1	1.4	0	1.7	9	1.0
N-50-200-24	0	30.3	0	6.6	0	0.7	1	3.2	9	1.0
W-40-200-12	0	27.7	0	6.5	1	7.26	0	5.9	9	1.1
W-40-200-24	0	32.4	1	1.9	0	1.9	0	1.9	9	2.4
W-50-200-12	0	39.3	1	9.1	0	8.1	1	7.2	8	2.1
W-50-200-24	0	39.7	0	1.8	2	0.5	0	1.5	8	2.3
Avg.	0	28.5	0.3	6.3	0.5	3.6	0.5	3.8	8.8	1.3
Overall	0	33.5	0.1	12.3	0.5	9.1	0.8	6.6	8.7	0.8

Table 4
Served tasks.

Name	Less technicians				More technicians			
	Te	Ta	Ta/All	Ta/Te	Te	Ta	Ta/All	Ta/Te
N-40-50	3	20.80	0.42	6.93	6	37.90	0.76	6.32
N-50-50	3	16.70	0.33	5.57	6	32.10	0.64	5.35
W-40-50	3	21.20	0.42	7.07	6	38.30	0.77	6.38
W-50-50	3	16.50	0.33	5.50	6	30.40	0.61	5.07
N-40-100	6	53.20	0.53	8.87	12	82.90	0.83	6.91
N-50-100	6	38.13	0.38	6.35	12	70.78	0.71	5.90
W-40-100	6	56.33	0.56	9.39	12	85.66	0.86	7.14
W-50-100	6	42.30	0.42	7.05	12	74.70	0.75	6.23
N-40-200	12	104.70	0.52	8.73	24	179.00	0.90	7.46
N-50-200	12	88.22	0.44	7.35	24	155.13	0.78	6.46
W-40-200	12	101.56	0.51	8.46	24	177.72	0.89	7.41
W-50-200	12	94.90	0.47	7.91	24	159.70	0.80	6.65

the larger service area because more time is needed to get from one task to another. When 6 technicians are available, the number of served tasks obviously increases, although the multiplier tends to be slightly less than 2. It should also be noted that more tasks can be served when the time windows are wide. Similar trends are observed for the instances of size 100 and 200.

Table 5
Branch-and-Price (B&P) vs TS-AM-I.

Name	#Opt	CPU (s)	
		B&P	TS-AM-I
N-40-10	5	0.4	146.5
N-50-10	5	0.7	39.6
W-40-10	5	0.3	182.8
W-50-10	5	0.6	321.9
N-40-15	5	1.1	398.6
N-50-15	5	0.7	910.9
W-40-15	5	1.0	491.2
W-50-15	5	0.5	279.9
N-40-20	5	3.4	950.2
N-50-20	5	0.3	863.5
W-40-20	5	13.8	202.3
W-50-20	5	1.7	14.2
N-40-25	5	44.8	1080.5
N-50-25	5	0.4	554.3
W-40-25	5	91.8	833.6
W-50-25	5	5.8	1457.0
N-40-30	3	69.4	321.6
N-50-30	5	4.6	1731.9
W-40-30	2	25.2	1976.7
W-50-30	5	15.1	1025.0
N-40-35	2	241.4	1251.5
N-50-35	4	22.9	358.4
W-40-35	2	177.7	2058.0
W-50-35	3	28.3	2297.6
N-40-40	2	280.1	2136.0
N-50-40	4	30.4	1050.6
W-40-40	1	239.9	1077.3
W-50-40	4	178.7	2036.1
N-40–45	0	–	–
N-50-45	4	67.5	376.8
W-40-45	0	–	–
W-50-45	3	233.8	497.3

5.4. Comparison with optimal solutions

Optimal solutions obtained with a branch-and-price algorithm are reported in Mathlouthi et al. (2018) based on similar subsets of instances (except that each subset contains only 5 instances and 3 technicians are involved in each subset). This algorithm was able to routinely solve instances with up to 25 tasks and TS-AM-I also found the optimum in each case. The branch-and-price algorithm began to strive for instances with 30 tasks and was not able to solve any instance with 50 tasks or more (within 24 h of computation time). Again, TS-AM-I was able to optimally solve all instances with a known optimum.

The detailed results are shown in Table 5, where #Opt is the number of optimal solutions. This is followed by the average CPU time in seconds of both methods over instances solved to optimality: time to prove optimality for the branch-and-price algorithm and time to reach the optimum for TS-AM-I. As we can see, the branch-and-price algorithm solves the smallest instances in much less time than TS-AM-I. On the other hand, many instances with 30 to 45 tasks cannot be solved within 24 h of computation time. When an optimal solution is found, though, the computation time is much smaller when compared to TS-AM-I. Thus, the branch-and-price algorithm exhibits a large variance in performance, even for two instances of the same size, which is typical of exact methods. It should also be noted that TS-AM-I was coded in Java and that we did not put so much emphasis on efficiency issues.

6. Conclusion

We proposed a metaheuristic approach to address a TRSP for which exact methods can only solve small instances. Given the practical interest of this type of problem, the proposed methodology opens the way for further progress in this area. The meta-

heuristic is based on a tabu search enhanced with an adaptive memory, where the evaluation of each solution in memory is driven by both its cost and its contribution to diversity. Our algorithm reached the optimum on each try for instances with less than 50 tasks. It can also solve larger instances, as indicated by the results reported in this paper for instances with up to 200 tasks. Among many possible avenues of research, we now want to explore the application of our metaheuristic to dynamic variants of our TRSP, where new service requests must be integrated in real-time into the current routes.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Financial support was provided by the Canadian Natural Sciences and Engineering Research Council. This support is gratefully acknowledged.

References

Blakeley, F., Argüello, B., Cao, B., Hall, W., Knolmajer, J., 2003. Optimizing periodic maintenance operations for Schindler Elevator Corporation. Interfaces 33 (1), 67–79.

Bostel, N., Dejax, P., Guez, P., Tricoire, F., 2008. Multiperiod planning and routing on a rolling horizon for field force optimization logistics. In: Golden, B., Raghavan, S., Wasil, E. (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, pp. 503–525.

Cordeau, J.-F., Laporte, G., Pasin, F., Ropke, S., 2010. Scheduling technicians and tasks in a telecommunications company. J. Schedul. 13 (4), 393–409.

Gendreau, M., Guertin, F., Potvin, J.-Y., Taillard, É., 1999. Parallel tabu search for real-time vehicle routing and dispatching. Transp. Sci. 33 (4), 381–390.

Gendreau, M., Hertz, A., Laporte, G., 1994. A tabu search heuristic for the vehicle routing problem. Manage. Sci. 40 (10), 1276–1290.

Glover, F., 1989. Tabu search-Part I. ORSA J. Comput. 1 (3), 190–206.

Hashimoto, H., Boussier, S., Vasquez, M., Wilbaut, C., 2011. A GRASP-based approach for technicians and interventions scheduling for telecommunications. Ann. Oper. Res. 183 (1), 143–161.

Kawaguchi, S., Fukuyama, Y., 2016. Reactive tabu search for job-shop scheduling problems. In: Proceedings of the 11th International Conference on Computer Science Education (ICCSE), pp. 97–102.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. Oper. Res. Perspect. 3, 43–58.

Mathlouthi, I., Gendreau, M., Potvin, J.-Y., 2017. Branch-and-price for a multi-attribute technician routing and scheduling problem. Technical Report CIRRELT-2017-56, Interuniversity Research Center on Entreprise Networks, Logistics and Transportation.

Mathlouthi, I., Gendreau, M., Potvin, J.-Y., 2018. Mixed integer linear programming for a multi-attribute technician routing and scheduling problem. Inf. Syst. Oper. Res. 56 (1), 33–49.

Mendoza, J.-E., Montoya, A., Guéret, C., Villegas, J.-G., 2017. A parallel matheuristic for the technician routing problem with conventional and electric vehicles. In: 12th Metaheuristics International Conference (MIC). Barcelona, Spain.

Or, I., 1976. Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking. Technical report, PhD dissertation, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

Pillac, V., Guéret, C., Medaglia, A., 2012. On the dynamic technician routing and scheduling problem. In: 5th International Workshop on Freight Transportation and Logistics (ODYSSEUS), Mikonos, Greece.

Pillac, V., Guéret, C., Medaglia, A., 2013. A parallel matheuristic for the technician routing and scheduling problem. Optim. Lett. 7 (7), 1525–1535.

Pillac, V., Guéret, C., Medaglia, A., 2018. A fast reoptimization approach for the dynamic technician routing and scheduling problem. In: Amadeo, L., Talbi, E.-G., Yalaoui, F. (Eds.), Recent Developments in Metaheuristics, Springer, pp. 347–367..

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Comput. Oper. Res. 34 (8), 2403–2435.

Potvin, J.-Y., Rousseau, J.-M., 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. Eur. J. Oper. Res. 66 (3), 331–340.

Rochat, Y., Taillard, É.D., 1995. Probabilistic diversification and intensification in local search for vehicle routing. J. Heurist. 1 (1), 147–167.

Tabitha, J., Rego, C., Glover, F., 2009. Multistart tabu search and diversification strategies for the quadratic assignment problem. IEEE Trans. Syst. Man Cybern. A Syst. Humans 39 (3), 579–596.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., Potvin, J.-Y., 1997. A tabu search heuristic for the vehicle routing problem with soft time windows. Transp. Sci. 31 (2), 170–186.

Tang, H., Miller-Hooks, E., Tomastik, R., 2007. Scheduling technicians for planned maintenance of geographically distributed equipment. Transp. Res. E Logist. Transp. Rev. 43 (5), 591–609.

Tsang, E., Voudouris, C., 1997. Fast local search and guided local search and their application to British Telecom's workforce scheduling problem. Oper. Res. Lett. 20 (3), 119–127.

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2014. A unified solution framework for multi-attribute vehicle routing problems. Eur. J. Oper. Res. 234 (3), 658–673.

Weigel, D., Cao, B., 1999. Applying GIS and OR techniques to solve Sears technician-dispatching and home delivery problems. Interfaces 29 (1), 112–130.

Zamorano, E., Stolletz, R., 2017. Branch-and-price approaches for the multiperiod technician routing and scheduling problem. Eur. J. Oper. Res. 257 (1), 55–68.