J Heuristics (2017) 23:471–500
DOI 10.1007/s10732-017-9347-8

Iterated local search for workforce scheduling and routing problems

Fulin Xie[1] · **Chris N. Potts**[1] · **Tolga Bektaş**[2]

Received: 31 July 2016 / Revised: 14 April 2017 / Accepted: 26 June 2017 / Published online: 18 July 2017

Abstract The integration of scheduling workers to perform tasks with the traditional vehicle routing problem gives rise to the workforce scheduling and routing problems (WSRP). In the WSRP, a number of service technicians with different skills, and tasks at different locations with pre-defined time windows and skill requirements are given. It is required to find an assignment and ordering of technicians to tasks, where each task is performed within its time window by a technician with the required skill, for which the total cost of the routing is minimized. This paper describes an iterated local search (ILS) algorithm for the WSRP. The performance of the proposed algorithm is evaluated on benchmark instances against an off-the-shelf optimizer and an existing adaptive large neighbourhood search algorithm. The proposed ILS algorithm is also applied to solve the skill vehicle routing problem, which can be viewed as a special case of the WSRP. The computational results indicate that the proposed algorithm can produce high-quality solutions in short computation times.

Keywords Workforce scheduling · Vehicle routing · Iterated local search

✉ Chris N. Potts
C.N.Potts@soton.ac.uk

Fulin Xie
fx1g12@soton.ac.uk

Tolga Bektaş
T.Bektas@soton.ac.uk

[1] School of Mathematical Sciences, CORMSIS, University of Southampton, Southampton, UK

[2] Southampton Business School, CORMSIS, University of Southampton, Southampton, UK

1 Introduction

The workforce scheduling and routing problem (WSRP) and its variants are commonly faced by many service providers, and have applications of home health care, field technician scheduling, security personnel routing and manpower allocation.

The term WSRP is coined by Castillo-Salazar et al. (2012), and refers to a class of optimization problems where service personnel are required to carry out tasks at different locations. For example, nurses visiting patients at their homes, and technicians performing maintenance jobs in different companies can each be modeled as a WSRP. As service personnel need to travel between different locations, minimizing their distances and times for travel is usually considered as one of the objectives when making operational decisions. This results in a routing problem of finding a set of least cost routes for a given workforce, where each route consists of a sequence of locations. Sometimes, tasks have associated time windows, within which service must start. This type of problem can be modeled as an extension of the vehicle routing problem with time windows (VRPTW), which is a well-known variant of the classical vehicle routing problem (VRP).

Service personnel often specialize in different skill domains, and possess skills at different levels. The tasks themselves have different skill requirements. For example, in the telecommunications industry, tasks may include maintenance, installation, construction and repair jobs, and technicians are trained in skills that allow them to only be able to service a subset of these tasks. Thus, skill compatibility must be taken into account to ensure that tasks are performed only by qualified personnel. The associated scheduling problem involves the assignment of tasks to service personnel. In some applications, tasks can be outsourced to a third party, albeit at the expense of additional cost, if appropriate resources are not available to provide the required service, or better operational performance can be achieved. The version of the WSRP that we consider allows for outsourcing.

Due to its complexity, most of the existing research on the WSRP has aimed at developing efficient heuristic solution algorithms. However, most of them are sophisticated and highly problem specific. In this paper, a simple heuristic algorithm based on iterated local search (ILS) is proposed to solve the WSRP. ILS is one of the most conceptually simple and robust algorithms (Burke et al. 2010). The essential idea of ILS is that when the local search is trapped at a local optimum, the ILS perturbs the previously visited local optimum instead of generating a new initial solution, and then restarts the local search from this modified solution (Lourenço et al. 2003). Although the ILS has a very simple framework, it has been successfully applied to a wide variety of optimization problems including the graph coloring problem (Chiarandini and Stützle 2002), the job shop scheduling problem (Lourenço 1995) and the vehicle routing problem (Hashimoto et al. 2008; Chen et al. 2010; Walker et al. 2012; Penna et al. 2013; Michallet et al. 2014). However, no study has been reported on the application of the ILS to the WSRP, which is the aim of this paper. The contribution of the paper is a fast and simple algorithm for the WSRP with the objective of minimizing the total travel cost and outsourcing cost. The proposed algorithm is also applied to solve the skill vehicle routing problem (Skill VRP). To the best of our knowledge, this is also the first ILS approach for the Skill VRP.

The remainder of the paper is organized as follows. Section 2 reviews the related literature on the WSRP. A formal definition of the problem is presented in Sect. 3. Section 4 gives a description of the proposed ILS. Computational results for benchmark instances are presented in Sect. 5. The paper ends with some concluding remarks in Sect. 6.

2 Literature review

Recent studies on the WSRP include the work of Kovacs et al. (2012). They present an adaptive large neighbourhood search (ALNS) algorithm to solve the service technician routing and scheduling problem (STRSP). In this problem, tasks are associated with time windows and skill requirements, outsourcing tasks is allowed, and team building may be required in order to fulfill skill requirements of difficult tasks. The objective is to minimize the total operational cost comprising the routing and outsourcing cost. The scheduling aspect of this problem is adapted from the study of Cordeau et al. (2010), which considers a technician and task scheduling problem arising in a large telecommunications company. Cordeau et al. (2010) focus on the construction of teams and the assignment of tasks to teams without considering routing costs between tasks. Their problem is solved by using a construction heuristic and an ALNS algorithm. Pillac et al. (2013) extend the study of Kovacs et al. (2012) by taking tools and spare parts into account, where each task must be carried out by a technician with the required skills, tools, and spare parts, and within the prescribed time window. The problem is solved by a matheuristic consisting of a parallel version of ALNS algorithm and a mathematical programming based post-optimization procedure.

Xu and Chiu (2001) also consider a field technician scheduling problem arising in the telecommunications industry. The objective is to maximize the number of jobs scheduled to technicians, while accounting for each job's priority and skill constraints. Three different heuristic approaches, namely, a greedy heuristic, a local search algorithm, and a greedy randomized adaptive search procedure (GRASP) are proposed to solve the problem. Castillo-Salazar et al. (2015) describe a greedy heuristic to address the WSRP with five types of time-dependent constraints, which model the relationship between tasks, e.g. one task needs to start after the completion of another task.

A variant of the WSRP is the skill vehicle routing problem (Skill VRP), which is introduced by Cappanera et al. (2011). The Skill VRP differs from other problems reviewed above in two aspects: (1) tasks do not have associated time windows, and (2) the routing costs depend both on the traveling distance and the technician in such a way that increasing the skill level of the technician causes an increase in costs. The use of technician-dependent routing costs is motivated by practical applications, since high-skilled employees usually have higher salaries than those with only basic skills. The Skill VRP is also studied by Schwarze and Voß (2012), but their study incorporates load balancing and resource utilization when constructing tours for service vehicles. Their motivation for proposing this model is their finding that many Skill VRP solutions usually use only a subset of vehicles, and a considerable number of tasks are assigned to vehicles with technicians that have higher skills than necessary.

Some studies have considered stochastic elements in the WSRP. For example, Weintraub et al. (1999) study a scheduling and routing problem for service vehicles belonging to an electric utility company in Chile, where service requests are stochastic. Pillac et al. (2012) also consider a technician routing and scheduling problem with stochastic service requests, which is solved by a parallel adaptive large neighbourhood search (pALNS) and a multiple plan approach. Binart et al. (2016) solve a field service routing problem with stochastic travel and service times using a two-stage stochastic programming model. Finally, Chen et al. (2015) describe a technician routing problem with experience-based service times, where technicians learn over time, which results in service times being reduced as experience increases.

Other problems closely related to the WSRP are the site-dependent vehicle routing problem with time windows (Cordeau and Laporte 2001; Cordeau et al. 2004), the home health care scheduling problem (Blais et al. 2003; Bertels and Fahle 2006; Akjiratikarl et al. 2007) and the manpower allocation problem (Dohn et al. 2009).

3 Problem definition

In this section, we first provide a formal description of the WSRP that we address. We then formulate a mixed integer programming (MIP) model for our problem.

The WSRP is defined on a complete graph $G = (V, A)$, where $V = \{0, 1, \ldots, n+1\}$ is a set of vertices and $A = \{(i, j) : i, j \in V, i \neq j\}$ is a set of arcs. The vertex 0 denotes the depot and vertex $n + 1$ is a copy of the depot, and $C = V \backslash \{0, n + 1\}$ represents the set of vertices that each have a unique task. Depending on the context, we refer to a task i or a vertex i for any $i \in C$. A set K of technicians are available to perform the tasks. Each technician is specialized in a number of skill domains at different proficiency levels. Each task $i \in C$ has an associated service duration d_i, a time window $[e_i, l_i]$ within which service should commence, and a skill requirement. The depot and its copy also have time windows, which define the earliest departure time e_0 and the latest return time l_{n+1} of any technician. Also, the route duration of each technician must not exceed a given time D. Each arc $(i, j) \in A$ has an associated cost c_{ij} and travel time t_{ij}.

In the studies of Cordeau et al. (2010) and Kovacs et al. (2012), each technician's skills and each task's skill requirements are described by skill matrices, which are used to determine if a single technician or a team of technicians would be able to perform a given task. In this paper, we do not consider the possibility of building a team of technicians, and thus simply define a binary parameter q_i^k, where $q_i^k = 1$ if technician $k \in K$ is qualified to perform task $i \in C$, and $q_i^k = 0$ otherwise. The values of q_i^k can be easily computed based on technicians' skills and tasks' skill requirements. Finally, any task $i \in C$ can be outsourced by incurring a cost o_i, in the event that resources are insufficient or too expensive to undertake all of the tasks.

The WSRP can be formulated as a mixed integer programming model that contains the following binary variables:

$$x_{ij}^k = \begin{cases} 1 & \text{if arc } (i, j) \text{ is traversed by technician } k, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall (i, j) \in A,\ k \in K;$$

$$y_i = \begin{cases} 1 & \text{if task } i \text{ is outsourced,} \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i \in V;$$

and the continuous variable $b_i^k, \forall i \in V,\ k \in K$, that lies within the interval $[e_i, l_i]$ if technician k does not perform task i; otherwise, it is the time at which service of task i commences, or the leaving time and returning time of technician k from and to the depot when $i = 0$ and $n + 1$ respectively.

The mathematical model is presented as follows:

$$\text{minimize} \quad \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k + \sum_{i \in C} o_i y_i \tag{1}$$

subject to:

$$\sum_{k \in K} \sum_{j \in V} x_{ij}^k + y_i = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{j \in V} x_{ij}^k \le q_i^k \qquad \forall k \in K,\ \forall i \in C \tag{3}$$

$$\sum_{j \in V} x_{0,j}^k = 1 \qquad \forall k \in K \tag{4}$$

$$\sum_{i \in V} x_{i,n+1}^k = 1 \qquad \forall k \in K \tag{5}$$

$$\sum_{i \in V} x_{ih}^k - \sum_{j \in V} x_{hj}^k = 0 \qquad \forall k \in K,\ \forall h \in C \tag{6}$$

$$b_i^k + (d_i + t_{ij}) x_{ij}^k \le b_j^k + l_i (1 - x_{ij}^k) \qquad \forall k \in K,\ \forall (i, j) \in A \tag{7}$$

$$e_i \le b_i^k \le l_i \qquad \forall k \in K,\ \forall i \in V \tag{8}$$

$$b_{n+1}^k - b_0^k \le D \qquad \forall k \in K \tag{9}$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall k \in K,\ \forall (i, j) \in A \tag{10}$$

$$y_i \in \{0, 1\} \qquad \forall i \in C \tag{11}$$

$$b_i^k \ge 0 \qquad \forall k \in K,\ \forall i \in V. \tag{12}$$

The objective function (1) minimizes the total operational cost comprising routing and outsourcing cost. Constraints (2) ensure that each task is either visited exactly once or outsourced, while constraints (3) guarantee that the tasks can only be performed by technicians satisfying the skill requirements. Constraints (4) and (5) ensure that each technician departs from the depot and returns to the copy of the depot after completing their service. Constraints (6) are the typical flow conservation equations. Constraints (7) set the time variables b_i^k, while constraints (8) enforce the time window restrictions. Constraints (9) guarantee that the route duration for each technician is no more than

the maximum time allowed. Constraints (10) and (11) represent the binary restrictions on variables x_{ij}^k and y_i, and (12) are the non-negativity constraints on the variables b_i^k.

4 Iterated local search

This section describes our proposed iterated local search (ILS) algorithm for solving the WSRP. The ILS consists of three main components: initial solution construction, local search procedure and perturbation mechanism. They are combined into a multi-start framework as given in Algorithm 1. At each iteration of the main loop between lines 3 to 18, an initial feasible solution s is constructed for the ILS loop (lines 6 to 14). At each ILS iteration, the local search procedure takes as input the solution s, and returns an improved solution s', which is accepted as the new best current solution if it is feasible and has a value $f(s')$ that is strictly smaller than that of the incumbent solution $\bar{s}$, denoted by $f(\bar{s})$. Then a new starting solution s for the local search procedure is generated by perturbing on the incumbent solution $\bar{s}$ (line 12). The ILS loop repeats until the maximum number of iterations without improvement $MaxIt_{NI}$ is met. Then the incumbent solution $\bar{s}$ replaces the global best solution s^* if $f(\bar{s}) < f(s^*)$. This procedure repeats until a predefined number $MaxIt$ of iterations have been executed.

Algorithm 1 Iterated Local Search

```
1: procedure ILS
2:     It ← 1, f(s*) ← +∞
3:     for It ← 1 to MaxIt do
4:         generate initial solution s
5:         set s̄ ← s, It_NI ← 0
6:         while ( It_NI < MaxIt_NI) do
7:             s' ← Local Search (s)
8:             if ((s' is feasible ) and (f(s') < f(s̄))) then
9:                 s̄ ← s'
10:                It_NI ← 0
11:            end if
12:            s ← Perturb(s̄)
13:            It_NI ← It_NI + 1
14:        end while
15:        if (f(s̄) < f(s*)) then
16:            s* ← s̄
17:        end if
18:    end for
19:    Return s*
20: end procedure
```

4.1 Search space

A number of studies have shown that an efficient exploration of infeasible solutions can contribute significantly to the performance of a heuristic (Cordeau et al. 1997; Glover and Hao 2011; Cordeau et al. 2001; Vidal et al. 2012, 2013). We follow the same line of

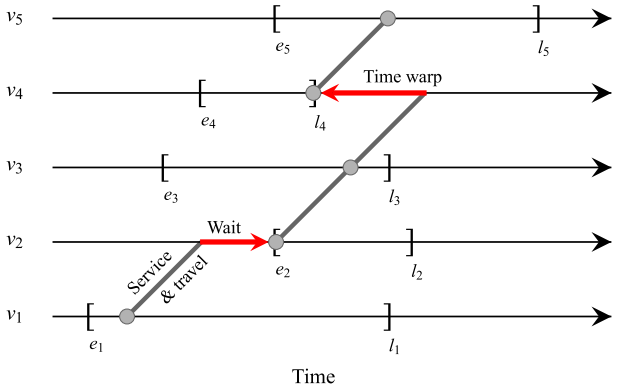

Fig. 1 Illustration of waiting time and time warp

thought here and allow the ILS to search infeasible, as well as feasible solutions, where the constraint violations in the former relate to the route duration and time window constraints. However, the skill requirement constraint is always respected, since it is concerned with the scheduling aspect of the WSRP and its relaxation would enlarge the search space dramatically. A solution s is therefore evaluated by an augmented cost function, which is defined by

$$f(s) = c(s) + \alpha d(s) + \beta w(s), \tag{13}$$

where $c(s)$ is the total operational cost as defined in (1), and $d(s)$ and $w(s)$ are the total violations of duration and time window constraints, which are weighted by parameters α and β, respectively.

The time window violation is measured based on a method proposed by Nagata et al. (2010). If there is a late arrival to a customer $i \in C$ at time $a_i > l_i$, then it is assumed that there is a penalty for the delay $a_i - l_i$, and that service starts at time l_i. In case of an early arrival at time $a_i < e_i$, then the technician has to wait until time e_i, but the waiting time is not penalized. The same method is used by Vidal et al. (2013), who refer to the penalty as 'time warp'. Figure 1 illustrates the waiting time and time warp of a route with visits involving five vertices $v_1, \ldots, v_5$. The horizontal axis corresponds to time, while the vertical axis presents the sequence of visits. The dots on each line show the start time of each visit, and the brackets on each line indicate the time window of the corresponding task. As seen in Fig. 1, there are no penalties associated with tasks v_1, v_3, and v_5 as the visits are made within the respective time windows. The bold line displays a possible schedule having a waiting time period at vertex v_2 and a time warp at vertex v_4.

4.2 Move evaluation

Most local search heuristics spend the largest part of the overall computational effort on move evaluation (Vidal et al. 2015). Efficient move evaluation techniques are there-

fore crucial for improving algorithm performance, particularly when the search space involves infeasible solutions.

The operational cost $c(s)$ consists of the outsourcing cost and the total traveling distance which can be computed in amortized $O(1)$ time (Kindervater and Savelsbergh 1997). However, it takes $O(n)$ to compute the penalties $d(s)$ and $w(s)$ in (13).

Nagata et al. (2010) propose an evaluation technique to compute the violation of time window constraints in amortized $O(1)$ time for most traditional neighbourhood operators including 2-opt, inter-route swaps, and inter-route inserts. Vidal et al. (2013) extend this technique to allow the evaluation of both duration and time windows violations not only for inter-route but also for intra-route operators. A preprocessing phase is required to develop relevant data for their evaluation techniques, and the data must be updated once the route under consideration has been modified. Our ILS incorporates the technique proposed by Vidal et al. (2013) to compute the violation penalties for infeasible solutions.

4.3 Initial solution construction

Our procedure for constructing a feasible solution includes the following steps. The existence of a feasible solution is guaranteed due to the possibility of outsourcing. First, a task list L_1 is created as follows. The first task in the list is selected at random. The remaining entries in the list are constructed by sorting the remaining tasks of C in non-decreasing order of the angle they make with a line drawn from the depot to the randomly selected first task on L_1. Then, a technician list L_2 is constructed by sorting the technicians of K in non-increasing order of the number of tasks they are qualified to perform. We then randomly select a task $i \in C$ from the list L_1 and insert it into the cheapest feasible position of the route of the first technician on list L_2. If the insertion violates feasibility, we insert i into the following technician's route. In the case where no feasible route can be constructed that incorporates task i, we set i to be outsourced. The procedure repeats by inserting tasks sequentially into technicians' routes following the above steps, yielding a feasible solution that consists of technicians' routes and a list of outsourced tasks.

4.4 Local search procedure

Our local search procedure consists of an inter-route search operator, an intra-route search operator, and an update mechanism of the weight parameters α and β using in (13).

The inter-route search uses a single neighbourhood structure called Swap and Relocate that removes two paths, each containing at most two tasks from two different routes, and then exchanges them. One of these paths may contain zero tasks, which results in the path from the other route being relocated. Figure 2 gives an example of this operator which removes two successive vertices v_2 and v_3 from route r_1 and one vertex v_6 from route r_2, and then exchanges them. This neighbourhood structure is extended to allow an outsourced task to be swapped or relocated into the route of one of the technicians. When considering new routes created by this operator, the skill

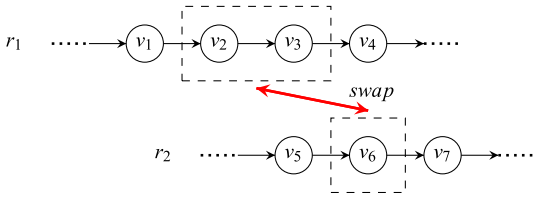

Fig. 2 Example of the swap and relocate operator

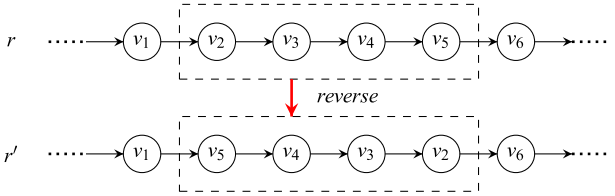

Fig. 3 Example of the 2-opt operator

requirement constraints must be always respected, but any violations of duration and time window constraints are allowed.

The intra-route search consists of three neighbourhood structures, namely, opt1, opt2 and 2-opt (Croes 1958), that operate on a single route. Operator opt1 removes one task and inserts it into another position on the same route, while operator opt2 is similar but removes and inserts two adjacent customers on a route. Operator 2-opt reverses the order of a sequence of successive visits on a route. Figure 3 provides an example of the 2-opt operator which removes a path consisting of four vertices $\{v_2, v_3, v_4, v_5\}$ from route r, reverses the order of visits on this path, and then inserts the path back into the same position to form a new route r'. The cost of r' is evaluated by the method described in Sect. 4.2.

The inter-route search and the intra-route search can be combined in different ways within the local search procedure. To test the effect of the search strategy on the performance of the algorithm, we investigate the three following strategies:

1. Execute only the inter-route search operator;
2. Execute both the inter-route and intra-route search operators at each iteration of the local search procedure;
3. Apply the intra-route search as a post-optimization procedure on the locally optimal solution returned by the inter-route search.

After each iteration of the local search, the weight parameters α and β are adjusted according to the duration violation $d(s)$ and the time window violation $w(s)$ of the incumbent solution s as follows. If $d(s) = 0$, then the parameter α is divided by a factor $1 + \delta$; otherwise, it is multiplied by $1 + \delta$, where $\delta > 0$ is a parameter that controls the strength of adjustment. The same rule applies to the parameter β with respect to $w(s)$. The initial values of α and β are both set to 1, as suggested by a number of studies that have similar cost functions and weight parameters (Cordeau et al. 2001, 1997; Ibaraki et al. 2008; Nagata et al. 2010).

The structure of the local search procedure is illustrated in Algorithm 2. The current best solution s' is set to the incumbent solution s. Then s is taken as input by the

SearchStrategy function, which applies inter-route or intra-route search depending on the search strategy selected and returns an improved solution $\hat{s}$ if such a solution exists. If $f(\hat{s}) < f(s')$, then $\hat{s}$ replaces s' as the current best solution. Then, the duration violation $d(\hat{s})$ and time window violation $w(\hat{s})$ are computed, and parameters α and β are adjusted accordingly by the control mechanism described above. The pre-processed data for the routes that have been modified at the current iteration are updated. This procedure repeats until the local search becomes trapped at a locally optimal solution.

Algorithm 2 Local Search Procedure

```
1: procedure LocalSearch
2:     input solution s
3:     set α = 1 and β = 1
4:     set s' = s
5:     set LocalOptimumFound = false
6:     while (LocalOptimumFound = false) do
7:         ŝ ← SearchStrategy(s)
8:         if (f(ŝ) < f(s')) then
9:             s' ← ŝ, s ← ŝ
10:            Compute d(ŝ) and w(ŝ), and update α and β
11:            Update PreprocessData
12:        else
13:            set LocalOptimumFound = true
14:        end if
15:    end while
16:    return s'
17: end procedure
```

4.5 Perturbation mechanism

The perturbation mechanism uses a random cross exchange operator, which removes two paths from two randomly selected routes and exchanges them. Figure 4 gives an example of the perturbation operator which removes a path of four successive visits from route r_1 and a path of two successive visits from route $r_2 \neq r_1$, and then exchanges them. Violations of duration and time window constraints are allowed, but the skill requirement constraint must be respected. The perturbation procedure is always carried out on the best solution found thus far, and applies the random cross exchange operator p times, where p is a positive integer denoting the perturbation strength.

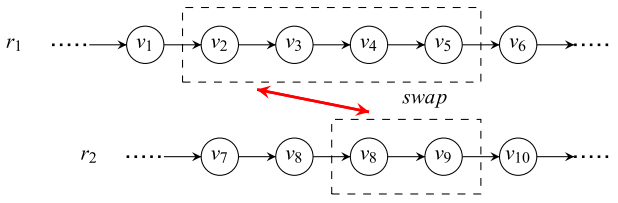

Fig. 4 Example of the random cross exchange operator

The perturbation strength p is a crucial parameter of the ILS. If p is too small, the local search may not be able to escape from a locally optimal solution. If p is too large, the ILS may behave similar to a random restart algorithm, making it difficult to discover better quality solutions (Lourenço et al. 2003). In order to determine the most appropriate value of p, we developed an adaptive mechanism, which adjusts p according to the number of consecutive iterations without improvement, denoting by It_{NI}. Let γ be a trigger for the adjustment of p. More precisely, whenever It_{NI} has increased by γ, the value of p will be increased by 1 until it reaches the upper bound $\bar{p}$, where $\bar{p}$ is used to prevent excessively large values of p to be chosen. For example, if $\gamma = 10$ and $\bar{p} = 5$, then p starts from 1 and increases by 1 when $It_{NI} \in \{10, 20, 30, 40\}$.

4.6 Reducing outsourcing cost

As the cost of outsourcing a task is usually higher than that of serving it by internal resources, reducing the outsourcing cost is considered as an objective within the algorithm. This is achieved by a simple mechanism embedded in the perturbation procedure of the proposed ILS. At the beginning of the perturbation procedure, we check the list of outsourced tasks. If it is not empty, we randomly select a task and insert it to the cheapest position of the current solution, and then proceed with the perturbation procedure; otherwise, we only apply the random cross exchange operator. The insertion of outsourced tasks and the perturbation procedure is likely to produce an infeasible solution, which will be improved by the local search procedure. Infeasible solutions are evaluated by a cost function defined in (13), and weight parameters α and β are dynamically adjusted based on the rule described in Sect. 4.4. If the local search procedure cannot repair the infeasibility during the first few iterations, the weight parameters will be adjusted to large values, such that the cost of scheduling a task to a technician becomes greater than the cost of outsourcing it. As a consequence, the local search tends to repair the infeasibility by simply outsourcing the relevant tasks. In order to avoid the overuse of the outsourcing option, we force the local search procedure to always select improved solutions with lower outsourcing costs, even if solutions with higher outsourcing costs but lower overall costs exist.

5 Computational results

This section presents results of our computational tests conducted to assess the performance of the proposed ILS. The ILS algorithm is coded in C++, and run on a personal computer with Intel Core i5-3570 3.40 GHz processor and 4 GB Memory (RAM). The MIP model is implemented on the same machine, and solved by the commercial solver CPLEX 12.6. Our ILS results are compared with existing solutions of an ALNS algorithm (Kovacs et al. 2012), where the reported ALNS results are based on the average of five runs of the algorithm. To maintain consistency and provide a fair comparison, we also perform five random runs of the ILS for each instance tested and report the obtained results.

5.1 Test instances

The experiments are conducted using the technician routing and scheduling problem (TSRP) instances introduced by Kovacs et al. (2012). These instances are adapted from the Solomon's benchmark instances (Solomon 1987) for the VRPTW and the test instances provided for the ROADEF 2007 challenge. They are available online at: http://prolog.univie.ac.at/research/STRSP/.

The set of instances of Kovacs et al. (2012) are generated using 12 instances of Solomon (1987), namely, R101, R103, R201, R203, C101, C103, C201, C203, RC101, RC103, RC201, RC203, where R, C and RC represent the random, clustered and a mix of random and clustered geographical setting, respectively. Instance sets with prefixes R1, C1 and RC1 have a short scheduling horizon, while those with prefixes R2, C2 and RC2 have a long scheduling horizon. The final two digits in the name of the instance indicate the time window density. In the 01 instances, all customers are associated with time windows, while in the 03 sets, only 50% of customers have time windows. In terms of the skill requirements, Kovacs et al. (2012) generate three types of skill requirement matrices shown by 5×4, 6×6, and 7×4 based on the ROADEF data, where the rows of the matrices correspond to skill domains, and the columns correspond to skill levels under each skill domain. The customer data of Solomon's instances are randomly paired with the skill data, which results in a total of 36 test instances. All instances have 100 customers and a single depot. For each instance, Kovacs et al. (2012) define a 'team' and a 'no team' version. As our study does not consider the possibility of team building, we only use the 'no team' version of instances in our experiments. For each instance, there are two sets of technician data: one has a sufficient number of technicians that feasibility can be achieved without outsourcing, while the other has limited technicians such that it is impossible to service all tasks without the use of the outsourcing option. The outsourcing cost of a task i is defined as $o_i = 200 + \mu_i^{1.5}$, where μ_i measures the difficulty of task i, and is calculated as the sum of the skill requirement for i in the skill matrix. The outsourcing cost is always higher than the cost of assigning a task to a technician.

5.2 Parameter setting

The ILS requires five input parameters as follows: $MaxIt$; $MaxIt_{NI}$; δ, which is the factor used to adjust weight parameters of duration and time window violations; $\bar{p}$, which is the upper bound that is used in the perturbation mechanism; and γ is the adjustment factor of the perturbation strength. The value of $MaxIt_{NI}$ is defined by Penna et al. (2013) as

$$MaxIt_{NI} = |C| + \lambda |K|, \tag{14}$$

where $|C|$ is the number of customers, $|K|$ is the number of technicians, and λ is a weight parameter determining the influence of $|K|$ on the value of $MaxIt_{NI}$. Thus, instead of finding the most appropriate value for $MaxIt_{NI}$, the value of λ is examined. Extensive parameter tuning suggests that a parameter setting shown in Table 1 performs well.

Table 1 Parameter setting for the proposed ILS

Parameter	*MaxIt*	δ	$\bar{p}$	γ	λ
Value	5	0.5	5	20	10

5.3 Performance measurement

The proposed ILS is evaluated against the MIP model and the ALNS (Kovacs et al. 2012) using benchmark instances. To compare the ILS and ALNS solutions, we compute the relative percentage difference defined as

$$\text{Imp}^{\text{S}}_{\text{A/I}} = \frac{v^{\text{S}}(\text{ALNS}) - v^{\text{S}}(\text{ILS})}{v^{\text{S}}(\text{ALNS})} \times 100 \tag{15}$$

where $v^{\text{S}}(\text{ALNS})$ and $v^{\text{S}}(\text{ILS})$ represent values of the ALNS and ILS solutions respectively, and $\text{S} = \{-, *, +\}$ denotes the minimum, mean, and maximum values over five random runs of the algorithm. For example, $\text{Imp}^{-}_{\text{A/I}}$ represents the relative percentage difference between the values of the best solutions found by the ALNS and ILS over five random runs. A positive value of $\text{Imp}^{\text{S}}_{\text{A/I}}$ indicates an improvement of the ILS over ALNS; otherwise, the cost of the ILS solution is greater or equal to that of the ALNS solution.

By replacing $v^{\text{S}}(\text{ALNS})$ of the expression (15) with $v^{\text{S}}(\text{CPLEX})$, we obtain the relative percentage difference between the values of the ILS solution and the optimal solution produced by CPLEX, denoted by $\text{Imp}^{\text{S}}_{\text{C/I}}$. There is no difference between $v^{-}(\text{CPLEX})$, $v^{*}(\text{CPLEX})$ and $v^{+}(\text{CPLEX})$, as they all refer to the value of the optimal solution found by CPLEX.

In addition to the comparison of solution values, we compare the computational times required by our ILS and the ALNS. Kovacs et al. (2012) run their ALNS on a Pentium D computer with two 3.2 GHz CPUs and 4 GB memory (the algorithm only uses one CPU), which is different from our machine that is used to implement the ILS and MIP model. In order to provide a fair comparison of computational speed, we scale the reported CPU times according to the speed factors provided in the report of Dongarra (2014). The report does not cover the two computers considered in our experiments. Thus, we use a slower but similar computer (Pentium IV with 3.0 GHz) available in Dongarra (2014) instead of the computer used by Kovacs et al. (2012), and use a speed factor of 1573 Mflop/s (millions of floating-point operations per second). As there is no suitable substitute available in Dongarra (2014), we apply the same software used by Dongarra (2014) to record the speed factor of our computer, which yields 2462 Mflop/s. Based on the speed factors, the reported CPU times of the ALNS are adjusted by multiplying a factor of (1573/2462), when comparing with the ILS times.

5.4 Evaluation of search strategies

Results on comparing the three search strategies described in Sect. 4.4 are shown in Table 2. Columns headed Avg. show the average solution values produced by the ALNS

Table 2 Evaluation of local search strategies

			ALNS		Strategy 1			Strategy 2			Strategy 3		
Instances	$\|C\|$	$\|K\|$	Avg.	CPU	Avg.	$\text{Imp}^*_{\text{A/I}}$	CPU	Avg.	$\text{Imp}^*_{\text{A/I}}$	CPU	Avg.	$\text{Imp}^*_{\text{A/I}}$	CPU
C101_5x4	100	17	1111.08	66.50	1106.01	0.46	28.67	1100.36	0.96	29.38	1108.14	0.26	30.96
C103_5x4	100	17	1037.33	84.61	1028.28	0.87	36.96	1035.82	0.15	37.44	1032.09	0.50	40.46
C201_5x4	100	8	1180.93	54.29	1157.58	1.98	10.30	1157.86	1.95	16.19	1157.58	1.98	12.62
C203_5x4	100	8	1049.30	89.78	1070.44	−2.01	23.86	1055.75	−0.62	43.34	1057.83	−0.81	26.43
R101_5x4	100	25	1685.85	84.25	1671.28	0.86	45.40	1663.13	1.35	51.85	1670.08	0.94	53.90
R103_5x4	100	25	1249.91	101.77	1240.41	0.76	64.01	1240.81	0.73	59.36	1237.72	0.98	62.53
R201_5x4	100	7	1448.93	57.22	1461.82	−0.89	25.47	1436.38	0.87	39.05	1443.02	0.41	25.57
R203_5x4	100	7	1106.12	84.18	1127.99	−1.98	31.12	1097.38	0.79	49.92	1098.64	0.68	30.84
RC101_5x4	100	22	1716.07	78.25	1692.06	1.40	39.29	1694.78	1.24	47.47	1690.19	1.51	48.71
RC103_5x4	100	22	1354.11	91.41	1365.55	−0.84	52.43	1346.18	0.59	48.28	1348.31	0.43	50.39
RC201_5x4	100	9	1607.25	58.71	1619.35	−0.75	28.55	1605.45	0.11	36.47	1602.75	0.28	26.03
RC203_5x4	100	9	1166.50	87.01	1177.42	−0.94	29.65	1165.32	0.10	46.27	1167.93	−0.12	34.12
C101_6x6	100	16	1004.82	69.76	998.15	0.66	40.97	983.91	2.08	45.82	993.71	1.11	36.34
C103_6x6	100	16	897.86	88.17	920.12	−2.48	52.71	911.33	−1.50	53.24	910.20	−1.37	58.97
C201_6x6	100	7	821.55	55.65	832.92	−1.38	13.95	830.36	−1.07	29.01	832.69	−1.36	20.70
C203_6x6	100	7	703.10	99.75	727.77	−3.51	39.21	703.28	−0.02	47.66	708.82	−0.81	36.18
R101_6x6	100	26	1667.43	96.78	1655.83	0.70	60.47	1658.88	0.51	72.55	1658.82	0.52	74.52
R103_6x6	100	29	1231.49	116.00	1225.62	0.48	74.87	1224.92	0.53	79.70	1222.90	0.70	79.47
R201_6x6	100	7	1270.26	62.02	1288.10	−1.40	51.63	1290.01	−1.55	54.62	1278.29	−0.63	47.75
R203_6x6	100	7	951.84	99.46	948.33	0.37	50.72	929.96	2.30	101.70	933.32	1.95	55.20

Table 2 continued

Instances	$\|C\|$	$\|K\|$	ALNS		Strategy 1			Strategy 2			Strategy 3		
			Avg.	CPU	Avg.	$\text{Imp}^*_{\text{A/I}}$	CPU	Avg.	$\text{Imp}^*_{\text{A/I}}$	CPU	Avg.	$\text{Imp}^*_{\text{A/I}}$	CPU
RC101_6x6	100	24	1683.96	90.42	1676.27	0.46	62.59	1672.12	0.70	51.39	1671.86	0.72	55.13
RC103_6x6	100	24	1310.95	103.68	1327.05	−1.23	69.85	1301.77	0.70	61.91	1315.62	−0.36	64.58
RC201_6x6	100	8	1406.95	61.32	1392.39	1.04	43.80	1372.90	2.42	52.14	1377.50	2.09	49.11
RC203_6x6	100	8	1016.71	91.39	1028.09	−1.12	56.16	1013.42	0.32	78.29	1016.73	0.00	56.64
C101_7x4	100	17	1398.95	58.65	1374.40	1.76	17.32	1382.87	1.15	24.74	1381.59	1.24	23.25
C103_7x4	100	17	1239.22	75.26	1232.41	0.55	26.91	1232.61	0.53	27.12	1233.67	0.45	25.22
C201_7x4	100	8	1282.18	49.54	1265.58	1.29	8.72	1269.68	0.97	17.83	1262.94	1.50	10.71
C203_7x4	100	8	1151.27	76.83	1164.43	−1.14	20.34	1150.27	0.09	32.04	1142.22	0.79	22.92
R101_7x4	100	28	1793.95	88.52	1785.47	0.47	40.15	1787.10	0.38	50.00	1797.41	−0.19	43.76
R103_7x4	100	28	1375.09	102.08	1352.08	1.67	41.53	1368.30	0.49	35.89	1346.73	2.06	43.04
R201_7x4	100	10	1410.90	59.31	1414.88	−0.28	18.74	1403.89	0.50	29.16	1403.94	0.49	18.48
R203_7x4	100	10	1166.94	80.29	1197.09	−2.58	21.24	1176.64	−0.83	36.54	1166.52	0.04	28.34
RC101_7x4	100	23	1844.37	75.55	1845.04	−0.04	29.33	1825.72	1.01	32.97	1818.02	1.43	28.72
RC103_7x4	100	23	1455.33	85.04	1441.02	0.98	38.42	1442.00	0.92	35.12	1446.39	0.61	39.64
RC201_7x4	100	9	1701.25	52.78	1729.81	−1.68	21.52	1703.28	−0.12	20.11	1704.20	−0.17	16.21
RC203_7x4	100	9	1241.65	73.82	1243.55	−0.15	20.93	1234.72	0.56	37.47	1235.11	0.53	22.78
Average			1298.37	79.17	1299.57	−0.21	37.16	1290.81	0.54	44.78	1290.93	0.51	38.89

and the ILS using three different strategies over five random runs. The corresponding relative percentage differences between the values of the ALNS solutions and ILS solutions are reported in columns titled $\text{Imp}^*_{\text{A/I}}$.

Comparing Strategy 1 with Strategy 2 and Strategy 3, it can be seen that by applying the intra-route search, the solution quality improves significantly from −0.21% to 0.54% and 0.51%, with the average computational time increasing accordingly. The intra-route search is seen to be especially useful on the R2, C2 and RC2 types of instances, which are characterized by a long scheduling horizon and a low number of technicians, where each route contains a relatively high number of tasks. Comparing Strategy 2 with Strategy 3, the difference between the average $\text{Imp}^*_{\text{A/I}}$ values is only 0.03%. However, the average computational time of Strategy 2 is about 13% higher than that of Strategy 3. Therefore, Strategy 3, which applies the intra-route search as a post-optimization procedure on the local optimum returned by the inter-route search, is recommended based on efficiency and effectiveness, and is used in the remainder of our tests.

5.5 Comparison of performance

This section presents the results of evaluating our ILS against the MIP model and the ALNS using benchmark instances containing 25, 50, and 100 tasks. In the tables presented hereafter, the first group of columns shows the instance identifier, the number of tasks $|C|$, and the maximum number of technicians $|K|$. Columns Opt. and Avg. show, for each instance, the optimal solution value found by CPLEX, and the average solution values found by the ALNS and ILS over five random runs. Columns $\text{Imp}^*_{\text{C/I}}$ and $\text{Imp}^*_{\text{A/I}}$ give the relative percentage differences between the values of the ILS solutions and the CPLEX solutions and the ALNS solutions, respectively. The average number of outsourced tasks, the average number of technicians used, and the average CPU time in seconds are reported in the columns headed $|C_o|$, $|K^*|$, and CPU, respectively. Emboldening in the ILS columns is used to highlight values that correspond to an improvement over the corresponding values of the ALNS.

Table 3 gives experimental results on small instances containing 25 tasks. Compared to CPLEX, our ILS algorithm consistently finds optimal solutions in all five random runs for 19 out of 23 instances and produces an overall average gap of −0.18% over all instances. Moreover, the average number of outsourced tasks given by the ILS is exactly the same as that for CPLEX. Compared to ALNS, our ILS algorithm gives better solutions for four instances, in particular RC101_5 × 4 and RC101_6 × 6, for which the solutions found by the ILS improve the ALNS solutions by 9.32% and 12.56% respectively. The significant improvement on these two instances is achieved by the reduction in the number of outsourced tasks. To test the statistical significance between the performances of ALNS and ILS, we conduct the two-tailed Wilcoxon test on the paired samples between the average solution values obtained by ALNS and ILS. The test is performed at a 95% significance level, where a p value of less than 0.05 indicates the rejection of the null hypothesis, which says that there is no significant difference between the results of ALNS and ILS. The p value of the Wilcoxon test for instances containing 25 tasks is 0.24, which suggests that the performances of ALNS

Table 3 Comparison of exact, ALNS and ILS solutions on small instances with 25 tasks

Instance	$\|C\|$	$\|K\|$	CPLEX				ALNS			ILS					
			Opt.	$\|C_o\|$	$\|K^*\|$	CPU	Avg.	$\|C_o\|$	CPU	Avg.	$\text{Imp}^*_{\text{C/I}}$	$\text{Imp}^*_{\text{A/I}}$	$\|C_o\|$	$\|K^*\|$	CPU
C101_5x4	25	4	271.70	0.00	4.00	0.16	271.70	0.00	1.63	272.96	−0.46	−0.46	0.00	4.00	0.11
C201_5x4	25	2	863.08	3.00	2.00	0.09	863.08	3.00	2.45	863.08	0.00	0.00	3.00	2.00	0.09
C203_5x4	25	2	835.83	3.00	1.00	214.45	835.83	3.00	0.00	835.83	0.00	0.00	3.00	1.00	0.15
R101_5x4	25	4	2195.04	9.00	4.00	0.34	2195.04	9.00	1.26	2195.04	0.00	0.00	9.00	4.00	0.22
R201_5x4	25	2	1091.07	3.00	2.00	0.86	1092.41	3.00	1.45	**1091.07**	0.00	0.12	3.00	2.00	0.03
RC101_5x4	25	4	862.21	2.00	4.00	13.51	950.81	2.40	1.41	**862.21**	0.00	9.32	2.00	4.00	0.37
RC201_5x4	25	3	465.25	0.00	3.00	1.12	465.25	0.00	1.71	465.31	−0.01	−0.01	0.00	3.00	0.06
C101_6x6	25	4	927.35	3.00	3.00	0.14	927.35	3.00	1.77	927.35	0.00	0.00	3.00	3.00	0.09
C201_6x6	25	2	1217.10	4.00	1.00	0.13	1217.10	4.00	4.56	1217.10	0.00	0.00	4.00	1.00	0.01
C203_6x6	25	2	930.60	3.00	1.00	5.73	930.60	3.00	1.86	930.60	0.00	0.00	3.00	1.00	0.03
R101_6x6	25	4	2857.05	12.00	4.00	0.17	2977.63	12.60	1.28	**2868.19**	−0.39	3.68	12.00	4.00	0.30
R201_6x6	25	2	1377.42	4.00	1.00	0.53	1377.42	4.00	1.76	1422.57	−3.28	−3.28	4.00	2.00	0.05
RC101_6x6	25	4	1361.80	4.00	4.00	1.95	1557.44	5.00	1.47	**1361.80**	0.00	12.56	4.00	4.00	0.23
RC201_6x6	25	3	1228.89	3.00	2.00	22.79	1228.89	3.00	1.40	1228.89	0.00	0.00	3.00	2.00	0.14
C101_7x4	25	4	789.08	2.00	4.00	0.16	789.08	2.00	1.86	789.08	0.00	0.00	2.00	4.00	0.06
C103_7x4	25	4	671.06	2.00	3.00	3993.12	671.06	2.00	2.03	671.06	0.00	0.00	2.00	3.00	0.11
C201_7x4	25	2	738.35	2.00	2.00	0.05	738.35	2.00	1.60	738.35	0.00	0.00	2.00	2.00	0.02
C203_7x4	25	2	684.98	2.00	2.00	190.13	684.98	2.00	1.86	684.98	0.00	0.00	2.00	2.00	0.03
R101_7x4	25	4	2447.74	10.00	4.00	0.09	2447.74	10.00	1.27	2447.74	0.00	0.00	10.00	4.00	0.12
R201_7x4	25	2	959.51	2.00	2.00	0.19	959.51	2.00	1.49	959.51	0.00	0.00	2.00	2.00	0.07
R203_7x4	25	2	849.47	2.00	2.00	496.32	849.47	2.00	1.88	849.47	0.00	0.00	2.00	2.00	0.03
RC101_7x4	25	4	1669.63	6.00	4.00	0.50	1669.63	6.00	1.36	1669.63	0.00	0.00	6.00	4.00	0.09
RC201_7x4	25	3	967.60	2.00	3.00	5.46	967.60	2.00	2.00	967.60	0.00	0.00	2.00	3.00	0.08
Average			1141.82	3.61	2.70	215.13	1159.48	3.70	1.71	1144.32	−0.18	0.95	3.61	2.74	0.11

and ILS on this set of instances are similar. This can be explained by the fact that both ALNS and ILS can solve a large majority of small instances to optimality. Perhaps the most significant feature of ILS is the speed with which it produces good-quality solutions, and it is significantly faster than the ALNS. With an average CPU time of 0.11 s, it only requires 7% of the time used by the ALNS. Although our computer is faster, the effect of the computer speed is negligible compared to the improvement on CPU times.

Table 4 presents results of the experiments on instances with 50 tasks. Of the 12 instances, our ILS algorithm discovers optimal solutions for seven and yields an overall average deviation of −0.14% in comparison to CPLEX. The average deviation of the ILS from the ALNS in terms of the solution values is 0.67%, and it finds better solutions for five instances. The p value of the Wilcoxon test for this set of instances is 0.06, which is very close to the margin of significance. This suggests that when the problem size increases to 50, our ILS tends to perform better than the ALNS. Using the computer speed factors, the average computation time of the ALNS is adjusted to 4.77 s, which is still considerably greater than the 1.89 s for ILS.

For instances with 100 tasks, a time limit of 7200 s is imposed on CPLEX. Tables 5 and 6 report computational results on instances with limited and unlimited technicians, respectively. The third to fifth columns of each table are associated to the results of the MIP model solved by CPLEX, where the columns Best and Gap present, for each instance, the value of the optimal or best solution found by CPLEX within the time limit, and the percentage gap of the LP bound with respect to the best solution value. In addition, we report the minimum and maximum solution values found by the ALNS and ILS over five random runs in columns titled Min. and Max., and the corresponding percentage differences between the values of ALNS and ILS solutions are presented in columns $\text{Imp}^{-}_{\text{A/I}}$ and $\text{Imp}^{+}_{\text{A/I}}$ respectively. Proven optimal solutions are underlined.

Of the 36 instances with unlimited technicians, CPLEX is only able to find optimal solutions for 9, and for the 36 instances with limited technicians, the model finds optimal solutions for 5 instances within the required time limit. This indicates that instances with limited technicians tend to be more difficult to solve than those with unlimited technicians, as the former problem considers the additional set of decisions concerning the selection of tasks to be outsourced.

A comparison of ILS and ALNS on instances with 100 tasks and limited technicians is given in Table 5. Of the 36 instances, our ILS algorithm outperforms ALNS in 17. In particular, for instances R101_5 × 4, RC101_6 × 6 and RC101_7 × 4, the solutions found by the ILS are between 5% and 8% better in cost than those for ALNS. The significant improvement on these instances can be explained by the reduced use of the outsourcing option by the ILS. The average number of outsourced tasks of the ILS solutions is 9.76, which is about 3% less than the value of the ALNS solutions. To determine the statistical significance between the numbers of outsourced tasks produced by the ILS and ALNS on this set of instances, we conduct a two-tailed Wilcoxon test and a p value of 0.004 is obtained. This confirms that our ILS uses significantly less outsourcing option than the ALNS, and also implies that the proposed mechanism of reducing outsourcing cost (described in Sect. 4.6) is effective. The average percentage difference between the ILS and ALNS solution values is 0.82%. Comparing the worst solutions found during five random runs, the ILS improves the

Table 4 Comparison of exact, ALNS and ILS solutions on medium instances with 50 tasks

Instance	$\|C\|$	$\|K\|$	CPLEX				ALNS			ILS					
			Opt.	$\|C_o\|$	$\|K^*\|$	CPU	Avg.	$\|C_o\|$	CPU	Avg.	$\text{Imp}^*_{\text{C/I}}$	$\text{Imp}^*_{\text{A/I}}$	$\|C_o\|$	$\|K^*\|$	CPU
C101_5x4	50	6	830.00	1.00	6.00	2.61	838.11	1.00	6.98	**830.00**	0.00	0.97	1.00	6.00	1.09
C201_5x4	50	4	859.54	1.00	4.00	0.59	859.54	1.00	7.60	859.54	0.00	0.00	1.00	4.00	0.78
R101_5x4	50	6	4507.87	19.00	6.00	1.20	4540.34	19.20	7.22	**4511.36**	−0.08	0.64	19.00	6.00	6.84
R201_5x4	50	4	1107.51	1.00	4.00	107.52	1107.51	1.00	7.49	1112.25	−0.43	−0.43	1.00	4.00	2.13
C101_6x6	50	6	1154.84	3.00	5.00	251.94	1181.31	3.00	7.68	**1154.84**	0.00	2.24	3.00	5.00	1.95
C201_6x6	50	4	1203.93	3.00	3.00	0.42	1203.93	3.00	8.15	1203.93	0.00	0.00	3.00	3.00	0.66
R101_6x6	50	6	5190.35	22.00	6.00	1.22	5362.77	23.00	6.13	**5190.32**	0.00	3.22	22.00	6.00	2.62
R201_6x6	50	4	1647.07	3.00	3.00	794.06	1647.00	3.00	8.27	1649.95	−0.17	−0.18	3.00	3.00	1.99
C101_7x4	50	6	1356.54	3.00	6.00	7.61	1367.75	3.00	7.71	1367.75	−0.83	0.00	3.00	6.00	1.26
C201_7x4	50	4	1312.21	3.00	3.00	0.42	1312.21	3.00	7.87	1312.21	0.00	0.00	3.00	3.00	0.37
R101_7x4	50	6	4463.80	18.00	6.00	1.15	4540.34	18.60	7.22	**4469.31**	−0.12	1.56	18.00	6.00	2.01
R201_7x4	50	4	1553.23	3.00	4.00	129.01	1553.23	3.00	7.19	1553.23	0.00	0.00	3.00	4.00	0.93
Average			2098.91	6.67	4.67	108.15	2126.17	6.82	7.46	2101.22	−0.14	0.67	6.67	4.67	1.89

Table 5 Comparison of the ILS and the ALNS on benchmark instances with 100 tasks and limited technicians

Instance	$\|K\|$	CPLEX			ALNS						ILS								
		Best	Gap	CPU	Avg.	Min.	Max.	$\|C_o\|$	$\|K^*\|$	CPU	Avg.	$\text{Imp}^*_{\text{A/I}}$	Min.	$\text{Imp}^-_{\text{A/I}}$	Max.	$\text{Imp}^+_{\text{A/I}}$	$\|C_o\|$	$\|K^*\|$	CPU
C101_5x4	8	5857.35	17.54	7200.00	5733.75	5656.63	5806.55	23.40	8.00	34.90	**5691.28**	0.74	**5589.76**	1.18	**5780.75**	0.44	22.80	8.00	33.65
C103_5x4	8	7874.13	92.02	7200.00	2782.20	2644.65	2869.64	7.60	8.00	49.49	2830.77	−1.75	2650.69	−0.23	2921.15	−1.79	7.80	8.00	38.05
C201_5x4	4	2755.52	0.00	14.40	2755.52	2755.52	2755.52	6.00	4.00	35.84	2781.37	−0.94	2755.52	0.00	2800.21	−1.62	6.00	4.00	11.40
C203_5x4	4	3630.92	46.67	7200.00	2392.50	2389.37	2393.62	6.00	4.00	70.46	2393.88	−0.06	**2383.01**	0.27	2407.41	−0.58	6.00	4.00	15.89
R101_5x4	12	5446.89	0.01	4229.83	5895.38	5582.58	6181.52	22.60	12.00	54.30	**5572.16**	5.48	**5561.83**	0.37	**5590.55**	9.56	21.00	12.00	62.46
R103_5x4	12	4856.22	81.38	7200.00	1845.25	1710.25	2020.48	2.00	12.00	60.09	**1765.16**	4.34	**1658.23**	3.04	**1943.30**	3.82	1.60	12.00	63.82
R201_5x4	4	3295.67	23.10	7200.00	2854.30	2838.50	2865.75	6.00	4.00	39.27	2866.39	−0.42	2839.54	−0.04	2887.93	−0.77	6.00	4.00	14.14
R203_5x4	4	4042.66	50.47	7200.00	2332.23	2332.23	2332.23	6.00	4.00	68.40	2349.24	−0.73	2335.76	−0.15	2360.57	−1.22	6.00	4.00	12.86
RC101_5x4	11	5441.33	61.97	7200.00	5164.84	5127.79	5262.36	18.20	11.00	50.49	**4983.98**	3.50	**4909.89**	4.25	**5093.47**	3.21	17.40	11.00	46.74
RC103_5x4	11	6142.71	87.10	7200.00	2348.06	2170.57	2490.12	4.20	11.00	57.80	2421.65	−3.13	2301.86	−6.05	2499.04	−0.36	4.60	11.00	39.24
RC201_5x4	5	3579.44	25.17	7200.00	3091.67	3088.23	3093.56	6.00	5.00	40.73	**3090.03**	0.05	**3083.49**	0.15	3100.00	−0.21	6.00	5.00	15.79
RC203_5x4	5	3579.20	43.86	7200.00	2540.35	2516.16	2550.62	6.00	5.00	67.95	**2530.63**	0.38	**2512.64**	0.14	2568.56	−0.70	6.00	4.80	12.54
C101_6x6	8	7660.86	9.61	7200.00	7762.94	7731.07	7791.08	32.00	8.00	36.72	**7695.83**	0.86	**7660.86**	0.91	**7783.33**	0.10	31.80	8.00	30.17
C103_6x6	8	9050.71	91.86	7200.00	5028.83	4980.70	5136.21	18.20	8.00	55.31	5066.61	−0.75	**4971.66**	0.18	5195.96	−1.16	18.40	8.00	37.08
C201_6x6	4	3315.30	6.02	7200.00	3299.56	3278.07	3328.01	9.60	3.40	35.35	3313.45	−0.42	3298.68	−0.63	3331.26	−0.10	9.20	3.80	21.10
C203_6x6	4	5443.79	63.09	7200.00	2465.90	2460.17	2468.71	6.00	3.00	67.12	2479.44	−0.55	2468.53	−0.34	2484.72	−0.65	6.00	3.00	23.56
R101_6x6	13	5944.91	0.01	2038.29	6152.29	5955.17	6322.82	23.00	13.00	62.19	**6005.32**	2.39	**5948.71**	0.11	**6083.98**	3.78	22.20	13.00	56.05
R103_6x6	13	8799.51	85.42	7200.00	2329.25	2251.64	2404.57	4.40	12.00	72.51	**2290.01**	1.68	**2225.67**	1.15	**2383.03**	0.90	4.20	12.20	61.65
R201_6x6	4	5169.21	48.79	7200.00	3536.70	3503.40	3574.97	8.60	4.00	38.66	3574.46	−1.07	3510.36	−0.20	3633.35	−1.63	8.80	4.00	32.37
R203_6x6	4	5595.34	62.06	7200.00	2446.18	2437.28	2481.77	6.00	3.00	71.59	2462.68	−0.67	2443.97	−0.27	2504.36	−0.91	6.00	3.00	23.00
RC101_6x6	12	7621.03	70.53	7200.00	5466.44	5276.34	5771.99	18.60	12.00	57.54	**5029.94**	7.99	**4975.34**	5.70	**5142.08**	10.91	16.40	12.00	43.80
RC103_6x6	12	6878.60	85.93	7200.00	2349.57	2263.83	2522.71	3.20	12.00	64.98	**2257.78**	3.91	**2113.03**	6.66	**2337.45**	7.34	2.80	12.00	45.74
RC201_6x6	4	7017.96	60.69	7200.00	4519.95	4422.86	4656.79	12.60	4.00	33.34	4550.99	−0.69	4490.33	−1.53	**4608.61**	1.03	12.60	4.00	31.23

Table 5 continued

Instance	$\|K\|$	CPLEX			ALNS						ILS								
		Best	Gap	CPU	Avg.	Min.	Max.	$\|C_o\|$	$\|K^*\|$	CPU	Avg.	$\text{Imp}^*_{\text{A/I}}$	Min.	$\text{Imp}^-_{\text{A/I}}$	Max.	$\text{Imp}^+_{\text{A/I}}$	$\|C_o\|$	$\|K^*\|$	CPU
RC203_6x6	4	5683.37	62.73	7200.00	2673.72	2649.51	2730.78	6.00	3.00	62.53	2686.83	−0.49	2671.23	−0.82	**2719.03**	0.43	6.00	3.00	19.58
C101_7x4	9	5208.99	9.11	7200.00	5257.90	5208.30	5307.12	19.00	9.00	35.89	5284.48	−0.51	5246.13	−0.73	5360.98	−1.01	19.00	9.00	19.88
C103_7x4	9	6657.08	86.65	7200.00	2117.44	2020.40	2173.38	2.60	9.00	49.85	**2059.98**	2.71	**2009.86**	0.52	**2163.05**	0.48	2.00	9.00	24.70
C201_7x4	4	2773.41	0.00	97.79	2779.37	2773.41	2803.21	5.00	4.00	32.51	2808.29	−1.04	2781.07	−0.28	2830.03	−0.96	5.00	4.00	8.11
C203_7x4	4	3048.56	41.76	7200.00	2282.15	2261.37	2301.73	5.00	4.00	61.64	2297.16	−0.66	2262.00	−0.03	2366.11	−2.80	5.00	4.00	9.98
R101_7x4	14	5079.67	0.00	1798.64	5381.35	5239.81	5437.66	18.80	14.00	55.20	**5238.11**	2.66	**5127.29**	2.15	**5283.80**	2.83	17.80	14.00	33.24
R103_7x4	14	5603.74	82.34	7200.00	2215.84	2104.92	2314.30	3.40	14.00	63.37	2222.76	−0.31	2139.77	−1.66	2333.64	−0.84	3.40	14.00	33.70
R201_7x4	5	2790.76	13.57	7200.00	2679.38	2672.96	2682.23	5.00	5.00	41.29	**2678.32**	0.04	**2664.93**	0.30	2693.43	−0.42	5.00	5.00	9.51
R203_7x4	5	3209.88	43.55	7200.00	2209.80	2199.10	2229.67	5.00	5.00	66.75	2223.96	−0.64	2209.32	−0.46	2242.78	−0.59	5.00	5.00	10.09
RC101_7x4	12	5692.20	36.43	7200.00	5799.77	5531.06	6367.47	20.40	12.00	52.57	**5440.59**	6.19	**5373.05**	2.86	**5556.76**	12.73	18.40	12.00	29.21
RC103_7x4	12	7660.38	88.19	7200.00	2674.54	2586.03	2820.48	5.00	12.00	57.89	**2615.16**	2.22	2591.39	−0.21	**2653.73**	5.91	5.00	12.00	24.52
RC201_7x4	5	3278.57	25.12	7200.00	2936.28	2919.83	2945.46	5.00	5.00	40.06	**2934.44**	0.06	**2910.73**	0.31	2948.77	−0.11	5.00	5.00	9.44
RC203_7x4	5	3739.80	51.34	7200.00	2285.17	2277.62	2301.26	5.00	5.00	58.82	2308.80	−1.03	2305.62	−1.23	2314.51	−0.58	5.00	4.40	10.14
Average		5261.82	45.95	6427.19	3510.73	3439.37	3597.12	10.04	7.54	52.87	3466.72	0.82	3416.16	0.43	3525.21	1.24	9.76	7.53	28.18

ALNS solutions by 1.24%, which indicates that our ILS is more stable than the ALNS when performing multiple runs. The average computational time required by ALNS is 52.87 s, which is equivalent to 33.78 s after applying the conversion factor, and is 16.59% higher than that of ILS.

Table 6 provides a comparison of ILS and ALNS on large instances with unlimited technicians. The average number of outsourced tasks is not reported in this table, as these instances have enough technicians to avoid outsourcing. The ILS algorithm outperforms ALNS in 30 out of 36 instances, and improves the best solutions for 24 instances. Of the 9 instances that are solved to optimality by CPLEX, our ILS algorithm finds optimal solutions for 5 of them. The average percentage difference between the ILS and ALNS solution values is 0.64%. Moreover, the ILS solutions tend to have smaller deviations within five random runs since the overall average values of $\text{Imp}^{-}_{\text{A/I}}$ and $\text{Imp}^{+}_{\text{A/I}}$ are both greater than 0. In terms of speed, ALNS requires an average solution time of 79.17 s, which is equivalent to 50.58 s under the adjustment of computer speeds, but is still 20% higher than the average CPU time required by ILS. Lastly, we conduct a two-tailed Wilcoxon test on the solution values of all large instances containing 100 tasks and a p value of 0.02 is obtained. This indicates that our ILS has a significantly better performance than the ALNS on the set of large instances since the p value is less than the chosen significance level 0.05.

5.6 Skill VRP instances

The proposed ILS algorithm is also applied to solve a set of Skill VRP instances, which are generated based on the benchmark instances of Solomon (1987) and the skill pattern introduced by Cappanera et al. (2011). As the Skill VRP does not involve time window and capacity constraints, we use only the geographical information of Solomon's instances to generate three types of geographical data for Skill VRP instances, namely, R, C and RC, which represent the random, clustered and a mixed of random and clustered geographical setting, respectively. Similar to Cappanera et al. (2011) and Schwarze and Voß (2012), we consider a skill set with three levels 1, 2 and 3, where skill 1 denotes the lowest level, and skill 3 the highest. Each task $i \in C$ is associated with a skill requirement $s_i \in \{1, 2, 3\}$, which must be fulfilled by a technician $k \in K$ having a skill level $\hat{s}_k \geq s_i$, where $\hat{s}_k \in \{1, 2, 3\}$. The skill data is randomly generated according to the four patterns introduced by Cappanera et al. (2011), as given in the Table 7, where each row of values represent a pattern that indicates the distribution of skill requirements over tasks. For example, the first pattern {50, 10, 40} indicates that a task i has a skill requirement $s_i = 1$ with probability 0.5, $s_i = 2$ with probability 0.1 and $s_i = 3$ with probability 0.4. For each combination of skill pattern and geographical data, we generated three random instances, which results in a total of 36 instances. All the instances have two sizes, where one has 20 tasks and the other has 30 tasks. Each instance has a set of three technicians $K = \{1, 2, 3\}$, where each technician $k \in K$ is specialised at a different skill level $\hat{s}_k \in S$; for example, $\hat{s}_1 = 1$, $\hat{s}_2 = 2$ and $\hat{s}_3 = 3$.

In the Skill VRP, the routing costs depend on both the traveling distance and the technician, such that the increasing skill level of the technician causes increasing costs. Thus, for each arc $(i, j) \in A$ and each technician $k \in K$, we follow the approach of

Table 6 Comparison of exact, ALNS and ILS solutions on benchmark instances with 100 tasks and unlimited technicians

Instance	$\|K\|$	CPLEX			ALNS					ILS							
		Best	Gap	CPU	Avg.	Min.	Max.	$\|K^*\|$	CPU	Avg.	$\text{Imp}^*_{\text{A/I}}$	Min.	$\text{Imp}^-_{\text{A/I}}$	Max.	$\text{Imp}^+_{\text{A/I}}$	$\|K^*\|$	CPU
C101_5x4	17	1096.85	0.00	2640.69	1111.08	1098.71	1128.02	13.00	66.50	**1107.76**	0.30	**1097.67**	0.09	**1117.56**	0.93	12.60	33.70
C103_5x4	17	2604.28	81.10	7200.00	1037.33	1018.61	1049.41	11.60	84.61	**1026.51**	1.04	**1012.86**	0.56	**1045.80**	0.34	11.00	38.63
C201_5x4	8	1157.56	0.00	9.44	1180.93	1158.97	1228.88	7.40	54.29	**1157.56**	1.98	**1157.56**	0.12	**1157.56**	5.80	7.00	9.21
C203_5x4	8	2449.89	72.07	7200.00	1049.30	1046.93	1052.83	5.00	89.78	1059.58	−0.98	1054.21	−0.70	1068.72	−1.51	5.20	21.17
R101_5x4	25	1652.13	0.00	6093.49	1685.85	1687.68	1697.20	20.00	84.25	**1668.00**	1.06	**1658.93**	1.70	**1676.64**	1.21	20.40	43.66
R103_5x4	25	2604.28	81.10	7200.00	1249.91	1238.67	1282.28	14.80	101.77	**1243.54**	0.51	**1235.05**	0.29	**1253.46**	2.25	15.20	58.66
R201_5x4	7	1569.42	18.54	7200.00	1448.93	1440.30	1462.62	7.00	57.22	**1436.37**	0.87	**1431.16**	0.63	**1447.84**	1.01	6.60	27.61
R203_5x4	7	2477.25	68.27	7200.00	1106.12	1098.00	1123.08	5.80	84.18	**1100.75**	0.49	**1097.55**	0.04	**1105.09**	1.60	6.00	23.91
RC101_5x4	22	2503.98	42.91	7200.00	1716.07	1702.51	1729.75	16.40	78.25	**1695.67**	1.19	**1673.94**	1.68	**1710.32**	1.12	16.80	51.32
RC103_5x4	22	6810.66	89.74	7200.00	1354.11	1337.99	1388.13	12.40	91.41	1355.40	−0.09	**1321.66**	1.22	**1383.61**	0.33	12.20	59.19
RC201_5x4	9	1849.86	24.94	7200.00	1607.25	1601.89	1610.75	8.60	58.71	**1606.08**	0.07	**1589.24**	0.79	1620.59	−0.61	8.00	27.02
RC203_5x4	9	2724.35	72.45	7200.00	1166.50	1161.53	1178.64	6.00	87.01	**1165.81**	0.06	1162.95	−0.12	**1169.24**	0.80	6.00	24.84
C101_6x6	16	972.89	0.00	2044.98	1004.82	989.21	1028.72	11.40	69.76	**988.43**	1.63	**972.89**	1.65	**1001.62**	2.63	11.40	45.79
C103_6x6	16	3897.18	88.27	7200.00	897.86	893.94	907.65	10.40	88.17	911.62	−1.53	900.82	−0.77	933.90	−2.89	10.40	53.15
C201_6x6	7	821.55	0.00	664.21	821.55	821.55	821.55	4.00	55.65	826.42	−0.59	**821.55**	0.00	832.56	−1.34	4.60	42.76
C203_6x6	7	2164.58	74.26	7200.00	703.10	689.60	750.12	4.00	99.75	**693.50**	1.36	**689.60**	0.00	**699.19**	6.79	4.00	44.15
R101_6x6	26	1648.27	0.00	7100.00	1667.43	1658.27	1672.57	19.80	96.78	**1660.15**	0.44	**1657.55**	0.04	**1664.28**	0.50	19.80	67.27
R103_6x6	26	15070.30	94.47	7200.00	1231.49	1223.63	1243.49	14.00	116.00	**1225.80**	0.46	**1216.08**	0.62	**1238.01**	0.44	14.40	73.23
R201_6x6	7	1462.90	23.69	7200.00	1270.26	1261.94	1279.81	6.00	62.02	**1270.25**	0.00	1265.56	−0.29	**1278.94**	0.07	5.80	55.72
R203_6x6	7	3329.47	79.29	7200.00	951.84	932.35	964.54	5.40	99.46	**933.41**	1.94	**930.74**	0.17	**940.98**	2.44	4.00	56.79
RC101_6x6	24	2317.68	40.98	7200.00	1683.96	1679.13	1690.06	15.60	90.42	**1674.61**	0.56	**1663.30**	0.94	**1682.62**	0.44	16.00	61.94
RC103_6x6	24	5614.42	87.81	7200.00	1310.95	1281.55	1331.41	11.80	103.68	**1309.18**	0.14	1297.09	−1.21	**1331.28**	0.01	12.00	86.38

Table 6 continued

Instance	$\|K\|$	CPLEX			ALNS					ILS							
		Best	Gap	CPU	Avg.	Min.	Max.	$\|K^*\|$	CPU	Avg.	$\text{Imp}^*_{\text{A/I}}$	Min.	$\text{Imp}^-_{\text{A/I}}$	Max.	$\text{Imp}^+_{\text{A/I}}$	$\|K^*\|$	CPU
RC201_6x6	8	1849.86	24.94	7200.00	1406.95	1395.40	1411.48	6.40	61.32	**1380.38**	1.89	**1367.89**	1.97	**1394.90**	1.17	6.40	52.16
RC203_6x6	8	2337.76	72.19	7200.00	1016.71	1001.04	1030.15	5.40	91.39	**1014.51**	0.22	1003.81	−0.28	**1024.80**	0.52	5.40	49.44
C101_7x4	17	1357.05	0.00	281.50	1398.95	1357.05	1462.16	15.20	58.65	**1370.78**	2.01	**1357.05**	0.00	**1381.59**	5.51	14.80	22.31
C103_7x4	17	3506.12	82.13	7200.00	1239.22	1215.70	1264.17	13.20	75.26	**1233.60**	0.45	1220.19	−0.37	**1256.67**	0.59	13.20	27.96
C201_7x4	8	1256.30	0.00	2.78	1282.18	1256.30	1302.56	8.00	49.54	**1263.00**	1.50	**1256.30**	0.00	**1289.68**	0.99	8.00	10.56
C203_7x4	8	2401.88	67.78	7200.00	1151.27	1150.85	1152.94	7.80	76.83	**1145.36**	0.51	**1137.07**	1.20	**1150.85**	0.18	7.60	17.89
R101_7x4	28	1764.78	0.00	2875.72	1793.95	1776.46	1813.53	21.40	88.52	**1787.22**	0.38	1781.13	−0.26	**1796.48**	0.94	22.00	39.28
R103_7x4	28	7412.24	87.71	7200.00	1375.09	1346.80	1399.95	16.20	102.08	**1349.32**	1.87	**1337.92**	0.66	**1373.27**	1.91	16.40	43.54
R201_7x4	10	1411.36	5.83	7200.00	1410.90	1398.14	1427.95	9.20	59.31	**1406.55**	0.31	1401.68	−0.25	**1412.46**	1.08	9.40	19.36
R203_7x4	10	2476.19	67.43	7200.00	1166.94	1164.90	1169.27	8.00	80.29	**1164.89**	0.18	**1160.51**	0.38	1170.12	−0.07	8.00	18.45
RC101_7x4	23	1902.94	17.32	7200.00	1844.37	1821.90	1859.17	17.80	75.55	**1822.30**	1.20	**1805.39**	0.91	**1837.41**	1.17	17.60	36.51
RC103_7x4	23	3367.93	77.17	7200.00	1455.33	1435.63	1477.84	13.40	85.04	**1436.23**	1.31	**1427.40**	0.57	**1450.97**	1.82	13.60	38.30
RC201_7x4	9	1823.30	17.64	7200.00	1701.25	1697.82	1705.48	9.00	52.78	1706.24	−0.29	**1697.82**	0.00	1727.00	−1.26	9.00	15.34
RC203_7x4	9	2070.17	60.02	7200.00	1241.65	1239.45	1249.72	7.20	73.82	**1235.76**	0.47	**1230.63**	0.71	**1238.74**	0.88	8.00	16.81
Average		2826.05	45.00	6003.13	1298.37	1285.57	1315.22	10.79	79.17	1289.79	0.64	1280.35	0.35	1301.80	1.05	10.80	39.28

Table 7 Distribution of skill requirements over tasks

Pattern	Skill		
	1 (%)	2 (%)	3 (%)
1	50	10	40
2	50	20	30
3	40	40	20
4	30	30	40

Schwarze and Voß (2012) by defining a skill dependent routing cost c_{ij}^k by

$$c_{ij}^k = c_{ij}\theta\hat{s}_k, \tag{16}$$

where c_{ij} is the traveling distance of arc $(i, j) \in A$, and θ is a weight parameter of the skill level $\hat{s}_k$ of the technician $k \in K$. Following the suggestion of Schwarze and Voß (2012), we set $\theta = 1$ in our experiments.

5.7 Results for skill VRP instances

The above Skill VRP instances are solved by using our ILS, and the results are compared with the solutions obtained from a basic MIP model of Cappanera et al. (2011) that is solved by using CPLEX 12.6. A time limit of 7200 s is imposed on CPLEX, and for instances not solved to optimality, we report the best values of the solutions found within this time limit.

Table 8 presents results of the experiments for instances with 20 tasks. Of the 36 instances tested, CPLEX finds optimal solutions for 27 and exceeds the time limit for 9 instances. The solutions produced by our ILS algorithm are exactly the same as the optimal or best solutions found by CPLEX for all instances. The average computational time of our ILS is 0.08 s, which is negligible compared to the time used by CPLEX.

Table 9 shows results of the experiments on instances with 30 tasks. For this size of instances, CPLEX is only able to find optimal solutions for 10 out of 36 instances. Among these 10 instances, our ILS can produce optimal solutions for 9, with the exception being instance R_4_1 for which our ILS found slightly worse solutions that have an average gap of −0.66% to that of CPLEX. Of the remaining instances that are not solved to optimality by CPLEX, our ILS produces better solutions for 6 and equal cost solutions for 20 compared to the best solutions found by CPLEX within the time limit. The average percentage difference between the values of our ILS solutions and CPLEX solutions is 0.74%. The average computational time required by our ILS is 0.48 s, which is also negligible compared to the time used by CPLEX.

Table 8 The comparison of exact and ILS solutions on Skill VRP instances with 20 tasks

Instance	CPLEX				ILS			
	Best	Gap	$\|K^*\|$	CPU	Avg.	$\text{Imp}^*_{\text{C/I}}$	$\|K^*\|$	CPU
C_1_1	370.93	0.00	1.00	122.82	370.93	0.00	1.00	0.07
C_1_2	367.55	4.79	2.00	7200.00	367.55	0.00	2.00	0.12
C_1_3	367.55	8.85	2.00	7200.00	367.55	0.00	2.00	0.09
C_2_1	370.93	0.00	1.00	353.27	370.93	0.00	1.00	0.06
C_2_2	367.55	11.36	2.00	7200.00	367.55	0.00	2.00	0.17
C_2_3	367.55	9.63	2.00	7200.00	367.55	0.00	2.00	0.08
C_3_1	370.93	0.00	1.00	239.05	370.93	0.00	1.00	0.06
C_3_2	367.55	4.38	2.00	7200.00	367.55	0.00	2.00	0.15
C_3_3	367.55	0.00	2.00	2742.05	367.55	0.00	2.00	0.17
C_4_1	370.93	0.00	1.00	28.27	370.93	0.00	1.00	0.09
C_4_2	367.55	0.00	2.00	1307.39	367.55	0.00	2.00	0.13
C_4_3	370.93	0.00	1.00	1648.17	370.93	0.00	1.00	0.06
R_1_1	781.09	0.00	2.00	8.78	781.09	0.00	2.00	0.07
R_1_2	772.50	0.00	2.00	73.35	772.50	0.00	2.00	0.10
R_1_3	710.16	0.00	2.00	18.66	710.16	0.00	2.00	0.04
R_2_1	781.09	0.00	2.00	33.40	781.09	0.00	2.00	0.11
R_2_2	729.12	0.00	2.00	76.47	729.12	0.00	2.00	0.06
R_2_3	710.16	0.00	2.00	68.49	710.16	0.00	2.00	0.09
R_3_1	777.40	0.00	3.00	191.36	777.40	0.00	3.00	0.04
R_3_2	702.26	0.00	2.00	143.98	702.26	0.00	2.00	0.09
R_3_3	748.90	0.00	2.00	323.30	748.90	0.00	2.00	0.14
R_4_1	787.01	0.00	1.00	6.76	787.01	0.00	1.00	0.08
R_4_2	787.01	0.00	1.00	15.46	787.01	0.00	1.00	0.09
R_4_3	755.20	0.00	2.00	56.62	755.20	0.00	2.00	0.07
RC_1_1	658.21	0.00	1.00	27.38	658.21	0.00	1.00	0.05
RC_1_2	658.21	1.89	1.00	7200.00	658.21	0.00	1.00	0.03
RC_1_3	570.00	0.00	2.00	2034.46	570.00	0.00	2.00	0.03
RC_2_1	658.21	0.00	1.00	83.90	658.21	0.00	1.00	0.03
RC_2_2	658.21	8.77	1.00	7200.00	658.21	0.00	1.00	0.04
RC_2_3	570.00	4.21	2.00	7200.00	570.00	0.00	2.00	0.03
RC_3_1	658.21	0.00	1.00	786.79	658.21	0.00	1.00	0.06
RC_3_2	658.21	13.29	1.00	7200.00	658.21	0.00	1.00	0.04
RC_3_3	570.00	0.00	2.00	893.34	570.00	0.00	2.00	0.03
RC_4_1	658.21	0.00	1.00	26.89	658.21	0.00	1.00	0.05
RC_4_2	658.21	0.00	1.00	3180.79	658.21	0.00	1.00	0.03
RC_4_3	658.21	0.00	1.00	3042.84	658.21	0.00	1.00	0.03
Average	586.20	1.87	1.58	2287.06	586.20	0.00	1.58	0.08

Table 9 The comparison of exact and ILS solutions on Skill VRP instances with 30 tasks

Instance	CPLEX				ILS			
	Best	Gap	$\|K^*\|$	CPU	Avg.	$\text{Imp}^*_{\text{C/I}}$	$\|K^*\|$	CPU
C_1_1	439.45	2.87	1.00	7200.00	439.45	0.00	1.00	0.33
C_1_2	432.86	25.93	1.00	7200.00	432.86	0.00	2.00	0.11
C_1_3	429.52	31.50	2.00	7200.00	429.52	0.00	2.00	0.37
C_2_1	439.45	11.92	1.00	7200.00	439.45	0.00	1.00	0.40
C_2_2	432.86	26.22	2.00	7200.00	432.86	0.00	2.00	0.12
C_2_3	429.52	32.25	2.00	7200.00	429.52	0.00	2.00	0.39
C_3_1	439.45	23.09	2.00	7200.00	439.45	0.00	1.00	0.41
C_3_2	440.30	28.08	2.00	7200.00	439.45	0.19	1.00	0.42
C_3_3	449.16	32.15	2.00	7200.00	439.45	2.16	1.00	0.49
C_4_1	439.45	0.00	1.00	530.36	439.45	0.00	1.00	0.23
C_4_2	475.39	28.19	2.00	7200.00	439.45	7.56	1.00	0.51
C_4_3	505.33	32.30	2.00	7200.00	439.46	13.03	1.00	0.56
R_1_1	956.66	0.00	2.00	122.79	956.66	0.00	2.00	0.57
R_1_2	912.68	0.00	2.00	1319.63	912.68	0.00	2.00	0.60
R_1_3	812.97	0.00	2.00	797.14	812.97	0.00	2.00	0.56
R_2_1	956.66	0.00	2.00	766.26	956.66	0.00	2.00	0.67
R_2_2	921.17	7.18	3.00	7200.00	912.68	0.92	2.00	0.71
R_2_3	812.97	0.00	2.00	6507.53	812.97	0.00	2.00	0.57
R_3_1	964.61	0.28	3.00	7200.00	964.61	0.00	3.00	0.82
R_3_2	896.06	4.19	2.00	7200.00	896.06	0.00	2.00	0.71
R_3_3	890.27	4.01	2.00	7200.00	890.27	0.00	2.00	0.71
R_4_1	973.72	0.00	2.00	25.01	980.11	−0.66	2.00	0.74
R_4_2	981.64	0.00	1.00	3549.01	981.64	0.00	1.00	0.43
R_4_3	919.09	0.00	2.00	7036.77	919.09	0.00	2.00	0.73
RC_1_1	928.31	6.57	1.00	7200.00	928.31	0.00	1.00	0.40
RC_1_2	928.31	37.44	1.00	7200.00	928.31	0.00	1.00	0.34
RC_1_3	773.66	30.98	2.00	7200.00	773.66	0.00	2.00	0.47
RC_2_1	928.31	23.40	1.00	7200.00	928.31	0.00	1.00	0.48
RC_2_2	928.31	21.32	2.00	7200.00	928.31	0.00	1.00	0.34
RC_2_3	773.66	34.10	2.00	7200.00	773.66	0.00	2.00	0.46
RC_3_1	920.74	25.82	2.00	7200.00	920.74	0.00	2.00	0.50
RC_3_2	928.31	39.12	1.00	7200.00	928.31	0.00	1.00	0.37
RC_3_3	842.32	33.62	2.00	7200.00	842.32	0.00	2.00	0.52
RC_4_1	928.31	0.00	1.00	1429.17	928.31	0.00	1.00	0.43
RC_4_2	928.31	17.15	1.00	7200.00	928.31	0.00	1.00	0.33
RC_4_3	953.07	26.58	2.00	7200.00	920.74	3.39	2.00	0.48
Average	753.14	16.29	1.75	5813.44	749.06	0.74	1.58	0.48

6 Conclusion

This paper presents an iterated local search (ILS) algorithm for solving the workforce scheduling and routing problem (WSRP). We have examined different combinations of neighbourhood structures, and results show that the strategy of applying the intra-route search as a post-optimization procedure for the inter-route search provides effective and efficient performance. The proposed ILS is evaluated against a mixed integer programming (MIP) model and an adaptive larger neighbourhood search (ALNS) algorithm (Kovacs et al. 2012) on benchmark instances with up to 100 tasks. Computational experiments indicate that the proposed algorithm can produce solutions that are within an average gap of 1% to the optimal values in at most 40 s on average for all instances tested here. Compared to other heuristic approaches (Kovacs et al. 2012; Castillo-Salazar et al. 2015) for the similar problems, the proposed ILS has a relatively simple structure and a small number of parameters.

The proposed ILS algorithm is also applied to solve a set of Skill VRP instances, and results show that our algorithm is able to find optimal or near-optimal solutions in less than 0.5 s on average for all instances tested. Although the proposed algorithm is designed for solving the workforce scheduling and routing problem, it can be easily adapted to tackle other types of scheduling and routing problems.

Acknowledgements This research was partially supported by the AA (Automobile Association), United Kingdom. This support is gratefully acknowledged. We thank the three anonymous referees for their valuable comments, which helped to improve the quality and clarity of the paper.

References

Akjiratikarl, C., Yenradee, P., Drake, P.R.: PSO-based algorithm for home care worker scheduling in the UK. Comput. Ind. Eng. **53**(4), 559–583 (2007)

Bertels, S., Fahle, T.: A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. Comput. Oper. Res. **33**(10), 2866–2890 (2006)

Binart, S., Dejax, P., Gendreau, M., Semet, F.: A 2-stage method for a field service routing problem with stochastic travel and service times. Comput. Oper. Res. **65**, 64–75 (2016)

Blais, M., Lapierre, S.D., Laporte, G.: Solving a home-care districting problem in an urban setting. J. Oper. Res. Soc. **54**(11), 1141–1147 (2003)

Burke, E.K., Curtois, T., Hyde, M., Kendall, G., Ochoa, G., Petrovic, S., Vázquez-Rodríguez, J.A., Gendreau, M.: Iterated local search vs. hyper-heuristics: towards general-purpose search algorithms. In: IEEE Congress on Evolutionary Computation, Barcelona, Spain, pp. 1–8 (2010)

Cappanera, P., Gouveia, L., Scutellà, M.G.: The skill vehicle routing problem. In: Pahl, J., Reiners, T., Voß, S. (eds.) Network Optimization. Lecture Notes in Computer Science, vol. 6701, pp. 354–364. Springer, Berlin, Heidelberg (2011)

Castillo-Salazar, J.A., Landa-Silva, D., Qu, R.: A survey on workforce scheduling and routing problems. In: Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling, Son, Norway, pp. 283–302 (2012)

Castillo-Salazar, J.A., Landa-Silva, D., Qu, R.: A greedy heuristic for workforce scheduling and routing with time-dependent activities constraints. In: Proceedings of the 4th International Conference on Operations Research and Enterprise Systems, Lisbon, Portugal, pp. 367–375 (2015)

Chen, P., Huang, H.K., Dong, X.Y.: Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem. Expert Syst. Appl. **37**(2), 1620–1627 (2010)
Chen, X., Thomas, B.W., Hewitt, M.: The technician routing problem with experience-based service times. Omega **61**, 49–61 (2015)
Chiarandini, M., Stützle, T.: An application of iterated local search to graph coloring problem. In: Johnson, D.S., Mehrotra, A., Trick, M. (eds.) Proceedings of the Computational Symposium on Graph Coloring and its Generalizations, Ithaca, NY, pp. 112–125 (2002)
Cordeau, J.F., Laporte, G.: A tabu search algorithm for the site dependent vehicle routing problem with time windows. Inf. Syst. Oper. Res. **39**(3), 292–298 (2001)
Cordeau, J.F., Gendreau, M., Laporte, G.: A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks **30**(2), 105–119 (1997)
Cordeau, J.F., Laporte, G., Mercier, A.: A unified tabu search heuristic for vehicle routing problems with time windows. J. Oper. Res. Soc. **52**(8), 928–936 (2001)
Cordeau, J.F., Laporte, G., Mercier, A.: Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. J. Oper. Res. Soc. **55**(5), 542–546 (2004)
Cordeau, J.F., Laporte, G., Pasin, F., Ropke, S.: Scheduling technicians and tasks in a telecommunications company. J. Sched. **13**(4), 393–409 (2010)
Croes, G.A.: A method for solving traveling-salesman problems. Oper. Res. **6**(6), 791–812 (1958)
Dohn, A., Kolind, E., Clausen, J.: The manpower allocation problem with time windows and job-teaming constraints: a branch-and-price approach. Comput. Oper. Res. **36**(4), 1145–1157 (2009)
Dongarra, J.J.: Performance of various computers using standard linear equations software. Technical report CS-89-85, Electrical Engineering and Computer Science Department, University of Tennessee (2014)
Glover, F., Hao, J.K.: The case for strategic oscillation. Ann. Oper. Res. **183**(1), 163–173 (2011)
Hashimoto, H., Yagiura, M., Ibaraki, T.: An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. Discrete Optim. **5**(2), 434–456 (2008)
Ibaraki, T., Imahori, S., Nonobe, K., Sobue, K., Uno, T., Yagiura, M.: An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. Discrete Appl. Math. **156**(11), 2050–2069 (2008)
Kindervater, G.A.P., Savelsbergh, M.W.P.: Vehicle routing: handling edge exchanges. In: Aarts, E.H., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 337–360. Wiley, Chichester (1997)
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. **15**(5), 579–600 (2012)
Lourenço, H., Martin, O., Stützle, T.: Iterated local search. In: Glover, F., Kochenberger, G.A. (eds.) Handbook of Metaheuristics, International Series in Operations Research & Management Science, pp. 320–353. Springer, Boston (2003)
Lourenço, H.R.: Job-shop scheduling: computational study of local search and large-step optimization methods. Eur. J. Oper. Res. **83**(2), 347–364 (1995)
Michallet, J., Prins, C., Amodeo, L., Yalaoui, F., Vitry, G.: Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services. Comput. Oper. Res. **41**, 196–207 (2014)
Nagata, Y., Bräysy, O., Dullaert, W.: A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. Comput. Oper. Res. **37**(4), 724–737 (2010)
Penna, P.H.V., Subramanian, A., Ochi, L.S.: An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. J. Heuristics **19**(2), 201–232 (2013)
Pillac, V., Guéret, C., Medaglia, A.L.: On the dynamic technician routing and scheduling problem. In: Proceedings of the 5th International Workshop on Freight Transportation and Logistics (ODYSSEUS 2012), Mykonos, Greece (2012)
Pillac, V., Guéret, C., Medaglia, A.L.: A parallel matheuristic for the technician routing and scheduling problem. Optim. Lett. **7**(7), 1525–1535 (2013)
Schwarze, S., Voß, S.: Improved load balancing and resource utilization for the skill vehicle routing problem. Optim. Lett. **7**(8), 1805–1823 (2012)
Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper. Res. **35**(2), 254–265 (1987)
Vidal, T., Crainic, T.G., Gendreau, M., Lahrichi, N., Rei, W.: A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems. Oper. Res. **60**(3), 611–624 (2012)

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Comput. Oper. Res. **40**(1), 475–489 (2013)

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: Time-window relaxations in vehicle routing heuristics. J. Heuristics **21**(3), 329–358 (2015)

Walker, J.D., Ochoa, G., Gendreau, M., Burke, E.K.: Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework. In: Hamadi, Y., Schoenauer, M. (eds.) Learning and Intelligent Optimization, pp. 265–276. Springer, Berlin, Heidelberg (2012)

Weintraub, A., Aboud, J., Fernandez, C., Laporte, G., Ramirez, E.: An emergency vehicle dispatching system for an electric utility in Chile. J. Oper. Res. Soc. **50**(7), 690–696 (1999)

Xu, J., Chiu, S.Y.: Effective heuristic procedures for a field technician scheduling problem. J. Heuristics **7**(5), 495–509 (2001)