Applied Intelligence
https://doi.org/10.1007/s10489-018-1250-y

A hyper-heuristic with two guidance indicators for bi-objective mixed-shift vehicle routing problem with time windows

Binhui Chen[1] · **Rong Qu**[1] · **Ruibin Bai**[2] · **Wasakorn Laesanklang**[3,4]

Abstract

In this paper, a Mixed-Shift Vehicle Routing Problem is proposed based on a real-life container transportation problem. In a long planning horizon of multiple shifts, transport tasks are completed satisfying the time constraints. Due to the different travel distances and time of tasks, there are two types of shifts (*long shift* and *short shift*) in this problem. The unit driver cost for *long shifts* is higher than that of *short shifts*. A mathematical model of this Mixed-Shift Vehicle Routing Problem with Time Windows (MS-VRPTW) is established in this paper, with two objectives of minimizing the total driver payment and the total travel distance. Due to the large scale and nonlinear constraints, the exact search showed is not suitable to MS-VRPTW. An initial solution construction heuristic (EBIH) and a selective perturbation Hyper-Heuristic (GIHH) are thus developed. In GIHH, five heuristics with different extents of perturbation at the low level are adaptively selected by a high level selection scheme with the Hill Climbing acceptance criterion. Two guidance indicators are devised at the high level to adaptively adjust the selection of the low level heuristics for this bi-objective problem. The two indicators estimate the objective value improvement and the improvement direction over the Pareto Front, respectively. To evaluate the generality of the proposed algorithms, a set of benchmark instances with various features is extracted from real-life historical datasets. The experiment results show that GIHH significantly improves the quality of the final Pareto Solution Set, outperforming the state-of-the-art algorithms for similar problems. Its application on VRPTW also obtains promising results.

Keywords Hyper-heuristic · Mixed-shift vehicle routing problem with time windows · Bi-objective · Container transportation

1 Introduction

The early research of the Vehicle Routing Problem (VRP) can be traced back to [1], shown as an essential issue with tremendous effect to the economy and society. In the classical Vehicle Routing Problem with Time Windows (VRPTW) [2], at the beginning of a planning horizon, a fleet of identical vehicles leave a center depot to visit/service a sequence of customers with certain demands, composing a number of so-called *routes*. Every customer is visited exactly once, satisfying the constraints (time window) specified by the customers. The sum of customer demands on each route cannot exceed the capacity of a vehicle, and all vehicles have to return the depot before the end of the planning horizon. The most common objectives in VRPTW are minimization of the number of vehicles used and minimization of the total travel distance.

1.1 Vehicle routing problem variants

Based on the VRPTW model, a large number of classic VRP variants have been proposed with diverse *side constraints* from practical scenarios. In this section, only the most relevant variants to our study are reviewed. In Vehicle Routing Problem with Pickups and Deliveries (VRPPD) [3], a service demand consists of picking up shipments from a

✉ Binhui Chen
Binhui.Chen@nottingham.ac.uk

Rong Qu
Rong.Qu@Nottingham.ac.uk

Ruibin Bai
Ruibin.Bai@nottingham.edu.cn

Wasakorn Laesanklang
Wasakorn.lae@mahidol.ac.th

1 University of Nottingham, Nottingham, UK

2 University of Nottingham Ningbo China, Ningbo, China

3 Mahidol University, Bangkok, Thailand

4 Center of Excellence in Mathematics, CHE, Bangkok, Thailand

Published online: 09 August 2018

customer and the associated delivery to another customer. Especially, if the depot is the only one pickup point and all the customers are delivery destinations, or in another case, all the customers are pickup points while only the depot is the delivery location, the problem is called a *One-to-Many-to-One* problem. If the customers are pickup points as well as delivery points, the problem is *Many-to-Many*. Last but not least, it is a *One-to-One* problem when the pickup demand of a customer is the delivery demand of another specific customer [4].

Furthermore, if the shipments can be consolidated, the problem would be classified as *Less-than Truckload Transportation*; otherwise, it is a *Full Truckload Transportation (FTT)* problem [5]. Container transportation problem is a specific variant of *FTT*, where one truck can carry only one demand item (container). Zhang et al. [6] model the container transportation problem with a node-based network, which is commonly used in VRPTW. The model integrates all activities of completing the transportation of a container into a so-called *load node*. This method has been widely used in the VRPPD with high loading and unloading time [7, 8].

In some cases of VRP, the scheduling horizon is very long, e.g. in soft drink industry, grocery distribution and waste collection. Their scheduling is usually performed over multiple periods/shifts, and the associated problems are categorized as Multi-Period Vehicle Routing Problem (MPVRP) [9]. Especially, when there is a specific service frequency to each customer over the scheduling horizon, the problem is called a Periodic Vehicle Routing Problem (PVRP) [10]. In this case, each customer may be visited more than once. The solution of PVRP is a combination of service shifts of customers, instead of the scheduled routes of one single period.

Apart from the two objectives in VRPTW mentioned above, there are various other objectives widely used in VRPs, e.g. minimizing the travel time, the waiting time, and other operational cost, maximizing the balance of workload, and so on [11]. With the increasing concern to the environment in recent years, the carbon emission and petrol consumption have also been considered in the VRP community, leading to the Pollution-Routing Problem and Green Vehicle Routing Problem [12]. From the cost perspective, labor cost (driver salary) usually is the dominated component in the overall cost [13]. This is one of the reasons why minimizing the number of vehicles used is a primary objective in VRPs, as fewer vehicles require fewer drivers being hired. In addition, making use of fewer vehicles generally implies a lower fuel consumption and a higher utilization rate of the vehicle capacity. When more than one objective are considered in a VRP, it is called a Multi-Objective Vehicle Routing Problem (MOVRP).

1.2 Existing methods

After decades of study in VRP, both exact and approximate methods have been extensively investigated. Exact methods explore the solution space of a problem exclusively to find the optimal solution. However, a critical issue of such methods is the unrealistic computational time needed searching the enormous size of the solution space in real-world problems. On the other hand, approximate methods (or heuristics) do not guarantee the optimality of solutions produced, but generate a good approximation of the optimal solution in an acceptable computation time [14]. Metaheuristics and Hyper-Heuristics methods guide the search with various strategies, showing powerful performance in solving diverse large scale and complex VRPs [15].

Population-based metaheuristics, such as Evolutionary Algorithms, Scatter Search, and Ant Colony Optimization Algorithms, evolve a population of solutions [14]. Using population improves the diversification of searches, these type of methods show powerful exploration ability while achieve high quality solutions in multi-objective and highly constrained problems. However, larger population is hard to operate and may greatly affect algorithm performance. For example, in Genetic Algorithm, which is a widely used population-based metaheuristic in VRPs, it is hard to use crossover to partition the periods and routes in the solution representation (e.g. genotype/chromosome) for MPVRP. Besides, in large size problems, the long chromosome and the associated large solution population is hard to manage as well. Population-based metaheuristics are not suitable to large scale problems with complex structures and constraints such as the MPVRP considered in this paper.

Differently, in each iteration of *single solution-based* metaheuristics, only one solution is updated by employing neighbourhood operators at each move during the search. In different algorithms, such as Tabu search [16], Simulated Annealing [17], and Variable Neighbourhood Search [18], different strategies are used in the Acceptance Criterion and Neighbourhood Operator Selection.

Metaheuristic algorithms are often designed to address specific problems by striking a balance between the diversity and intensity of the search for the specific problems. In the literature, a large number of problem specific and knowledge intensive metaheuristics have been developed for VRPs [19, 20]. Differently, *Hyper-Heuristics* is a type of high level algorithms which aim to develop generic approaches beyond the problem specific metaheuristics [21, 22]. Hyper-Heuristics work at a higher level to generate or select a set of Low-Level Heuristics

(LLH) in a common framework, while the LLH execute the operations on problem solutions. Hyper-Heuristics focus on designing the high level framework, called High-Level Heuristic (HLH), instead of searching the specific solutions for the problem confronted. In a well-designed Hyper-Heuristics algorithm, its HLH would adaptively adjust the LLH used, creating proper algorithms for various searching scenarios for the given instances.

Hyper-heuristics approaches can be categorized to two classes: *Heuristic Selection* and *Heuristic Generation* [23]. *Heuristic Selection* methodologies choose existing heuristics from the LLH pool to tackle the problem given, while the methodologies of *Heuristic Generation* generate new heuristics using existing heuristics as the components. What's more, each above class can be further divided into two subcategories, namely *Construction Heuristic* and *Perturbation Heuristic*, according to the constructive or perturbative low level heuristics used. *Construction Heuristics* construct solutions using the given LLH, while *Perturbation Heuristics* produce new solutions by perturbing existing solutions. More details can be found in [24, 25].

As a classic combinatorial optimization problem, VRP is an essential application of hyper-heuristics. Garrido and Riff [26] propose an evolutionary hyper-heuristic for Dynamic Vehicle Routing Problem (DVRP). Each genotype in this evolutionary algorithm consists of a constructive heuristic, an improvement heuristic and an ordering heuristic. This generative construction hyper-heuristic adapts well to the dynamic scenario in DVRP. Both hyper-heuristics of [27] and [28] obtain competitive results in Capacitated Vehicle Routing Problem (CVRP). The former generates LLH by searching the space of heuristic component (i.e. neighbourhood structure, neighbourhood combination, local search configuration and acceptance criterion), while the latter adjusts the order of LLH to perturb the current solution, incorporating an adaptive ordering scheme in an Iterated Local Search framework. In [29], besides the selection of LLH, a Gene Expression Programming framework is also proposed to automatically generate the acceptance criterion for different problem instances. The proposed method shows promising results in DVRP and CVRP.

Vidal et al. [30] propose an unified hybrid genetic search framework (UHGS), which replaces the mutation with a unified local search (ULS). In ULS, the route-evaluation operators vary according to the change of problem *attributes*, aiming to provide a general-purpose solver for diverse VRP variants. UHGS produces results better than or close to the state-of-the-art results on benchmarks. However, the experiment results show that its computation time increases significantly in MPVRPs again due to the period and route partition problem as explained above on genetic algorithms. The long computation time impedes its application to large scale MPVRP.

Benefiting from decades of intensive research in VRP, a large number of excellent heuristics have been developed, providing sufficient LLH for designing high performance hyper-heuristics. Potvin and Rousseau [31] and Taillard et al. [32] propose the *2-opt** and *CROSS-exchange* heuristics, respectively, which show excellent performance in routing problems with time windows. However, when facing large-scale problems with complex structure, they often converge prematurely due to their relatively small change (low perturbation) to a solution in each iteration, thus the search is often stuck to local optimum.

Shaw [33, 34] proposes the Large Neighbourhood Search (LNS) heuristic which removes a number of nodes (e.g. demands/customers) from the current solution and then reinserts them to generate an updated new solution (*Destroy & Repair*). This heuristic brings greater changes (higher perturbation) to escape from local optimum and avoid premature convergence. It obtains the best results in several VRP variants, although a larger computation time is required in each iteration [35]. A similar strategy called *Ruin & Recreate* is proposed in [36].

Nagata and Bräysy [37] propose the Guided Ejection Search (GES) heuristic combining the ideas of LNS and Ejection Pool methods [38]. In each iteration of GES, one route is removed and then the nodes of the removed route are reinserted into the destroyed solution. Any infeasible partial solutions are accepted with penalties. GES outperforms the existing heuristics on minimizing the number of routes, but longer computation time for each iteration is needed. For more details, see [39, 40].

Much research on MOVRP have been done as well. In some of them, a set of *non-dominated solutions* based on Pareto Dominance [41] are generated, providing the decision maker a pool of candidate solutions as a reference (*Pareto Methods*). In the literature, the *Pareto Methods* are mainly used in Evolutionary Algorithms [42–45]. Differently, in the other research, one single optimal solution is pursued. In this case, either the problem objectives are projected into one single objective and the problem is solved as a single-objective problem (*Scalar Techniques*), or different priorities are assigned to objectives which are considered separately (*Non-Scalar and Non-Pareto Algorithms*). More methodologies for MOVRP can be found in [46].

In real-life, the vehicle scheduling of different types of shifts are usually considered separately as independent problems. In this paper, a real-world Mixed-Shift Vehicle Routing Problem with Time windows (MS-VRPTW) is studied. A construction heuristic and a selection perturbation hyper-heuristic, which combine the scheduling work of two types of shifts, are proposed for the MS-VRPTW. The

proposed algorithms integrate the independent resource for the two types of shifts, aiming to increase the utility of vehicles and reduce the scheduling stress for logistic companies. The algorithms are tested on a set of benchmark instances with different features.

The rest of this paper is organized as follows: Section 2 introduces the problem background and presents the mathematical problem model. Section 3 introduces the proposed solution methods. The benchmark instances and computation experiments are presented in Section 4. Section 5 shows the conclusions of this paper.

2 Problem definition & mathematical model

2.1 Problem description

The problem studied is a container transportation problem faced by a logistic company at Ningbo Port, which is the second largest port in China. Every day, the company has to transship a number of commodities, each consists of a number of containers. Every commodity has a specific service time constraint. These commodities are transited among 19 container terminals including harbors and dry ports (see Fig. 1). There is a fleet of 250 trucks, whose depot locates at the Ningbo coast. Every day, the trucks leave the depot with a list of transport tasks and return to the depot after completing all the tasks.

The management of transportation involves three levels of planning, namely: strategic planning, tactical planning, and operational planning [5]. Strategic level management focuses on the decisions of the locations of facilities (e.g. the locations of depots and fleets), while the key tactical issues are terminal operation specification, service selection and other mixed decision making. Strategic planning and tactical planning are the preconditions to transportation problems, and they are long-term and medium-term planning. The operational planning focuses on the Vehicle Routing and Scheduling Problem, which is the major issue of the Ningbo Port problem.

As one truck in the Ningbo Port can carry only one container at a time, one container represents one transport task. Completing a transport task consists of loading the container to the truck at the source terminal, transporting the container from the source to the destination terminal and then unloading the container over there. The well-known Planning Domain Description Language (PDDL) is a complex descriptive system providing a standard and flexible formalism for various AI planning domains including the VRPs [48]. It is supported by state-of-the-art planning methodologies, producing high quality solutions in various planning problems. However, those methods have not shown to perform effectively or efficiently in solving large size real-life problems [49]. To simplify the problem model and make the prevailing neighbourhood search heuristics applicable, the node-based method of [7],

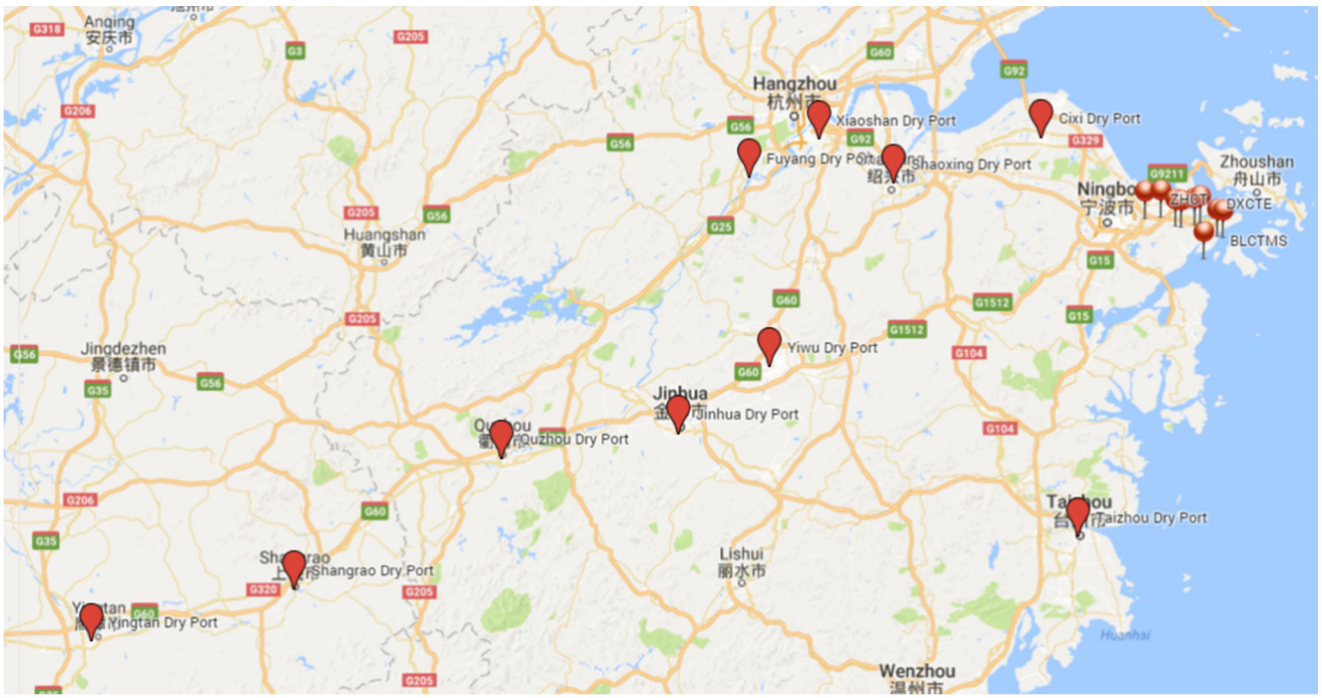

Fig. 1 The locations of 19 container terminals of the logistic company (screenshot taken from Google Maps [47]). The balloon icons represent dry ports and the ball stick icons represent harbors. The nine harbors are located along the coast of Ningbo City, while the 10 dry ports are either inland or far from the Ningbo coast

instead of the PDDL, is employed for formulating the problem of this paper. A *task node* integrates the three activities to represent the service of a transport task. The service time of a task is the total time of the three activities.

From Fig. 1, we can find that the tasks associated with the dry ports are *long-distance* tasks (LDT), while those transportation between harbors are *short-distance* tasks (SDT). In the Ningbo Port, the service time of a SDT is less than seven hours, and all the harbors can be reached in less than 2 hours from the depot. On the other hand, because the service time of LDT and the travel time between the dry port and the depot is quite long, the average time of completing a LDT is longer than 13 hours. In some studies, the exact path between two points is also considered, i.e. the problem of Path Planning [50]. Since the paths among the terminals and the depot are fixed by the company in our problem, the drivers cannot change the fixed path when completing a task or going to the next task. Vehicle routing considering path planning presents an interesting and different integrated problem, thus is in the scope of our future research.

The Ningbo Port company sets up two types of working shifts: *short shift* and *long shift*. A *short shift* is 12 hours, meaning a day is divided into two short shifts (day shift and night shift). In the day shift, drivers drive trucks away from the depot, and drivers of the night shift return the trucks to the depot after completing their tasks. The two drivers using the same truck (called *one-driver truck*) have a shift-change in the middle of a day at a terminal. Shift-change cannot happen within a task node, so the shift-change terminal is either the last destination terminal of the day shift or the first source terminal of the night shift. Differently, a *long shift* is 24 hours. In this case, two drivers are assigned to one single truck (*double-driver truck*) at the same time. With this arrangement, the two drivers can drive the truck in turn, satisfying the associated regulations on continuous working hours in Labor Law.

The two types of shifts are associated with two different driver salary schemes, which lead to different overall operational cost to the company. In a working day, two drivers are required for one truck of either type. The difference between the two types of trucks is that the two drivers of a *one-driver truck* route work separately within their own *short shifts*, while both drivers of a *double-driver truck* route have to stay in the truck during the whole *long shift*. Correspondingly, the unit payment to the drivers of *double-driver trucks* is higher for their longer shift length. SDT can be completed in a *short shift* using *one-driver trucks*, while LDT must be completed with *double-driver trucks* in *long shifts* due to the long service time. When optimizing the assignment of LDT and SDT, considering both types of trucks simultaneously can reduce the overall number of trucks used, consequently minimizing the overall total operational cost of driver payment.

The truck scheduling for both types of shifts are combined in this study. Currently, the company handles LDT and SDT with two separate scheduling systems, resulting to inefficient use of trucks and lots of task lateness in busy seasons. This low efficiency schedule is mainly caused by the two separate scheduling systems which do not share the limited truck resource. In our study, the two scheduling systems are integrated to increase the efficiency of the scheduling and the utility of trucks. *Artificial task* which represents the driver shift-change between two *short shifts* is thus proposed. The routes of a truck in two consecutive *short shifts* are thus converted to one route in a *long shift*. To the best of our knowledge, this is the first time the Mixed-Shift Vehicle Routing Problem with Time Windows (MS-VRPTW) is proposed in the literature. In the Ningbo Port, the trucks in the fleet are identical and can be appointed to be either *one-driver* or *double-driver* according to the commodity situation.

An example schedule of a working day (with one *long shift* or two *short shifts*) is presented in Fig. 2 to illustrate our proposed model. There are in total eight routes, three for *one-driver* trucks and five for *double-driver* trucks. We can see that, LDT (represented by rectangles) only appear in *double-driver* truck routes, while SDT (solid circles) exist in both *one-driver* truck routes and *double-driver* truck routes. The hollow circles in the top three routes are *artificial tasks*.

The fourth route in Fig. 2 explains why the LDT require *double-driver* trucks. Considering the travel time leaving and returning to the depot, completing a LDT takes more than 12 hours (maximum length of a *short shift*). In addition, if the distance between two LDT is small, more than one LDT might be serviced in one *double-driver* route. For instance, in the last route, as the destination of the first LDT is the source of the second LDT, the travel distance and time between the two tasks is zero. In this case, the two LDTs can be completed by one *double-driver* truck, leading to a more efficient use of vehicles.

Another special case of LDT is the rectangle in the seventh route. It represents a type of task which require short service time but can only be finished in *double-driver* routes. Because their time windows are narrow (i.e. 3 hours in this example) and across the middle of a working day, the shift-change between *short shifts* cannot be done when completing these type of tasks. Therefore, these type of tasks can only be assigned to *double-driver* trucks.

In different real-life scenarios, the shift lengths, the number of task types and the number of shift types might be different from that of the Ningbo Port problem. However, the method of using artificial tasks is still applicable, which integrates the scheduling and routing with different shift settings into one model. Therefore, the model of MS-VRPTW can cover various practical cases from real scenarios.

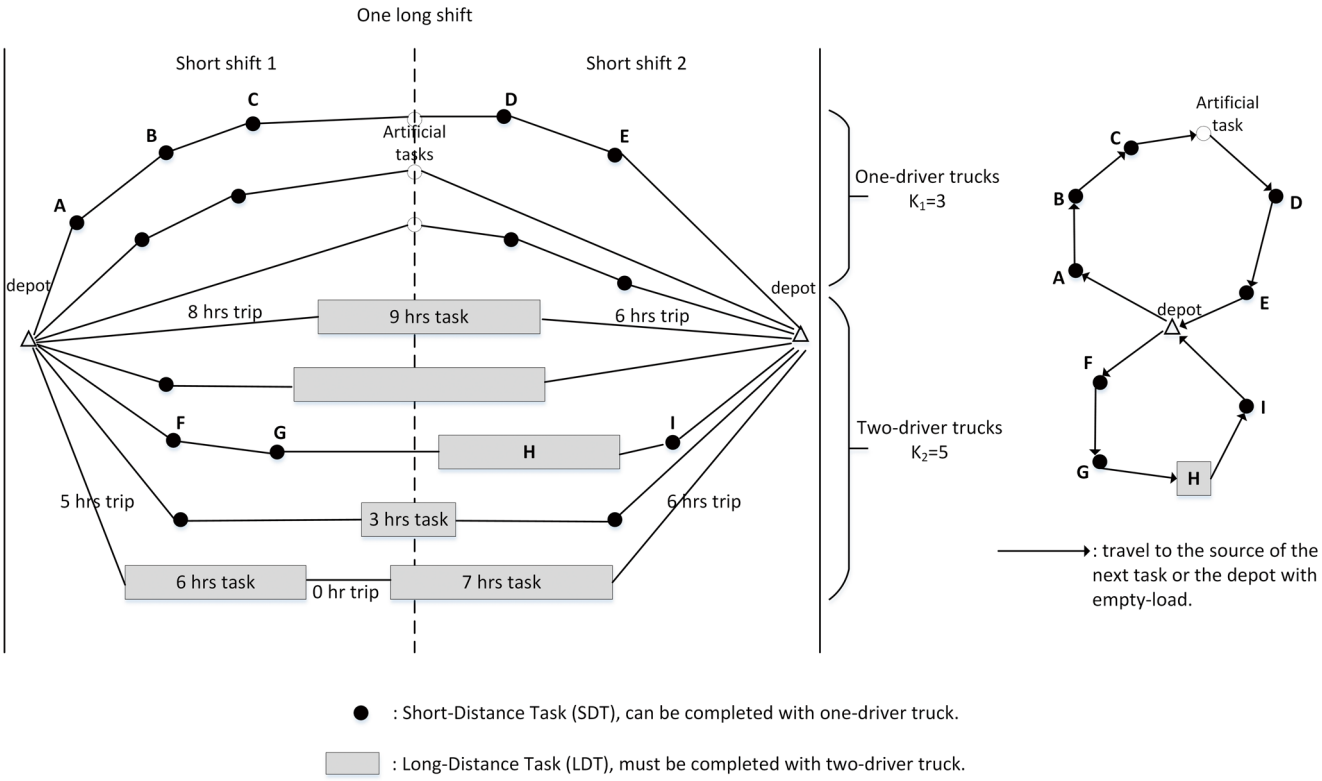

Fig. 2 A schedule example of one *long shift* (two *short shifts*). Three *one-driver* trucks and five *double-driver* trucks are used in this schedule. The right subgraph presents the first and sixth routes in the schedule

2.2 Mathematical model

To model the MS-VRPTW, a number of notations are defined, see Table 1.

The MS-OPVRPTW can be formally defined as follows.

Objective:

$$Minimize \qquad DP = \sum_{s\in S}\left(P_o \cdot K_O^s + P_d \cdot K_D^s\right) \tag{1}$$

$$Minimize \qquad TD = \sum_{p\in P}\sum_{s\in S}\sum_{i\in N\cup W}\sum_{j\in N\cup W} c_{ij}\cdot x_{ij}^{sp} \tag{2}$$

Subject to:

$$\sum_{p\in P}\sum_{s\in S}\sum_{i\in W\cup N\setminus\{0\}} x_{ij}^{sp} = 1, \qquad \forall j \in N\setminus\{0\} \tag{3}$$

$$\sum_{p\in P}\sum_{s\in S}\sum_{j\in W\cup N\setminus\{0\}} x_{ij}^{sp} = 1, \qquad \forall i \in N\setminus\{0\} \tag{4}$$

$$\sum_{i\in W\cup N\setminus\{0\}} x_{ij}^{sp} = \sum_{f\in W\cup N\setminus\{0\}} x_{jf}^{sp}, \qquad \forall j \in W\cup N\setminus\{0\},\ s\in S, p\in P \tag{5}$$

$$K_O^s + K_D^s \le K, \qquad \forall s \in S \tag{6}$$

$$\sum_{j\in N\cup W} x_{0j}^{sp} = \begin{cases} K_O^s & \forall s\in S, p = O \\ K_D^s & \forall s\in S, p = D \end{cases} \tag{7}$$

$$\sum_{i\in N\cup W} x_{i0}^{sp} = \begin{cases} K_O^s & \forall s\in S, p = O \\ K_D^s & \forall s\in S, p = D \end{cases} \tag{8}$$

$$\sum_{p\in P}\sum_{j\in N} x_{wj}^{sp} = K_O^s \qquad \forall w\in W, s\in S \tag{9}$$

$$\sum_{p\in P}\sum_{i\in N} x_{iw}^{sp} = K_O^s \qquad \forall w\in W, s\in S \tag{10}$$

$$\sum_{j\in N} x_{wj}^{sO} = K_O^s \qquad \forall w\in W, s\in S \tag{11}$$

$$\sum_{i\in N} x_{iw}^{sO} = K_O^s \qquad \forall w\in W, s\in S \tag{12}$$

$$x_{wv}^{sp} = 0, \qquad \forall w, v\in W, s\in S, p\in P \tag{13}$$

Table 1 Definition of Notations

Input Parameters:	
S	The set of time-continuous working shifts. Here one shift is 24 hours (*long shift*).
$P = \{O, D\}$	The set of truck types. O represents that the truck used is a *one-driver* truck, while D means it is a *double-driver* truck.
P_o, P_d	The operating cost of using a *one-driver* truck (P_o) and that of a *double-driver* truck (P_d). They are mainly determined by the payments to the drivers.
K	The fleet size, which is the number of available trucks.
$[Y_s, Z_s]$	Time window of shift $s \in S$.
$N = \{0, 1, 2, ..., n\}$	Set of $n + 1$ nodes. Each node represents a task except node 0, which is the depot.
$[a_i, b_i]$	The time window for node $i \in N$. The time window for the depot is zero at the boundary of a shift. If a truck arrives at the source of i early, it has to wait until a_i.
A	Set of arcs. Each arc(i, j) represents node j being immediately serviced/visited after node i.
c_{ij}	The cost of traveling from node i to node j. If both nodes are tasks, it is the travel distance from the destination of i to the source of j. Otherwise, it is the distance from the depot to the first source or from the last destination to the depot. These travels are empty-load with no container carried.
t_{ij}	The travel time from node i to node j. When both nodes are tasks, t_{ij} is the travel time from the destination of i to the source of j. Otherwise, it is the travel time from the depot to the first source or from the last destination to the depot.
l_i	The time for servicing node i, which includes the loading time, transportation time (from pick-up source to delivery destination) and unloading time. The service time of the depot is zero.
W	Artificial task set. Artificial tasks ($w \in W$) can only be found in *one-driver* routes, representing the shift-change (e.g. the hollow circles in Fig. 2). Artificial task's service time (l_w) and loaded travel distance are zero (i.e. its source and destination are the same terminal). The source and destination of an artificial task must be either the last destination of the day shift or the first source of the night shift in that route. The time window of w is the mid-line of workday, i.e. $[a_w, b_w]$ = [8pm,8pm].
Variables:	
T_i	The time of arrival at node i.
B_i	The time to begin the service of node i.
x_{ij}^{sp}	A binary decision variable for nodes i, $j \in N \cup W$, $s \in S$, $p \in P$. Its value is 1 when arc(i, j) is included in the solution in shift s by a truck type p, otherwise its value is 0.
$K_O^s \in \{0, 1, ..., K\}$	An integer variable of the number of *one-driver* trucks used in shift $s \in S$.
$K_D^s \in \{0, 1, ..., K\}$	An integer variable of the number of *double-driver* trucks used in shift $s \in S$.

$$T_j = \sum_{p \in P} \sum_{s \in S} ((B_i + l_i + t_{ij}) \cdot x_{ij}^{sp} + (Y_s + t_{0j}) \cdot x_{0j}^{sp}), \quad \forall i \in N \backslash \{0\} \tag{14}$$

$$B_j = T_j + max\{a_j - T_j, 0\}, \quad \forall j \in N \backslash \{0\} \tag{15}$$

$$x_{ij}^{s0} \cdot (B_i + l_i + t_{i0}) \leq x_{i0}^{sp} \cdot Z_s, \quad \forall i \in N \cup W, \; s \in S, \; p \in P \tag{16}$$

$$a_i \leq B_i \leq b_i - l_i, \quad \forall i \in N \backslash \{0\} \tag{17}$$

$$x_{ij}^{sp} \in \{0, 1\} \quad \forall i, j \in N \cup W, s \in S, p \in P \tag{18}$$

$$K_O^s \in \{0, 1, ..., K\} \quad \forall s \in S \tag{19}$$

$$K_D^s \in \{0, 1, ..., K\} \quad \forall s \in S \tag{20}$$

MS-VRPTW is a bi-objective problem. The first objective is minimizing the total driver payment (*DP*), see Eq. 1, which depends on the number and types of trucks used. It is notable that, the cost of a driver for *double-driver* truck is 1.5 times of a driver of *one-driver* truck in our study (i.e. $P_o = 1$, $P_d = 1.5$). Minimizing the total travel distance (*TD*) (2) is the other objective. Actually, the target of *TD* is to minimize the empty-load travel distance as the total loaded travel distance in an instance is fixed. *DP* focuses on the operational cost, and *TD* concentrates on the utility of trucks which actually pursues a higher heavy-loaded travel distance rate in total travel distance.

Constraints (3) and (4) denote that every task node can be visited exactly once and all the tasks must be visited. Constraint (5) specifies that a task may only be serviced after the previous task is completed. Constraints (3)–(5) together make sure arcs over more than one shift are unacceptable. Constraint (6) guarantees the number of trucks used is not larger than the fleet size.

Constraints (7) and (8) place the limits on *one-driver* truck ($p = O$) and *double-driver* truck ($p = D$). Constraints (9)–(12) guarantee that there must be K_O^s artificial tasks completed on the routes of *one-driver* trucks, while there is no artificial task on the routes of *double-driver* trucks. In addition, constraint (13) guarantees each route of *one-driver* truck has only one artificial task.

Equation (14) defines the arrival time at a task node. Equation (15) defines the beginning time of servicing a task node. This time is calculated by the arrival time plus the waiting time at the source of a task. Equations (14) and (15) enforce the correct successive relationship between consecutive nodes. Constraints (14)–(16) together define the time windows of shifts. Constraint (17) represents the time constraint on each task. The domains of the decision variables are presented in (18)–(20).

From this mixed integer programming (MIP) model, we can find that the MS-VRPTW is a large-scale and tightly constrained non-linear problem. In MS-VRPTW, the size of solution space is decided by the number of tasks (n), the number of shifts ($|S|$) and the size of the fleet (K). Since there are $|S| \cdot K$ possible routes in a solution, which are either *one-driver* or *double-driver*, and each route has $n!$ permutations of tasks, the size of the search space is $2^{|S| \cdot K} \cdot n!$. In real-life, a logistic company may face hundreds to thousands of containers to be transited, leading to a highly complex problem with huge solution space.

3 Solution methodologies for bi-objective mixed-shift vehicle routing problem with time windows

3.1 Exact search

In our study, exact search method is first implemented to address MS-VRPTW using a successful and widely used optimization solver, CPLEX. To address this bi-objective MIP problem with CPLEX, the objectives of the mathematical model has to be slightly modified since CPLEX is not a tool for multi-objective models. To this end, three different configurations are employed to linearly combine the two objectives into one (called *decomposition* in some research, see formula (21)). The configurations represent three scenarios in the modified objective: 1) *DP* has the same weight as *TD*, 2) *DP* dominates *TD* and 3) *TD* dominates *DP*. Considering the different ranges of *DP* and *TD*, the three configurations are $\{a = 200, b = 1\}$, $\{a = 10000, b = 1\}$ and $\{a = 4, b = 1\}$, respectively.

$$Minimize \qquad a \cdot DP + b \cdot TD \tag{21}$$

The CPLEX script of exact search has been run on a high performance computer system. Considering the scale of this problem, a large number of computation resources have been assigned, which were 16 cores (2.6 GHz), 100 GB memory and 24 hours runtime limit for each experiment instance. However, the output of CPLEX shows that even with large amounts of computation resources, it is still very hard to obtain satisfying solutions for MS-VRPTW with exact search methods. CPLEX was out of memory within 10 minutes in all the three configurations. This observation indicates that exact search is not realistic for solving this large-scale tightly constrained nonlinear problem due to massive computation resources required for computation time and memory. It is no doubt that there may exist exact methods which can work better than CPLEX in this problem, however, the requirement of extensive computation resource still remains. Therefore our studies focus on developing efficient approximate approaches for MS-VRPTW.

3.2 Initial solution construction heuristic

Solomon [2] develops four classic construction heuristics for VRPTW, among which the Insertion Heuristic in general shows the best performance. Given a set of candidates to be assigned (e.g. customers, demands), in each iteration, a candidate is inserted to an insertion position in the existing routes using *Insertion Selection Schemes*. During the construction, if all existing routes are full, a new empty route will be created. The Insertion Selection Schemes used in existing routes and the newly created empty routes can be different. These steps are repeated until all candidates are assigned, obtaining a complete solution.

Insertion Heuristic is widely applied to diverse VRP variants using various Insertion Selection Schemes. Chen et al. [51] propose an emergency-based construction heuristic for the Open Periodic Vehicle Routing Problem with Time Windows. In that heuristic, tasks with higher emergency are dealt with a higher priority. Based on the emergency-based construction heuristic, we propose an Emergency Level-Based Insertion Construction Heuristic (EBIH) for MS-VRPTW.

In EBIH, all the tasks are classified into LDT or SDT following the definitions given in Section 2.1. Then they are further categorized according to their emergency levels. When a task i can be completed in shift s according to its time window, the task is either *optional* or *mandatory*. To be precise, if i can be completed in s and later shift(s), i

is an *optional* task in shift s; otherwise, i is a *mandatory* task to s. So, to each shift, four sets of available tasks would be assigned, which are *mandatory LDT*, *optional LDT*, *mandatory SDT*, and *optional SDT*.

The four sets of tasks are considered in order in EBIH. It is easy to understand that we should assign *mandatory* tasks first. Because the delay of tasks may cause the containers missing the vessel appointed and greatly increase the operational cost of the company. Besides, SDT can be completed with both *one-driver* trucks and *double-driver* trucks while LDT can only use *double-driver* trucks, which means SDT have more insertion options than LDT when constructing a solution. Therefore, LDT is relatively harder to assign than SDT and should be assigned earlier.

In practice, logistic companies usually complete tasks as early as possible to avoid leaving many tasks to the following shifts and increasing later scheduling pressure. In real-life, extra commodities might be added in real time. Reducing the remainder tasks and leaving more available trucks for later shifts can also enhance the stability of the scheduling system. In EBIH, after arranging all mandatory tasks, if there still are available trucks in the fleet, optional tasks will be inserted to the current shift until all trucks are ran out. The order of task sets being assigned shift by shift is: *mandatory LDT* $\rightarrow$ *mandatory SDT* $\rightarrow$ *optional LDT* $\rightarrow$ *optional SDT*.

Faced with a set of tasks to be inserted and a large number of potential insertion positions, the Insertion Selection Scheme used determines the performance of an Insertion Heuristic. The scheme of *Greedy Strategy* always executes the insertion bringing the least cost increase among all candidate insertions. The routes constructed with this scheme are relatively *tighter*. Less trucks would be employed with this strategy, but requiring more computation time to evaluate all possible candidates. Differently, *First Feasible Strategy* adopts the first feasible insertion to a task given. It takes less evaluation time but more trucks would be used in the solution generated.

When choosing the Insertion Selection Schemes, a trade-off between efficiency and effectiveness should be made. The key issue in the scheduling is that all tasks must be completed with the limited trucks. Thus, in EBIH, *Greedy Strategy* is adopted for mandatory tasks. This setting aims to guarantee the urgent tasks' assignment first. On the other hand, to avoid long computation time, *First Feasible Strategy* is applied to the insertion of optional tasks. In addition, because the tasks with long service time are often too *big* to be inserted into the routes with existing tasks, the task with the longest service time will be selected as the first task in the newly created new route.

The performance of EBIH is tested on instances with diverse sizes and features. The test results are presented in Section 4.2.1.

3.3 A selective perturbation hyper-heuristic with two guidance indicators

To further reduce the operational cost of the company, based on the initial solution generated by EBIH, an improvement Hyper-Heuristic with Two Guidance Indicators (GIHH) is developed. GIHH is a Selection Perturbation Hyper-Heuristic, which selects perturbative low level heuristics (LLH) adaptively based on the changes of a problem scenario. Two guidance indicators are proposed to guide the selection of LLH. Considering the large scale and complex multi-level solution structure in MS-VRPTW, only one solution is updated in each algorithm iteration (single solution-based).

3.3.1 High-level heuristic

Algorithm 1 The GIHH framework

Input: An initial feasible solution (S) produced by EBIH, a set of LLH (H), Stopping Criterion ($NONIMP$).
Step 1. **Set up the initial parameters and *ARCH*.**
$Weight \leftarrow \{1, ..., 1\}$
$ScoreA \leftarrow \{0, ..., 0\}$
$ScoreB \leftarrow \{0, ..., 0\}$
$ARCH \leftarrow S$
Step 2.
while Stopping Criterion is not met **do**
Step 2.1: **Solution Selection**
Randomly select a solution from $ARCH$ as the current solution S_c.
Step 2.2: **Low Level Heuristic Selection and Execution.**
Select a LLH (h_i) from H according to $weight_i$ with *Roulette Wheel Rule*, see Eq. 22; Execute h_i on the current solution, obtaining a new solution: $S' \leftarrow h_i(S_c)$
Step 2.3: **Accept or Reject (Hill Climbing).**
if S' is non-dominated in $ARCH$ **then**
Add S' into $ARCH$ and remove all dominated solutions.
else
Reject S'.
end if
Update $ScoreA_i$ and $ScoreB_i$, recording the contribution of h_i to the solution improvement. More details are presented in Section 3.3.2.
Step 2.4: **Weight Adjustment.**
After a predefined number (SEG) of iterations, $Weight$ of H is updated according to $ScoreA$ and $ScoreB$, see Section 3.3.2.
end while
Output: A solution set $ARCH$.

$$Pr_i = \frac{weight_i}{\sum_{j \in H} weight_j} \qquad \forall i \in H \qquad (22)$$

Algorithm 1 introduces the high level framework of GIHH. The input contains an initial feasible solution, a set of given LLH (H, introduced in Section 3.3.3) and the stopping criterion. To this bi-objective problem, GIHH is a *Pareto Method* whose output is a solution archive (*ARCH*) consisting of *non-dominated* solutions. The small range of *DP* reduces the diversity of *DP*, leading to a relatively small number of non-dominated solutions. Thus, no limit is set to the size of *ARCH*, which means all non-dominated solutions found will be stored. In addition, to increase the diversification of the search, different solutions with the same objective values are stored in *ARCH*.

In each iteration, one LLH is chosen and applied to a chosen solution (S_c), generating an updated solution. During the loop, to diversify the search, S_c is randomly selected from $ARCH$ in Step 2.1. The stopping criterion is set as when *ARCH* is not being updated in a predefined number (*NONIMP*) of iterations.

In GIHH, three scalars ($Weight$, $ScoreA$ and $ScoreB$) are defined to guide the selection of LLH, generating better problem solutions. The LLH executed in an iteration is chosen with the *Roulette Wheel Rule* (Step 2.2). To avoid the probabilities of LLH converging to zero and the corresponding LLH never being called at all, a minimal probability limit of 5% is applied to every LLH. *ScoreA* and *ScoreB* are two guidance indicators, which record the performance of LLH in previous search history from two different aspects respectively. *Weight* is updated based on *ScoreA* and *ScoreB*. All these three scalars are adjusted adaptively during search (in Steps 2.3 and 2.4), details in Section 3.3.2.

Because the ranges of the two objectives in MS-VRPTW are significantly different, that is, the range of *DP* is remarkably smaller than that of *TD*, a small change on *DP* is usually accompanied by a great fluctuation on *TD* in a solution. To further investigate this issue, in addition to the Hill Climbing acceptance criteria, a *Record-to-Record Travel* (RRT) [52] acceptance criterion is also implemented in our study. RRT accepts the worst solutions (S') of deteriorated quality from the current solution (S_c) in a predefined range. The comparison of experiment results are presented in Section 4.2.3.

3.3.2 Guidance indicators and weight adjustment scheme

$ScoreA_i$ stores the accumulated rewards to h_i according to the change of objective values from S_c to S', recording the performance of h_i on improving solution quality. In each iteration, if S' is acceptable, reward 1 is added to $ScoreA_i$, otherwise no reward is added. Therefore, a larger $ScoreA_i$ represents a greater contribution of h_i to generating new non-dominated solutions. This indicator emphasizes LLH's contribution on solution quality improvement.

$ScoreB_i$ is a specially designed indicator for this bi-objective problem, which indicates which objective h_i inclines to improve (improvement direction). In MS-VRPTW, a Pareto Solution Set with uniform distribution and good convergence on the Pareto Front is expected, instead of the solutions within local regions. During the search, the improvement on both of the two objectives is pursued. When updating $ScoreB_i$, the objective values of S_c and S' are compared. If S' is better than S_c on *DP*, $ScoreB_i$ is increased by one; If S' is better than S_c on *TD*, $ScoreB_i$ is decreased by one. A positive $ScoreB_i$, thus, means the inclination of improving *DP* (generated more improved solutions on *DP*) to h_i, while a negative one indicates that of improving *TD*.

$Weight_i$ is updated once in every *SEG* iterations (called a *Segment*) to avoid over-fitting. It is adjusted according to the feedback from the search history ($ScoreA_i$ and $ScoreB_i$). The update is a two-phase procedure. The first phase is guided by $ScoreA_i$, see Eq. 23.

$$weight_i^t = \alpha \cdot weight_i^{t-1} + \beta \cdot \frac{ScoreA_i}{Applied\ Times\ of\ h_i} \qquad (23)$$

In the second update phase, to find the improvement *DEVIATION* by Eq. (24) between the two objectives, the newly generated non-dominated solutions are compared with the first S_c in the last Segment, obtaining the number of the non-dominated solutions with improved *DP* (*DP_IMP*) and that of improved *TD* (*TD_IMP*). If *DP* was improved more times in the last Segment ($DEVIATION > 0$), then the $weight_i$ of those LLH with *TD* inclination should be increased by using Eq. 25, obtaining a higher probability being selected in the current Segment. The similar operations are made when $DEVIATION < 0$. This procedure aims to balance the improvement direction.

$$DEVIATION = \frac{DP\_IMP - TD\_IMP}{(DP\_IMP + TD\_IMP) \cdot 0.5} \qquad (24)$$

$$weight_i^t \mathrel{+}= \gamma \cdot \frac{ScoreB_i}{Applied\ Times\ of\ h_i} \cdot DEVIATION \quad when\ (DEVIATION < 0\ And\ ScoreB_i < 0)\ Or\ (DEVIATION > 0\ And\ ScoreB_i > 0) \qquad (25)$$

The three coefficients (α, β, γ) in Eqs. 23 and 25 determine the response speed to the search feedback and the influence of each guidance component on updating $weight_i$, subject to $\alpha + \beta + \gamma = 1$.

3.3.3 Low-level heuristics

Five LLH are adopted in GIHH. Each LLH changes the current solution to a certain extent, obtaining updated solutions. Heuristics with large changes perturb the operated solution dramatically. They increase the search diversity and avoid trapping to local optimum, but longer computation time is needed usually to produce a new feasible solution. Heuristics with small changes use relatively less computation time in each iteration, however, their common deficits are easily to stuck to local optimum and premature search. Previous research shows that properly combining heuristics can improve the performance of search [53].

- ***Inter-Route 2-opt****. Lin [54] proposes the λ-*opt* route improvement heuristic which removes and reconnects λ edges in a route. This classic heuristic brings relatively small changes to a solution, obtaining good results in various VRPs. Potvin and Rousseau [31] develop an improved *2-opt* heuristic (*2-Opt**) which keeps the direction of each route segment during reconnection. This heuristic is devised for *Traveling Salesman Problem* at first, but shows excellent performance in various routing problems with time windows. In GIHH, *Inter-Route 2-Opt** removes two edges from different routes and reconnects them while keeping the directions of associated route segments. Notice that the edges modified can be the starting or ending points of routes, which means two routes being connected into one route is possible.
- ***Inter-Route CROSS-exchange***. Taillard et al. [32] propose *CROSS-exchange* which swaps two route segments from two different routes while keeping their directions. This heuristic brings relatively small perturbation as well. The length of a route segment can be zero, e.g. when one of the two operated route segments is empty, the execution of *Inter-Route CROSS-exchange* actually relocates a route segment from one route to another route.
- ***Intra-Route CROSS-exchange***. In this heuristic, the swapping strategy of *CROSS-exchange* is applied to one single route.
- ***Large Neighbourhood Search (LNS)***. In GIHH, Random Selection is used in the *Destroy* heuristic of LNS to remove q randomly chosen tasks. Then the removed tasks are reinserted into the destroyed solution using a greedy *Repair* heuristic. This heuristic always executes the insertion causing the least increase on the travel distance. Obviously, comparing all possible insertion positions for all the q tasks is time-consuming. To balance the solution quality and the computation time, the value of q is defined as $min\{5\% \cdot n, 10\}$, where n is the total number of tasks.
- ***Guided Ejection Search (GES)***. To further reduce the number of trucks used and optimize *DP*, GES is employed in GIHH. The main ideas of GES have been summarized in Section 1.2. Using LNS and GES obtains larger change to solutions and greater perturbation in search, at the cost of longer execution time.

4 Experiments & analysis

4.1 Benchmark dateset

To evaluate the proposed algorithms in different scenarios, a benchmark of 24 instances with various features are generated (available at http://www.cs.nott.ac.uk/~pszrq/benchmarks.htm). The instances are extracted from the company's historical dataset. In these instances, each item represents a commodity, which consists of its commodity ID, source terminal, destination terminal, available time to transport, deadline of completing the tasks, and the number of containers in this commodity. Notice that the number of containers in a commodity can be larger than one, meaning finishing one commodity transportation may need to complete multiple transport tasks.

A categorization scheme similar to [55] is adopted to define the features of the instances. Firstly, to a LDT, if its time window is smaller than 20 hours, it will be classified as an *emergent* task. The time window for SDT is smaller than 10 hours. These two values are suggested by the port company's coordinator. In addition, index B (26) is used to measure the total throughput balance at terminals in each instance.

$$B = \frac{1}{|V|} \sum_{i \in V} |I_i - O_i| \tag{26}$$

Here, V is the set of terminals, is composed of the harbors and dry ports. I_i and O_i respectively represent the number of incoming and outgoing tasks at terminal i. A smaller B represents a more balanced throughput in the instance. Based on these, four types of features are used to create the benchmark instances.

- ***Tight*** instance: 70%-80% tasks in the instance are emergent.
- ***Loose*** instance: less than 30% tasks in the instance are emergent.
- ***Balanced*** instance: the value of B in the instance is smaller than 30.
- ***Unbalanced*** instance: the value of B in the instance is larger than or equal to 30.

According to the time of receiving transhipment requests before their deadlines in practice, two types of scheduling

horizons (two and four days) are set for the instances. Based on this setting, we created in total eight combinations of features. They represent a comprehensive dataset of instances with various commodity emergencies and workload balance. For each combination, three instances are generated in sizes of small, medium and large, respectively. The details of instances are presented in Table 2. The last column provides the total *loaded travel* distances which are fixed in instances. These instances are generated based on the problem characteristics at Ningbo Port, e.g. the geographical distribution of the terminals and the lengths of shifts, and can be used as a set of benchmark instances with diverse features for testing the solution methods of other MS-VRPTWs.

4.2 Comparison experiments

4.2.1 Initial solutions

Table 3 presents the initial solutions produced by EBIH, obtained on a PC with i7-3820 3.60GHz CPU and 16.0 GB memory. Feasible solutions can be obtained within an acceptable time for all instances. The computation time of generating a solution grows rapidly along with the number of tasks in the instance. The highest requirement of truck happens on instance TU2-3, where 71 *one-driver* trucks and 171 *double-driver* trucks are used.

4.2.2 Parameter setting and complexity discussion

GIHH adaptively employs LLH according to the search, with relatively few parameters to set. The parameters are tuned one by one, while the others are fixed.

In Eqs. 23 and 25, a large α means a low response speed to the change in the search space, often leading to slow convergence. However, high-quality solutions may be skipped over when the response speed is too high. On the other hand, high response speed usually leads to premature convergence. Our preliminary experiments show that, the setting of $\alpha = 0.5$ makes a good trade-off between convergence speed and solution quality. The values of β and γ determine the influence of the two guidance indicators to update $weight_i$. The setting of $\beta = 0.4$, $\gamma = 0.1$ is adopted based on preliminary experiments, indicating that *ScoreA* has a greater influence than *ScoreB* in GIHH.

Table 2 Features of the benchmark instances

Instance	Configuration	No. of Shifts	No. of Commodities	No. of Tasks	Loaded TD
LB2-1	Loose Balanced	2	50	145	27,474
LB2-2	Loose Balanced	2	100	566	122,878
LB2-3	Loose Balanced	2	200	697	179,802
LU2-1	Loose Unbalanced	2	50	390	78,891
LU2-2	Loose Unbalanced	2	100	551	132,220.5
LU2-3	Loose Unbalanced	2	200	768	196,460
TB2-1	Tight Balanced	2	50	245	44,674
TB2-2	Tight Balanced	2	100	446	98,062.5
TB2-3	Tight Balanced	2	200	779	163,255
TU2-1	Tight Unbalanced	2	50	364	55,854
TU2-2	Tight Unbalanced	2	100	529	97,656.5
TU2-3	Tight Unbalanced	2	200	895	190,897.5
LB4-1	Loose Balanced	4	50	156	39,471
LB4-2	Loose Balanced	4	100	578	121,575.5
LB4-3	Loose Balanced	4	200	976	175,464
LU4-1	Loose Unbalanced	4	50	395	97,047
LU4-2	Loose Unbalanced	4	100	670	150,680.5
LU4-3	Loose Unbalanced	4	200	1077	283,463
TB4-1	Tight Balanced	4	50	321	69,536
TB4-2	Tight Balanced	4	100	536	118,923
TB4-3	Tight Balanced	4	200	914	185,164.5
TU4-1	Tight Unbalanced	4	50	389	92,008
TU4-2	Tight Unbalanced	4	100	606	127,203
TU4-3	Tight Unbalanced	4	200	886	185,556.5

The shifts adopted are *long shifts*

Table 3 Initial solutions produced by EBIH

	LB2-1	LB2-2	LB2-3	LU2-1	LU2-2	LU2-3
DP	83.5	393	456.5	221	341.5	520.5
TD	22955.5	144926	140842.5	69096	93044	174560.5
time (s)	6	369	869	125	464	1176
	TB2-1	TB2-2	TB2-3	TU2-1	TU2-2	TU2-3
DP	147.5	315.5	484.5	252.5	374.5	554.5
TD	46032	98749	153719.5	71517.5	109143.5	145165
time (s)	29	215	1017	87	284	1407
	LB4-1	LB4-2	LB4-3	LU4-1	LU4-2	LU4-3
DP	131.5	382	547.5	294.5	475.5	671.5
TD	47028	113011	169075.5	101569	149.47	234230.5
time (s)	5	220	841	130	374	1711
	TB4-1	TB4-2	TB4-3	TU4-1	TU4-2	TU4-3
DP	289	411	579	349.5	533	569
TD	76690	125146.5	150932	105797.5	144385	188010.5
time (s)	52	229	1063	92	198	799

When updating $weight_i$, a smaller *SEG* would change $weight_i$ more frequently, when *SEG* is too large, the feedbacks cannot change in time. *SEG* is set to 80 in GIHH empirically. In addition, $NONIMP = 150$ is used as the stopping criterion to strike a balance between the computation time and the quality of results.

When assessing the computational complexity of meta-heuristics and hyper-heuristics, time complexity cannot be determined since these approximate algorithms do not guarantee finding the global optimal solution within a given time limit. Whether or not the algorithm procedure would terminate depends on the applied problem and specific definition of its stopping criterion (e.g. the definition of $NONIMP$ in GIHH). Therefore, the CPU time and objective function evaluations on benchmark are often used to compare the computational complexity of approximate methods in research. In this study, the algorithms with the above parameter setting are compared from the aspects of computational time and iterations at high level, and the results and associated analysis are presented in the next subsection. As only one solution is updated in each iteration, with the task node-based solution representation, the space complexity of GIHH is $O(K \cdot |S| \cdot n)$, where K is the fleet size, $|S|$ is the length of the planning horizon and n is the number of tasks to be assigned.

4.2.3 Comparison experiment results and analysis

Impacts of the guidance indicators To evaluate the influence of the two proposed guidance indicators in GIHH, two variants (GIHH-A and RHH) of GIHH with different guidance indicator settings are developed for comparison. In GIHH-A, only $ScoreA$ is adopted, while in RHH, LLH are randomly chosen without any guidance. Our preliminary experiments show that increasing the computation time does not improve the results significantly, so all the three algorithms use the same stopping criterion.

Table 4 presents the comparison of GIHH, GIHH-A and RHH. All the results are obtained in 20 runs. In the literature, to compare the performance of Pareto Methods, various quality indicators are proposed. Most of them focus on the comparison on the Pareto Set approximation [56]. One of the most widely used indicators is *Hyper-Volume*, which considers the convergence, uniformity and spread over the Pareto Front produced. Previous studies have shown that a Pareto Set with a larger hyper-volume is likely to have a better trade-off among multiple objectives [57]. To compare the three algorithm variants, the hyper-volumes of the *ARCH*s obtained are calculated and presented in Table 4. In our study, the *reference points* used in calculating hyper-volume are the initial feasible solutions generated by EBIH. It can be found that, comparing the three algorithms from multiple aspects, most of the best results are produced by GIHH.

Among the three variants, RHH produced the worst hyper-volumes with the most iterations, while its standard deviation obtained is the largest. This shows that, when the High-Level Heuristic is random selection with no guidance, the algorithm would take more iterations to converge with a lower stability. However, it may have a higher probability of finding better solutions against objective (2), i.e. with the best *DP*.

It can be found that from Table 4, GIHH-A and GIHH obtained remarkably better solutions (higher **HV**) than RHH.

Table 4 Results comparison among algorithms with different indicator configurations

		LB2-1	LB2-2	LB2-3	LU2-1	LU2-2	LU2-3	TB2-1	TB2-2
RHH	Iteration	1479	4831	5557	4096	5782	9012	2459	4919
	Ave HV	0.8176	1.0964	0.6745	0.6652	0.4826	0.8086	0.2598	0.4856
	Best HV	0.8966	1.1303	0.6936	0.6924	0.5472	0.8504	0.2941	0.5132
	Best DP	**53**	**247**	**336**	**166**	**256.5**	**369.5**	131	**224.5**
	Best TD	11685	53531.5	73506	34004	57693	76342.5	30498	68167.5
	S.D.	4.32%	2.63%	**1.03%**	2.14%	2.90%	2.20%	1.65%	1.98%
GIHH-A	Iteration	1322	4911	3701	3121	5441	9320	2973	4945
	Ave HV	0.8456	1.1277	0.6943	0.6842	**0.5299**	**0.8632**	0.2754	0.4950
	Best HV	0.8933	1.1424	**0.7201**	0.7093	**0.5575**	**0.8837**	**0.3044**	0.5259
	Best DP	53.5	250	339	166.5	259	371	**130.5**	226
	Best TD	11491.5	50565	68909	**32365**	**54509**	**71227**	**29811**	**66195**
	S.D.	3.08%	1.21%	1.70%	2.80%	**1.73%**	1.39%	1.66%	2.11%
GIHH	Iteration	1538	4586	5125	3486	4732	7958	2837	6516
	Ave HV	**0.8578**	**1.1321**	**0.6967**	**0.6914**	0.5134	0.8539	**0.2792**	**0.5036**
	Best HV	**0.9203**	**1.1624**	0.7182	**0.7234**	0.5516	0.8771	0.3000	**0.5275**
	Best DP	**53**	248.5	339	**166**	261	372	131	227.5
	Best TD	**11332**	**50382.5**	**67728**	32409	55549	71609.5	29961	66208.5
	S.D.	**3.00%**	**0.85%**	1.51%	**1.30%**	1.86%	**1.34%**	**1.31%**	1.42%
		TB2-3	TU2-1	TU2-2	TU2-3	LB4-1	LB4-2	LB4-3	LU4-1
RHH	Iteration	6763	2821	5170	7334	1648	4827	8474	2880
	Ave HV	0.5908	0.4493	0.6217	0.4574	0.6443	0.7181	0.8271	0.3262
	Best HV	0.6146	0.4710	0.6385	0.4888	**0.6823**	0.7507	0.8560	0.3574
	Best DP	**357.5**	183	268	**430.5**	**91**	**268**	374.5	237.5
	Best TD	88763.5	51263.5	64919	92812	27610.5	58122	80004.5	74178
	S.D.	2.02%	1.42%	1.60%	2.09%	2.08%	**1.66%**	**2.20%**	**1.34%**
GIHH-A	Iteration	6228	2800	4933	9713	1389	4199	6691	2257
	Ave HV	**0.6005**	**0.4649**	0.6463	**0.5357**	0.6429	0.7429	0.8360	0.3407
	Best HV	**0.6305**	0.4824	0.6531	**0.5396**	0.6665	0.7629	**0.8784**	0.3625
	Best DP	364.5	**181**	268	432.5	93	269	**374**	236
	Best TD	**84585**	50878	62694	**85413.5**	27465	57612	75555	73566
	S.D.	**1.66%**	1.29%	**0.52%**	**0.40%**	**1.57%**	1.87%	3.01%	1.70%
GIHH	Iteration	5360	2592	4975	8695	1371	3883	6009	3071
	Ave HV	0.5972	0.4622	**0.6480**	0.5176	**0.6517**	**0.7495**	**0.8362**	**0.3423**
	Best HV	0.6203	**0.4863**	**0.6700**	0.5333	0.6728	**0.7747**	0.8692	**0.3838**
	Best DP	364	**181**	**267**	431	92	**268**	380	**233.5**
	Best TD	85127	**50544**	**62691**	85813.5	**27409**	**57437**	**75285**	**73073**
	S.D.	1.67%	**1.21%**	1.10%	1.16%	1.72%	2.16%	2.25%	2.05%
		LU4-2	LU4-3	TB4-1	TB4-2	TB4-3	TU4-1	TU4-2	TU4-3
RHH	Iteration	5032	8915	1180	3807	7649	2349	3508	7349
	Ave HV	0.7547	0.9387	0.3778	0.3634	0.6021	0.2371	0.3648	0.6134
	Best HV	0.7824	0.9765	0.3885	0.3831	0.6357	0.2454	0.3800	0.6334
	Best DP	**321.5**	**476.5**	226	**323.5**	**409**	287	397	**405.5**
	Best TD	80535.5	75804	55991	90690.5	92126	88214	111090	112465
	S.D.	1.52%	2.57%	0.78%	1.06%	1.95%	0.50%	1.22%	1.71%
GIHH-A	Iteration	3029	6789	921	2817	5487	2229	2386	4919
	Ave HV	0.7659	0.9759	0.3770	0.3761	0.6247	**0.2577**	0.3743	0.6239
	Best HV	0.7962	0.9891	0.3909	0.3871	0.6643	**0.2620**	0.3888	0.6470
	Best DP	327.5	484.5	**225**	325.5	**409**	**284.5**	398	412
	Best TD	**76664**	71859	55920	89808.5	88744	86548	**110743**	**106921**

Table 4 (continued)

	S.D.	2.18%	**1.00%**	0.78%	**0.54%**	1.97%	**0.32%**	0.85%	1.80%
GIHH	Iteration	3701	5341	1131	3098	5745	2122	2871	5055
	Ave HV	**0.7774**	**0.9768**	**0.3865**	**0.3795**	**0.6343**	0.2554	**0.3756**	**0.6246**
	Best HV	**0.8050**	**0.9997**	**0.3974**	**0.3981**	**0.6674**	0.2613	**0.3938**	**0.6480**
	Best DP	322.5	480	**224.5**	**323.5**	412	285.5	**393.5**	407
	Best TD	**75138**	**70811.5**	**55606**	**89397**	**86690**	**86482**	110926	107892
	S.D.	**1.50%**	1.38%	**0.73%**	0.57%	1.82%	0.37%	**0.74%**	**1.34%**

Ave HV and **Best HV** are the average and best hyper-volumes, respectively. **Best DP** and **Best TD** are the best found objective values, while **S.D.** is standard deviations. **Iteration** is the average iterations in the 20 runs. Best results are in bold

Using $ScoreA$ significantly improves the quality of the produced solution set. Generally, GIHH-A and GIHH used less iterations but longer computation time to obtain the output. This can be observed in Fig. 3. GIHH-A and GIHH may have less average iterations than RHH (blue columns), but their computation time (red crosses) are longer on all the eight sample instances. Because the unit computation time of LNS and GES are significantly longer than the other LLH, this observation indicates that, compared to RHH, GIHH-A and GIHH employed these two LLH with greater perturbation more frequently during the search.

Between GIHH-A and GIHH, the latter obtained a higher average and the best hyper-volume on most instances with the guidance of $ScoreB$, while no obvious increase on iteration time and computation time is found. This can also be observed from Fig. 3. GIHH promotes the overall search performance and stability with the help of the two proposed guidance indicators.

With regard to the features of instances, *Loose* instances have broader time windows than *Tight* instances, which means more scheduling options and larger solution space. Thus, when the sizes of instances are similar, the *Loose* instances require more iterations and computation time to converge in all the three algorithms. In addition, comparing the iteration time, GIHH-A and GIHH work better on *Loose* instances, see Fig. 4. It can be found that, compared to RHH, the reduction of iterations is higher on *Loose* instances than on *Tight* instances, except GIHH on the LB4 instances. When the feature of throughput imbalance at terminals changes, no obvious difference is found.

Note that, in the *ARCH* generated by GIHH, each non-dominated point on the Pareto Front may have 20-40 different solutions on average. The number of different solutions with the same objectives stored does not affect the value of hyper-volume. Experiment results show that storing different solutions with the same objective values does not significantly increase the hyper-volume of a solution archive, but it boosts the diversification of the solution set. Those solutions provide the logistic company coordinator more reference solutions.

Impacts of solution selection and acceptance criterion In each iteration of GIHH, the solution to be operated (S_c) is randomly selected from *ARCH*, aiming to increase the diversity in search. To justify the function of the random selection scheme, an algorithm with deterministic selection of S_c (named GIHH-D) is also implemented in our research. With this deterministic scheme, in *ARCH*, the solution farthest from the reference point will be selected as S_c. Because all solutions are derived from the initial solution (reference point), this deterministic scheme means that the solution with the highest improvement on both objectives will be selected. T-test is conducted on the output of GIHH and GIHH-D. The results are presented in Table 5.

In addition, as mentioned in Section 3.3.1, another variant adopting the Record-to-Record Travel acceptance criterion (GIHH-RRT) is also compared with GIHH. In GIHH-RRT, comparing to S_c, a worse solution would be accepted as long as the deterioration of objective value is less than $0.01 \cdot TD(S_c)$ on *TD* and less than 1.5 on *DP*. Acceptance criterion in a perturbative algorithm should balance the diversification and intensification of search, while RRT can increase the diversification of search greatly. Its output is compared with that of *Hill Climbing* criterion presented in Table 5.

From Table 5, it can be found that GIHH outperforms the other two algorithms. On the one hand, using the deterministic scheme to select the solution to be updated (GIHH-D) decreases the diversity of search, leading to significantly worse output than GIHH on most instances (19/24). On the other hand, accepting worse solutions (GIHH-RRT) does not improve the final search result on all instances. As the two objectives have remarkably different ranges, accepting worse solutions would bring great fluctuation and deterioration to S_c in the search. This observation indicates that, in MOVRP, when the difference

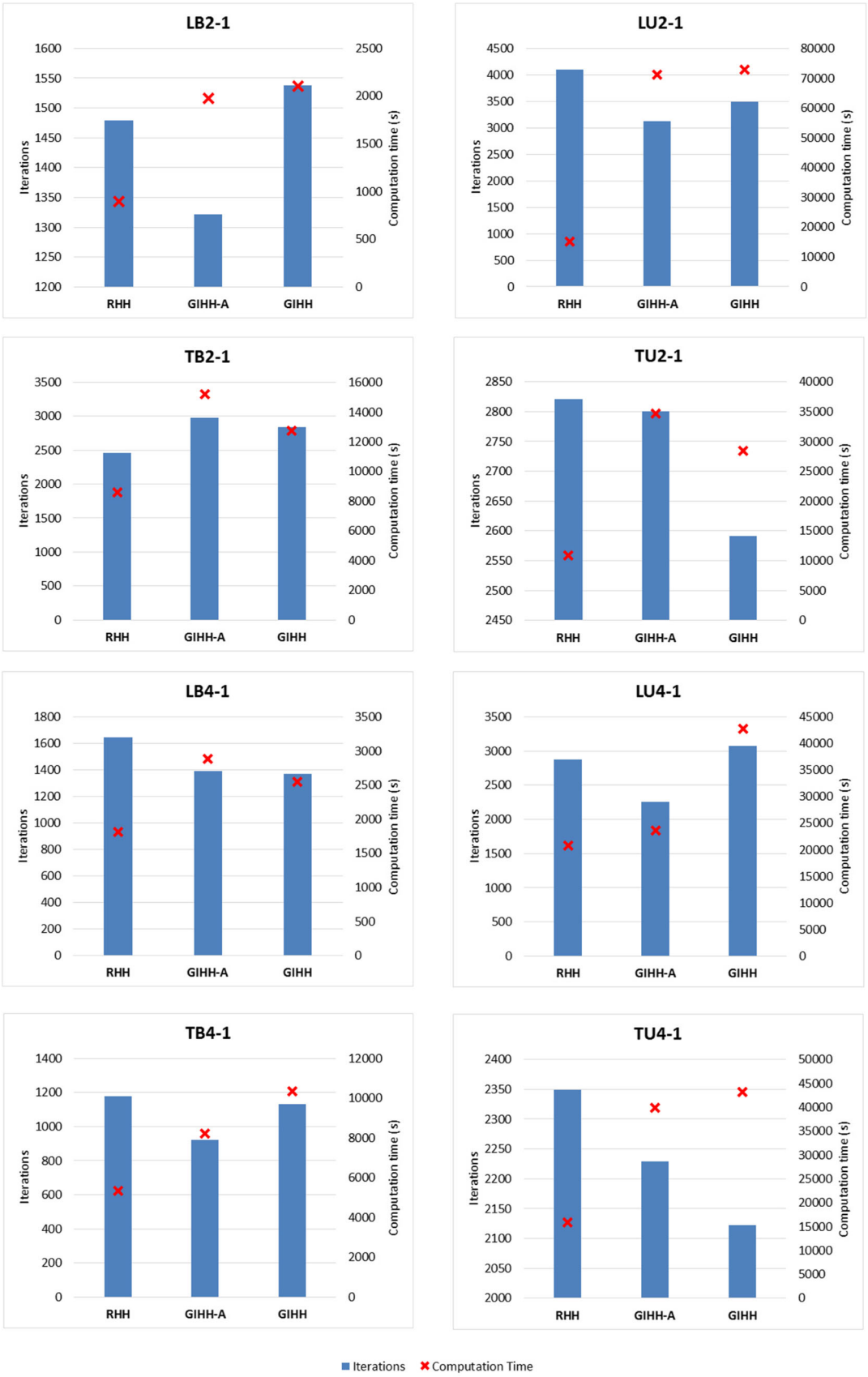

Fig. 3 The iteration times and computation time of the three algorithms on eight sample instances

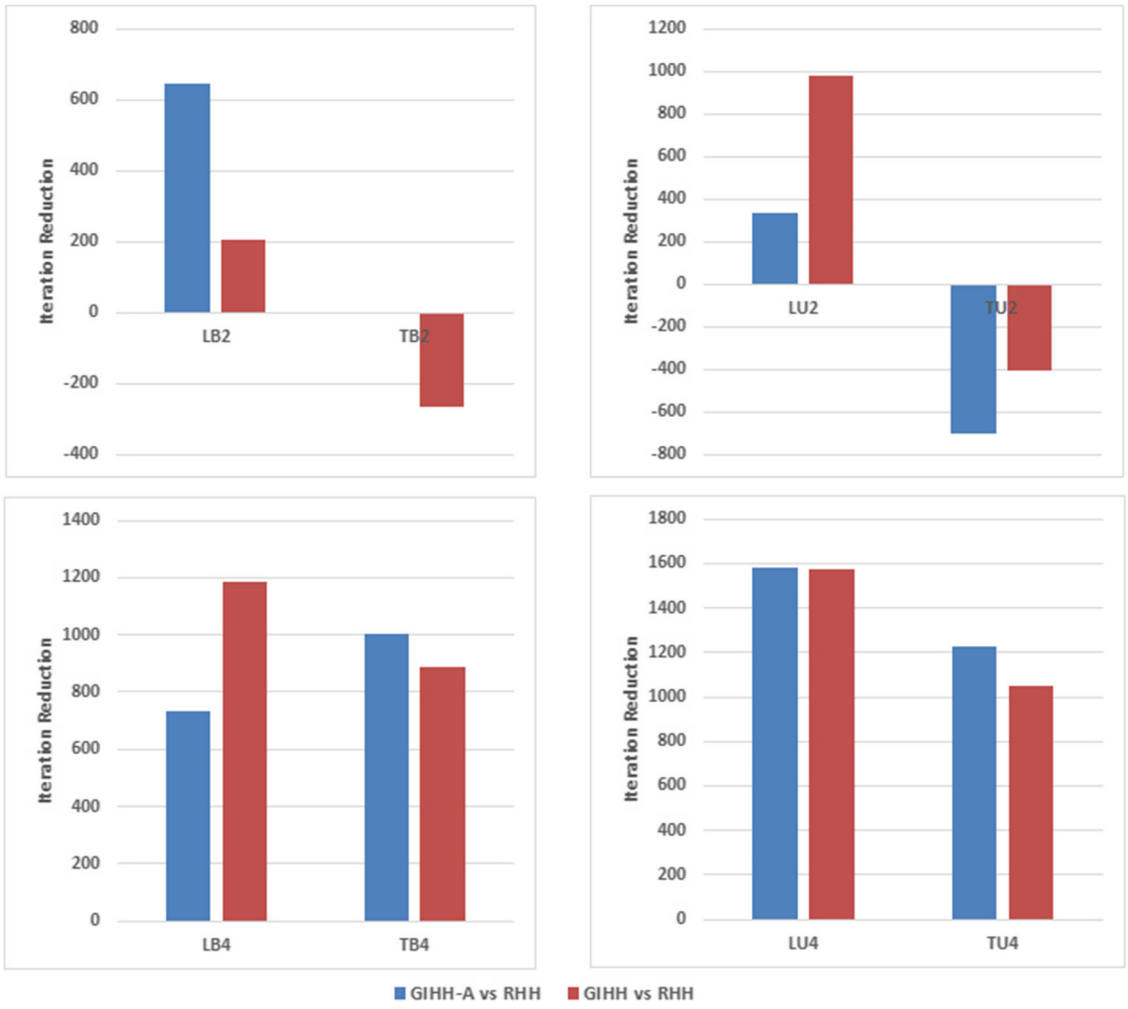

Fig. 4 Comparison of the iteration time reduction. The bars indicate the number of reduction of iterations. Longer bars represent greater reduction, while the negative values indicate more iterations than RHH

Table 5 T-test results with GIHH. Y means GIHH generates significantly better solutions, while N represents it does not

	LB2-1	LB2-2	LB2-3	LU2-1	LU2-2	LU2-3	TB2-1	TB2-2	TB2-3	TU2-1	TU2-2	TU2-3
GIHH-D vs GIHH	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y
GIHH-RRT vs GIHH	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	LB4-1	LB4-2	LB4-3	LU4-1	LU4-2	LU4-3	TB4-1	TB4-2	TB4-3	TU4-1	TU4-2	TU4-3
GIHH-D vs GIHH	N	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y
GIHH-RRT vs GIHH	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

in the ranges of objectives is big, accepting solutions of lower quality does not improve the search. Besides, our experiments also show that GIHH is more stable than the other two algorithms with smaller standard deviations.

Comparison with the state-of-the-art algorithms MS-VRPTW is a newly introduced model in the literature, there is thus no existing algorithm applied to it yet. Three state-of-the-art algorithms (RVNS [58], FVNS [59] and ALNS [60, 61]) are adopted and applied to MS-VRPTW in our study. Both RVNS and FVNS use the Variable Neighbourhood Search framework and produce the best solutions in PVRP. Apart from the neighbourhood structures used, a main difference between them is that the order of shaking operators employed is fixed in FVNS, while they are randomly selected in RVNS. ALNS produces the best results for VRPPD with Adaptive Large Neighbourhood Search. The experiments show that GIHH outperforms the three algorithms on both solution quality and computation time in MS-VPRTW, especially on larger instances. Their result deterioration is presented in Table 6.

Possible causes for these results include the following. Firstly, the neighbourhood structure employed in GIHH are highly effective. FVNS and RVNS only use the small perturbation neighbourhood operators (e.g. λ-opt, CROSS, relocation). With these smaller neighborhood structures, it is hard or needs a long time to escape from the local optimum in this nonlinear constrained problem. On average, 65% more computation time is required by FVNS and RVNS comparing to GIHH. Large perturbation operators are used in ALNS but are lacking of intensive exploitation. Secondly, without the guidance of specific indicators, e.g. $ScoreB$, the solutions generated are more likely to cluster, leading to a low hyper-volume. In addition, the three algorithms compared are problem specific metaheuristics. Different from hyper-heuristics, their performance may decline drastically for different instances even in the same problem. For example, both FVNS and ALNS obtain better results than GIHH on LU2 instances.

An observation from the results of FVNS and RVNS is that, they both produce many more solutions with the same objective values than GIHH. The small perturbation operators tend to generate a large number of solutions with small differences but of the same objective values in the solution archive. Comparing VNS and RVNS, the former performs better in MS-VRPTW with a fixed order of the neighbourhood operators of low perturbation to high perturbation. ALNS outperforms VNS and RVNS on the

Table 6 Solution deterioration comparing with the results of GIHH

		Average HV	Best HV	Best DP	Best TD	S.D.
FVNS	LB2	–4.86%	–4.76%	–3.19%	–0.88%	–0.54%
	LU2	2.44%	1.80%	–0.27%	1.13%	0.34%
	TB2	–15.13%	–15.90%	–3.39%	–0.40%	0.31%
	TU2	–9.25%	–9.75%	–2.76%	–2.01%	0.04%
	LB4	–5.70%	–5.82%	–2.29%	–1.04%	0.16%
	LU4	–1.30%	–2.59%	–1.19%	–1.22%	–0.67%
	TB4	–9.85%	–7.51%	–1.14%	–2.42%	–0.63%
	TU4	–13.85%	–12.14%	–3.50%	–0.72%	–0.30%
RVNS	LB2	–13.14%	–16.50%	–5.76%	–12.41%	–4.17%
	LU2	–10.27%	–12.93%	–2.31%	–8.50%	–9.05%
	TB2	–19.36%	–24.08%	–4.90%	–5.67%	–1.03%
	TU2	–14.92%	–18.46%	–5.63%	–6.04%	–2.52%
	LB4	–22.80%	–15.06%	–5.09%	–8.64%	–3.73%
	LU4	–29.90%	–23.98%	–4.49%	–11.38%	–4.40%
	TB4	–30.88%	–19.17%	–2.67%	–7.42%	–2.29%
	TU4	–27.02%	–21.87%	–3.99%	–4.61%	–1.04%
ALNS	LB2	–5.48%	–5.33%	–0.07%	–7.97%	0.22%
	LU2	–8.82%	–6.97%	1.17%	–9.59%	–0.48%
	TB2	–6.72%	–7.31%	0.57%	–3.77%	0.33%
	TU2	–8.47%	–7.69%	–1.03%	–4.68%	–0.19%
	LB4	–5.70%	–5.03%	–1.28%	–3.38%	0.59%
	LU4	–7.09%	–10.43%	–1.05%	–5.22%	0.39%
	TB4	–4.64%	–4.59%	–0.11%	–2.29%	0.08%
	TU4	–18.55%	–17.37%	–2.28%	–4.20%	–0.23%

The values in table are the objective differences divided by the GIHH objective values

objective DP with the help of large perturbation, while has a higher stability than GIHH.

Results on VRPTW benchmarks To evaluate its performance in other problems, GIHH is applied to classic VRPTW on the Solomon Benchmarks [2]. The VRPTW is the basis of many other complex VRPs, while the Solomon Benchmarks have been extended and adopted in the research of many other VRP variants as well. An equal priority is given to the two objectives, the number of vehicles used (NV) and the total travel distance (TD), in the VPRTW model of our study. The results obtained are compared with the best known solutions to date, see Table 7 in Appendix. It can be found that, GIHH obtains solutions the same as or close to the best known solutions (which are optimal actually) on the instances with clustered customers (C1 and C2). On the randomly and mixed distributed instances (R1, R2 and RC1, RC2), GIHH produces solutions close to the best known ones, and nine new non-dominated solutions are found. Considering that most of those best known solutions are generated by customized problem-specific algorithms with sufficient computation resources, the results of GIHH are satisfying.

5 Conclusions

This study defines a new bi-objective Mixed-Shift Vehicle Routing Problem with Time Windows (MS-VRPTW), which arises from a real-life container transportation problem between short-distance and long-distance terminals. Due to the big difference between the completion time of the transportation tasks, two types of shifts (*long-shift* and *short-shift*) with different operational costs are defined in this problem. The two objectives of this problem are minimizing the total driver payment and minimizing the total travel distance. A mathematical model of MS-VRPTW is proposed in this paper.

Using the proposed *artificial node*, the scheduling of two types of shifts is combined into one model. To the best of our knowledge, this is the first mixed-shift VRP model in the literature. Our investigation shows that it is unrealistic to tackle MS-VRPTW with exact search approaches even if a huge amount of computation resources is given. A hyper-heuristic is thus developed for MS-VRPTW. The proposed method showed to increase the utilization rate of trucks and reduce the operational cost of the logistic company.

In the proposed method, firstly, an initial feasible solution is generated using an Emergency Level-Based Insertion Construction Heuristic (EBIH). Then, a Hyper-Heuristic with two Guidance Indicators (GIHH) is proposed to improve the solutions. GIHH is a selection perturbation hyper-heuristic, adapting a set of Low-Level Heuristics (LLH) with different extents of perturbation to the problem solution. Two indicators are proposed to guide the LLH selection adaptively along with changes during the search, which evaluate LLH's contribution to the solution quality improvement and the improvement direction, respectively.

To test the generality and performance of the proposed algorithms, a set of diverse benchmark problem instances is created based on a dataset derived from the real-world problem, considering the features of commodity emergency and workload balance. On all the benchmark instances, EBIH produced feasible solutions within an acceptable time. The experiment results show that, in different environments, the two proposed guidance indicators significantly improve the performance and stability of search for this bi-objective problem, producing solutions with higher hyper-volumes. In terms of the acceptance criterion and the selection scheme of solution, it is shown that, when the ranges of objectives are vastly different in the Multi-Objective Vehicle Routing Problem, the *Hill Climbing* acceptance criterion outperforms the acceptance criterion of accepting worse solutions (*Record-to-Record Travel*). Research also finds that randomly selecting the next current solution can increase the diversity of search, bringing better results than deterministic selection in MS-VRPTW. GIHH outperforms three state-of-the-art algorithms for PVRP and VRPPD on both the computation time and the quality of solutions generated. Comparing to the best known solutions to date, GIHH also produces promising results in the classic VRPTW.

In our future work, the MS-VRPTW model could be extended to other mixed-shift problems. The proposed algorithms can be applied to more practical complicated multi-objective optimization problems. Hybrid methodologies combining GIHH and exact methods can be another promising research direction.

Acknowledgements This research is supported by the National Natural Science Foundation of China (Grant No. 71471092), Zhejiang Natural Science Foundation (Grant No. LR17G010001), Ningbo Science & Technology Bureau (Grandt No. 2014A35006) and School of Computer Science, The University of Nottingham. We are grateful for the access to the University of Nottingham High Performance Computing Facility.

Appendix

Table 7 Results of GIHH on the Solomon's instances

Instance	Best Known			GIHH	
	NV	TD	Ref.	Best Found	
				NV	TD
C101	10	828.94	[62]	**10**	**828.94**
C102	10	828.94	[62]	**10**	**828.94**
C103	10	828.06	[62]	10	828.94
C104	10	824.78	[62]	10	825.65
C105	10	828.94	[62]	**10**	**828.94**
C106	10	828.94	[62]	**10**	**828.94**
C107	10	828.94	[62]	**10**	**828.94**
C108	10	828.94	[62]	**10**	**828.94**
C109	10	828.94	[62]	**10**	**828.94**
C201	3	591.56	[62]	**3**	**591.56**
C202	3	591.56	[62]	**3**	**591.56**
C203	3	591.17	[62]	**3**	**591.17**
C204	3	590.6	[62]	**3**	**590.6**
C205	3	588.88	[62]	**3**	**588.88**
C206	3	588.49	[62]	**3**	**588.49**
C207	3	588.29	[62]	**3**	**588.29**
C208	3	588.32	[62]	**3**	**588.32**
R101	19	1650.80	[62]	**19**	**1650.80**
	20	1642.87	[63]	20	1643.34
R102	17	1486.12	[62]	17	1489.33
	18	1476.06	[53]	18	1490.72
R103	13	1292.67	[62]	13	1367.27
				14	**1219.89**
R104	9	1007.31	[62]		
	10	974.24	[64]	10	1007.27
	11	971.5	[65]	11	994.85
R105	14	1377.11	[62]	14	1381.88
	15	1346.12	[66]	15	1360.78
R106	12	1252.03	[62]	12	1270.45
	13	1234.6	[67]	13	1243.72
R107	10	1104.66	[62]		
	11	1051.84	[66]	11	1077.24
				12	**1050.06**
R108	9	960.88	[62]		
	10	932.1	[68]	10	956.22
R109	11	1194.73	[62]		
	12	1013.2	[69]	12	1168.18
	13	1151.84	[63]	13	1157.61
R110	10	1118.84	[62]		
	11	1112.21	[68]	11	1153.83
	12	1068	[67]	12	1081.88
R111	10	1096.72	[62]	**11**	**1087.5**
	12	1048.7	[67]	12	1062.58
R112	9	982.14	[62]		
	10	953.63	[70]	10	958.7
R201	4	1252.37	[62]	4	1282.75
	5	1190.52	[53]		
R202	3	1191.7	[62]	3	1239.82
	4	1091.21	[64]	4	1098.06
R203	3	939.503	[62]	3	968.67
	4	905.72	[53]	4	935.55
R204	2	825.52	[62]		
	3	766.91	[53]	3	767.52
R205	3	994.42	[62]	3	1059.91
	5	954.16	[63]	**4**	**964.02**
R206	3	906.142	[62]	3	930.80
R207	2	890.61	[62]		
	3	814.78	[70]	3	843.88
R208	2	726.82	[62]	2	741.75
	4	698.88	[71]	**3**	**708.9**
R209	3	909.16	[62]	3	962.08
	5	860.11	[63]	4	871.63
R210	3	939.37	[62]	3	978.11
	4	935.01	[53]	4	948.95
R211	2	885.71	[62]		
	3	794.04	[53]	3	804.16
	4	761.1	[68]		
RC101	14	1696.94	[62]		
	15	1619.8	[71]	15	1633.10
RC102	12	1554.75	[62]		
	13	1470.26	[64]	13	1497.43
	14	1466.84	[63]	14	1467.25
RC103	11	1261.67	[62]	11	1265.86
RC104	10	1135.48	[62]	10	1136.49
RC105	13	1629.44	[62]		
	14	1589.91	[64]	14	1623.54
	15	1513.7	[63]	15	1524.14
RC106	11	1424.73	[62]	**12**	**1396.59**
	13	1371.69	[64]	13	1376.99
RC107	11	1230.48	[62]	11	1254.68
	12	1212.83	[63]	12	1233.58
RC108	10	1139.82	[62]	10	1200.69
	11	1117.53	[63]	11	1131.23
RC201	4	1406.94	[62]	4	1457.87
	6	1134.91	[64]	**5**	**1310.44**
RC202	3	1365.64	[62]	3	1546.3
	4	1181.99	[68]	4	1192.54
RC203	3	1049.62	[62]	3	1097.32
	4	957.10	[53]		

Table 7 (continued)

Instance	Best Known			GIHH	
	NV	TD	Ref.	Best Found	
				NV	TD
RC204	3	798.46	[62]	3	829.13
RC205	4	1297.65	[62]	4	1298.90
	5	1233.46	[53]	5	1240.45
RC206	3	1146.32	[62]	3	1156.06
	4	1107.40	[53]	**4**	**1107.19**
RC207	3	1061.14	[62]	3	1135.61
	4	1032.78	[53]	4	1033.78
RC208	3	828.14	[62]	3	830.06

The solutions equal to best known results and the newly found non-dominated solutions are shown in bold

References

1. Dantzig GB, Ramser JH (1959) The truck dispatching problem. Manag Sci 6(1):80–91
2. Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper Res 35(2):254–265
3. Eksioglu B, Vural AV, Reisman A (2009) The vehicle routing problem: a taxonomic review. Comput Ind Eng 57(4):1472–1483
4. Golden BL, Raghavan S, Wasil EA (2008) The vehicle routing problem: latest advances and new challenges. Springer Science & Business Media, vol. 43
5. Wieberneit N (2008) Service network design for freight transportation: a review. OR spectrum 30(1):77–112
6. Zhang R, Yun WY, Kopfer H (2010) Heuristic-based truck scheduling for inland container transportation. OR spectrum 32(3):787–808
7. Wang X, Regan AC (2002) Local truckload pickup and delivery with hard time window constraints. Transp Res B Methodol 36(2):97–112
8. Chen J, Bai R, Qu R, Kendall G (2013) A task based approach for a real-world commodity routing problem. In: 2013 IEEE Workshop on Computational Intelligence In Production And Logistics Systems (CIPLS) IEEE, Conference Proceedings, pp 1–8
9. Mourgaya M, Vanderbeck F (2007) Column generation based heuristic for tactical planning in multi-period vehicle routing. Eur J Oper Res 183(3):1028–1041
10. Dayarian I, Crainic TG, Gendreau M, Rei W (2016) An adaptive large-neighborhood search heuristic for a multi-period vehicle routing problem. Transportation Research Part E: Logistics and Transportation Review 95:95–123
11. Cordeau JF (2000) The VRP with time windows. Montréal: Groupe détudes et de recherche en analyse des décisions
12. Erdoğan S, Miller-Hooks E (2012) A green vehicle routing problem. Transportation Research Part E: Logistics and Transportation Review 48(1):100–114
13. Bektaś T, Laporte G (2011) The pollution-routing problem. Transp Res B Methodol 45(8):1232–1250
14. Talbi EG (2009) Metaheuristics: from design to implementation. Wiley, vol 74
15. Bräysy O, Gendreau M (2001) Metaheuristics for the vehicle routing problem with time windows. Report STF42 A, vol. 1025, 3–38
16. Laporte G, Gendreau M, Potvin J-Y, Semet F (2000) Classical and modern heuristics for the vehicle routing problem. Int Trans Oper Res 7(4-5):285–300
17. Aarts EH, Lenstra JK (1997) Local search in combinatorial optimization. Princeton University Press
18. Hansen P, Mladenović N, Pérez JAM (2010) Variable neighbourhood search: methods and applications. Ann Oper Res 175(1):367–407
19. Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys (CSUR) 35(3):268–308
20. Gendreau M, Tarantilis CD (2010) Solving large-scale vehicle routing problems with time windows: The state-of-the-art. CIRRELT
21. Burke E, Kendall G, Newall J, Hart E, Ross P, Schulenburg S (2003) Hyper-heuristics: An emerging direction in modern search technology. In: Handbook of metaheuristics. Springer, pp 457–474
22. Bai R, Blazewicz J, Burke EK, Kendall G, McCollum B (2012) A simulated annealing hyper-heuristic methodology for flexible decision support. 4OR: A Quarterly Journal of Operations Research 10(1):43–66
23. Burke EK, Gendreau M, Hyde M, Kendall G, Ochoa G, Özcan E, Qu R (2013) Hyper-heuristics: a survey of the state of the art. J Oper Res Soc 64(12):1695–1724
24. Qu R, Burke EK, McCollum B, Merlot LT, Lee SY (2009) A survey of search methodologies and automated system development for examination timetabling. J Sched 12(1):55–89
25. Burke EK, Hyde M, Kendall G, Ochoa G, Özcan E, Woodward JR (2010) A classification of hyper-heuristic approaches. In: Handbook of metaheuristics. Springer, pp. 449–468
26. Garrido P, Riff MC (2010) Dvrp: a hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic. J Heuristics 16(6):795–834
27. Sabar NR, Ayob M, Kendall G, Qu R (2012) Grammatical evolution hyper-heuristic for combinatorial optimization problems. strategies 3:4
28. Walker JD, Ochoa G, Gendreau M, Burke EK (2012) Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework. In: LION. Springer, pp. 265–276
29. Sabar NR, Ayob M, Kendall G, Qu R (2015) A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems. IEEE Transactions on Cybernetics 45(2):217–228
30. Vidal T, Crainic TG, Gendreau M, Prins C (2014) A unified solution framework for multi-attribute vehicle routing problems. Eur J Oper Res 234(3):658–673
31. Potvin J-Y, Rousseau J-M (1995) An exchange heuristic for routeing problems with time windows. J Oper Res Soc 46(12):1433–1446
32. 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
33. Shaw P (1997) A new local search algorithm providing high quality solutions to vehicle routing problems. APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK
34. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In: International Conference on Principles and Practice of Constraint Programming. Springer, pp. 417–431
35. Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. Comput Oper Res 34(8):2403–2435

36. Schrimpf G, Schneider J, Stamm-Wilbrandt H, Dueck G (2000) Record breaking optimization results using the ruin and recreate principle. J Comput Phys 159(2):139–171
37. Nagata Y, Bräysy O (2009) A powerful route minimization heuristic for the vehicle routing problem with time windows. Oper Res Lett 37(5):333–338
38. Lim A, Zhang X (2007) A two-stage heuristic with ejection pools and generalized ejection chains for the vehicle routing problem with time windows. INFORMS J Comput 19(3):443–457
39. Nagata Y, Tojo S (2009) Guided ejection search for the job shop scheduling problem. In: European Conference on Evolutionary Computation in Combinatorial Optimization. Springer. Conference Proceedings, pp. 168–179
40. Nagata Y, Kobayashi S (2010) Guided ejection search for the pickup and delivery problem with time windows. In: European Conference on Evolutionary Computation in Combinatorial Optimization. Springer, Conference Proceedings, pp. 202–213
41. Ehrgott M (2006) Multicriteria optimization. Springer Science & Business Media
42. Lourens T (2005) Using population-based incremental learning to optimize feasible distribution logistic solutions. Thesis
43. Coello CAC, Lamont GB, Van Veldhuizen DA et al (2007) Evolutionary algorithms for solving multi-objective problems Springer, vol. 5
44. Ghoseiri K, Ghannadpour SF (2010) Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. Appl Soft Comput 10(4):1096–1107
45. Yu B, Yang ZZ (2011) An ant colony optimization model: The period vehicle routing problem with time windows. Transportation Research Part E: Logistics and Transportation Review 47(2):166–181
46. Jozefowiez N, Semet F, Talbi E-G (2008) Multi-objective vehicle routing problems. Eur J Oper Res 189(2):293–309
47. Maps G (2018). Google maps https://www.google.com/maps/d/u/0/edit?hl=en&mid=1F7Ap9EO3MyzFudUQ4_Mu48ZlhSY&ll=30.08120233597248%2C117.0881652999999&z=7, accessed: 2018-05-11
48. Chrpa L, Magazzeni D, McCabe K, McCluskey TL, Vallati M (2016) Automated planning for urban traffic control: Strategic vehicle routing to respect air quality limitations. Intelligenza Artificiale 10(2):113–128
49. Allard T, Gretton C (2015) A realistic multi-modal cargo routing benchmark. In: Workshops at the 29th AAAI Conference on Artificial Intelligence
50. Kiesel S, Burns E, Wilt CM, Ruml W (2012) Integrating vehicle routing and motion planning. In: ICAPS
51. Chen B, Qu R, Ishibuchi H (2017) Variable-depth adaptive large neighbourhood search algorithm for open periodic vehicle routing problem with time windows. In: International Conference on Harbor, Maritime & Multimodal Logistics Modelling and Simulation (HMS 2017), Conference Proceedings, pp. 25–34
52. Dueck G (1993) New optimization heuristics: The great deluge algorithm and the record-to-record travel. J Comput Phys 104(1):86–92
53. Chen B, Qu R, Bai R, Ishibuchi H (2016) An investigation on compound neighborhoods for vrptw. In: International Conference on Operations Research and Enterprise Systems. Springer, Cham, pp 3–19
54. Lin S (1965) Computer solutions of the traveling salesman problem. The Bell System Technical Journal 44(10):2245–2269
55. Bai R, Xue N, Chen J, Roberts GW (2015) A set-covering model for a bidirectional multi-shift full truckload vehicle routing problem. Transp Res B Methodol 79:134–148
56. Li M, Yang S, Liu X (2014) Diversity comparison of pareto front approximations in many-objective optimization. IEEE Transactions on Cybernetics 44(12):2568–2584
57. Bradstreet L (2011) The hypervolume indicator for multi-objective optimisation: calculation and use. University of Western Australia
58. Pirkwieser S, Raidl GR (2008) A variable neighborhood search for the periodic vehicle routing problem with time windows. In: Proceedings of the 9th EU/meeting on metaheuristics for logistics and vehicle routing. Troyes, France, pp 23–24
59. Hemmelmayr VC, Doerner KF, Hartl RF (2009) A variable neighborhood search heuristic for periodic routing problems. Eur J Oper Res 195(3):791–802
60. Ropke S, Pisinger D (2006) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp Sci 40(4):455–472
61. Masson R, Lehuédé F, Péton O (2013) An adaptive large neighborhood search for the pickup and delivery problem with transfers. Transp Sci 47(3):344–355
62. SINTEF (2018) Best known solution values for solomon benchmark http://www.sintef.no/Projectweb/TOP/VRPTW/Solomon-benchmark/100-customers/, accessed: 2018-05-11
63. Alvarenga GB, Mateus GR, De Tomi G (2007) A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. Comput Oper Res 34(6):1561–1584
64. Tan K, Chew Y, Lee L (2006) A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. Comput Optim Appl 34(1):115–151
65. Küç ükoğ lu İ, Öztürk N (2014) An advanced hybrid meta-heuristic algorithm for the vehicle routing problem with backhauls and time windows. Computers & Industrial Engineering
66. Kallehauge B, Larsen J, Madsen OB (2006) Lagrangian duality applied to the vehicle routing problem with time windows. Comput Oper Res 33(5):1464–1487
67. Cook W, Rich JL (1999) A parallel cutting-plane algorithm for the vehicle routing problem with time windows. Computational and Applied Mathematics Department, Rice University, Houston, TX, Technical Report
68. Ombuki B, Ross BJ, Hanshar F (2006) Multi-objective genetic algorithms for vehicle routing problem with time windows. Appl Intell 24(1):17–30
69. Chiang W-C, Russell RA (1997) A reactive tabu search metaheuristic for the vehicle routing problem with time windows. INFORMS Journal on computing 9(4):417–430
70. Rochat Y, Taillard ÉD (1995) Probabilistic diversification and intensification in local search for vehicle routing. J Heuristics 1(1):147–167
71. Ursani Z, Essam D, Cornforth D, Stocker R (2011) Localized genetic algorithm for vehicle routing problem with time windows. Appl Soft Comput 11(8):5375–5390

Binhui Chen holds BSc and MSc degrees from Fuzhou University, China. Currently, he is a PhD student in the School of Computer Science, University of Nottingham, UK. His research interests include combinatorial optimization problem model and algorithm, data mining and transporatation problems.

Dr. Rong Qu is an Associated Professor at the University of Nottingham. She is a member of the Automated Scheduling, Optimsation And Planning (ASAP) Research Group. She received her PhD in Computer Science from the University of Nottingham in 2002, and BSc in Computer Science and Its Applications from XiDian University in 1996. Dr. Qu's main research interests include the modelling and optimisation algorithms for scheduling and optimisation algorithms in transport scheduling in logistics, personnel scheduling, telecommunication network routing, portfolio optimisation, and timetabling problems, etc. by using evolutionary algorithms, mathematical programming, constraint programming in operational research and artificial intelligence, and hybridisations of these techniques.

Prof. Ruibin Bai holds BSc and MSc degrees from Northwestern Polytechnic University, China and a PhD from University of Nottingham UK. He is now a professor in the School of Computer Science and leads the Artificial Intelligence and Optimisation (AIOP) group. He is an IEEE senior member and Associate Editor for Networks, an ISI indexed journal. His current research interests include computational intelligence, machine learning, operations research, modelling, scheduling and optimisation with a special focus on transportation systems and digital healthcare.

Wasakorn Laesanklang received his PhD in Computer Science from the University of Nottingham, UK. Currently, he is a faculty member of the Department of Mathematics, Faculty of Science, Mahidol University, Thailand. He also works as a researcher for the Centre of Excellence in Mathematics, CHE, Thailand. His research interests are mixed integer programming model, transportation problems and optimization techniques.