Journal of Heuristics
https://doi.org/10.1007/s10732-019-09412-1

Three multi-start data-driven evolutionary heuristics for the vehicle routing problem with multiple time windows

Slim Belhaiza[1] · **Rym M'Hallah**[2] · **Ghassen Ben Brahim**[3] · **Gilbert Laporte**[4]

Received: 8 August 2018 / Revised: 10 January 2019 / Accepted: 1 April 2019

Abstract

This paper considers the vehicle routing problem with multiple time windows. It introduces a general framework for three evolutionary heuristics that use three global multi-start strategies: ruin and recreate, genetic cross-over of best parents, and random restart. The proposed heuristics make use of information extracted from routes to guide customized data-driven local search operators. The paper reports comparative computational results for the three heuristics on benchmark instances and identifies the best one. It also shows more than 16% of average cost improvement over current practice on a set of real-life instances, with some solution costs improved by more than 30%.

Keywords Evolutionary search · Genetic algorithm · Vehicle routing problem with multiple time windows · Local search

This work was partly funded by the Canadian Natural Sciences and Engineering Research Council under Grant 2015-06189. This support is gratefully acknowledged. Thanks are due to the referees for their valuable comments.

✉ Slim Belhaiza
slimb@kfupm.edu.sa

Rym M'Hallah
rymmha@yahoo.com

Ghassen Ben Brahim
gbrahim@pmu.edu.sa

Gilbert Laporte
gilbert.laporte@cirrelt.ca

1 Department of Mathematics and Statistics, King Fahd University of Petroleum and Minerals, Dhahran, Kingdom of Saudi Arabia

2 Department of Statistics and Operations Research, Kuwait University, Kuwait City, Kuwait

3 Department of Computer Science, Prince Mohammad Bin Fahd University, Dhahran, Kingdom of Saudi Arabia

4 CIRRELT and HEC Montréal, Montreal, Canada

Published online: 08 April 2019

1 Introduction

This paper considers the vehicle routing problem with multiple time windows (VRPMTWs), an extension of the VRP where deliveries at customers location may take place during one of several time windows. The multiple time window constraints further complicate the VRP and render exact approaches intractable. Solving the VRPMTW requires the application of approximate algorithms with proven efficiency on highly combinatorial problems, such as genetic algorithms (GA) and variable neighborhood search (VNS).

Genetic algorithms are adaptive metaheuristic techniques inspired from population genetics. Their main concepts were defined by Holland (1975) while their practical efficiency in solving complex problems was shown by De Jong (1975), Goldberg (1989), Prins (2004) and Vidal et al. (2013a, b), for example. These algorithms consider a population of chromosomes and apply natural selection mechanisms to proliferate the individuals of the population. Their application requires defining a solution representation, as well as selection, crossover, and mutation mechanisms. The representation encodes a solution as a chromosome. The selection mechanism allows the fittest chromosomes to survive and proliferate. The crossover strengthens the good genes of a population, thus creating offspring with better genetic traits than those of their parents. Finally, the mutation adapts an individual to its environment.

Variable neighborhood search is an effective and efficient heuristic scheme. It explores various neighborhoods with different structures or focal points. It perturbates the current local minimum and changes the neighborhood type to intensify and diversify the search. It helps identifying the main characteristics of the problem and balancing the two key components of an efficient cooperative algorithm (Liu and Zhou 2013; Sánchez-Oro et al. 2014). It produces local optima that are closer to the global optimum in a more efficient and direct way than other metaheuristics (Hansen et al. 2010).

This paper proposes three multi-start data-driven evolutionary heuristics for the VRPMTW. The proposed heuristics bring four main methodological contributions. First, they keep track not only of the current and best solutions, but also of a pool of best solutions which constitutes the GA's population. This feature is motivated by the successful hybridization of GA with VNS for the flow time no-wait flow shop problem (Yang et al. 2008). Second, they enhance standard VNS with a ruin-and-recreate procedure that helps VNS escape from suboptimal valleys. This procedure has yielded substantial benefits on the VRP with multiple routes (Azi et al. 2014). Third, they apply a multiple restart strategy which reduces the risk of cycling. This feature has enhanced the results of the periodic VRP with time spread constraints on services (Michalet et al. 2014). Fourth, they categorize the VRPMTW infeasibilities into different types, and handle each of them with the appropriate customized local search operators. This feature has proved to be efficient for the Dial-a-ride problem with time windows (Belhaiza et al. 2017). The ruin-and-recreate procedure, the multiple restart strategy and the customized data-driven local search operators are useful additions with respect to the hybrid variable neighborhood tabu search (HVNTS) and the hybrid genetic variable neighborhood search (HGVNS) heuristics proposed in Belhaiza et al. (2014, 2017), as validated by our experimental results.

Section 2 provides some background on the VRPMTW, reviews its literature and examines recent GA-based, VNS-based and data-driven heuristics used in the VRP context. Section 3 describes the diversification strategies. Section 4 introduces the three proposed multi-start data-driven heuristics and the VRPMTW infeasibilities. Section 5 details the algorithmic steps of our proposed heuristics, including the data-driven local search operators. Section 6 analyzes the performance of the three different heuristics on VRPMTW benchmark instances, and reports computational results of the best heuristic on VRPTW and VRPMTW benchmark and real-life instances. Section 7 concludes the paper.

2 The VRPMTW

The VRPMTW is defined on a directed graph $G = (V, A)$, where V is the vertex set and A is the arc set. The vertex set is partitioned into $V = \{\mathcal{I}, \mathcal{D}\}$, where $\mathcal{I} = \{1, \ldots, n\}$ is a set of customers and $\mathcal{D}$ is a set of depots. We consider a set $\mathcal{K} = \{1, \ldots, m\}$ of m vehicles. Each vehicle $k \in \mathcal{K}$ has a finite capacity $Q_k > 0$. It leaves its depot $d_k \in \mathcal{D} = \{1, \ldots, \bar{d}\}$, visits a subset N_k of customers, and returns to d_k. Each customer $i \in \mathcal{I}$ is characterized by its demand $q_i \geq 0$, service time $s_i \geq 0$, and ordered set of $\bar{p}_i$ alternative delivery time windows $\mathcal{W}_i = \{w_1, \ldots, w_{\bar{p}_i}\}$, where l_{p_i}, u_{p_i} are the upper and lower bounds of time window w_{p_i}, and $0 < l_1 \leq l_{p_i} < u_{p_i} \leq u_{\bar{p}_i} < \infty$. A customer can only be visited by one vehicle. A vehicle reaching a customer i before l_{p_i}, waits until l_{p_i} to start delivery.

A vehicle $k \in \mathcal{K}$ is assigned to a unique route, which is a closed path in the graph G. To each route of a vehicle k correspond a total traveled distance, a total travel time, and a total travel duration D_k. The total duration D_k equals the sum of service, waiting and travel times. We denote by X a solution to the VRPMTW. The objective is to minimize the total cost $h(X)$ of delivering the demand of all customers. The objective $h(X)$ is proportional to either the total travel time $h_1(X)$ or to the total duration $h_2(X)$. It includes the vehicle fixed costs, which are expressed in time units and are a function of their capacities.

2.1 Mixed 0-1 linear programming formulation

Along the lines of Favaretto et al. (2007) and Belhaiza et al. (2014), the VRPMTW can be formulated as a mixed 0-1 linear program. We define in Table 1 the variables and the parameters used in our formulation. The objective is to minimize the total travel time, in addition to the total waiting and service time multiplied by a binary parameter B, equal to 1 if and only if total duration is to be minimized, plus the sum of all used vehicles' fixed costs F^k converted to time units.

$$\min \quad h(X) = \sum_{k\in\mathcal{K}}\sum_{i\neq j} t_{ij}x_{ij}^k + B\left(\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}} w_i^k + \sum_{i\in\mathcal{I}} s_i\right) + \sum_{k\in\mathcal{K}} F^k r^k$$

subject to

$$(1) \quad \sum_{k\in\mathcal{K}} z_i^k = 1, \qquad i \in V,$$

$$(2) \quad \sum_{j\in V} x_{ji}^k = \sum_{j\in V} x_{ij}^k \qquad i \in V \text{ and } k \in \mathcal{K},$$

$$(3) \quad 2x_{ij}^k \leq z_i^k + z_j^k, \qquad i,\ j \in V \text{ and } k \in \mathcal{K},$$

$$(4) \quad \sum_{k\in\mathcal{K}}\sum_{j\in V} x_{ij}^k \leq 1, \qquad i \in V,$$

$$(5) \quad \sum_{k\in\mathcal{K}}\sum_{j\in V} x_{ji}^k \leq 1, \qquad i \in V,$$

$$(6) \quad y_{ij}^k \leq Q_k x_{ij}^k, \qquad i,\ j, \in V \text{ and } k \in \mathcal{K},$$

$$(7) \quad \sum_{j\in V} y_{dj}^k - \sum_{i\in V} y_{id}^k = q_d^k e_d^k, \qquad d \in \mathcal{D} \text{ and } k \in \mathcal{K},$$

$$(8) \quad \sum_{j\in V} y_{ji}^k - \sum_{j\in V} y_{ij}^k \geq q_i z_i^k, \qquad i \in \mathcal{I} \text{ and } k \in \mathcal{K},$$

$$(9) \quad b_d^k \geq l_d - M(1 - z_d^k), \qquad d \in \mathcal{D} \text{ and } k \in \mathcal{K},$$

$$(10) \quad a_d^k \leq u_d + M(1 - z_d^k), \qquad d \in \mathcal{D} \text{ and } k \in \mathcal{K},$$

$$(11) \quad b_i^k \geq a_i^k + w_i^k + s_i - M(1 - z_i^k), \qquad i \in \mathcal{I} \text{ and } k \in \mathcal{K},$$

$$(12) \quad a_j^k \geq b_i^k + t_{ij} - M(1 - x_{ij}^k), \qquad i,\ j \in V \text{ and } k \in \mathcal{K},$$

$$(13) \quad a_i^k + w_i^k \geq l_i^p - M(1 - z_i^k) - M(1 - v_i^p), \quad i \in \mathcal{I},\ p \in W_i \text{ and } k \in \mathcal{K},$$

$$(14) \quad a_i^k + w_i^k \leq u_i^p + M(1 - z_i^k) + M(1 - v_i^p), \quad i \in \mathcal{I},\ p \in W_i \text{ and } k \in \mathcal{K},$$

$$(15) \quad \sum_{p=1}^{p_i} v_i^p = 1, \qquad i \in \mathcal{I} \cap \mathcal{D},$$

$$(16) \quad r^k \geq z_i^k, \qquad i \in V \text{ and } k \in \mathcal{K},$$

$$(17) \quad y_{ij}^k,\ w_i^k,\ q_d^k,\ d_k,\ a_i^k,\ b_i^k \geq 0,$$

$$(18) \quad r^k,\ x_{ij}^k,\ v_i^p,\ z_i^k \text{ binary.}$$

Conditions (1) assert that every customer is assigned to exactly one vehicle. Conditions (2) assert that every route of a vehicle k starts and ends at the depot, and that the number of arcs leaving a customer's node i is equal to the number of arcs entering it.

Table 1 Definition of the parameters and variables

	Type	Description
Parameter		
t_{ij}	Real	Travel time associated with the arc $(i,\ j)$
q_i	Real	Demand associated with vertex customer i
l_i^p	Real	Time window p lower bound at customer i
u_i^p	Real	Time window p upper bound at customer i
Q_k	Real	Capacity of vehicle k
D_k	Real	Maximum duration of the route of vehicle k
s_i	Real	Service time at customer i
e_d^k	Binary	Equal to 1 if and only if vehicle k starts and ends at depot d
B	Binary	Equal to 1 if and only if total duration is to be minimized
F^k	Real	Fixed cost in time units of using vehicle k
M	Real	Arbitrary large constant
Variable		
x_{ij}^k	Binary	Equal to 1 if and only if arc $(i,\ j)$ is traversed by vehicle k
y_{ij}^k	Real	Equal to the flow carried on arc $(i,\ j)$
r^k	Binary	Equal to 1 if and only if vehicle k is used
v_i^p	Binary	Equal to 1 if and only if customer i is served within its time window p
w_i^k	Real	Waiting time of vehicle k at customer i
z_i^k	Binary	Equal to 1 if and only if customer i is assigned to vehicle k
q_d^k	Real	Total demand loaded in vehicle k at depot d
d_k	Real	Route duration of vehicle k
a_i^k	Real	Arrival time of vehicle k at customer i
b_i^k	Real	Departure time of vehicle k from customer i

Conditions (3) assert that every arc $(i,\ j)$ can be visited by vehicle k only if z_i^k and z_j^k are both equal to 1. Conditions (4) and (5), respectively, assert that every customer i can only be visited by one entering arc and one leaving arc. Conditions (6) assert that the flow on arc $(i,\ j)$ is upper-bounded by the capacity Q_k of the vehicle k visiting this arc. Conditions (7) assert that the demand of all customers visited by vehicle k is satisfied, while conditions (8) assert that the demand of each customer assigned to the route of a vehicle k is satisfied. Conditions (9) assert that the departure time of vehicle k from the depot d does not exceed l_d, and conditions (10) assert that the arrival time of vehicle k at the depot d is less or equal to u_d. Conditions (11) assert that the departure time of vehicle k from customer i, when customer i is assigned to vehicle k, is at least equal to the arrival time of vehicle k at customer i, plus the waiting time of vehicle k and the service time at customer i. Conditions (12) assert that the arrival time of vehicle k at customer j is equal to the departure time from customer i, plus the travel time t_{ij} on arc $(i,\ j)$, if this arc is assigned to vehicle k. Conditions (13) and (14) assert that the arrival time of vehicle k plus its waiting time at

customer i is within the time window $\left[l_i^p, u_i^p\right]$, when customer i is served by vehicle k during time window p. Conditions (15) assert that exactly one single time window is chosen for every customer i. Conditions (16) assert that a given customer i can be served by vehicle k only if this vehicle is used. Finally, conditions (17) and (18) set the non-negativity and binary conditions. Due to their very large numbers of variables and conditions, it is common knowledge that VRP mixed 0-1 linear formulations can be solved to optimality only for a limited number of customers and vehicles.

2.2 Related literature

The literature on the VRPMTW is rather limited. Favaretto et al. (2007) described an ant colony heuristic (ACH) for the VRPMTW, and Belhaiza et al. (2014) proposed a hybrid variable neighborhood tabu search (HVNTS). Experimental results showed that HVNTS outperforms ACH and reduces the optimality gap on two sets of benchmark instances. Beheshti et al. (2015) solved the multi-objective VRP with multiple prioritized time windows, where the distributor has a set of non-overlapping time windows ranked by each customer according to their preference. They proposed a cooperative co-evolutionary multi-objective quantum-genetic algorithm to construct a Pareto front. They applied their method to a real-life case study and compared their solution to that of the distributor. They concluded that their algorithm significantly improves the operational efficiency and yields a better Pareto set. Belhaiza and M'Hallah (2016) proposed a Pareto non-dominated based approach. They modeled the VRP aspect from a game theoretic perspective and tested it on the benchmark instances presented in Belhaiza et al. (2014). Belhaiza et al. (2017) proposed a hybrid genetic variable neighborhood search heuristic (HGVNS). Their hybridization is based on the addition of genetic crossover operators within the shaking phase of VNS. This algorithm helped improve many of the best known solutions on a set of VRPMTW benchmark instances. Nevertheless, it suffers from early convergence due to the lack of efficient diversification strategies compared with the new multi-start evolutionary heuristic we will propose in this paper. Belhaiza (2018b) showed that Nash equilibria correspond to VRP solutions when positive utilities are assigned to agents' preferences. The proposed approach was tested on multiple-depot real-life instances. The results demonstrated not only the applicability of the approach to large-scale industrial problems, but also the savings that it induces. Ferreira et al. (2018) proposed a simple VNS heuristic which explores only feasible solutions for the VRPMTW. Although, it displays some promising preliminary results, the paper does not report new best solutions on the benchmark instances. Hoogeboom and Dullaert (2018) proposed an efficient variable neighborhood evaluation framework for the VRPMTW which proved to be competitive with HVNTS and HGVNS. We compare our results to theirs in Sect. 6.

2.3 GA and VNS applications in VRP

Blanton and Wainwright (1993) proposed a hybrid GA with a greedy heuristic. Tangiah (1995) developed a GA to obtain initial solutions for their hybrid heuristics consisting of simulated annealing, tabu search, and λ-exchange route improvement. Potvin and

Bengio (1996) derived a specialized methodology for merging a pair of solutions into a single one that is likely to be feasible. Berger et al. (2003) proposed a two-population GA, the first focusing on minimizing the total traveled distance, and the second on minimizing the violations of the temporal constraints. Vidal et al. (2013a, b) classified the literature according to the problem characteristics and constraints, pointed out the most successful approaches and explained their key strategies. They deduced guidelines for the design of heuristics for related problems. Genetic algorithms (GA) have been extensively applied to the VRP and its variants. Mirabi (2015) used three genetic operators to produce new offspring for the multi-depot periodic VRP. Despite its wide application in optimization, GA has a poor local search capability that limits its success as a stand alone heuristic for VRP. In addition, it suffers from premature convergence. Bouziyane et al. (2016) proposed a hybrid genetic-VNS algorithm for the static and dynamic vehicle VRP with soft time windows.

Similarly, VNS has been widely applied to the VRP and the VRPTW. Neighborhood-centered search methods like iterative local search (ILS) (Michalet et al. 2014), multiple restart ILS (Lourenço et al. 2015), and VNS improve when combined with simple local search operators and diversification strategies. A more recent application (Defryn and Sörensen 2017) is a two-level VNS based heuristic for the clustered capacitated VRP. Customers of a cluster are served sequentially by the same vehicle. The two-level VNS devotes one level to the clusters and the other to the customers of a cluster. The most recent application (Hoogeboom and Dullaert 2018) proposed a granular tabu-VNS based heuristic for the VRP with arrival time diversification, with an application to the real-life cash in transit problem (CIT). They used a multiple-time windows' routing approach to obtain new best-known solutions for all benchmark instances from the literature for the periodic vehicle routing problem with time window and time spread constraints on services, decreasing the average distance by 29%.

2.4 Machine learning and data-driven applications in VRP

Machine learning (ML) and data-driven based techniques have been used in conjunction with heuristic techniques with the goal of achieving improved solutions to the VRP and its variants. Zennaki and Ech-Cherif (2008) proposed a support vector machine (SVM) based learning approach to guide heuristic techniques to explore the more promising areas of the search space. Experimental results demonstrated that SVM is very effective in improving the quality of solutions, as well as in reducing the computation time. Along the same line, Al-Duoli and Rabadi (2014) used ML techniques to guide heuristic algorithms perform efficient search. They use inductive decision tree to improve the performance of the meta-heuristic randomized priority search (meta-RaPS). The performance of this model is yet to be assessed. Rasku et al. (2016) proposed a feature based ML approach by dynamically and adaptively configure existing VRP solver algorithms. This approach is an extension of the Steinhaus (2015) method. The authors proposed a comprehensive set of 386 features to describe the problem types and properties which will further be used to help configure three meta-heuristic capacity VRP (CVRP) solvers. Results on 168 CVRP benchmark instances showed that simulated annealing benefited the most from the use of the

proposed set of features. Calvet et al. (2016) introduced a statistical-based learning to solve the multi-depot VRP. The proposed model includes a statistical learning step to study the correlation between historical customer features and problem parameters such as the customer expenditures, etc. The output of this learning step will be used in case of new problem instance. Fahrettin (2016) used ML techniques to incorporate knowledge discovery during the process of solving complex problems and the use of historical data to develop models aiming at performing either of the following: (1) explore the search space more efficiently, (2) tune existing heuristic configuration parameters, and (3) predict problem related parameters based on customer identified features. ML was also used in fine tuning meta-heuristic configuration parameters with the goal of enhancing the overall solution of VRP. For example, Cooray and Rupasinghe (2017) suggested improving the impact of the mutation step of the genetic algorithm (GA) by applying K-means clustering technique. Results of this approach on a variant of VRP (EMVRP) showed some improvement in the overall solution quality compared to untuned GA. To the best of our knowledge, data-driven local search operators were previously used only by Belhaiza (2018a) for the dial-a-ride problem with time windows DARPTW. The DARPTW is a variant of the VRP with time windows, maximum duration and maximum transit time constraints, and where every customer has two different stops: a pick-up and a drop. In the DARPTW context, four kinds of DARPTW infeasibilities were defined. Three of these relate to the non-satisfaction of the pick-up and drop precedence constraints proper to the DARPTW. The local search phase is guided by the infeasibility types of the routes to prioritize the selection of operators. Nazari et al. (2018) developed a deep reinforcement learning based model to solve the classical VRP. Their approach consists of training a single model to search for a near-optimal solution by considering a sampled set of instances while monitoring some cost metrics and within VRP problem constraints. Simulation results showed that this approach outperforms traditional heuristics in terms of quality and execution time.

3 Diversification strategies

In this paper, the three algorithms we propose avoid cycling via a *multiple restart*. They also hybridize VNS with *genetic crossovers*, *ruin-and-recreate* and *tabu search* that ensure diversification. The combination of these different diversification strategies guards against GA's premature convergence by applying a *ruin-and-recreate* procedure, and against VNS' stagnation in a large cluster of attraction basins or confinement in a wide basin. In summary, they benefit from the flexibility of VNS and the strengths of both data-driven local search and multiple restart techniques. Section 3.1 presents the *Restart* strategy. Section 3.2 represents the *Ruin-and-Recreate* strategy. Section 3.3 presents the *Genetic Crossovers*. Section 3.4 presents the *Shaking* procedure. Section 3.5 presents the *Tabu Pool* strategy.

3.1 Restart

The pool of best solutions L is updated in a first-in first-out manner, every time a new incumbent is found. This allows L to evolve during the local search phase. The *restart* resets the current solution X to the first entry of L. The pool is reasonably large to avoid revisiting the same solutions and to avoid cycling. It is also reasonably small to allow the restoration of some good solutions and the exploration of neighborhoods that are closer to the global optimum.

3.2 Ruin-and-recreate

The *ruin-and-recreate* modifies the current solution X. It removes from the route of each vehicle k a number n'_k of customers randomly selected between $\lceil |I_k|/3 \rceil$ and $\lfloor 2|I_k|/3 \rfloor$. It then inserts unassigned customers to the vehicles iteratively.

3.3 Genetic crossovers

The *genetic crossovers* use the pool L of the last ψ best solutions found during the local search as their initial population. They apply a single-parent and a two-parent crossover. The single-parent crossover selects a solution from L, which becomes the parent. For each route k of this parent, this crossover makes the offspring inherit a sequence of ξ customers of one of its parent's vehicle k, where ξ is randomly generated from $\{1, \ldots, |I_k|\}$, as Fig. 1 illustrates. From an evolutionary perspective, the mutation ratio of each route k equals $(|I_k| - \xi)/|I_k|$. The dashed arcs show that all customers in the sequence i to $i + \xi$ are inherited by the offspring. The two-parent crossover selects a pair of parents from L to generate an offspring. For each vehicle k, the offspring inherits the clients assigned to the same vehicle k in both parents.

Regardless of its type, the crossover completes the offspring using insertion operators. It then subjects the offspring to the backward slack time BST fitting procedure (Belhaiza et al. 2014), which tries to make it feasible while minimizing the total waiting time. The offspring is then subject to mutation, which alters its quality and fitness.

3.4 Shaking

The shaking phase selects a solution X'' from the neighborhood $N_\kappa(X)$, where κ is chosen from the discrete uniform distribution on $[1, \bar{\kappa}]$ and $N_\kappa(X)$ is centered around X. The $\bar{\kappa}$ neighborhoods cover an important part of the search space. They perturb portions of the current solution and maintain other potions, with size varying according to the neighborhood. Since these perturbations involve exchanging segments of routes, the length of the exchanged segments determines the size of the neighborhood. Specifically, we use $\bar{\kappa} = 5$ neighborhood types:

- N_1 swaps two segments from two non-empty routes.
- N_2 swaps two segments from two non-empty routes and swaps the orientation of each segment.

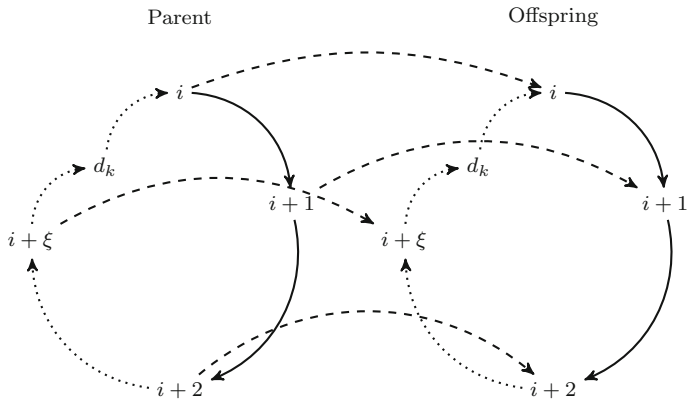

Fig. 1 Single-parent crossover

- N_3 swaps a pair of customers within a single non-empty route.
- N_4 swaps a pair of customers between two non-empty routes.
- N_5 swaps two segments between two non-empty routes, and randomly inverts one of them.

3.5 Tabu pool

The assignment of a customer to a given vehicle is the most basic representation of a gene. The genes of a solution encode its characteristics. A move to a routing solution X' with exactly the same set of genes is approved only if it decreases the overall best solution cost. The move is rejected otherwise. This principle accounts for the genetic characteristics of the routing solution. The pool of best solutions L is maintained using a first-in-first-out mechanism, which makes it subject to an evolution during the execution of the algorithm.

4 Three heuristic algorithms

This section presents our three heuristics and the three VRPMTW infeasibility types. We refer to the three multi-start data-driven evolutionary heuristics by MSDEH1, MSDEH2 and MSDEH3, respectively. Section 4.1 presents MSDEH1, the hybrid genetic data-driven variable neighborhood search heuristic. Section 4.2 presents MSDEH2, the hybrid data-driven variable neighborhood tabu search heuristic. Finally, Sect. 4.3 presents MSDEH3, the hybrid genetic data-driven variable neighborhood tabu search heuristic.

4.1 MSDEH1

MSDEH1 is a hybrid genetic heuristic that uses evolutionary crossovers within a data-driven variable neighborhood search framework. In contrast to MSDEH2 and MSDEH3, MSDEH1 does not use a *tabu pool* of best solutions. MSDEH1 improves

upon HGVNS (Belhaiza et al. 2017) as it uses data-driven local search operators, and relies on *ruin-and-recreate* as a diversification strategy. *Restart*, *ruin-and-recreate*, and *genetic crossover* are the three diversification strategies used by MSDEH1.

4.2 MSDEH2

MSDEH2 is a hybrid heuristic that uses data-driven variable neighborhood search enhanced with a tabu pool of best solutions. In contrast to MSDEH1 and MSDEH3, MSDEH2 does not rely on *genetic crossovers* as diversification strategy. MSDEH2 improves upon HVNTS (Belhaiza et al. 2014) as it uses *ruin-and-recreate* as a diversification strategy as well as data-driven local search operators guided by VRPMTW infeasibilities.

4.3 MSDEH3

MSDEH3 is a hybrid genetic heuristic that uses evolutionary crossovers and tabu search within a data-driven variable neighborhood search framework. In contrast to MSDEH1, MSDEH3 uses a *tabu pool* of best solutions.

5 VRPMTW infeasibilities and algorithmic steps

The data-driven local search operators address three types of VRPMTW infeasibilities. Type 1 does not involve any route. It is the most basic form of infeasibility consisting of a customer being left unserved. Type 2 relates to the capacity of a vehicle being exceeded. Type 3 relates to customers being served outside of all their time windows.

The proposed three heuristics generate a near global minimum X^* of fitness $f(X^*)$. The fitness $f(X)$ of X balances the value $h(X)$ of the objective function with the weighted penalties on the violations $\boldsymbol{\upsilon}$ of the time windows and $\boldsymbol{\varsigma}$ of the overload of the vehicles. The non-satisfaction of the time windows of client i is measured as $\upsilon_i = \max\{0, \min_{p_i=1,\ldots,\bar{p}_i}\{|a_i - l_{p_i}|, |a_i - u_{p_i}|\}\}$, where a_i is the starting time of delivery at customer i. The overload of the capacity of vehicle k is the positive surplus $\varsigma_k = \max\{0, \sum_{i\in\mathcal{I}_k} q_i - Q_k\}$. The violations υ_i and ς_k engender penalties $\theta\upsilon_i^\mu$, and $\rho\varsigma_k^\mu$, where θ, and ρ are the weights, and μ is an exponent. Setting $\mu > 1$ prohibits large violations.

Evidently, a penalty is strictly positive when its associated constraints are unsatisfied. Otherwise, it is set to zero. It follows that

$$f(X) = h(X) + \theta \sum_{i\in\mathcal{I}} \min_{p\in W_i} \upsilon_i^\mu + \rho \sum_{k\in\mathcal{K}} \varsigma_k^\mu. \tag{1}$$

Algorithm 1 provides the pseudo code of a standard MSDEH. Section 5.1 describes the initialization and insertion steps. Section 5.2 explains the iterative loop. Finally, Sect. 5.3 details the local search.

5.1 Initialization and insertion

The initialization step constructs an initial solution to the problem. Line 1 of Algorithm 1 sets both the counter ι of non-improving iterations and the elapsed runtime t to 0. Line 2 constructs an initial solution X as follows. It randomly assigns to a vehicle $k \in \mathcal{K}$ as many unassigned customers as the capacity Q_k allows. It then assigns each of the randomly selected customers to the positions that yield the least increase of the route cost among all possible positions in the route of k. Solution X does not necessarily serve all customers, nor does it ensure that deliveries occur during one of the customers' time windows.

When the initialization step generates a solution that leaves some customers unassigned, insertion is called before every iteration of the iterative loop. The insertion phase tries to insert every unassigned customer in every possible position of every route. If this phase succeeds in positioning the customer, the insertion is adopted; otherwise, the customer remains unassigned. In this way, the insertion reduces type 1 infeasibilities. If not, the insertion is to be tried again. Line 3 computes $f(X)$, the fitness of X. Finally, Line 4 initializes the incumbent X^* to X and the best fitness value $f(X^*)$ to $f(X)$.

5.2 Iterative loop

The three heuristic algorithms share the same structure. Hence, they are detailed in the single Algorithm pseudo-code 1. The iterative loop (Lines 5–26) stops when t exceeds a preset threshold time limit $\bar{t}$. It applies a steepest descent data-driven local search (Line 6) to X to obtain a solution X'. Consequently, two cases may rise: $f(X') < f(X)$ (i.e., X' improves X) and $Move(X') = True$, or the local search fails to improve X or $Move(X') = False$. In the former case, Lines 7–11 apply. Line 8 changes the focal point of the search to X' setting the current solution $X = X'$. Furthermore, when this new local minimum X' improves upon the incumbent X^*, Line 10 updates X^* and $f(X^*)$, and resets ι to 0 signaling a new incumbent. In the latter case, $Move(X') = False$ might be due to X' being part of the tabu pool L in MSDEH2 and MSDEH3, Lines 12–25 apply. Line 13 increments ι, the counter of non-improving trials. Consequently, when ι is not a multiple of $\bar{\nu}$, $\bar{\upsilon}$, and $\bar{\eta}$, Lines 21–23 apply a shaking procedure (Lines 21–23); otherwise, the algorithm calls one of three diversification strategies: restart when $\iota = 0(\bar{\nu})$ in the three heuristics, ruin and recreate when $\iota = 0(\bar{\upsilon})$ in the three heuristics, and GA when $\iota = 0(\bar{\eta})$ in MSDEH1 and MSDEH2, where $1 < \bar{\nu} < \bar{\upsilon} < \bar{\eta}$, and $a = r(b)$ means that $r \in \{0, 1, \ldots, b-1\}$ is the remainder of the division of a by b, with a, b, r all integer. The application of the diversification strategy to X yields a mutant X''. Finally, Line 24 resets X to X''. The updated X serves as the initial solution for the next iteration.

5.3 Data-driven local search

The steepest descent phase involves regular and data-driven customized local operators. Regular local search operators have been extensively detailed in the VRP literature

Algorithm 1 MSDEH Algorithms' Pseudo code.

Set $\iota = 0$, and $t = 0$.
Apply assignment and insertion operators to X.
Compute $f(X)$.
Set $X^* = X$ and $f(X^*) = f(X)$.
while $t < \bar{t}$ **do**
  Apply local search to get X'.
  if $Move(X') = True$, **then**
    set $X = X'$, and $f(X) = f(X')$;
    if $f(X') < f(X^*)$, **then**
      set $X^* = X'$, $f(X^*) = f(X')$, and $\iota = 0$;
    end if
  else
    set $\iota = \iota + 1$;
    if $\iota = 0(\bar{\eta})$, **then**
      apply genetic crossover to get a mutant X";
    else if $\iota = 0(\bar{\upsilon})$, **then**
      apply *ruin and recreate* to X to get X";
    else if $\iota = 0(\bar{\nu})$, **then**
      apply *restart* to get a solution X";
    else
      randomly generate κ from $\{1, \ldots, \bar{\kappa}\}$;
      apply shaking to X to get $X" \in N_\kappa(X)$;
    end if
    set X to X";
  end if
end while

(Toth and Vigo 2014). Therefore, this section only details the new data-driven local search operators we propose. The intuition behind these operators is that routes with infeasibilities could be avoided or repaired instead of being overlooked during the local search phase. The data-driven operators are based on the VRPMTW infeasibilities of types 2 and 3. A data-driven local search operator returns X', the first solution that improves X among all neighbors and maintains or reduces the total number of VRPMTW infeasibilities.

VRPMTW infeasibilities of type 2 deal with maximum capacity conditions. The satisfaction, or not, of these conditions is easy to assess during the local search phase. It is only a matter of computing the net variation of the used capacity of each route, and verifying that the maximum vehicle capacity is not exceeded. However, VRPMTW infeasibilities of type 3, which deal with time windows conditions, are much more complicated to handle. In the following, we present three short algorithms that need to be applied whenever a data-driven local search move is to be performed.

For a given route visit schedule, the first algorithm resets the customers' arrival times to their possible earliest values. The second algorithm computes the min-max forward slack time allowable for a given section of a route to remain feasible with respect to the time windows' conditions. The third algorithm computes the max-min backward slack time necessary for a given section of a route to become feasible with respect to the time windows' conditions.

5.3.1 Earliest arrival times

The BST fitting procedure (Belhaiza et al. 2014) makes the route visit schedule very tight to minimize its whole duration. While loose visit schedules allow feasible insertions and relocations, tight ones do not offer much flexibility. To assess the VRPMTW infeasibilities of type 3, the arrival times at all the customers of a given route should be reset to their earliest possible time. The main reason is that the earliest arrival times guarantee the largest forward slack times in the case of an insertion. Algorithm 2 provides the pseudo-code of arrival times reset procedure which simply resets the visit schedule to its earliest configuration. As usual, the customers' time windows are assumed to be pre-sorted in increasing order, based on the lower bound of each time window. For a given route k, Algorithm 2 initializes the departure from the depot a_0 at it earliest possible value given by the route time window lower bound l_0. It then computes the arrival time a_i at each customer i as the sum of the arrival time a_{i-1}, the service time s_{i-1}, and the travel time $t_{i-1,i}$. If the arrival time a_i is found to be less than at least one time window lower bound l_i^p, a waiting time w_i^p is added and the arrival time a_i is set to l_i^p. It is understood that all waiting times w_i^p are initialized at 0.

Algorithm 2 Algorithm 2 ResetArrivalTimes(k).

```
1: Set a_0 ← l_0;
2: for i = 1, ..., n_k do
3:     w_i^p ← 0;
4:     for p = 1, ..., p̄_i do
5:         if a_i < l_i^p then
6:             w_i^p ← l_i^p − a_i;
7:             a_i ← l_i^p;
8:             break;
9:         end if;
10:    end for;
11: end for;
12: return;
```

5.3.2 Forward slack times

The flexibility of the visit schedule with respect to insertions is assessed using one single parameter relatively to each of the customers' time windows. For a given customer i, we define $\bar{F}_i^p$ as the maximum allowable forward slack time needed to realize the feasibility of the arrival time at i with respect to her time window p. Algorithm 3 provides the pseudo-code of the forward maximum slack times computation. For a given route k, and a given customer index j, Algorithm 3 computes the maximum forward slack time for every customer i in the sequence of visits, noted $S_{j,l}^k$, from j to $l \leq n_k$. If the arrival time $a_i \leq u_i^p$, the maximum forward slack time $\bar{F}_i^p$ equals the difference between the upper bound u_i^p and the arrival time a_i, plus the waiting time w_i at i. If the arrival time $a_i > u_i^p$, the maximum forward slack time $\bar{F}_i^p$ is set to 0.

The maximum value among all maximum allowable forward slack times of a given customer $\bar{F}_i$, initialized to 0, indicates the maximum allowable stretch of the arrival time at i before it becomes infeasible with respect to the time windows of i. For each customer i, her maximum $\bar{F}_i$ is updated, if $\bar{F}_i < \bar{F}_i^p$. Finally, the minimum value $\bar{F}_{j,l}^k$ among all $\bar{F}_i$, initialized to a large value M, indicates the maximum allowable stretch of all the arrival times in the sequence such that none of them becomes infeasible. For each customer i, if $\bar{F}_{j,l}^k > \bar{F}_i$, $\bar{F}_{j,l}^k$ is updated.

Algorithm 3 Algorithm 3 ForwardSlackTimes(k, j, l).

```
1: Set w_i ← 0;
2: Set F̄^k_{j,l} ← M;
3: for i = j, ..., l do
4:     Set F̄_i ← 0;
5:     for p = 1, ..., p̄_i do
6:         w_i ← w_i + w_i^p;
7:         if a_i ≤ u_i^p then F̄_i^p ← u_i^p − a_i + w_i;
8:             if F̄_i < F̄_i^p then F̄_i ← F̄_i^p;
9:             end if;
10:        end if;
11:    end for;
12:    if F̄^k_{j,l} > F̄_i then F̄^k_{j,l} ← F̄_i;
13:    end if;
14: end for;
15: return;
```

Proposition 1 shows that any insertion of a customer, immediately before $S_{j,l}^k$, that does not cause a delay strictly larger than $F_{j,l}^k$, does not increase the number of VRPMTW infeasibilities of type 3 for the sequence $S_{j,l}^k$.

Proposition 1 *Let N_j^k be a set of ordered customers to be inserted immediately before section $S_{j,l}^k$. Let $\Delta_{j,l}^k \geq 0$ be the minimum delay in serving $S_{j,l}^k$ caused by the insertion of N_j^k. If $\Delta_{j,l}^k \leq F_{j,l}^k$, then the number of type 3 infeasibilities for the sequence $S_{j,l}^k$ remains the same.*

Proof Let w_i be the waiting time when i is reached. Two cases arise:

- If $w_i = 0$, delaying the service of the sequence $S_{j,l}^k$ yields a potential arrival time at i equal to $\Delta_{j,l}^k + a_i \leq F_{j,l}^k + a_i \leq F_i + a_i$. If p^* is its time window such that $F_i = F_i^{p^*} = \max_p F_i^p$, then $F_i + a_i = F_i^{p^*} + a_i = u_i^{p^*}$, yields $\Delta_{j,l}^k + a_i \leq u_i^{p^*}$. The arrival time at customer i is feasible with respect to its time window p^*. The number of type 3 infeasibilities remains unchanged.
- If $w_i > 0$, $a_i - w_i$ indicates the real arrive time at i if no waiting time needs to be observed. Delaying the service of the sequence $S_{j,l}^k$ yields a potential arrival time at i equal to $\Delta_{j,l}^k + a_i - w_i \leq F_{j,l}^k + a_i - w_i \leq F_i + a_i - w_i$. If p^* is its time window such that $F_i = F_i^{p^*} = \max_p F_i^p$, we have $F_i^{p^*} = u_i^{p^*} - a_i + w_i$. Hence,

$F_i + a_i - w_i = F_i^{p^*} + a_i - w_i = u_i^{p^*}$, yields $\Delta_{j,l}^k + a_i - w_i \leq u_i^{p^*}$. The arrival time at customer i is feasible with respect to its time window p^*. The number of type 3 infeasibilities remains unchanged.

In the two cases, if i is not served during any of her time windows, $F_i = \max_p F_i^p = 0$. Therefore $F_j^k = \min_i F_i = 0$. Hence, $\Delta_{j,l}^k$ can only be equal to 0. The arrival time at customer i remains unfeasible with respect to all of her time windows. The number of type 3 infeasibilities remains unchanged. □

5.3.3 Backward slack times

The backward slack time for a given section of a route, with respect to each of the customers' time windows, is also assessed by means of a single parameter. For a given customer i, we define $\underline{B}_i^p$ as the minimum necessary backward slack time to make the arrival time at i feasible with respect to its time window p. Algorithm 4 provides the pseudo-code of the backward maximum slack times computation. For a given route k, and a given customer index j, Algorithm 4 computes all the minimum backward slack times necessary for every customer i in the sequence of visit from j to n_k. If the arrival time $a_i > u_i^p$, the minimum backward slack time $\underline{B}_i^p$ equals the difference between the arrival time a_i and the upper bound of time window p. If the arrival time $a_i \leq u_i^p$, the minimum backward slack time $\underline{B}_i^p$ is set to 0. The minimum value among all minimum necessary backward slack times of a given customer $\underline{B}_i$ indicates the minimum stretch required for the arrival time at i before it becomes feasible with respect to at least one of its time windows. For each customer i, her minimum $\underline{B}_i$ is updated, if $\underline{B}_i > \underline{B}_i^p$. Finally, the maximum value $\underline{B}_{j,l}^k$ among all $\underline{B}_i$ indicates the maximum stretch required of all the arrival times in the sequence $S_{j,l}^k$, such that none of them remains infeasible. For each customer i, if $\underline{B}_{j,l}^k < \underline{B}_i$, $\underline{B}_{j,l}^k$ is updated.

Algorithm 4 Algorithm 4 BackwardSlackTimes(k, j).

```
1: Set B_{j,l}^k ← 0;
2: for i = j, ..., l do
3:    Set B_i ← M;
4:    for p = 1, ..., p̄_i do
5:       if a_i > u_i^p then B_i^p ← a_i − u_i^p;
6:       else B_i^p ← 0;
7:       end if;
8:       if B_i > B_i^p then B_i ← B_i^p;
9:       end if;
10:   end for;
11:   if B_{j,l}^k < B_i then B_{j,l}^k ← B_i;
12:   end if;
13: end for;
14: return;
```

Proposition 2 shows that any ejection of a customer that causes a hastening strictly larger than $B^k_{j,l}$ decreases the number of VRPMTW infeasibilities of type 3 to zero for the sequence $S^k_{j,l}$.

Proposition 2 *Let N^k_j be a set of ordered customers to be ejected immediately before section $S^k_{j,l}$. Let $\Gamma^k_j \geq 0$ be the maximum hastening in serving $S^k_{j,l}$ caused by the ejection of N^k_j. If $\Gamma^k_{j,l} \geq B^k_{j,l}$, then the number of type 3 infeasibilities for the sequence $S^k_{j,l}$ is reduced to* 0.

Proof Since $\Gamma^k_{j,l} \geq B^k_{j,l} \geq B_i$, hastening the service of the sequence $S^k_{j,l}$ yields a potential arrival time at i equal to $a_i - \Gamma^k_{j,l} \leq a_i - B^k_{j,l} \leq a_i - B_i$. Two cases arise:

- If i is not served during any of its time windows, let p^* be the closest to a_i. Therefore, $B_i = \min_p B^p_i = B^{p^*}_i > 0$. Hence, $a_i - B_i = a_i - B^{p^*}_i = u^{p^*}_i$, yields $a_i - \Gamma^k_{j,l} \leq u^{p^*}_i$. The arrival time at customer i becomes feasible with respect to its time window p^*. The number of type 3 infeasibilities is reduced by one.
- If i is served during its time window p^*, $B_i = \min_p B^p_i = B^{p^*}_i = 0$. Therefore, $a_i - \Gamma^k_{j,l} \leq a_i - B^k_{j,l} \leq a_i - B_i = a_i \leq u^{p^*}_i$. The arrival time at customer i remains feasible with respect to its time window p^*. The number of type 3 infeasibilities remains the same.

Hence, the number of type 3 infeasibilities for the sequence S^k_j is reduced to 0. □

5.3.4 Data-driven single route operators

For every vehicle $k \in \mathcal{K}$, the data-driven single-route search applies, in increasing order of complexity, up to three customer-exchange data-driven operators: the data-driven relocate operator (DR), the data-driven node relocate operator (DNR), and the data-driven arc relocate operator (DAR). It applies DNR only when DR fails to obtain a lower-cost solution and decrease the total number of VRPMTW infeasibilities. Similarly, it uses DAR only when DNR fails to decrease the cost and the total number of VRPMTW infeasibilities of the incumbent. All customers are eligible to be relocated with the data-driven single route operators, in particular those with infeasibilities of type 3, because they are served outside of one their time windows. The data-driven single route operators can reduce the cost of a route and also make it feasible. Customers with an infeasibility of type 2 are also eligible to be relocated, but the relocation cannot repair their type 2 infeasibilities since it cannot reduce the used capacity if it exceeds Q_k.

Data-driven relocate (DR) moves a customer $i \in I_k$ to a different position along the same route. DR computes the cost ϵ_{ij} of moving i to every possible position $j \in I_k$. As illustrated in Fig. 2, moving customer i immediately after customer j alters the cost by $\epsilon_{ij} = t_{i-1,i} + t_{i,i+1} + t_{j,j+1} - t_{j,i} - t_{i,j+1} - t_{i-1,i+1}$. This change consists of subtracting the cost of arcs $(i-1, i)$, $(i, i+1)$ and $(j, j+1)$ (dotted) and adding the costs of arcs (j, i), $(i, j+1)$ and $(i-1, i+1)$ (dashed). The type 3 infeasibility of

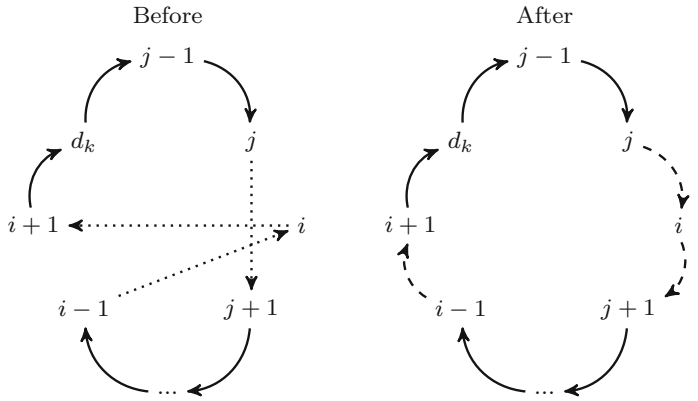

Fig. 2 Data-driven relocate (DR)

route k may be avoided or repaired under some conditions. If i is moved upwards in route k, one needs to verify that i can be served in at least one of its time windows, and the sequence of customers from $j+1$ to $i-1$ can be moved forward with all customers being eligible to be served in at least one of their time windows, respectively.

DR finds the best position j^* of i such that $\epsilon_{ij^*} = \max_{j' \in I_k}\{\epsilon_{ij'}\} > 0$, and the total number of infeasibilities in route k is maintained or reduced. Firstly, the expected arrival time at i, after applying the ResetArrivalTimes() Algorithm 2 needs to be feasible with at least one of its time windows: $a_j + s_j + t_{j,i} \leq u_i^p$, for at least one $p \in 1, \ldots, p_i$. Secondly, $\Delta^k_{j+1,i-1} = t_{j,i} + s_i + t_{i,j+1}$ which represents the delay in serving the sequence $S^k_{j+1,i-1}$, should be less or equal to $F^k_{j+1,i-1}$. If these conditions are satisfied, DR inserts i route k, just after j^*.

Data-driven node relocate (DNR) swaps the positions of two customers i and j of I_k if this reduces the total cost of the route, and reduces or maintains the total number of infeasibilities of route k. DNR computes, for every pair (i, j), $i \in I_k$, $j \in I_k$, $i \neq j$, the cost saving $\epsilon_{ij} = t_{i-1,i} + t_{i,i+1} + t_{j-1,j} + t_{j,j+1} - t_{i-1,j} - t_{j,i+1} - t_{j-1,i} - t_{i,j+1}$. As Fig. 3 shows, swapping i and j requires subtracting the costs of arcs $(i-1, i)$, $(i, i+1)$, $(j-1, j)$ and $(j, j+1)$ (dotted) and adding those of arcs $(i-1, j)$, $(j, i+1)$, $(j-1, i)$ and $(i, j+1)$ (dashed). Two cases arise. In the first case, $t_{i-1,j} + s_j + t_{j,i+1} < t_{i-1,i} + s_i + t_{i,i+1}$, the move results in the hastening of the service of the sequence $S^k_{i+1,j-1}$, as $\Gamma^k_{i+1,j-1} > 0$. DNR is performed only if $\Gamma^k_{i+1,j-1} \geq B^k_{i+1,j-1}$, and i could be served within one of its time windows. The second case where $t_{i-1,j} + s_j + t_{j,i+1} \geq t_{i-1,i} + s_i + t_{i,i+1}$, the move results in the delay of the service of the sequence $S^k_{i+1,j-1}$, as $\Delta^k_{i+1,j-1} > 0$. DNR is performed only if $\Delta^k_{i+1,j-1} \leq F^k_{i+1,j-1}$, and i could be served within one of its time windows. DNR swaps the pair of customers i and j with the largest ϵ_{ij}. Finally, since $\epsilon_{ij} > 0$, the sequence S^k_{j+1,n_k} does not incur any risk of having its number of type 3 VRPMTW infeasibilities increase.

DAR swaps the positions of two selected arcs $(i, i+1)$ and $(j, j+1)$ on the same route if this reduces the total cost of the route, and reduces or maintains the total number

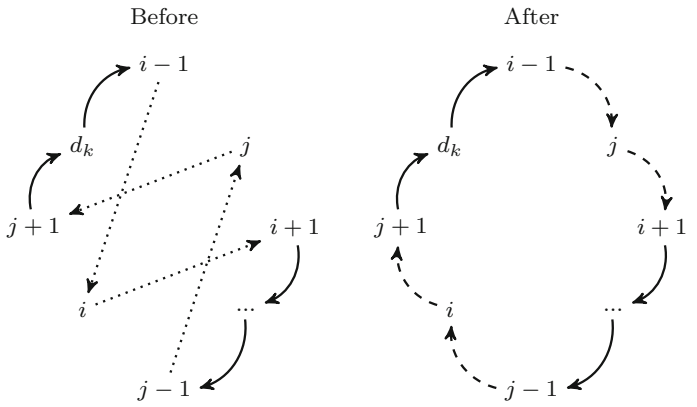

Fig. 3 Data-driven node relocate (DNR)

of infeasibilities. It first computes for all pairs of non-connected arcs in a route the saved cost $\epsilon_{ij} = t_{i-1,i} + t_{i+1,i+2} + t_{j-1,j} + t_{j+1,j+2} - t_{i-1,j} - t_{j+1,i+2} - t_{j-1,i} - t_{i+1,j+2}$. As Fig. 4 shows, swapping arcs $(i, i+1)$ and $(j, j+1)$ requires subtracting the costs of $(i-1, i)$, $(i+1, i+2)$, $(j-1, j)$ and $(j+1, j+2)$ (dotted) and adding those of $(i-1, j)$, $(j+1, i+2)$, $(j-1, i)$ and $(i+1, j+2)$ (dashed). Two cases arise. In the first case, $t_{i-1,j} + s_j + t_{j,j+1} + s_{j+1} + t_{j+1,i+2} < t_{i-1,i} + s_i + t_{i,i+1} + s_{i+1} + t_{i+1,i+2}$, the move results in the hastening of the service of the sequence $S^k_{i+2,j-1}$, as $\Gamma^k_{i+2,j-1} > 0$. DAR is performed only if $\Gamma^k_{i+2,j-1} \geq B^k_{i+2,j-1}$, and both i and $i+1$ could be served within one of their time windows. The second case where $t_{i-1,j} + s_j + t_{j,j+1} + s_{j+1} + t_{j+1,i+2} \geq t_{i-1,i} + s_i + t_{i,i+1} + s_{i+1} + t_{i+1,i+2}$, the move results in the delay of the service of the sequence $S^k_{i+2,j-1}$, as $\Delta^k_{i+2,j-1} > 0$. DAR is performed only if $\Delta^k_{i+2,j-1} \leq F^k_{i+2,j-1}$, and i and $i+1$ could be served within one of their time windows. DAR swaps the pair of arcs $(i, i+1)$ and $(j, j+1)$ with the largest ϵ_{ij}. Finally, since $\epsilon_{ij} > 0$, the sequence S^k_{j+2,n_k} does not incur any risk of having its number of type 3 VRPMTW infeasibilities increase.

5.3.5 Data-driven multiple route operators

The multiple-route improvement alters the routes of two vehicles with three operators: data-driven inter-route relocate (DIRR), data-driven inter-route swap (DIRS) and data-driven inter-route cross (DIRC). All customers are eligible to be relocated with the data-driven multiple route operators, in particular those with type 2 and type 3 infeasibilities. The data-driven multiple route operators can reduce the cost of a pair of routes, and also make them feasible with respect to the used capacity or the customers' time windows.

In the case of a type 2 infeasibility, if Q_k is exceeded by a quantity less than q_i, DIRR ejects customer i from route k and inserts it in a selected position j on a route k' such that the total number of infeasibilities in not increased. It first computes the cost saving $\epsilon^{kk'}_{ij} = t_{i-1,i} + t_{i,i+1} + t_{j,j+1} - t_{i-1,i+1} - t_{j,i} - t_{i,j+1}$. As Fig. 5 suggests, moving $i \in I_k$ immediately after customer $j \in I'_k$ involves subtracting the costs of

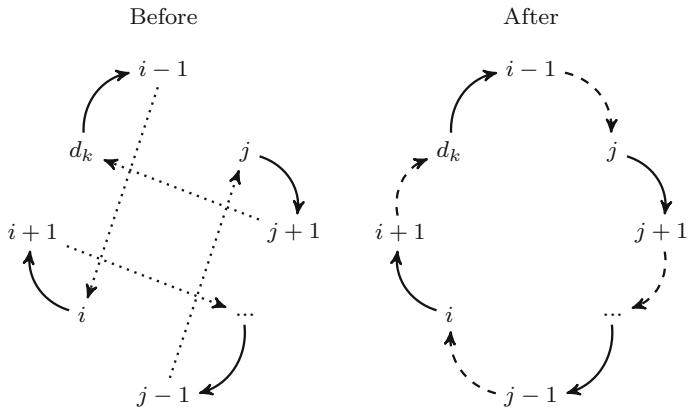

Fig. 4 Data-driven arc relocate (DAR)

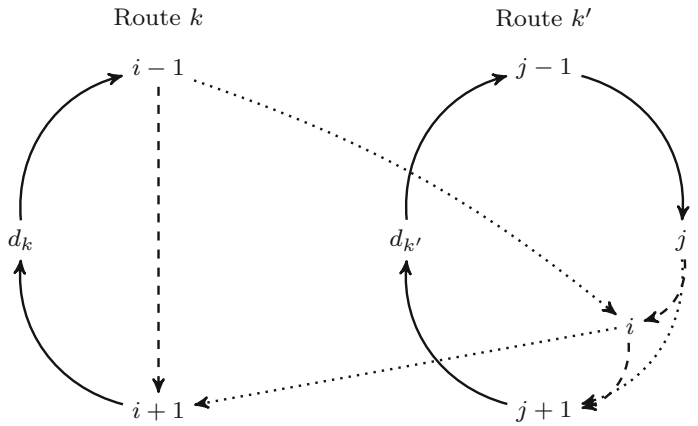

Fig. 5 Data-driven inter-route relocate (DIRR)

$(i-1,i)$, $(i,i+1)$ and $(j,j+1)$ (dotted) and adding those of $(i-1,i+1)$, (j,i) and $(i,j+1)$ (dashed). The move results in the delay of the service of the sequence $S^{k'}_{j+1,n_{k'}}$ by $\Delta^{k'}_{j+1,n_{k'}} > 0$. DIRR is performed only if $\Delta^{k'}_{j+1,n_{k'}} \leq F^{k'}_{j+1,n_{k'}} 0$ and i could be served within one of its time windows. DIRR chooses the position j that yields the largest $\epsilon^{kk'}_{ij}$, while maintaining or reducing the total number of infeasibilities.

In the case of a type 2 infeasibilities, if Q_k, or Q'_k, is exceeded by a quantity less than q_i, or $q_{i'}$, DIRS ejects customers i, in route k, and $i' \in I_{k'}$, in route k', and swaps their positions if the expected saved cost is the best among all possible similar moves, and the total number of VRPMTW infeasibilities is maintained or reduced. As Fig. 6 shows, exchanging customer $i \in I_k$ with customer $i' \in I'_k$ alters the cost by $\epsilon^{kk'}_{ii'} = t_{i-1,i} + t_{i,i+1} + t_{i'-1,i'} + t_{i',i'+1} - t_{i-1,i'} - t_{i',i+1} - t_{i'-1,i} - t_{i,i'+1}$ as it requires subtracting the costs of $(i-1,i)$, $(i,i+1)$, $(i'-1,i')$ and $(i',i'+1)$ (dotted) and adding those of $(i-1,i')$, $(i',i+1)$, $(i'-1,i)$ and $(i,i'+1)$ (dashed). Four cases may

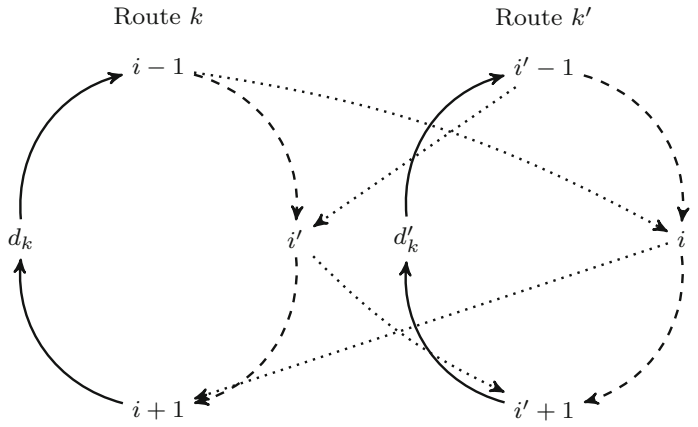

Fig. 6 Data-driven inter-route swap (DIRS)

arise. We detail the case where $t_{i-1,i'} + s_{i'} + t_{i',i+1} < t_{i-1,i} + s_i + t_{i,i+1}$. The move results in the hastening of the service of the sequence S^k_{i+1,n_k}, and the delay of the sequence $S^{k'}_{i'+1,n_{k'}}$, as $\Delta^{k'}_{i'+1,n_{k'}} > 0$. DIRS is performed only if $\Delta^{k'}_{i'+1,n_{k'}} \leq F^{k'}_{i'+1,n_{k'}}$, and both i and i' can be served within one of their time windows. The two cases where at least one of the routes has a delayed sequence, could be deduced from the case detailed. The last case where both routes have hastened sequences is trivial. DIRS is performed only if i and i' could be served during one of their time windows. DIRS computes the cost saving for all given pairs of customers of two routes, and swaps the pair with the largest $\epsilon^{kk'}_{ii'}$.

DIRC exchanges two arcs of a pair of routes if the expected saved cost is the largest among all possible similar moves, and the total number of VRPMTW infeasibilities is maintained or reduced. As Fig. 7 indicates, swapping $(i, i+1)$ from route k with $(i', i'+1)$ of route k' changes the cost by $\epsilon^{kk'}_{ii'} = t_{i-1,i} + t_{i+1,i+2} + t_{i'-1,i'} + t_{i'+1,i'+2} - t_{i-1,i'} - t_{i'+1,i+2} - t_{i'-1,i} - t_{i+1,i'+2}$ as it requires subtracting the costs of $(i-1, i)$, $(i+1, i+2)$, $(i'-1, i')$ (dotted) and $(i'+1, i'+2)$ and adding those of $(i-1, i')$, $(i'+1, i+2)$, $(i'-1, i)$ and $(i+1, i'+2)$ (dashed). Four cases may arise, particularly the case where $t_{i-1,i'} + s_{i'} + t_{i',i'+1} + s_{i'+1} + t_{i'+1,i+2} < t_{i'-1,i} + s_i + t_{i,i+1} + s_{i+1} + t_{i+1,i'+2}$. The move results in the hastening of the service of the sequence S^k_{i+2,n_k}, and the delay of the sequence $S^{k'}_{i'+2,n_{k'}}$, as $\Delta^{k'}_{i'+2,n_{k'}} > 0$,. DIRC is performed only if $\Delta^{k'}_{i'+2,n_{k'}} \leq F^{k'}_{i'+2,n_{k'}}$, and customers i, $i+1$, i' and $i'+1$ could be served within one of their time windows. The remaining cases can easily be deduced. DIRC computes $\epsilon^{kk'}_{ii'}$ for all pairs of arcs in two distinct routes and swaps the arcs with the largest $\epsilon^{kk'}_{ii'}$.

6 Computational results

The algorithms were implemented in C++ under MS Visual. The computational results were obtained on workstations with 3.3 GHz Intel Core i7 vPro processors, and 3.2

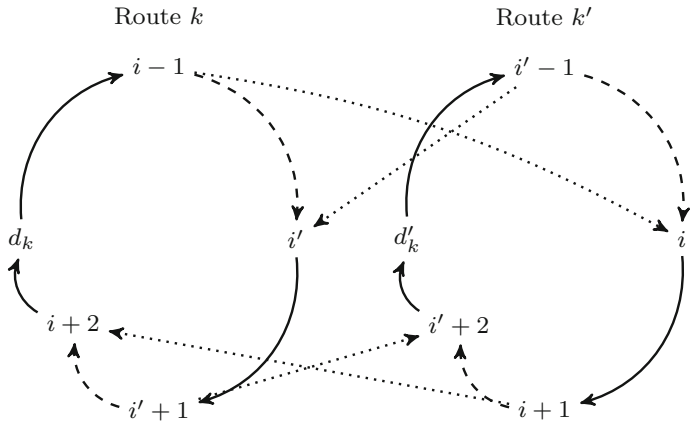

Fig. 7 Data-driven inter-route cross (DIRC)

GB RAM. The objective of the computational investigation is to motivate the common design of the three MSDEH algorithms, and to test their global performance on two sets of benchmark and real-life instances.

6.1 Tuning of MSDEH

To tune the parameters of MSDEH, we consider different designs and choose the ones that consistently yield the best performance on six representative instances (rm101, rm201, cm101, cm201, rcm101, and rcm201) of the VRPMTW benchmark set (Belhaiza et al. 2014). The designs correspond to

- two levels for the size of the tabu list $|L| = 0,\ 30$;
- four levels for *ruin and recreate*: $\bar{\upsilon} = 2500,\ 5000,\ 7500,\ 10{,}000$; and
- five levels for *genetic crossover*: $\bar{\eta} = \infty$ (when not applied), and $\bar{\eta} = 2500,\ 5000,\ 7500,\ 10{,}000$,

such that $\bar{\upsilon} \neq \bar{\eta}$. Every instance is replicated 30 times, with $\bar{t} = 100$ s, for each variant of MSDEH. The same seed is used for the same replication of the different designs. We test this large number of variants for the minimization of total duration h_2, which already accounts for h_1.

Table 2 reports the average $\bar{h}_2$ and minimum h_2^*, in time units, of the best feasible solutions along with the average runtime $t_{\bar{h}_2}$, in seconds, over all replications. It reveals that the best order of the restart strategies, is the one given in Algorithm 1, and that the best combinations of parameters are those where $|L| = 30$, $\bar{\eta} \in \{7500,\ 10{,}000\}$, $\bar{\upsilon} \in \{2500,\ 5000\}$, and $\bar{\nu} = 500$. In fact, these combinations of parameters yield the best averages, and improve many best known solution values. They provide computational evidence of the merit of the diversification approaches, paired with a tabu list of best solutions.

Table 2 Tuning of versions 1, 2 and 3 of MSDEH

$\|L\|$		$\bar{\eta}$	$\bar{\upsilon} = 2500$			$\bar{\upsilon} = 5000$			$\bar{\upsilon} = 7500$			$\bar{\upsilon} = 10{,}000$		
			$\bar{h}_2$	$t_{\bar{h}_2}$	h_2^*	$\bar{h}_2$	$t_{\bar{h}_2}$	h_2^*	$\bar{h}_2$	$t_{\bar{h}_2}$	h_2^*	$\bar{h}_2$	$t_{\bar{h}_2}$	h_2^*
1	0	2500				4778.2	46.4	4713.9	4775.2	44.3	4710.6	4775.4	47.5	4709.0
1	0	5000	4773.8	42.7	4712.9				4767.3	51.2	4713.6	**4765.0**	**50.4**	**4709.5**
1	0	7500	4771.3	47.4	4708.9	4766.8	51.1	4704.7				4765.8	49.7	4709.0
1	0	10,000	4771.1	46.2	4706.5	4767.8	51.4	4712.5	4766.6	51.6	4707.0			
2	30	∞	4760.8	72.3	4705.2	**4758.4**	**61.7**	**4699.1**	4763.3	69.2	4704.7	4762.4	69.5	4703.7
3	30	2500				4762.9	65.6	4699.1	4762.7	66.5	4696.1	4762.1	71.0	4702.2
3	30	5000	4765.7	68.7	4701.2				4764.0	68.6	4701.6	4764.8	68.5	4697.3
3	30	7500	**4761.0**	**71.1**	**4701.8**	**4759.9**	**66.5**	**4702.6**				4763.6	69.9	4700.6
3	30	10,000	**4760.6**	**69.7**	**4695.2**	**4760.8**	**68.9**	**4695.3**	4763.1	66.3	4703.2			

Bold values represent the best-known solution variable

Table 3 MSDEH3 Results on VRPTW instances

Class	n	PR(10)	PDR(5)	RTI(3)	NB-100(1)	NB-f_n(5)	HGSADC(5)	MSDEH3(10)
R1	100	1212.39	**1210.34**	1210.82	**1210.34**	**1210.34**	1210.69	1210.40
R2	100	957.72	955.74	952.67	952.08	**951.03**	951.51	952.30
C1	100	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**	**828.38**
C2	100	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**	**589.86**
RC1	100	1385.78	**1384.16**	1384.30	**1384.16**	**1384.16**	1384.17	1384.17
RC2	100	1123.49	1119.44	1119.72	**1119.24**	**1119.24**	**1119.24**	1119.71
	Avg.	1016.27	1014.65	1014.29	1014.01	**1013.84**	1013.98	1014.13

Bold values represent the best-known solution variable

6.2 Performance of MSDEH3

Hereafter, we use the MSDEH3 version of MSDEH, implemented with $|L| = 30$, $\bar{\eta} = 7500$, $\bar{\upsilon} = 5000$, and $\bar{\nu} = 500$. This version is tested on 56 homogeneous fleet VRPTW benchmark instances (Solomon 1983), 48 homogeneous fleet VRPMTW benchmark instances (Belhaiza et al. 2014) and on 25 real-life heterogeneous fleet VRPMTW instances.

The 56 VRPTW benchmark instances have $n = 100$ customers, randomly distributed (R1) and (R2), clustered (C1) and (C2), and randomly clustered (RC1) and (RC2). Each customer has a single time windows. The cost $h(X) = h_1(X)$, only includes the travel times. Table 3 summarizes and compares the best solution values, in distance units, obtained by MSDEH3 with the best solution values of six well known algorithms for the VRPTW:

- PR(10): ten runs of the adaptive large neighborhood search by Pisinger and Ropke (2007);
- PDR(5): five runs of the branch-and-price based large neighborhood search by Prescott-Gagnon et al. (2009);

- RTI(3): three runs of the arc-guided evolutionary algorithm by Repoussis et al. (2009);
- NB-100(1) and NB-f_n(5): one run and five runs of the hybrid GA based crossover of Nagata et al. (2010) with population of 100 and $f_n = n/20000$.
- HGSADC(5): five runs of the hybrid genetic with adaptive diversity management of Vidal et al. (2013a, b).

Table 3 shows that MSDEH3 competes very well with these algorithms as it stands very close to the best ones NB-100, NB-f_n and HGSADC with a gap not exceeding 0.03% with respect to the best one, i.e., NB-f_n.

The 48 benchmark instances have $n = 100$ customers, randomly distributed (rm), clustered (cm) and randomly clustered (rcm). For each of these classes, there are two classes of time windows: '1' for narrow and '2' for wide. Each customer has one to 10, more or less tight, non-overlapping time windows, as detailed in Belhaiza et al. (2014). The fixed cost is proportional to the vehicle capacity. It is equal to 200 for class '1' and to 1000, 700, and 1000 for $rm2$, $cm2$, and $rcm2$, respectively. The cost, in time units, $h(X) = h_2(X)$, which includes the travel, service and waiting times.

Tables 4 and 5 compare the best and average solution values, in time units, obtained over 30 consecutive runs of MSDEH3:

- m: number of vehicles;
- HVNTS: best upper bound obtained by Belhaiza et al. (2014);
- HGVNS: best upper bound obtained by Belhaiza et al. (2017);
- EAVNS: best upper bound obtained by Hoogeboom et al. (2018);
- Δ_f: average solution obtained by MSDEH3;
- Δ_t: average execution time in seconds;
- g_1 =percent relative improvement of MSDEH3 with respect to HVNTS;
- g_2 = percent relative improvement of MSDEH3 with respect to HGVNS;
- g_3 = percent relative improvement of MSDEH3 with respect to EAVNS;

Table 5 shows an overall average improvement of 1.33% for g_1, an overall average improvement of 0.97% for g_2, and an overall average improvement of 0.48% for g_3, with respect to HVNTS, HGVNS and EAVNS, respectively. Some of the bounds, such as for rcm1, were improved by as much as 2.91% and 2.46% for rcm107 and 2.90% and 1.85% for rcm108. In addition, MSDEH3 decreases the minimal number of vehicles needed for rm201, cm102, cm103, cm104 and cm107. Tables 4 and 5 show that MSDEH3 performs better than EAVNS on all classes of instances, except on the rm2 class where both heuristics obtain almost the same average best solution value. Over the drivers' work period, the rm2 class has few widely spread customers time windows. The rm2 has the lowest average number of time windows, with large distances between each pair of consecutive time windows. Therefore, the rm2 class is easier to solve than the remaining classes, which makes the best solutions found by both heuristics very close.

The real-life instances are provided by BeTeLL (2018), a Canadian transportation and logistics research oriented company. The instances are from five major Canadian cities: Vancouver, Edmonton, Toronto, Ottawa and Montréal. They correspond to furniture's delivery activities of five weekdays. For each day and city, the data include the customer, the mailing address, the number of items to be delivered, the estimated

Table 4 MSDEH3 Results I on VRPMTW instances

Instance	m	HVNTS	HGVNS	EAVNS	MSDEH3	Δ_f	Δ_t	g_1	g_2	g_3
rm101	10	4041.9	4027.1	4026.1	**4005.5**	4086.4	77.2	0.90	0.54	0.51
rm102	9	3765.1	3751.2	3774.8	**3749.8**	3784.0	71.9	0.41	0.04	0.66
rm103	9	3708.5	3703.0	3700.6	**3676.6**	3733.3	77.8	0.86	0.71	0.65
rm104	9	3718.0	3701.2	3707.1	**3682.5**	3736.7	81.3	0.95	0.51	0.66
rm105	9	3688.8	3687.2	3690.5	**3625.6**	3703.2	69.7	1.71	1.67	1.76
rm106	9	3692.9	3708.4	3714.8	**3622.6**	3733.4	67.9	1.90	2.31	2.48
rm107	9	3701.4	3692.8	3700.4	**3653.3**	3713.4	72.4	1.30	1.07	1.27
rm108	9	3729.1	3722.6	3738.1	**3687.1**	3722.6	78.2	1.13	0.95	1.36
Average	9.1	3755.7	3749.2	3756.6	3717.5	3784.3	74.6	1.02	0.85	1.04
rm201	**2**	4808.2	4805.4	**3888.9**	3901.2	3965.2	65.4	18.86	18.82	− 0.32
rm202	2	3739.0	3706.8	3721.9	**3706.8**	3763.6	63.5	0.86	0.00	0.41
rm203	2	3710.3	3696.9	3693.2	**3691.4**	3734.3	49.3	0.51	0.15	0.05
rm204	2	3691.9	3674.5	**3671.7**	3674.5	3715.0	49.3	0.47	0.00	− 0.08
rm205	2	3689.9	**3668.1**	3668.4	**3668.1**	3705.9	48.4	0.59	0.00	0.01
rm206	2	3703.4	3684.9	**3672.6**	3673.5	3710.9	45.8	0.81	0.31	− 0.02
rm207	2	3701.7	3664.3	**3662.4**	3664.3	3709.7	60.9	1.01	0.00	− 0.05
rm208	2	3682.8	3664.3	**3663.6**	3664.3	3705.9	82.3	0.50	0.00	− 0.02
Average	2	3840.9	3820.7	3705.3	3705.5	3751.3	58.1	3.52	3.01	0.00
cm101	10	12, 320	12, 319.1	12, 345.4	**12,319.1**	12, 461.9	75.5	0.01	0.00	0.21
cm102	**11**	12, 492.1	12, 410.7	12, 482.3	**12,382.4**	12, 475.7	48.3	0.88	0.23	0.80
cm103	**11**	12, 641.2	12, 632.4	12, 592.2	**12,572.3**	12, 688.4	65.5	0.55	0.48	0.16
cm104	**13**	13, 087.8	13, 098.0	**12,927.8**	12, 928.4	12, 981.2	81.3	1.22	1.29	0.00
cm105	10	12, 083.4	12, 027.0	12, 066.3	**12,023.1**	12, 150.0	68.3	0.50	0.03	0.36
cm106	10	12, 073.9	12, 059.0	12, 066.4	**12,059.0**	12, 160.4	77.7	0.12	0.00	0.06
cm107	**10**	12, 324.2	12, 318.0	**12,108.4**	12, 114.5	12, 209.5	65.8	1.70	1.65	− 0.05
cm108	10	11, 990.4	11, 986.0	11, 985.9	**11,985.5**	12, 037.8	76.5	0.04	0.00	0.00
Average	10.6	12, 376.6	12, 356.3	12, 321.8	12, 298.0	12, 395.6	69.9	0.64	0.47	0.19

Bold values represent the best-known solution variable

Table 5 MSDEH3 Results II on VRPMTW instances

Instance	m	HVNTS	HGVNS	EAVNS	MSDEH3	Δ_f	Δ_t	g_1	g_2	g_3
cm201	5	13,520.1	13,498.8	13,468.4	**13,444.9**	13,589.7	68.3	0.56	0.40	0.17
cm202	6	14,027.3	14,025.1	**14,020.2**	14,021.0	14,097.7	69.4	0.04	0.03	−0.01
cm203	5	13,497.2	**13,465.8**	13,486.5	**13,465.8**	13,613.1	69.1	0.23	0.00	0.15
cm204	5	13,359.8	**13,344.0**	13,356.9	**13,344.0**	13,439.2	75.6	0.12	0.00	0.10
cm205	4	12,884.1	**12,827.8**	12,896.8	**12,827.8**	12,967.5	71.5	0.44	0.00	0.54
cm206	4	12,767.7	**12,713.2**	12,733.4	**12,713.2**	12,822.5	62.4	0.43	0.00	0.16
cm207	4	13,009.7	12,963.7	12,963.7	**12,950.7**	13,050.5	66.2	0.45	0.10	0.10
cm208	4	12,788.1	**12,749.7**	12,756.8	**12,749.7**	12,851.9	67.5	0.30	0.00	0.06
Average	4.6	13,231.8	13,198.5	13,210.3	13,190.2	13,304	68.8	0.31	0.06	0.15
rcm101	10	4098.9	4081.2	4080.6	**4056.8**	4128.2	58.6	1.03	0.60	0.58
rcm102	10	4222.6	4188.3	**4184.3**	4188.3	4212.9	82.5	0.81	0.00	−0.10
rcm103	10	4174.3	4150.4	4148.3	**4132.5**	4175.5	72.8	1.00	0.43	0.38
rcm104	10	4156.3	4144.0	4141.2	**4128.3**	4163.9	69.0	0.67	0.38	0.31
rcm105	10	4216.7	4207.0	4208.2	**4114.9**	4177.7	88.0	2.41	2.19	2.22
rcm106	10	4219.9	4187.7	4191.8	**4167.6**	4222.2	75.0	1.24	0.48	0.58
rcm107	11	4542.4	4521.5	4516.5	**4410.1**	4479.8	92.0	2.91	2.46	2.36
rcm108	11	4614.5	4565.2	4566.2	**4480.6**	4563.0	93.7	2.90	1.85	1.87
Average	10.3	4280.7	4254.9	4254.6	4209.9	4265.4	79	1.65	1.06	1.05
rcm201	2	3783.6	**3783.2**	3800.1	**3783.2**	3909.7	40.0	0.01	0.00	0.44
rcm202	2	3847.1	**3779.4**	3822.9	**3779.4**	3862.0	68.9	1.76	0.00	1.14
rcm203	2	3721.9	**3722.0**	3771.7	**3722.0**	3801.5	66.2	0.00	0.00	1.32
rcm204	2	3726.5	3708.5	3716.0	**3691.7**	3738.0	64.6	0.93	0.45	0.65
rcm205	2	3754.5	3754.5	3756.0	**3709.7**	3771.1	72.6	1.19	1.19	1.23
rcm206	2	3812.7	3803.3	**3725.0**	3748.7	3804.9	73.0	1.68	1.44	−0.64
rcm207	3	4764.2	4761.5	**4757.1**	4760.9	4791.8	45.5	0.07	0.01	−0.08
rcm208	2	3791.4	3742.7	**3735.1**	3742.7	3830.0	86.0	1.28	0.00	−0.20
Average	2.1	3900.2	3881.9	3885.5	3867.3	3938.6	64.6	0.84	0.38	0.47
All	6.4	6897.7	6876.9	6855.7	6831.4	6906.5	69.2	1.33	0.97	0.48

Bold values represent the best-known solution variable

Table 6 Results of MSDEH3 on real-life instances

City	Day	n	m	$f(X)$	$f(X^*)$	g
Vancouver	Mon	259	12	1503	1342	10.72
	Tue	241	12	1467	1238	15.61
	Wed	243	12	1531	1302	14.97
	Thu	237	11	1449	1246	14.01
	Fri	272	12	1466	1332	9.14
Avg		250	11.8	1483	1292	12.89
Edmonton	Mon	98	6	502	405	19.24
	Tue	73	5	461	381	17.44
	Wed	78	5	642	556	13.33
	Thu	73	5	506	395	21.88
	Fri	93	6	446	356	20.11
Avg		83	5.4	511	419	18.11
Toronto	Mon	533	29	5370	4320	19.56
	Tue	549	29	4761	4371	8.19
	Wed	567	28	5670	4949	12.72
	Thu	503	28	5398	4488	16.86
	Fri	653	30	5296	5253	0.82
Avg		561	28.8	5299	4676	11.76
Ottawa	Mon	147	10	1247	1044	16.30
	Tue	137	10	790	610	22.76
	Wed	130	11	1351	1137	15.86
	Thu	78	7	770	525	31.87
	Fri	109	6	532	441	17.17
Avg		120	8.8	938	751	19.92
Montréal	Mon	357	18	2999	1475	50.83
	Tue	346	11	2476	2306	6.85
	Wed	248	10	1650	1529	7.36
	Thu	242	10	2018	1897	6.01
	Fri	393	20	4149	3513	15.34
Avg		561	13.8	2658	2144	19.36
Avg		315	13.7	2169	1856	16.41

service time, and the possible delivery time windows. The mailing addresses were geocoded and the shortest distances and travel times were obtained using the shortest path algorithm and a road map. In addition, the data includes each vehicle's capacity. Each customer is given the possibility of delivery during one or more of three time windows: [8:00, 11:30], [12:30, 16:00] and [17:00, 20:30]. These latter account for the company's contractual agreement with the drivers' work union: No delivery is initiated during lunch [11:30, 12:30] or peak hours [16:00, 17:00]. A driver's working hours, including lunch time, cannot exceed 10 h. The instances vary in size from 73 customers and five drivers to 653 customers and 30 drivers. For each day and city,

the company provided its delivery schedule, prepared and validated by experienced human dispatchers within 2 to 5 h.

Table 6 compares the delivery schedules of the dispatchers to those obtained by MSDEH3 when run for 60 s. Columns 1–4 display the "City", " Day", n, and m. Column 5 displays $f(X)$ the total distance, in kilometers, of the delivery schedule obtained by the company's dispatchers. Column 6 reports $f(X^*)$, which equals $h(X^*)$ as only feasible solutions are considered. In addition, the company minimizes the total distance traveled. Finally, Column 7 reports the percent relative gap $g = (f(X) - f(X^*))/f(X)\%$ between the solution values of the dispatchers and those of MSDEH.

Table 6 shows that, on average, MSDEH3 reduces the dispatchers' solution cost by 16.41% and by more than 18% for Edmonton, Ottawa, and Montréal. This improvement exceeds 50% on the Montréal Monday's instance and 30% on the Ottawa Tuesday's instance. Such reduction of the total distance traveled translates into shorter working hours for the drivers.

Figure 8 partly illustrates the routing optimization solution obtained on Monday's VRPMTW using BeTeLL's routing visualizer.

7 Conclusions

We have presented three multi-start data-driven evolutionary algorithms (MSDEH) for the vehicle routing problem with multiple time windows (VRPMTW). The proposed heuristics keep track not only of the current and best solutions, but also of a pool of best solutions which constitutes the GA's population. The heuristics also improve upon standard VNS with a ruin-and-recreate procedure that helps VNS escape from suboptimal valleys. They apply a multiple restart strategy which reduces the risk of cycling. Finally, they categorize the VRPMTW infeasibilities into different types and handle them with the appropriate customized local search operators. The new general heuristic framework, common to these three algorithms, is guided by multi-start strategies and data-driven local search operators. The hybridization schemes are tested and compared. The best heuristic MSDEH3 yields substantial savings of duration and total distance traveled on benchmark and real-life instances, respectively. A natural extension to our work is to study the impact of data-driven operators to other VRP variants, such as the periodic VRP (PVRP).

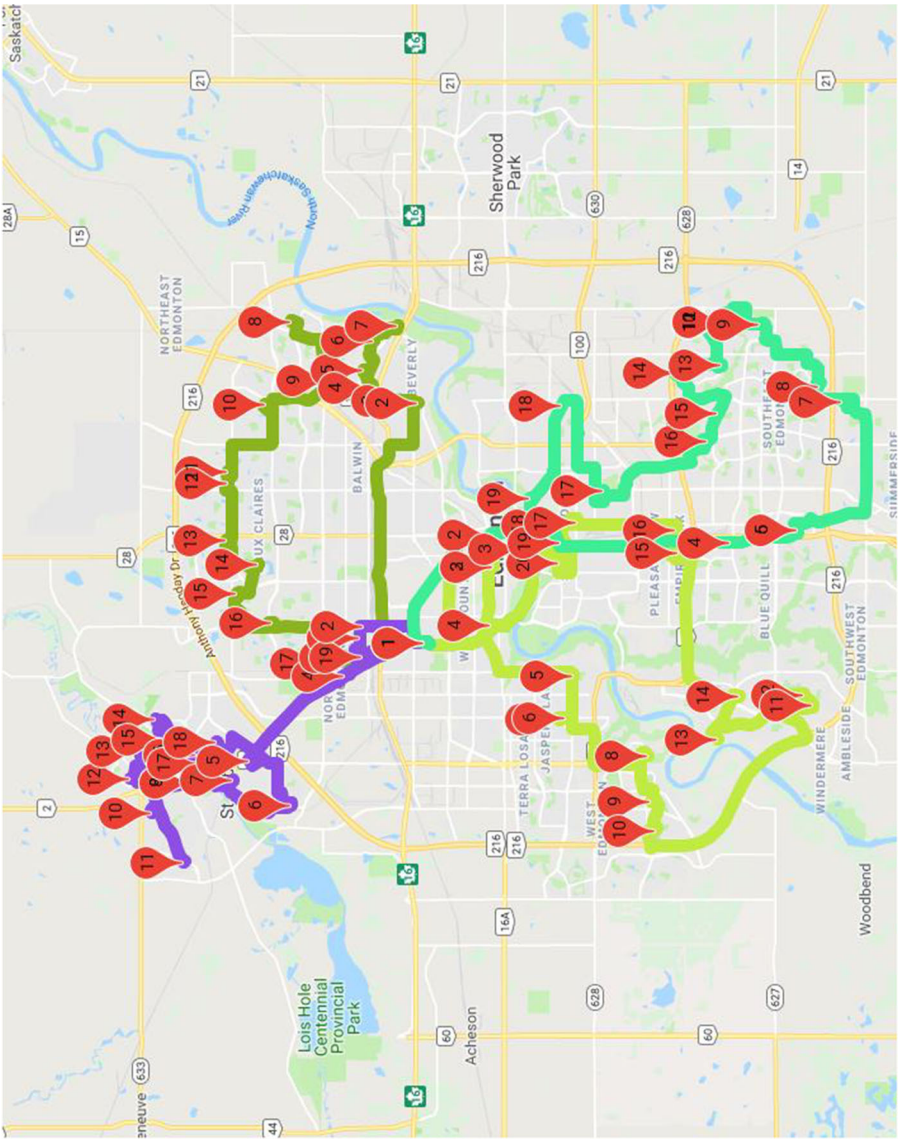

Fig. 8 Edmonton and region routing, Monday

References

Al-Duoli, F., Rabadi, G.: Data mining based hybridization of Meta-RaPS. Procedia Comput. Sci. **36**, 301–307 (2014)

Azi, N., Gendreau, M., Potvin, J.-Y.: An adaptive large neighborhood search for a vehicle routing problem with multiple routes. Comput. Oper. Res. **41**, 167–173 (2014)

Beheshti, A.K., Hejazi, S.R., Alinaghian, M.: The vehicle routing problem with multiple prioritized time windows: a case study. Comput. Ind. Eng. **90**, 402–413 (2015)

Berger, J., Barkaoui, M., Bräysy, O.: A route-directed hybrid genetic approach for the vehicle routing problem with time windows. Inf. Syst. Oper. Res. **41**, 179–194 (2003)

Belhaiza, S.: A data driven hybrid heuristic for the dial-a-ride problem with time windows. In: IEEE Symposium Series on Computational Intelligence SSCI 2017, Proceedings January, 1–8 (2018a)

Belhaiza, S.: A game theoretic approach for the real-life multiple-criterion vehicle routing problem with multiple time windows. IEEE Syst. J. **12**, 1251–1262 (2018b)

Belhaiza, S., M'Hallah, R.: A Pareto non-dominated solution approach for the vehicle routing problem with multiple time windows. In: Proceedings of the 2016 IEEE Congress on Evolutionary Computation CEC, pp. 3515–3524 (2016)

Belhaiza, S., Hansen, P., Laporte, G.: Hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. Comput. Oper. Res. **52**, 269–281 (2014)

Belhaiza, S., M'Hallah, R., Ben Brahim, G.: A new hybrid genetic variable neighborhood search heuristic for the vehicle routing problem with multiple time windows. In: IEEE Congress on Evolutionary Computation CEC, pp. 1319–1326 (2017)

BeTeLL.: Transportation & Logistics Labs., Canada (2018)

Bouziyane, B., Dkhissi, B., Cherkaoui, M.: A hybrid genetic algorithm for the static and dynamic vehicle routing problem with soft time windows. In: Proceedings of the Third IEEE International Conference on Logistics Operations Management GOL (2016)

Blanton, J.L. Jr., Wainwright, R.L.: Multiple vehicle routing with time and capacity constraints using genetic algorithms. In: Forrest, S. (ed.) Proceedings of the Fifth International Conference on Genetic Algorithm, pp. 452–459. Morgan Kaufmann Publisher, San Mateo, California (1993)

Calvet, L., Ferrer, A., Gomes, M.I., Juan, A., Masip, D.: Combining statistical learning with metaheuristics for the multi-depot vehicle routing problem with market segmentation. Comput. Ind. Eng. **94**, 93–104 (2016)

Cakir, F.: Data-centric solution methodologies for vehicle routing problems. PhD thesis, University of Iowa (2016)

Cooray, P., Rupasinghe, P.: Machine learning-based parameter tuned genetic algorithm for energy minimizing vehicle routing problem. J. Ind. Eng. **2017**, 1–13 (2017)

Defryn, C., Sörensen, K.: A fast two-level variable neighborhood search for the clustered vehicle routing problem. Comput. Oper. Res. **83**, 78–94 (2017)

De Jong, K.A.: An analysis of the behavior of a class of genetic adaptive systems. PhD dissertation, University of Michigan, Ann Arbor (1975)

Favaretto, D., Moretti, E., Pellegrini, P.: Ant colony system for a VRP with multiple time windows and multiple visits. J. Interdiscip. Math. **10**, 263–284 (2007)

Ferreira, H.S., Bogue, E.T., Noronha, T.F., Belhaiza, S., Prins, C.: Variable neighborhood search for vehicle problem with multiple time windows. Electron. Notes Discrete Math. **66**, 207–214 (2018)

Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, New York (1989)

Hansen, P., Mladenović, N., Moreno-Perez, J.A.: Variable neighbourhood search: methods and applications. Ann. Oper. Res. **175**, 367–407 (2010)

Hoogeboom, M., Dullaert, W.: Vehicle routing with arrival time diversification. Eur. J. Oper. Res. **275**, 93–107 (2018)

Hoogeboom, M., Dullaert, W., Lai, D., Vigo, D.: Efficient neighborhood evaluations for the vehicle routing problem with multiple time windows. Technical report OR-18-2 DEI, University of Bologna, Italy (2018)

Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)

Liu, L., Zhou, H.: Hybridization of harmony search with variable neighborhood search for restrictive single-machine earliness/tardiness problem. Inf. Sci. **226**, 68–92 (2013)

Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search: framework and applications. In: Gendreau, M., Potvin, J.-Y. (eds.) Handbook of Metaheuristics. Int. Ser. Oper. Res. Manag. Sci. **146**, 363–397 (2015)

Michalet, J., Prins, C., Amodeo, L., Yalaoui, F., Vitry, G.: Multi-start iterated local search for the periodic vehicle routing problem with time window and time spread constraints on services. Comput. Oper. Res. **41**, 196–207 (2014)

Mirabi, M.: A novel hybrid genetic algorithm for the multidepot periodic vehicle routing problem. Artif. Intell. Eng. Des. Anal. Manuf. AIEDAM **29**, 45–54 (2015)

Nagata, Y., Bräysy, O., Dullaert, W.: A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. Comput. Oper. Res. **37**, 724–737 (2010)

Nazari, M., Oroojlooy, A., Snyder, L., Takac, M.: Deep reinforcement learning for solving the vehicle routing problem. In: 35th International Conference on Machine Learning, Stockholm, PMLR 80 (2018)

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

Potvin, J.-Y., Bengio, S.: The vehicle routing problem with time windows part II: genetic search. INFORMS J. Comput. **8**, 165–172 (1996)

Prescott-Gagnon, E., Desaulniers, G., Rousseau, L.M.: A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. Networks **54**, 190–204 (2009)

Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Comput. Oper. Res. **31**(1985–2002), 624 (2004)

Repoussis, P.P., Tarantilis, C.D., Ioannou, G.: Arc guided evolutionary algorithm for the vehicle routing problem with time windows. IEEE Trans. Evol. Comput. **13**, 624–647 (2009)

Rasku, J., Kärkkäinen, T., Musliu, N.: Feature extractors for describing vehicle routing problem instances. In: OASIcs-OpenAccess Series in Informatics 50, Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2016)

Sánchez-Oro, J., Pantrigo, J.J., Duarte, A.: Combining intensification and diversification strategies in VNS. An application to the vertex separation problem. Comput. Oper. Res. **52**, 209–219 (2014)

Solomon, M.: Vehicle routing and scheduling with time window constraints: models and algorithms. Ph.D. Dissertation, Dept. of Decision Sciences, University of Pennsylvania (1983)

Steinhaus, M.: The application of the self-organizing map to the vehicle routing problem. PhD thesis, University of Rhode Island (2015)

Tangiah, S.: Vehicle routing with time windows using genetic algorithms. In: Chambers, L. (ed.) Application Handbook of Genetic Algorithms: New Frontiers, Volume II, pp. 253–277. CRC Press, Boca Raton (1995)

Toth, P., Vigo, D.: Vehicle Routing: Problems, Methods and Applications, 2nd edn. MOS-SIAM Series on Optimization, Philadelphia (2014)

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

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: Heuristics for multi-attribute vehicle routing problems: a survey and synthesis. Eur. J. Oper. Res. **231**, 1–21 (2013b)

Yang, N., Li. X-P., Zhu, J., Wang, Q.: Hybrid genetic-VNS algorithm with total flowtime minimization for the no-wait flowshop problem. In: International Conference on Machine Learning and Cybernetics (2008)

Zennaki, M., Ech-Cherif, A.: A new approach using machine learning and data fusion techniques for solving hard combinatorial optimization problems, vol. 1–5 (2008)