algorithms

MDPI

Article

A Hybrid Adaptive Large Neighborhood Heuristic for a Real-Life Dial-a-Ride Problem

Slim Belhaiza

Department of Mathematics and Statistics, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia; slimb@kfupm.edu.sa

Received: 15 December 2018; Accepted: 14 February 2019; Published: 16 February 2019

Abstract: The transportation of elderly and impaired people is commonly solved as a Dial-A-Ride Problem (DARP). The DARP aims to design pick-up and delivery vehicle routing schedules. Its main objective is to accommodate as many users as possible with a minimum operation cost. It adds realistic precedence and transit time constraints on the pairing of vehicles and customers. This paper tackles the DARP with time windows (DARPTW) from a new and innovative angle as it combines hybridization techniques with an adaptive large neighborhood search heuristic algorithm. The main objective is to improve the overall real-life performance of vehicle routing operations. Real-life data are refined and fed to a hybrid adaptive large neighborhood search (Hybrid-ALNS) algorithm which provides a near-optimal routing solution. The computational results on real-life instances, in the Canadian city of Vancouver and its region, and DARPTW benchmark instances show the potential improvements achieved by the proposed heuristic and its adaptability.

Keywords: adaptive large neighborhood search; genetic algorithms; impaired and elderly transportation; Dial-A-Ride problem; time windows

1. Introduction

The dial-a-ride problem (DARP) is a subfamily of Vehicle Routing Problems (VRPs) where customers' needs are met considering a fleet of heterogeneous vehicles. The most common application happens in senior or impaired people door-to-door transportation. The DARP's goal is to set a vehicle routing schedule with minimum cost meeting all requests and satisfying a number of conditions. These conditions include maximum vehicle capacity, maximum route duration and maximum customer transit time. Cases where requests are known at the time of planning define static DARP, while cases where some requests are received during routing operations define dynamic DARP. In addition, cases where servicing operations at stops start within pre-defined time windows are called DARP with time windows (DARPTW). Solving the DARPTW requires the application of approximate algorithms such as adaptive large neighborhood search (ALNS) and genetic algorithms (GA).

Ropke and Pisinger [1] proposed ALNS as an extension of the Large Neighborhood Search [2] for the Rich Pickup and Delivery Problem with Time Windows (RPDPTW). Its general principal is to remove requests from the routing solution and reinsert them at potentially more profitable positions. This is performed repeatedly with the help of adaptive destroy and repair operators. While the Large Neighborhood Search (LNS) heuristic uses a single destroy and repair operator, ALNS uses many of them. At each iteration, one or more specific destroy and repair operators are applied. The selection is based on the continuous updating of the operators' success rates. While most local search heuristics can only apply very small modifications to a given solution, ALNS covers a large search space, which defines the neighborhood of the current solution. Within one single iteration, ALNS can modify up to 30 to 40% of a solution. This revealed to be efficient with tightly constrained problems like the RPDPTW.

The common principles of GA are inspired from population genetics [3]. The generation of new individuals follows the general evolutionary scheme: selection, recombination and mutation. The selection phase is usually based on one or more characteristics that depend on the problem. The recombination phase selects parents' genes that may be inherited or not by their offspring. The mutation phase changes the offspring to maintain genetic diversity. Mutation happens with a marginal probability. Chevrier et al. [4] detailed local search operators adapted for the DARP within a multi-objective evolutionary hybrid heuristic applied to real-life instances. Masmoudi et al. [5] described a GA for the heterogeneous DARP (H-DARP). They proposed efficient operators adapted to the H-DARP: construction, crossover and local search operators.

Based on the strengths of ALNS and GA, this paper proposes a hybrid-ALNS heuristic for a real-life DARPTW. While every known heuristic contains both diversification and intensification elements, one can observe that single-solution based methods give more importance to intensification, and population-based methods give more importance to diversification. Through its main three features, the hybrid-ALNS heuristic offers a fair compromise between diversification and intensification search strategies. First, it keeps track not only of the current and best solutions, but also of a pool of best solutions which constitutes the genetic crossovers' input. Second, it enhances standard ALNS with newly customized operators. Third, it applies genetic crossover operators that diversify the solution neighborhood and help ALNS recover from suboptimal valleys.

Section 2 details the DARPTW and compiles a number of references on Neighborhood Search-based and Genetic Algorithm-based heuristics proposed in the DARP context. Section 3 details the proposed new algorithm and presents its phases. Section 4 compiles the experimental results on real-life and benchmark instances. Finally, Section 5 summarizes the paper.

2. The DARPTW

The DARPTW is commonly defined as a pick-up and delivery problem with maximum duration, maximum transit-time and time window constraints. The DARPTW involves a set of n customers and a set of m vehicles. The set of customers defines the set of $2n$ vertices V, connected through the arc set A. Hence, the DARPTW can be defined on a graph $G = (V, A)$. A customer i has a pick-up stop indicated by (i^+), and a drop stop indicated by (i_-). Each pick-up or drop has a non-negative service time s_i, and a time window: $[l_{i^+}, u_{i^+}]$ for a pick-up, or $[l_{i_-}, u_{i_-}]$ for a drop. A pick-up stop i^+ has a positive demand d_i and a maximum transit time r_i before arrival at its drop i_-, with a negative demand $-d_i$. A vehicle k has a work time window $[l_k, u_k]$ and a maximum capacity q_k. The total route duration of a vehicle k, from the beginning until the end of its duty, cannot exceed a maximum duration t_k. The DARPTW aims to design a set of m vehicle routes starting and ending at a depot D, while satisfying all conditions. Each customer i needs to be picked-up and dropped by the same vehicle, during the respective time windows. A route executed by a vehicle k should not exceed the vehicle's load capacity q_k and its maximum duration t_k. The waiting times are included in the route duration. Hence, if a vehicle arrives at a stop before the corresponding time window, a waiting time must be observed. The total duration is the sum of the total travel times, waiting times and service times.

2.1. Related Literature

Cordeau and Laporte [6], and, recently, Ho et al. [7] surveyed the most important research developments on the DARP, and classified the problem variants and the solution methodologies. Variable Neighborhood Search (VNS), Simulated Annealing (SA), ALNS and GA are among the most frequently used and most prolific solution approaches for the DARP and its different variants. Part of the literature proposed different hybridization of adaptive large Neighborhood search (ALNS) and variable neighborhood search (VNS).

2.2. VNS DARP Applications

Muelas et al. [8] described a VNS distributed algorithm for the DARP with a large scale. The algorithm used partitioning of requests and merging of routes. The algorithm proved to be efficient on a set of large-scale instances in San Francisco. Parragh et al. [9] detailed a VNS-based heuristic using different types of neighborhoods. The first type used simple exchange operators adapted to the DARP. The second used the ejection chain principle. The third dealt with the existence of arcs where the vehicle might be empty. Their heuristic achieved excellent results on different benchmark instances. Schilde et al. [10] used statistical information available about historical accidents, with stochastic solution approaches for the dynamic dial-a-ride problem (dynamic DARP). They presented a dynamic VNS paired with a dynamic stochastic VNS solution approach tested on a real-world road network problem. Experimental results demonstrated that, under some assumptions, using stochastic information about travel speeds yields significant improvements with respect to deterministic approaches. Belhaiza [11] proposed a hybridized VNS algorithm for the DARPTW. The heuristic combines evolutionary crossover operators with customized local search operators guided by four types of DARP infeasibilities.

2.3. ALNS DARP Applications

Parragh and Schmid described a large hybrid neighborhood heuristic for the DARP [12]. They used different neighborhoods to set many new best known solutions for a set of benchmark instances. Madsen et al. [13] detailed a heuristic based on insertions for the DARPTW with multiple objectives and capacities. The algorithm was implemented for new requests' online scheduling in a dynamic environment. Guerreiro et al. [14] described a bi-objective method for a public transportation dial-a-ride problem. Their two objectives are the minimization of the total transit time and the total waiting times under a maximum route duration constraint. Masson et al. [15] described an adaptive large neighborhood search (ALNS) algorithm for the DARP with transfer points, where users can transfer to different vehicles during their trip at predefined points. On real-life instances, the algorithm generated around 8% of cost savings Masmoudi et al. [16] studied a variant called the multi-depot multi-trip heterogeneous DARP (MD-MT-HDARP). They proposed three different metaheuristics: an improved ALNS, a hybrid Bees Algorithm with Simulated Annealing (BA-SA), and a hybrid Bees Algorithm with Deterministic Annealing (BA-DA). Experimental results showed the efficiency of the proposed algorithms, as well as their competitiveness with other algorithms on the MD-HDARP.

3. Hybrid ALNS DARP

We propose a hybrid-ALNS heuristic which combines diversification and intensification search strategies. For diversification, the proposed heuristic keeps track of a pool of best solutions which constitutes the Genetic Algorithm's pool of reinitializing solutions. For intensification, the heuristic is enhanced with newly customized ALNS operators. As it is often used for the VRP with Multiple Time Windows [17–20], the objective of our hybrid-ALNS DARP algorithm is to minimize the total duration of all routes. From the drivers' perspective, total duration minimization provides shorter working schedules while accounting for travel times. From the customers' perspective, total duration minimization reduces waiting times at each stop which allows them to spend less time in transit. This argument is confirmed by the computational results obtained by Belhaiza [11]. The four main steps of our ALNS-DARP are as follows:

- Step (1) Initialization and Insertion: inserts unassigned stops considering the relation between pick-ups and drops.
- Step (2) Repair and Destroy Iterative Loop: calls repair and destroy operators.
- Step (3) Simulated Annealing Features: applies simulated annealing (SA) acceptance mechanism and updates the operators' selection probabilities.

- Step (4) GA Crossovers: applies genetic crossover operators to the current solution X, to obtain a solution X''.

3.1. Initialization and Insertion

The initialization step constructs an initial solution X considering the relation between pick-ups and drops. The construction of X assigns to a vehicle $k \in \mathcal{K}$ as many stops as long as the solution remains feasible. It chooses, among all non-assigned pick-ups and among all their possible positions in the route of k, the pick-up and its corresponding position that yield the least increase of the route cost. Once a pick-up is selected, it attempts to find a suitable position to its corresponding drop in the same route without exceeding the capacity q_k and the duration t_k. The construction phase reiterates until all routes can no longer take any additional stop. The obtained solution X_0 may not necessarily serve all stops, or ensure that deliveries occur during the stops time windows. In this case, insertion operations are performed before every iteration of the repair and destroy iterative loop. For every pick-up or drop left unassigned, it attempts to insert it in all possible positions of every route.

With $h(X)$ as the total duration of all routes, the fitness function $f(X)$ adds to $h(X)$ the sum of the weighted penalties on the violations v of the time windows, ς of the overload of the vehicles, and ζ of their duration overtime. The non-satisfaction of the time windows of stops i^+ or i_- is measured as $v_i = \max\{|a_i - l_{p_i}|, |a_i - u_{p_i}|\}\}$, where a_i is the starting time of delivery at customer i. The overload of the capacity of vehicle k is the positive surplus $\varsigma_k = \max\{0, \sum_{i \in I_k} q_i - q_k\}$. The violations v_i, ς_k and ζ_k yield the penalties βv_i^μ, $\beta \varsigma_k^\mu$ and ζ_k^μ where β is a penalty weight, 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

$$f(X) = h(X) + \beta \sum_{i \in I} \min_{p \in W_i} v_i^\mu + \beta \sum_{k \in \mathcal{K}} \varsigma_k^\mu + \beta \sum_{k \in \mathcal{K}} \zeta_k^\mu. \quad (1)$$

3.2. Repair and Destroy Iterative Loop

The iterative loop improves the initial solution X through repeated and alternated intensification and diversification search operations. For intensification, it applies a set of repair and destroy operators to X to obtain a solution X'. Two cases may rise: the repair and destroy is accepted or rejected. In the first case, the focal point of the search is changed to X' setting the current solution $X = X'$. Furthermore, when this new local minimum X' improves X^*, X^* and $f(X^*)$ are updated. In the second case, the counter of non-improving trials is incremented. For diversification, when the counter reaches a preset number of iterations without improvement, genetic crossovers are called. The iterative loop stops when the execution time t exceeds a preset threshold time limit $\bar{t}$.

The current paper proposes a hybrid-ALNS algorithm, with operators not only based on infeasibilities as in [11], but rather on a destruction and repair mechanism which updates the success rate of each given operator. The repair and destroy phase combines regular and data-driven operators. Regular operators, such as 1-opt, 2-opt and X-opt, are provided with data on route infeasibilities and operators success history. The intuition behind these operators is that routes with infeasibilities could be repaired or avoided instead of being overlooked. A repair operator builds a new solution from X while maintaining or improving its overall feasibility, while the destroy operators randomly remove parts of it. While operators are called based on their increasing order of complexity in [11], the primary selection of a specific operator is performed through a roulette wheel mechanism in the current paper. Moreover, no records on the success rates of the operators, and no weights are assigned to them in HG-DARP [11], while a weight τ_g which measures the success rate is associated with a given operator g. We mainly use roulette wheel selection for its simplicity, as it allows operators with higher weights to have higher probabilities of selection. Considering that the weight of a given operator may follow a

different updating scheme depending on the problem being solved, roulette wheel selection adapts to these differences.

At the beginning of the single and multi-route repair-destroy phase, all weights are initialized to 1 and the probability of selecting the operator g equals $\frac{1}{G}$, where G is overall number of single or multi-route operators used. The weights are adjusted after each iteration of the repair and destroy loop following their scores. If an operator g is applied, its weight τ_g is updated depending on its success or failure to improve the overall or incumbent solution. If the new solution improves the overall best solution, the score of the operator is $\eta_g = 2$. If the new solution improves the incumbent solution, the score of the operator is $\eta_g = 1$. Otherwise, the score of the operator is $\eta_g = 0$. If the operator improves the overall or incumbent solution, its weight is increased: $\tau_g = [(1-\epsilon)\tau_g + \epsilon\eta_g]/o_g$, where $\epsilon \in [0,1]$ is the reaction factor controlling how quickly the weight reacts to score changes, and o_g is the number of times operator g was called. For example, if an operator g succeeds on its first call to improve the overall best solution, its weight τ_g increases from $\frac{1}{G}$ to $\frac{1+\epsilon(2G-1)}{G}$. While if it succeeds on its first call to improve the incumbent solution, its weight increases from $\frac{1}{G}$ to $\frac{1+\epsilon(G-1)}{G}$. Otherwise, if it fails to make any improvement, its weight decreases from $\frac{1}{G}$ to $\frac{1-\epsilon}{G}$. In the following, we detail the ALNS operators we implemented.

3.2.1. Single Route Repair and Destroy Operators

For every vehicle $k \in \mathcal{K}$, the data-driven single-route repair-destroy phase applies up to three insertion-removal operators: the repair-destroy relocate operator (RDR), the repair-destroy node operator (RDN), and the repair-destroy arc operator (RDA). All customers are eligible to be relocated, in particular those with time windows, as they are served outside of one their time windows. The single route operators can reduce the cost of a route and also make it feasible.

RDR detects any drop appearing before its corresponding pick-up within a single route and tries to move it after in the route order. If the move is accepted, this repairs the route. If the move is rejected, both stops, pick-up and drop, are ejected from the route. Theoretically, as a given route may contain a maximum of $2n$ stops to be relocated in $2n-1$ possible positions, the complexity order of RDR is $o(4mn^2)$. Figure 1 shows how (j_-) is moved to a position after (j_+).

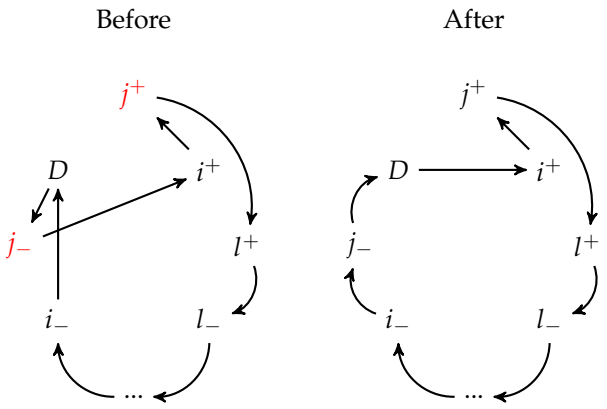

Figure 1. Repair-Destroy Relocate (RDR).

RDN scans all routes for any pick-up appearing after its corresponding drop in a single route and tries to interchange their positions in the route order. If the move is accepted, this repairs the route. If the move is rejected, both stops are ejected from the route. As a given route may contain a maximum of $2n$ stops to be interchanged with $2n-1$ remaining stops, the theoretical complexity order of RDN is also $o(4mn^2)$. Figure 2 shows how (i^+) is interchanged with (i_-).

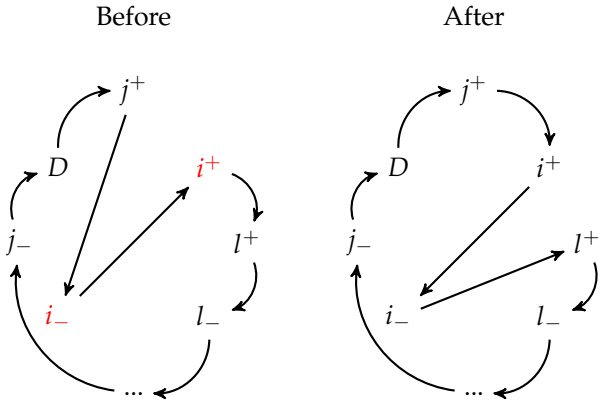

Figure 2. Repair-Destroy Node (RDN).

RDA detects any drop appearing before its corresponding pick-up and exchanges the visit order of the ingoing or outgoing arc, with any other arc in the route. If the move is accepted, this repairs the route. If the move is rejected, both stops are ejected from the route. As a given route may contain a maximum of $2n + 1$ arcs to be interchanged with $2n$ remaining ones, the theoretical complexity order of RDA is $o(4mn^2)$. Figure 3 shows how the position of arc (j_-, i_-) is exchanged with the position of arc (l^+, i^+).

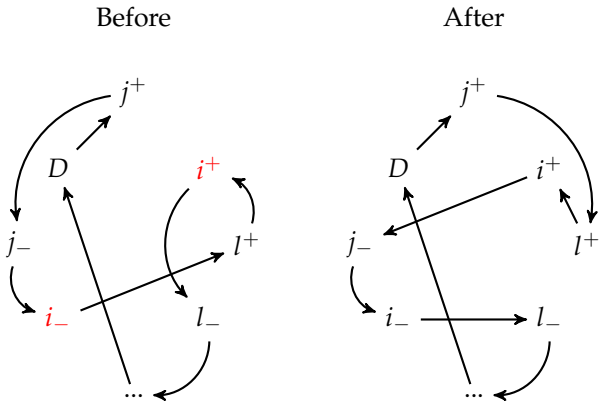

Figure 3. Repair-Destroy Arc (RDA).

3.2.2. Multiple Route Repair and Destroy Operators

The multiple-route repair and destroy are applied to the routes of two vehicles with three different operators: the repair-destroy inter-route relocate (RDIR), the repair-destroy inter-route swap (RDIS) and the repair-destroy inter-route cross (RDIC). All customers are eligible to be relocated with the destroy-repair multiple route operators, in particular those with time windows' infeasibilities.

RDIR detects any drop in a different route than its corresponding pick-up. It attempts two different relocate moves. Firstly, it attempts to insert the drop stop in its corresponding pick-up route. Secondly, if that fails, it tries to insert the pick-up in its corresponding drop route. If the move is accepted, this partially repairs both routes k and k'. If the move is rejected, both stops are ejected from their corresponding routes. Therefore, the two routes are partially repaired or destroyed. Theoretically, as every given route may contain a maximum of $2n$ nodes to be moved to a maximum of $2n - 1$ possible positions in the second route, the complexity order of RDIR is $o(4m^2n^2)$. Figure 4 shows how the drop (v_-) in route k is moved to route k' and inserted in a position after its corresponding drop (v^+).

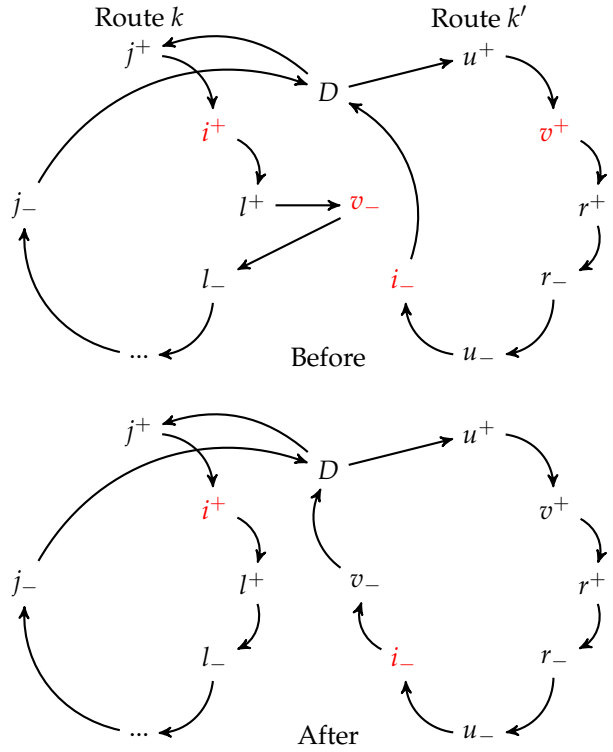

Figure 4. Repair-Destroy Inter-route Relocate (RDIR).

RDIS seeks through all routes for different pairs of pick-ups, drops or non-related pick-up and drop. Firstly, it tries to exchanges two drops (i_-) and (j_-) if one's pick-up is scheduled in the other's route, and vice versa. Secondly, it tries to exchange two pick-ups (i^+) and (j^+) if one's drop is scheduled in the other's route, and vice versa. Finally, it tries to exchange two non-related drop (i_-) and pick-up (j^+) if (i^+) is scheduled in (j^+)'s route and (j_-) is scheduled in (i_-)'s route. If the move is accepted, this partially repairs both routes. If the move is rejected, the two pairs of stops are ejected from their routes. As every given route may contain a maximum of $2n$ nodes to be interchanged with $2n-1$ possible nodes in the second route, the theoretical complexity order of RDIS is $o(4m^2n^2)$. Figure 5 shows how the drop (v_-) from route k is exchanged with the drop (i_-) from route k'.

RDIC seeks through all routes for a particular configuration that involves pairs of drops, pick-ups, or non-related drops and pick-ups requiring to be shifted to the same route. Firstly, it attempts to shift the arc (i^+, j^+) to the route containing (i_-) or (j_-), or attempts to shift the arc (i_-, j_-) to the route containing i^+ or j^+. Secondly, it attempts to shift the arc (i^+, j_-) to the route containing (i_-) or (j^+), or attempts to shift the arc (i_-, j^+) to the route containing i^+ or j_-. If the shifting is accepted, this repairs both routes. Otherwise, the stops are ejected from their routes. As every given route may contain a maximum of $2n+1$ arcs to be interchanged with $2n+1$ possible arcs in the second route, the theoretical complexity order of RDIC is $o(4m^2n^2)$. Figure 6 shows how the arc (u_-, v_-) is shifted from route k to route k'.

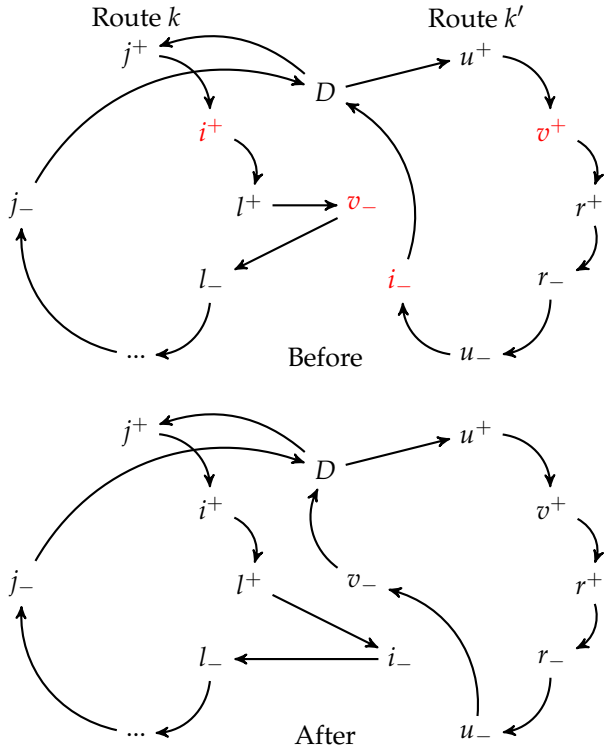

Figure 5. Repair-Destroy Inter-route Swap (RDIS).

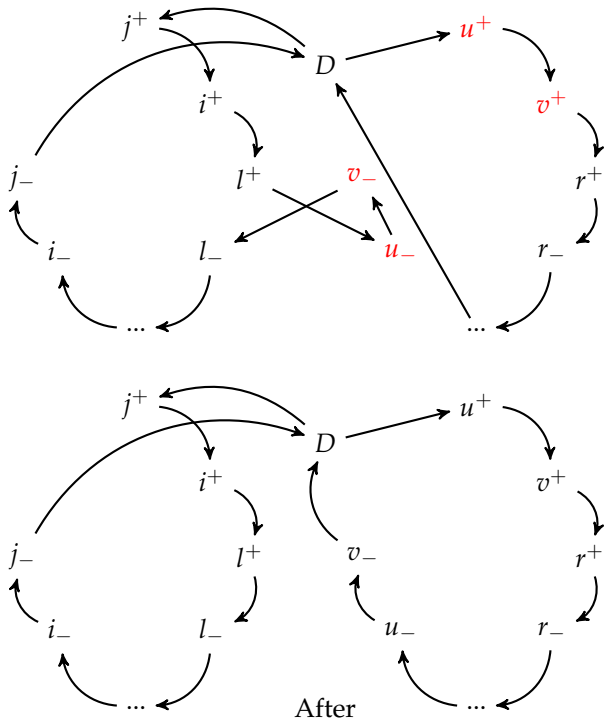

Figure 6. Repair-Destroy Inter-route Cross.

3.3. Simulated Annealing Features

Along the lines of Canca et al. [21], we consider a standard two-parameter exponential function for our temperature cooling process. The first parameter is the start temperature T_0. The second parameter is the cooling rate $\alpha \in (0, 1)$.

Hybrid-ALNS starts with a temperature T_0. The initial temperature T_0 is such that a solution X' with fitness $f(X')$ that is less than 10% larger than $f(X_0)$, the fitness value of the initial solution X_0, has a probability larger than 0.5 of being accepted. In other words, if $f(X') - f(X_0) \leq 10\ \% f(X_0)$, then $e^{-\frac{[f(X')-f(X_0)]}{T_0}} \geq 0.5$. Therefore, $T_0 \geq -\frac{[f(X')-f(X_0)]}{ln(0.5)}$.

The SA current temperature T follows the cooling scheme $T = T_0\alpha^{\iota}$ at every temperature cycle ι. We consider a new temperature cycle, with a newly updated temperature, for each 1000 iterations, which offers a fair compromise between a too fast and a too slow simulated annealing. If the repair move is accepted, the corresponding operator's weight τ_g is adjusted, and the number of calls o_g is incremented. If not, o_g is solely incremented.

3.4. GA Crossovers

To escape from sub-optimal valleys, genetic crossovers are called whenever the best solution cannot be improved during a number of $\iota_{\max} = 5$ of consecutive temperature cycles, i.e., 5000 iterations. The genetic crossover's input consists of the pool of best solutions stored during the execution of the algorithm. Its output consists of solutions' offspring which share the same genes with their parents. Within a given route k, a gene encodes the service by a given stop i^+ or i_-. Therefore, if two solution parents use the same route k to serve the same set or sequence of stops, this characteristic is most likely to be inherited by their offspring. Figure 7 shows how the stops i^+, k^+ and l^+ are served by the same route k in both parents. Therefore, the solution offspring obtained inherits this gene. After a number of iterations, if a better overall solution is not obtained, a simple reset is performed using one of the elements in the pool of best solutions.

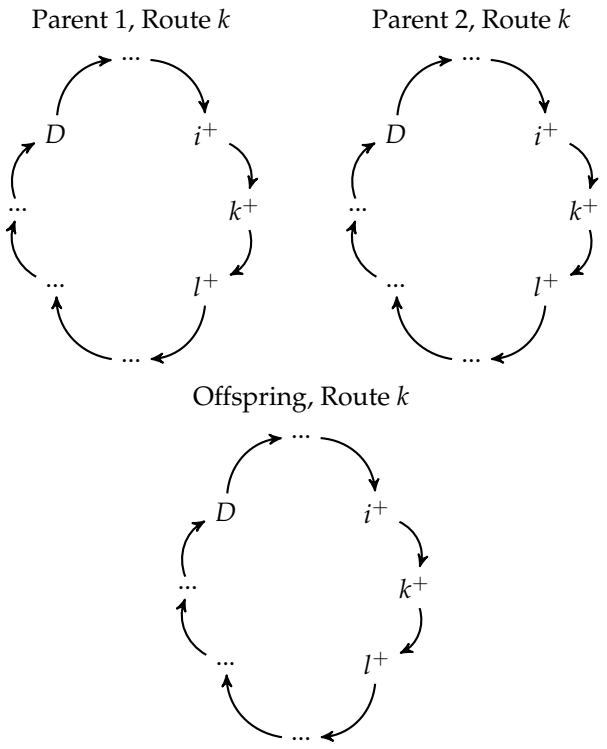

Figure 7. Two parents' crossover.

4. Experimental Results

The experimental results were performed on two sets of real-life and benchmark instances using a C++ implementation, under MS Windows on workstations with 3.3GHz Intel Core i7 vPro processors, and 8 GB RAM.

4.1. Results on Real-Life Instances

The real-life instances are collected on 10 consecutive days, representing two consecutive weeks, in the Canadian city of Vancouver, its region and suburbs: Richmond, Surrey, Coquitlam, North Vancouver, etc. The physical addresses of all pick-up and drop points were first geocoded, i.e., converted to longitude and latitude coordinates. Based on Google's map and distance API, these geocodes were used to compute the travel times and distances between each pair of nodes in the given day's graph. The test instances involve from 30 stops up to 80 stops. They detail the demand for impaired people and elderly transportation by a private company. In these instances, the first n stops represent the pick-ups and the last n stops represent their corresponding drops. The demand is around 25 requests per business day. It peaks on Thursdays and drops on Fridays. The company does not take requests for weekends. The company has 12 medium size shuttle buses equipped for elderly and reduced mobility passengers. The maximum capacity of each bus is of eight passengers at the same time, as a large passage corridor between the seats and an empty storage space for wheel chairs and other luggage are needed. Each route has a maximum duration of 360 min, as any driver is not supposed to spend more than six consecutive hours on duty. The time windows for the customers are very large for the pick-ups, but tight for their corresponding drops, with exactly one hour span. For example, a customer could be picked-up at any time between 7:00 am and 7:00 pm, but needs to be dropped between 10:00 a.m. and 11:00 a.m. at his destination, as he may have a medical appointment. If a customer needs to be picked-up again and taken home, it is treated as a new request: a new pick-up with a new drop. Each customer's maximum transit time is of 90 min, as some passengers need to be picked-up quite far from their destination. The service time at each stop is of 5 to 10 min. The company uses a transportation optimization software for its routing activity. These instances were provided by BeTeLL, a transportation optimization and research oriented company based in Montreal, QC, Canada.

In Table 1, the column "Day" indicates the day of the instance. The column "n" indicates the number of customers. The column "m" indicates the number of vehicles used. The column "$f(X)$" indicates the total duration in minutes of the routes scheduled by the company's current optimization software. The columns "$f(X^*)$", "Avg" and "Time" indicate the best and average solution values, and the average execution times (in seconds) obtained by ALNS (without GA) and Hybrid-ALNS, respectively. The column "Δ_f%" indicates the gap $\frac{f(X)-f(X^*)}{f(X^*)}$ in percentage between the best solution obtained with Hybrid-ALNS and the company's overall route duration.

The results in Table 1 show that our Hybrid-ALNS heuristic was able to save about 5.25% on average routing durations compared to the business solution currently used by the company. These results were obtained in less than 37 s in average, which could give the company the required flexibility to adapt to re-routing requests or deal with dynamic ones in the future. The results in Table 1 also show that Hybrid-ALNS outperforms ALNS (without GA). Hybrid-ALNS provides not only better solution quality, but also better general results.

Table 1. Results on real-life instances.

Week		Instance			ALNS			Hybrid-ALNS			
1	Day	n	m	$f(X)$	$f(X^*)$	Avg	Time	$f(X^*)$	Avg	Time	$\Delta_f\%$
	Mon	25	7	1615.15	1578.18	1683.67	19	1561.42	1622.56	17	−3.33
	Tue	24	7	1706.94	1629.88	1728.79	43	1586.15	1707.20	44	−7.08
	Wed	24	7	1671.41	1587.15	1749.41	56	1574.33	1681.28	57	−5.81
	Thu	37	9	2444.96	2286.73	2507.29	63	2213.38	2472.65	67	−9.47
	Fri	19	4	1350.88	1291.68	1303.11	11	1291.68	1303.85	12	−4.38
2	Day	n	m	$f(X)$	$f(X^*)$	Avg	Time	$f(X^*)$	Avg	Time	$\Delta_f\%$
	Mon	24	4	1308.32	1170.07	1178.57	24	1164.79	1180.01	26	−10.57
	Tue	26	5	1474.19	1461.20	1498.50	25	1451.65	1487.40	24	−1.53
	Wed	28	6	1743.93	1734.75	1798.12	29	1710.61	1776.45	27	−1.91
	Thu	40	10	2619.14	2619.03	2725.26	79	2518.58	2642.31	82	−3.84
	Fri	15	5	1131.32	1079.74	1112.85	13	1079.74	1086.86	13	−4.19
	Avg	26.2	6.4	1706.62	1643.84	1728.56	36.2	1615.23	1696.06	36.9	−5.25

Figure 8 illustrates the routing optimization solution obtained on week 2 Monday's solution using BeTeLL's routing visualizer.

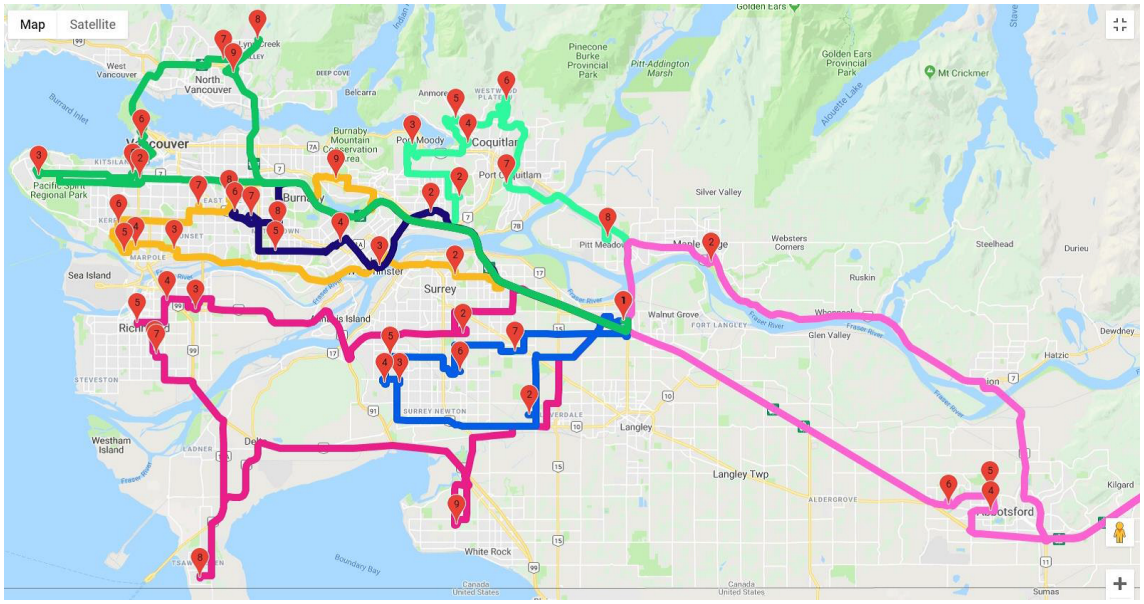

Figure 8. Vancouver and Region routing, Week 2, Monday.

4.2. Results on Benchmark Instances

The 20 DARPTW benchmark instances proposed by Cordeau and Laporte [22] involve from $n = 24$ up to $n = 144$ customers. In Table 2, column "Instance" indicates the name of the instance, column "n" indicates the number of customers and column "m" indicates the number of vehicles used by the heuristics for the DARPTW proposed by Parragh and Schmid [12] and Braekers et al. [23]. The column "m^*" indicates the number of vehicles used by HG-DARP heuristic [11] and Hybrid-ALNS. The columns "HGDARP(m)" and HGDARP(m^*) indicate the best known route durations with m and m^* obtained with HG-DARP. The columns "HALNS(m)" and "HALNS(m^*)"indicate the best route duration with m and m^* obtained with Hybrid-ALNS. The columns "$Avg_f(m)$" and "$Avg_f(m^*)$" indicate the average total route duration obtained with m and m^*, respectively, over 10 randomly seeded runs of Hybrid-ALNS. The columns "$Avg_t(m)$" and "$Avg_t(m^*)$" indicate the average execution

time, with m and m^*, respectively, in seconds. The columns "$\Delta_f(m)\%$" and "$\Delta_f(m^*)\%$" indicate the gaps $\frac{HGDARP(m)-HALNS(m)}{HGDARP(m)}$ and $\frac{HGDARP(m^*)-HALNS(m^*)}{HALNS(m^*)}$ in percentage.

Hybrid-ALNS improved the best total duration on all instances with respect to HG-DARP. For instance pr02, Hybrid-ALNS reaches a minimum total duration lower by more than 12% when four vehicles are used, and by more than 7.6 % when five vehicles are used. For an average computational time below 5 min (300 s), these results show that the proposed hybrid-ALNS algorithm improves by 3.91% to 5.03%, on average, the overall best solution values reached by HG-DARP.

Table 2. Duration minimization results

Instance	n	m	HGDARP(m)	HALNS(m)	$\mathbf{Avg_f(m)}$	$\mathbf{Avg_t(m)}$	$\mathbf{\Delta_f(m)\%}$	$\mathbf{m^*}$	HGDARP(m^*)	HALNS(m^*)	$\mathbf{Avg_f(m^*)}$	$\mathbf{Avg_t(m^*)}$	$\mathbf{\Delta_f(m^*)\%}$
pr01	24	3	703.27	698.95	703.1	8.0	−0.61	2	726.86	724.24	732.3	5.6	−0.36
pr02	48	5	1564.15	1444.29	1491.5	32.3	−7.66	4	1621.06	1423.18	1536.8	32.5	−12.21
pr03	72	7	2309.99	2146.57	2196.0	116.8	−7.07	6	2213.77	2137.73	2214.3	121.4	−3.43
pr04	96	9	2970.00	2728.70	2788.8	261.9	−8.12	8	2916.30	2754.73	2810.5	203.2	−5.54
pr05	120	11	3564.57	3354.95	3412.2	374.4	−5.88	10	3555.90	3335.39	3401.6	342.0	−6.20
pr06	144	13	4303.50	4094.97	4143.5	523.4	−4.85	12	4359.60	4172.64	4231.3	526.4	−4.29
pr07	36	4	1133.30	1113.74	1151.7	24.3	−1.73	3	1117.00	1077.77	1103.4	30.7	−3.51
pr08	72	6	2188.70	2102.65	2140.1	81.6	−3.93	6	2188.70	2102.65	2140.1	81.6	−3.93
pr09	108	8	3248.60	3037.96	3076.4	334.5	−6.48	8	3248.60	3037.96	3076.4	334.5	−6.48
pr10	144	10	4556.70	4289.93	4325.2	513.6	−5.85	10	4556.70	4289.93	4325.2	513.6	−5.85
pr11	24	3	691.20	659.28	664.5	8.1	−4.62	2	675.10	668.81	675.7	8.6	−0.93
pr12	48	5	1350.50	1293.22	1308.4	132.0	−4.24	3	1339.30	1318.87	1338.9	128.0	−1.53
pr13	72	6	2138.20	2046.17	2066.0	210.5	−4.30	6	2138.20	2046.17	2066.0	210.5	−4.30
pr14	96	9	2820.30	2649.97	2661.4	408.1	−6.04	8	2696.70	2610.73	2648.8	409.5	−3.19
pr15	120	10	3493.40	3255.86	3290.1	587.1	−6.80	9	3374.05	3209.40	3350.1	533.3	−4.88
pr16	144	11	4100.10	4045.14	4088.5	712.3	−1.34	10	4046.10	3971.28	4001.2	603.6	−1.85
pr17	36	4	1062.10	992.39	997.4	18.3	−6.56	3	1009.44	1000.23	1008.0	17.5	−0.91
pr18	72	6	2144.70	2070.42	2083.8	124.5	−3.46	5	2111.10	2087.70	2088.9	117.1	−1.11
pr19	108	8	3157.00	2999.28	3022.1	322.2	−5.00	7	3060.70	3010.45	3048.8	314.8	−1.64
pr20	144	10	4248.10	3989.57	4051.6	654.3	−6.09	10	4248.10	3989.57	4051.6	654.3	−6.09
Avg	86.4	7.4	2587.42	2450.7	2483.1	272.4	−5.03	6.6	2560.16	2448.47	2492.5	259.4	−3.91

5. Conclusions

In this paper, we have proposed a hybrid adaptive large neighborhood search heuristic algorithm (Hybrid-ALNS) for the dial-a-ride problem with time windows (DARPTW). Hybrid-ALNS heuristic offers a compromise between diversification and intensification search strategies. It keeps track of a pool of best solutions which constitutes the genetic crossover's input used to diversify the pool of reinitializing solutions and help ALNS recover from suboptimal valleys. Hybrid-ALNS is also enhanced with newly customized operators that are not only based on DARP infeasibilities, but also on a destruction and repair mechanism which updates their respective success rates.

The proposed heuristic was applied to a real-life impaired and elderly transportation in the city of Vancouver and its region, with the objective to improve the overall performance of vehicle routing operations. The computational results have shown potential improvements exceeding 5% of the total route durations on the real-life instances, and around 4% to 5% on DARPTW benchmark instances.

Author Contributions: S.B. is the sole author of this article, therefore, conceptualization, implementation, validation, writing—review and editing, were carried out by himself.

Funding: This research received no external funding.

Acknowledgments: The author is grateful to BeTeLL for providing the refined real-life data.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Ropke, S.; Pisinger, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transp. Sci.* **2006**, *40*, 455–472. [CrossRef]
2. Shaw, P. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming—CP98*; Maher, M., Puget, J.F., Eds.; Springer: Berlin/Heidelberg, Germany, 1998; pp. 417–431.

3. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial intelligence*; MIT Press: Cambridge, MA, USA, 1992.
4. Chevrier, R.; Liefooghe, A.; Jourdan, L.; Dhaenens, C. Solving a dial-a-ride problem with a hybrid evolutionary multi-objective approach: application to demand responsive transport. *Appl. Soft Comput.* **2012**, *12*, 1247–1258. [CrossRef]
5. Masmoudi, M.A.; Braekers, K.; Masmoudi, M.; Dammak, A. A hybrid genetic algorithm for the heterogeneous dial-a-ride problem. *Comput. Oper. Res.* **2017**, *81*, 1–13. [CrossRef]
6. Cordeau, J.F.; Laporte, G. The dial-a-ride Problem: models and algorithms. *Ann. Oper. Res.* **2007**, *153*, 29–46. [CrossRef]
7. Ho, S.C.; Szeto, W.Y.; Kuo, Y.-H.; Leung, J.M.Y.; Petering, M.; Tou, T.W.H. A survey of dial-a-ride problems: Literature review and recent developments. *Transp. Res. Part B Methodol.* **2018**, *111*, 395–421. [CrossRef]
8. Muelas, S.; LaTorre, A.; Pena, J.-M. A distributed VNS algorithm for optimizing dial-a-ride problems in large-scale scenarios. *Transp. Res. Part C Emerging Tech.* **2015**, *54*, 110–130. [CrossRef]
9. Parragh, S.; Doerner, K.; Hartl, R. Variable neighborhood search for the dial-a-ride problem. *Comput. Oper. Res.* **2010**, *37*, 1129–1138. [CrossRef]
10. Schilde, M.; Doerner, K.F.; Hartl, R.F. Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *Eur. J. Oper. Res.* **2014**, *238*, 18–30. [CrossRef] [PubMed]
11. Belhaiza, S. A Data Driven Hybrid Heuristic for the Dial-A-Ride Problem with Time Windows. In Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence SSCI, Honolulu, HI, USA, 27 November–1 December 2017.
12. Parragh, S.; Schmid, V. Hybrid large neighborhood search for the dial-a-ride problem. *Comput. Oper. Res.* **2013**, *40*, 490–497. [CrossRef] [PubMed]
13. Madsen, O.B.G.; Raven, H.F.; Rygaard, J.M. A branch-and-cut algorithm for a realistic dial-a-ride problem. *Transp. Res. Part B Methodol.* **2015**, *81*, 267–288.
14. Guerreiro, F.; Pezzella, F.; Pisacane, O.; Trollini, L. Multi-objective optimization in dial-a-ride public transportation. *Transp. Res. Procedia* **2014**, *3*, 299–308. [CrossRef]
15. Masson, R.; Léhudé, F.; Péton, O. The dial-a-ride problem with transfer. *Comput. Oper. Res.* **2013**, *41*, 12–23. [CrossRef]
16. Masmoudi, M.A.; Hosny, M.; Braekers, K.; Dammak, A. Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transp. Res. Part E Logist. Transp. Rev.* **2016**, *96*, 60–80. [CrossRef]
17. Belhaiza, S.; Hansen, P.; Laporte, G. Hybrid variable neighborhood tabu Search heuristic for the vehicle routing problem with multiple time windows. *Comput. Oper. Res.* **2014**, *52*, 269–281. [CrossRef]
18. 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), Vancouver, BC, Canada, 24–29 July 2016; pp. 3515–3524.
19. Belhaiza, S.; M'Hallah, R.; Brahim, B.G. A new hybrid genetic variable neighborhood search heuristic for the vehicle routing problem with multiple time windows. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 5–8 June 2017; pp. 1319–1326.
20. Belhaiza, S. A Game Theoretic Approach for the Real-Life Multiple-Criterion Vehicle Routing Problem With Multiple Time Windows. *IEEE Syst. J.* **2018**, *12*, 1251–1262. [CrossRef]
21. Canca, D.; De-Los-Santos, A.; Mesa, J.A.; Laporte, G. The railway network design, line planning and capacity problem: An adaptive large neighborhood heuristic. *Adv. Intell. Syst. Comput.* **2018**, *572*, 198–219.
22. Cordeau, J.F.; Laporte, G. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transp. Res. Part B Methodol.* **2003**, *37*, 579–594. [CrossRef]
23. Braekers, K.; Caris, A.; Janssens, G.K. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transp. Res. Part B Methodol.* **2014**, *67*, 166–186. [CrossRef]