Citation for published version:
Shelbourne, B, Battarra, M & Potts, CN 2017, 'The Vehicle Routing Problem with Release and Due Dates', INFORMS Journal on Computing, vol. 29, no. 4, pp. 705 - 723. https://doi.org/10.1287/ijoc.2017.0756

DOI:
10.1287/ijoc.2017.0756

Publication date:
2017

Document Version
Peer reviewed version

Link to publication

The final publication is available at pubsonline.informs.org via https://doi.org/10.1287/ijoc.2017.0756.

Submitted to *INFORMS Journal on Computing*
manuscript (Please, provide the mansucript number!)

The vehicle routing problem with job availability constraints

BC Shelbourne
School of Mathematical Sciences, University of Southampton, Highfield, Southampton SO17 1BJ, UK, bcs1g11@soton.ac.uk

M Battarra
School of Management, University of Bath, Bath, BA2 7AY, UK, m.battarra@bath.ac.uk

CN Potts
School of Mathematical Sciences, University of Southampton, Highfield, Southampton SO17 1BJ, UK, c.n.potts@soton.ac.uk

A novel extension of the classical vehicle routing and scheduling problem is introduced that integrates aspects of machine scheduling into vehicle routing. Associated with each customer order is a release date that defines the earliest time that the order is available to leave the depot for delivery, and a due date that indicates the time by which the order should ideally be delivered to the customer. The objective is to minimize a convex combination of the operational costs and customer service level, represented by the total distance traveled and the total weighted tardiness of delivery, respectively. A path-relinking algorithm (PRA) is proposed to address the problem, and a variety of benchmark instances are generated to evaluate its performance. The PRA exploits the efficiency and aggressive improvement of neighborhood search, but relies on a new path-relinking procedure and advanced population management strategies to navigate the search space effectively. To provide a comparator algorithm to the PRA, we embed the neighborhood search into a standard iterated local search algorithm (ILS). Extensive computational experiments on the benchmark instances show that the newly defined features have a significant and varied impact on the problem, and the performance of the PRA dominates that of the ILS algorithm.

Key words: vehicle routing and scheduling; weighted tardiness; release dates; path relinking; hybrid population-based metaheuristics.

1. Introduction

The vehicle routing problem (VRP) is a classical NP-hard combinatorial optimization problem. Starting with the paper of Dantzig and Ramser (1959), it has been studied widely for more than 50 years. The class of VRPs traditionally involve minimizing the total distance traveled by a number of vehicles to visit a set of customers. Commonly, the vehicles are assumed to operate from a single depot, and each customer's demand must be satisfied through a single visit from one of the vehicles, although various studies propose a significant variety of additional features and constraints. There are direct applications in transportation and logistics, and others in a variety of areas including manufacturing, communications, and the military. A more detailed discussion of application areas is given in the book of Toth and Vigo (2002), and more recently by Hoff et al. (2010).

In both the vehicle routing and production scheduling literature, a trend toward simultaneously addressing a greater proportion of the operational supply-chain is observed. Some notable examples include supply chain scheduling problems (Hall and Potts 2003), production-routing problems (Boudia et al. 2007), location-routing problems (Nagy and Salhi 2007), and inventory-routing problems (Coelho et al. 2014). Many of these consider integrating operational decisions about distribution with others from different elements of the supply-chain. A problem class of particular interest combines machine scheduling to process customer orders and vehicle routing to deliver the processed orders. A variety of these integrated problems are proposed in the literature, and some examples are described by Chang and Lee (2004), Chen and Vairaktarakis (2005), and Ullrich (2013). Reviews of these problems, and the wider class of integrated machine scheduling and distribution problems, are given by Chen (2010) and Ullrich (2013).

The problems considered above motivate us to define a VRP where orders become available for delivery to the customers at different times, which we call release dates. Delivery of the orders therefore requires a traditional VRP to be solved, except that the departure time of each vehicle from the depot is dependent on the release dates of the orders on its route. Moreover, the objective is to minimize a convex combination of transportation cost and customer service level, measured as the total distance traveled and total weighted tardiness associated with delivery, respectively. We refer to this problem as the VRP with job availability constraints (VRPJA). Situations in which release dates can arise include processing or production of the orders as described above, and also the arrival of the orders at the depot from a higher echelon VRP.

At the polar extremes of the objective, the VRPJA models the classical capacitated vehicle routing problem (CVRP), and a parallel machine total weighted tardiness problem with sequence-dependent set-up times (PWTPSST) and additional constraints. Although this generalization of the CVRP and the PWTPSST is of high practical and theoretical importance, to our knowledge it has not been addressed in the literature. A similar problem is being studied in parallel by Johar (2014), but no results have been published. Independently of this study, some preliminary results are presented by Cattaruzza et al. (2013) for a multi-trip VRP with hard time windows and release dates. The latter authors consider batched release dates, a purely transportation cost objective, and vehicles that can perform multiple routes.

We propose a novel path-relinking algorithm (PRA) to solve the VRPJA. This is based on a hybrid evolutionary framework rooted in scatter search and concepts proposed by Glover (1989) to enhance tabu search. Our algorithm utilizes the exploratory potential and adaptive memory of evolutionary algorithms, but relies on more intelligent solution recombination and population (diversity) management. A new relinking procedure is described to enable efficient exploration of the search space between the solutions in the population, and which introduces some controlled randomization. To intensify the search and improve the convergence of the algorithm, an efficient and powerful neighborhood search (NS) is applied to some solutions resulting from relinking. With the aim of assessing the performance of our proposed PRA against an alternative method, we extend the NS from the PRA to produce an iterated local search (ILS) algorithm. An ILS algorithm iteratively applies NS followed by a kick to escape the resulting local optima. Examples for a broad range of problem types show that ILS achieves competitive results (see Lourenço et al. 2010).

The remainder of the paper is organized as follows. § 2 defines the VRPJA formally and discusses its complexity. A thorough literature review is presented in § 3. In § 4, we describe the proposed PRA in detail. Computational experience is reported in § 5, where we also introduce the proposed set of benchmark instances and outline the comparator ILS algorithm. Lastly, § 6 concludes our findings.

2. Preliminaries

In this section, we introduce some notation and formally define the VRPJA. We also present some results on the worst-case complexity of the problem, and describe the results of a small experiment that assess its practical tractability.

2.1. Problem definition

The VRPJA is defined on a complete graph $G = (V, A)$, where the vertex set is $V = \{0, 1, \dots, n\}$ and arc set is $A = \{(i, j) \colon i, j \in V, i \neq j\}$. Vertex 0 corresponds to the depot, while $V' = V \backslash \{0\}$ represents the set of n customer vertices. For customer (vertex) i, the order to be delivered is characterized by an individual load $q_i \geq 0$, a due date $d_i \geq 0$, a release data $r_i \geq 0$, and a weight $w_i \geq 0$. A homogeneous fleet of m vehicles, each with capacity Q, is stationed at the depot (vertex) 0. Each arc $(i, j) \in A$ has an associated distance cost c_{ij} and travel time τ_{ij}. We assume each τ_{ij} includes the service time at vertex i (which is zero if i is the depot). If the triangle inequality holds for travel times, then it also holds for the modified τ_{ij} values that include equal service times.

A solution x to the VRPJA comprises m routes, one for each vehicle, where each customer is assigned to exactly one route and empty routes are allowed. Each route r is an elementary circuit in G that contains the depot. More precisely, for any route $r = 1, \dots, m$, let R^r be the set of customers visited in r and let $n^r = |R^r|$ be the number of customers in r. Then a route r can be represented by a permutation $(\sigma^r(1), \dots, \sigma^r(n^r))$ of the elements of R^r, where $\sigma^r(i)$ is the i^{th} customer to be visited, for $i = 1, \dots, n^r$. The route starts and ends at the depot, so we naturally set $\sigma^r(0) = \sigma^r(n^r + 1) = 0$. The solution is feasible if the capacity constraint $\sum_{i \in R^r} q_i \leq Q$ is satisfied for $r = 1, \dots, m$. Further, the vehicle for route r leaves the depot at time $\max_{i\in R^r} r_i$, so that the vehicle's earliest departure time is equal to the maximum release date of the orders for the customers visited in r.

The objective function f to be minimized is a convex combination of the total distance cost and total weighted tardiness, and is defined by

$$f(x) = \alpha \sum_{r=1}^{m} \sum_{i=0}^{n^r} c_{\sigma^r(i),\sigma^r(i+1)} + (1-\alpha) \sum_{r=1}^{m} \sum_{i\in R^r} w_i \max\{0, S_i - d_i\}, \tag{1}$$

where x is a solution, S_i is the time at which the service for customer i starts in solution x, $\max\{0, S_i - d_i\}$ is the tardiness for customer i and $0 \le \alpha \le 1$ defines the relative weight of the two objective function components. Since f is a non-decreasing function of the arrival times S_i, then vehicle idle time is never beneficial. Therefore, without loss of generality, we may assume that

$$S_{\sigma^r(i)} = \max_{j\in R^r}\{r_j\} + \sum_{j=1}^{i} \tau_{\sigma^r(j-1),\sigma^r(j)}, \forall\, i = 1, \ldots, n^r, r = 1, \ldots, m. \tag{2}$$

Sometimes, we allow solutions that are infeasible with respect to the capacity constraint and modify the objective defined in (1) to include a penalty cost. This yields a penalized objective function

$$f^p(x) = f(x) + \beta \sum_{r=1}^{m} \max\left\{0, \sum_{i\in R^r} q_i - Q\right\}, \tag{3}$$

where β is a parameter.

Figure 1 presents a small-sized instance I, together with two different solutions a and b. Instance I has $n = 5$, $m = 2$, $Q = 3$, and $q_i = w_i = 1$ for $i \in V'$. For each arc $(i, j) \in A$, we have $\tau_{ij} = d_{ij}$, and these values are shown on the edges. For each customer i, the values $[r_i, d_i]$ are given in the problem instance within Figure 1, and solutions a and b show the values (S_i, w_iT_i) for the given routing. The respective total distance traveled and total weighted tardiness are reported below the solutions. If $\alpha = 0.7$, then a is optimal and $f(a) = 7.6$ and $f(b) = 8.4$, but if $\alpha = 0.3$, then b is optimal and $f(a) = 8.4$ and $f(b) = 7.6$. Finally, if $\alpha = 0.5$, then both solutions are optimal and $f(a) = f(b) = 8$.

2.2. Problem complexity and tractability

In this section, we will establish the complexity of the VRPJA, and describe a small experiment attempting to solve a natural mixed integer programming formulation with a commercial solver.

We claim that the VRPJA is unary NP-hard for all values of α satisfying $0 \le \alpha \le 1$. For $\alpha = 1$, or for $0 < \alpha < 1$ and $d_i = \infty$ for $i \in V'$, then (1) reduces to the total distance traveled objective function. For this objective the problem is equivalent to the CVRP, which is unary NP-hard by reduction to the traveling salesman problem. For $\alpha = 0$, $Q = \infty$, $m = 1$, $r_i = 0$ and $w_i = 1$ for $i \in V'$, then the problem reduces to the total tardiness problem with sequence dependent set-up times. The total tardiness objective reduces to the makespan objective (Lenstra et al. 1977), and the

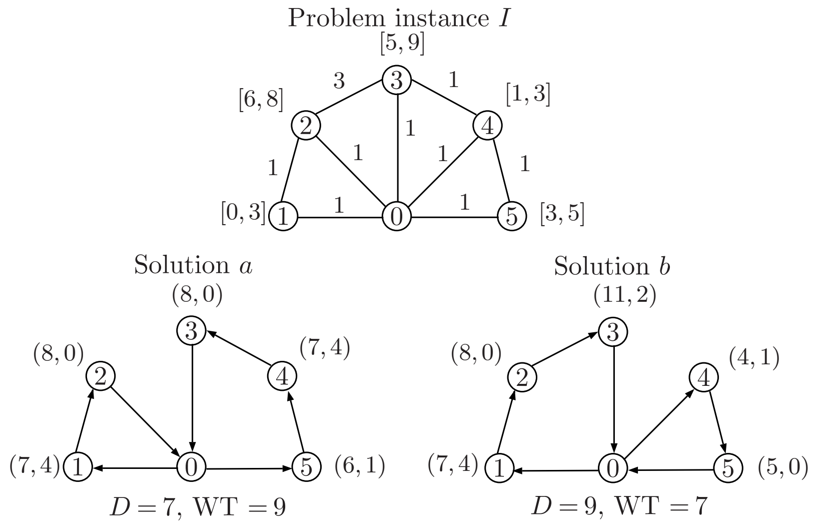

Figure 1 VRPJA instance and optimal solutions for different values of α.

resulting problem is unary NP-hard by reduction from the Hamiltonian path problem. Notice that the traveling salesman problem is reducible to the Hamiltonian path problem.

To explore the tractability of the VRPJA for exact methods, we attempt to solve the problem as a mixed integer program (MIP) using a commercial solver. We propose a classical *multi-commodity flow formulation* for the VRPJA, and this is provided in § A. We use the branch-and-cut algorithm of CPLEX 12.5.1.0, for which we trial various settings of the algorithm parameters. For instances generated with between 30 and 50 customers, where neighborhood search is used to provide an upper bound, the average relative gap between the best lower and upper bound after more than 5 hours of computation time is around 30%. This suggests that instances of these sizes or larger are challenging for exact methods, and that tighter bounds must be found for enumerative methods to be effective. We therefore rely on heuristic methods to tackle instances of practical size, and to provide tight upper bounds.

3. Literature review

In this section, we describe some VRPs with time windows that have similarities to the VRPJA. A review of some milestones in the development of heuristics for these VRPs is presented, where we seek to identify currently available techniques that are successful in providing high-quality solutions. Lastly, the main features of these heuristics are compared and contrasted.

3.1. Vehicle routing problems with time windows

One of the most widely-studied VRPs is the vehicle routing problem with (hard) time windows (VRPTW). In contrast to the VRPJA, the customer orders are assumed to be available for delivery immediately and (hard) time window constraints are defined by lower and upper limits on the start time of service at each customer. Additionally, the number of vehicles is usually unlimited and the objective is hierarchical, where minimizing the number of vehicles required is the primary objective and minimizing the total distance traveled is the secondary objective, and otherwise the number of vehicles is omitted from the objective. Notice that if a vehicle arrives at a customer before the lower limit of the customer's time window, then it must wait to begin service.

The vehicle routing problem with soft time windows (VRPSTW) is a generalization of the VRPTW, allowing violations of the time windows with penalties for earliness and tardiness. Again, if the number of vehicles is unlimited, then the objective is again hierarchical, where minimizing the number of vehicles is the primary objective and minimizing the total distance plus the weighted earliness and tardiness is the secondary objective. Fu et al. (2008) provide some motivations for the VRPSTW over the VRPTW, and a classification of six types of soft time windows. Popular examples are type 1 that relaxes the upper limit on the time window, type 4 that is the same as type 1 but also introduce (hard) time deadlines for each customer, and type 2 that relaxes both the upper and lower limits on the time windows. Note that the studies in the literature on the VRPSTW predominately use equal earliness and tardiness weights in the objective.

The significant practical value of VRPs motivates their study, including the design of exact algorithms that can prove optimality in reasonable computational time. For the VRPTW, this is currently possible for most benchmark instances with 100 customers, but many with 200 or more cause significant difficulties. For the interested reader, reviews of exact methods for the VRPTW are given by Desaulniers et al. (2010), and more recently by Desaulniers et al. (2014). The literature is generally sparser and more fragmented for the VRPSTW, and to the best of our knowledge a branch-and-cut algorithm has not been proposed for any variant. The intractability of the problem is demonstrated by the branch-and-price algorithm for the type 2 VRPSTW with a fixed number of vehicles proposed by Liberatore et al. (2010), which cannot solve most instances with 75 customers or more in one hour of computation time and for some instances finds no valid lower bound. The most successful exact approaches currently rely heavily on implicit route enumeration, and therefore instances that have a greater number of feasible routes are generally the most difficult.

As a result of the complexity of these VRPs, a significant variety of heuristics are proposed in the literature. The earliest of these are constructive and neighborhood search (NS) heuristics, and many can be found in the comprehensive survey of Bräysy and Gendreau (2005a). Metaheuristic frameworks have since received significant attention, as evidenced for the VRPTW by the surveys of Bräysy and Gendreau (2005b), Gendreau and Tarantilis (2010), and by Desaulniers et al. (2014), and for both hard and soft time windows in the extensive survey of Vidal et al. (2013a). The evolutionary algorithms paradigm receives particular attention, and dedicated surveys are given by Bräysy et al. (2004) and more recently by Potvin (2009). There is also a notable trend in these surveys towards hybridization of strategies and components of different metaheuristics, and this aims to balance the individual advantages and disadvantages.

3.2. Single solution based heuristics

Many of the earlier implementations of metaheuristics for the VRPTW and VRPSTW rely on single solution based and neighborhood-centric frameworks. These are often extensions of NS and differ in the techniques used to achieve diversification. Although historically the initial solution had a significant impact on the objective quality of the final solution in many of these heuristics, it is now common to use random or simple constructive procedures and the initial solution has little influence on the overall solution quality. We now discuss some important implementations of single solution based metaheuristics.

Tabu search (TS) is historically popular, with notable TS heuristics for the VRPTW and VRPSTW being proposed by Cordeau et al. (2001) and Fu et al. (2008), respectively. These heuristics both require a definition of solution attributes to construct the tabu list, and the TS and guided local search (GLS) hybrid algorithm of Cordeau et al. (2001) also relies on these for defining the guiding penalties. In contrast, the TS algorithm of Fu et al. (2008) uses a strategy of exploring a random set of neighbors from the union of several neighborhoods in each iteration. Solutions are also represented as a single giant tour with the depot between routes, and although this slightly increases the number of solutions it also results in greater connectivity of the search space. Cordeau et al. (2001) also increase the connectivity of the search space by relaxing a number of constraints, such as those defined by the vehicle capacities and time windows. The amounts of infeasibility are individually weighted and penalized in the objective function, and are periodically adapted to facilitate controlled exploration of the search space and to adjust the numbers of infeasible solutions. More details on different relaxations for the time windows can be found in the review of Vidal et al. (2015a).

Another elementary framework with some popularity is iterated local search (ILS), and ILS heuristics are proposed by Cordeau and Maischberger (2012) for the VRPTW and by Ibaraki et al. (2008) for the VRPSTW with convex penalties on time-window violations. The ILS algorithm of Ibaraki et al. (2008) uses an internal variable neighborhood search (VNS) algorithm that explores a variety of neighborhoods of increasing size, and Cordeau and Maischberger (2012) use a slightly improved version of the TS and GLS hybrid algorithm of Cordeau et al. (2001). The differences between these internal heuristics are notable, and contrast the efficient search of a range of neighborhoods with fewer neighborhoods and diversification techniques. These differences also require different types of perturbation or kick. Accordingly, Ibaraki et al. (2008) choose a random neighbor from a larger neighborhood than any used in the VNS algorithm, and Cordeau and Maischberger (2012) use a large neighborhood (LN) or ruin-and-recreate heuristic. Ibaraki et al. (2008) also describe an efficient route-first and schedule-second evaluation technique in the NS, which decomposes the problem into finding a set of routes and then scheduling the start times of customer services for each route. This approach is discussed in more detail in Section § 3.5.

3.3. Hybrid evolutionary algorithms

Recently, much of the research focus shifts to population-based metaheuristics, which have advantages and drawbacks. Specifically, these methods usually offer greater adaptive control over the balance of diversity and intensity, although they are often observed to converge to good-quality solutions slowly. Hybridization of both single-solution and population-based metaheuristics is often proposed to balance such strengths and weaknesses. Hybrid evolutionary algorithms (HEAs) are particularly popular, and use single solution-based or NS heuristics to improve solutions generated as combinations of solutions stored in a population. These HEAs are the subject of this section and some important contributions are discussed.

In the majority of the HEAs described below, the number of vehicles is fixed and if the objective is hierarchical it is not handled explicitly. If this is the case, then either a route minimization heuristic is applied initially to fix the number of routes as suggested by Nagata et al. (2010), or an iterative approach is used to increase or decrease the number of vehicles depending on whether a feasible solution is found as suggested by Hashimoto and Yagiura (2008). Notice that this increases the similarity between the objective function of the problem solved by these HEAs and the VRPJA.

A very early HEA is proposed by Taillard et al. (1997) for the type 1 VRPSTW with a fixed number of vehicles, and is described as a hybrid algorithm based on adaptive memory programming (AM), TS and GLS. The heuristic maintains a steady-state and elitist population of solutions, and in each iteration an offspring solution is constructed probabilistically from routes in the population. Each offspring is then improved by the TS and GLS hybrid algorithm before replacing the worst solution in the population if it has better objective value. The authors also suggest decomposing the instance, where the decomposition is based on a current solution, and they describe an iterative decomposition and recomposition (ID) framework using their guided TS to improve the initial population of solutions.

The heuristic of Taillard et al. (1997) can be characterized as a hybrid genetic algorithm (HGA) with a multi-parent crossover, but other studies typically give greater importance to the management and recombination of solutions in the population. For example, the HGA of Berger et al. (2003) for the VRPTW evolves two subpopulations independently, which individually minimize the total distance traveled, and the weighted infeasibility and number of vehicles required. Individual fitness functions are used for parent selection, and the subpopulations interact when a feasible solution with a lower number of vehicles is identified. A similar approach is described by Vidal et al. (2013b) in their HGA for the VRPTW and VRPSTW (see Vidal et al. 2014). Subpopulations for feasible and infeasible solutions are managed independently in their HGA, but the interaction is through selecting solutions for recombination from the union of these subpopulations.

Given the use of heuristics to improve the offspring solutions, the diversity of a population may fall quickly with the solutions becoming more similar. A simple approach which reduces the chances of this is suggested by Nagata et al. (2010) in their HGA for the VRPTW, and only considers replacing parent solutions with their offspring. If a distance between solutions is defined, then the diversity of a solution can be measured by its distance to solutions in the population. Vidal et al. (2013b) use this approach and introduce a measure of diversity into the fitness function, which is used for parent selection and population management. This permits the diversity of the population to be managed adaptively, and if the diversity falls, then more diverse offspring solutions will survive population management and be selected for recombination.

A variety of methods for recombining solutions are proposed, and often these technique vary quite significantly. For example, Hashimoto and Yagiura (2008) propose a path-relinking algorithm for the VRPTW. In the path-relinking (PR) procedure, the solution attributes are defined as the edges and the distance between two solutions as the number of different edges. By iteratively reducing the distance between an initial and a guiding solution using inter- and intra-route neighborhoods, then a trajectory of offspring solutions is constructed. Another type of specialized crossover operator is proposed by Nagata et al. (2010) that builds on the powerful edge assembly-based crossover (EAX) proposed for the CVRP and TSP in previous work by the first author. Similar to PR, the EAX has significantly less randomization and more structural consideration than most traditional crossover operators, and produces a set of offspring solutions. In contrast, a traditional ordered crossover (OX) is used by Vidal et al. (2013b), where solutions are represented as a single giant tour without the depot. The optimal partitioning into routes is then found using an adaptation of the dynamic programming split algorithm presented by Prins (2004), which runs in polynomial time.

3.4. Comparison of heuristics

Certain algorithmic strategies and components are frequently used in heuristics for the VRPs considered, and there are combinations of these which seem popular. To observe this, a comparison of the main features of the heuristics is presented in Table 1. The columns from left to right are: the algorithm; whether the heuristic can be applied to a version of the VRPSTW (as indicated by a "Y"); the main metaheuristic frameworks; how insert, swap, 2-Opt, 2-Opt* and large neighborhoods are used; details of NS as defined by the exploration strategy ,whether infeasible solutions are allowed (as indicated by a "Y") and the types of neighborhood pruning; the type of solution recombination and whether infeasible offspring solutions are produced and how they are treated; and the population management strategy.

From Table 1, we observe that all the heuristics feature NS, and except for F08 (Fu et al. 2008) they are hybrids. This suggests that NS is vital in achieving competitive performance, and that a variety of strategies in combination performs best for these complex problems. In the NS components, the exploration strategies are generally balanced between accepting the best-improving or first-improving neighbor. Interestingly, for V13 (Vidal et al. 2013b), this HGA achieves more div-

Table 1 Features of heuristics for the VRPTW and VRPSTW

	STW	Frameworks	Neighborhood search								Population		
			Neighborhoods					Str.	Infeas.	Prun.	Recombination		Management
			Ins.	Swap	2-O	2-O*	LNs				Type	Infeas.	
T97	Y	AM, ID, GLS, TS	+	+			−	B		A	Insertion		Steady-state, elit.
B03		HGA	+				x	F	Y	D	Insertion	Repair	Periodic, elit., tailored fit., subpops
F08	Y	TS	x	x	x	+		B	Y	R	—	—	—
H08		PRA	x	+		+		F	Y	S, D	PR	Accept	Steady-state, elit., unique
I08	Y	ILS, VNS	x′	x′	−	+		A+	Y	S, D	—	—	—
N10		HGA	x′	x′		+		F		D	EAX	Repair	Steady-state, elit., repl. parent
C12		ILS, GLS, TS	+				x	B	Y		—	—	—
V13	Y	ID, HGA	x′	x′	−	+		RF	Y	S, D	OX + split	Copy + rep.	Periodic, diversity in fit., subpops

Algorithms. T97: Taillard et al. (1997); B03: Berger et al. (2003); F08: Fu et al. (2008); H08: Hashimoto and Yagiura (2008); I08: Ibaraki et al. (2008); N10: Nagata et al. (2010); C12: Cordeau and Maischberger (2012); V13: Vidal et al. (2013b).
Frameworks. AM: Adaptive memory programming; GLS: Guided local search; HGA: Hybrid genetic algorithm; ID: Iterative decomposition-recomposition; ILS: Iterated local search; PRA: Path-relinking algorithm; TS: Tabu search.
Neighborhoods. Ins: Insert neighborhood; Swap: Swap neighborhood; 2-O: 2-Opt; 2-O*: 2-Opt*; LNs: Large neighborhoods.
Neighborhood features. −: Intra-route; +: Inter-route; x: Intra- and inter-route; ′: original and reverse subsequences both considered.
Neighborhood exploration. A+: Aspiration plus; B: Best improvement; F: First improvement; RF: Random first improvement.
Neighborhood pruning. A: Approximate; D: Dynamic; R: Random; S: Static.
Recombination. EAX: Edge assembly crossover; OX: Order-based crossover; PR: Path-relinking.

ersity by exploring neighbors from the union of a number of neighborhoods in a random order and accepting the first-improving neighbor. Also, for I08 (Ibaraki et al. 2008), their ILS and VNS hybrid algorithm uses an aspiration plus (A+) strategy, in which a neighborhood is explored until it is complete or the computation time taken approximately exceeds that used for preprocessing. It would appear that the benefits of efficiently searching neighborhood in each iteration outweigh potential improvements in objective value that may occur by finding the best-improving neighbor.

All of the more recent heuristics use controlled exploration of infeasible solutions, and except N10 (Nagata et al. 2010) they all explore infeasible solutions in the NS component. Alongside the adaptive control of the penalties on infeasibility, considering some infeasible solutions appears to be critical for instances with tight constraints and complex objectives. Note that all of these heuristics relax the time window constraints, and this increases the similarity between the VRPTW and VRPSTW, and therefore the VRPJA.

A variety of solution recombination approaches are suggested in the HEAs considered, and all except V13 (Vidal et al. 2013b) explicitly consider the structure of solutions more than traditional crossovers. This would suggest that diversity is important in creating offspring solutions, but often the loss of the structural properties of the solution from traditional crossovers obscures the translation of information from the parents to the offspring. It is also notable that both H08 (Hashimoto and Yagiura 2008) and N10 (Nagata et al. 2010) produce a set of offspring from each recombination, and V13 implicitly enumerates the set of offspring resulting from different partitions of the giant tour created by their crossover.

The management of the population in earlier HEAs is purely elitist, but different techniques are increasingly introduced to maintain and more carefully introduce diversity to the population. This more recent development appears to be important for achieving current performance levels and particularly for improving the robustness of the results. The adaptive control of population diversity as used in V13 (Vidal et al. 2013b) is a particularly important development, which can enable the population to escape the equivalent of local optima when the diversity of the population falls.

If infeasible offspring solutions are produced by the HEAs, then often a repairing NS is applied that attempts to find a feasible solution whilst retaining objective quality. For V13 (Vidal et al. 2013b), the HGA does not repair solutions automatically but a with a certain probability. In both V13 and B03 (Berger et al. 2003), the HGAs use infeasible solutions more explicitly and a separate subpopulation of infeasible solutions is evolved. These strategies and techniques suggest that infeasible solutions are still important in both the population and NS, although finding feasible solutions quickly can be helpful for problems with tight constraints such as the VRPTW.

3.5. Neighborhoods

For any NS, the time complexity of evaluating the objective value of a neighbor is critical to the efficiency. For the VRPTW, the difference in objective function for neighborhoods based on exchanging a bounded number of arcs can be evaluated in $O(1)$ time. Importantly, Kindervater and Savelsbergh (1997) prove that the use of preprocessing requiring $O(n)$ time allows the feasibility of the time windows (and route loads and durations) to be evaluated in $O(1)$ time. Furthermore, Vidal et al. (2013b) show that the use of preprocessing requiring $O(n)$ time permits a popular measure of time window infeasibility termed time-warp to be also evaluated in $O(1)$.

For the types of VRPSTW in which both earliness and tardiness are permitted, the most efficient NS approaches use the route-first and schedule-second technique. This is because even for a given route the schedule of customer service start times is non-trivial. Dynamic programming is used to construct piecewise-linear cost functions in a preprocessing phase, and structures such as heaps or balanced binary trees are used to store and query these efficiently. If only earliness or tardiness is permitted, then the schedule is trivial and dynamic programming is unnecessary, but the cost functions are still constructed. For all the types of soft time windows described and neighborhoods defined by a bounded number of arc exchanges, Ibaraki et al. (2008) show that the use of preprocessing requiring $O(n^2)$ time allows an evaluation that requires $O(\log n)$ time. Further, for any convex piecewise-linear penalty function of earliness and tardiness, these complexities increase to $O(n \sum_i h_i)$ and $O(\log \sum_i h_i)$ respectively, where h_i is the number of segments of the penalty function of customer $i \in V'$. Note that these general techniques are first described independently by Ibaraki et al. (2005), Ergun and Orlin (2006) and Hendel and Sourd (2006) for the VRP with general time windows, and the single machine weighted tardiness and earliness-tardiness scheduling problems respectively. More details on the implementations and other types of functions on earliness and tardiness can be found in the comprehensive review of Vidal et al. (2015b).

A direct comparison of the neighborhoods used in the heuristics described is complicated by the use of different exploration and pruning strategies. Instead a high-level view is adopted and five underlying neighborhoods are identified: *insert*, changing the position of a subsequence (of consecutive customers); *swap*, exchanging the positions of two distinct subsequences; *2-Opt*, removing two arcs and reconnecting the associated route segments differently; *2-Opt**, exchanging subsequences at the ends of two routes; and large neighborhoods, which usually have exponential size and include ruin-and-recreate heuristics. Most of the traditional neighborhoods proposed are subsets or compositions of subsets of the first four of these neighborhoods. Although 2-Opt* is the least general and a subset of inter-route 2-Opt, it is included separately due to its apparent importance and frequency of use.

We notice in Table 1 that a considerable number of neighborhoods are frequently used, and this applies even in the HEAs. This again hints at the underlying complexity of the problems, and the difficulty of designing an efficient and effective NS. A greater variety of neighborhoods are generally used for the VRPSTW, and this may be attributable to the increased complexity and size of the search space. Subsets of the insert and swap neighborhoods are the most popular, and the use of both inter- and intra-route neighborhoods is also increasing. The intra-route subset of 2-Opt is used in three of the four heuristics for the VRPSTW, and one of these heuristics is also the most recently proposed of those considered for the VRPTW. Exploring neighbors resulting from considering the subsequences involved in the move in reverse order is also gaining popularity for both insert and swap. Possible reasons for this include instances with wider time windows that allow such a reversal to be reduce the objective value, and the additional diversification, especially when used in VNS and first-improvement methods.

LNs are used in three of the heuristics, and a general trend is observed toward larger neighborhoods. For this reason, considerable efforts are made to achieve efficiency in NS and this is evidenced by the range of pruning techniques that either statically or dynamically restrict the neighbors considered, and the exploration and evaluation strategies that are used. Uniquely, Taillard et al. (1997) propose selecting a set of neighbors in each iteration using an approximate evaluation that requires $O(1)$ time, and Fu et al. (2008) randomly restrict the neighborhoods. These general issues again reveal the important balance between the greater potential for improving solutions and the efficiency of exploring the neighborhood.

4. Path-relinking algorithm

PRAs and the closely related scatter search are EAs incorporating concepts for strategic recombination of solutions and population diversity management. Formalized by Glover et al. (2000), these frameworks share many features of HGAs, such as a population of solutions, recombination, and NS to improve offspring solutions. In contrast, solutions are recombined in an intelligent manner by progressively adapting an initial solution to involve more features of a set of guiding solutions. This is termed path-relinking or simply relinking. At each step of relinking a solution is created, and the resulting solutions form a trajectory connecting the initial and guiding solutions. A more detailed discussion of the variations, design choices and some applications of PRAs are given by Resende et al. (2010).

We propose a PRA for the VRPJA, and a general outline is given in Algorithm 1. Notable features of our method are the successful hybridization of concepts from different evolutionary frameworks, the efficient and effective NS, and the novel relinking procedure. The algorithm relies on various parameters: β is an infeasibility penalty that is used in equation (3) (see § 4.5 for our method of computing β), μ is the reduced size of the population after updating, λ is the number of additional solutions in the population before the population is reduced in size, γ is the number of relinkings that are performed without improving the best-known solution until the population is refreshed, κ is the number of offspring that are generated between updates to the value of β, and $T_{\max}$ is the computation time limit that is used as a termination criterion.

Algorithm 1 Path-relinking algorithm

```
1: Set the parameter values $\mu$, $\lambda$, $\gamma$, $\kappa$, $T_{\max}$
2: Generate an initial population $P$ and compute the value of $\beta$
3: while running time $< T_{\max}$
4:     Select initial and guiding solutions $x_I$ and $x_G$
5:     Construct trajectory from $x_I$ to $x_G$ and select a subset $S$ of these solutions (relinking)
6:     Improve solutions in $S$ by applying NS
7:     Set $P = P \cup S$
8:     if $|P| \geq \mu + \lambda$ then
9:         Reduce population size to $\mu$ (population management)
10:    if Number of relinkings without improving both best and best feasible solution $\geq \gamma$ then
11:        Refresh population
12:    if Number of offspring since most recent penalty update or population refresh $\geq \kappa$ then
13:        Update infeasibility penalty $\beta$
14: return Best feasible solution
```

A population comprising a set of feasible and infeasible solutions P is evolved through iterative relinking, application of NS and population management. Two parent solutions x_I and x_G are selected and relinked (§ 4.3), and a subset S of resulting offspring solutions are improved through NS (§ 4.1). These improved solutions then enter P if not already present. P is reduced to size μ when it reaches size $\mu + \lambda$, by iteratively removing the solution with worst fitness (§ 4.4). Evaluation of the fitness, used in population reduction and selection of solutions for relinking, follows the approach developed in Vidal et al. (2012). More precisely, fitness is measured as a combination of rank in the population for both the objective function value and diversity (§ 4.2). If the best feasible solution and best solution do not improve after γ relinkings, then the population is refreshed (§ 4.4). Infeasible solutions encountered during the search are penalized by the level of infeasibility, as defined in (3). The associated infeasibility penalty weight β is adaptively updated each time κ offspring are generated since β has been updated or the population has been refreshed (§ 4.5). The algorithm terminates once the running time exceeds $T_{\max}$.

4.1. Neighborhood search

We apply a NS to improve and evaluate solutions before considering their introduction to the population. This is an aggressive improvement procedure which enables fast progression to solutions with low objective values. The neighborhoods used are popular for both VRPs and machine scheduling problems. In hybrid approaches such as our PRA, NS is applied frequently and represents a large proportion of the computation time. In the later parts of the section, we describe an efficient move evaluation procedure and a memory structure to reduce this time requirement.

Let n^r and $(\sigma^r(1),\dots,\sigma^r(n^r))$ be defined as in § 2.1. We consider subsets of four well-known neighborhood structures, where the sizes are restricted by the parameters s_1 and s_2.

- N_1 - Insert (or relocate): Remove $(\sigma^r(i),\dots,\sigma^r(j))$ and insert immediately after $\sigma^{r'}(i') \notin (\sigma^r(i-1),\sigma^r(i),\dots,\sigma^r(j))$, where $1 \le i \le j \le n^r$, $0 \le i' \le n^{r'}$, $j-i < s_1$ and $1 \le r, r' \le m$. The reverse order of $(\sigma^r(i),\dots,\sigma^r(j))$ is also considered.
- N_2 - Swap: Exchange the positions of $(\sigma^r(i),\dots,\sigma^r(j))$ and $(\sigma^{r'}(i'),\dots,\sigma^{r'}(j'))$, where $1 \le i \le j < n^r$, $1 \le i' \le j' < n^{r'}$, $j-i < s_1$, $j'-i' < s_1$, $1 \le r \le r' \le m$, and the subsequences are disjoint. The reverse order of either or both subsequences are also considered.
- N_3 - Inter-route 2-Opt: Reverse the order of $(\sigma^r(i),\dots,\sigma^r(i'))$, where $1 \le i < i' \le n^r$, $i'-i < s_2$ and $1 \le r \le m$.
- N_4 - 2-Opt*: Exchange $(\sigma^r(i),\dots,\sigma^r(n^r))$ and $(\sigma^{r'}(i'),\dots,\sigma^{r'}(n^{r'}))$, where $1 \le i \le n^r+1$, $1 \le i' \le n^{r'}+1$, $i > 1$ or $i' > 1$, $i \le n^r$ or $i' \le n^r$ and $1 \le r < r' \le m$. By not considering $i = i' = 1$ and $i = i' = n^r+1$, the cases where all customers are exchanged from r and r' and no customers are exchanged from r and r', respectively, are eliminated.

The neighborhoods above can be described in terms of either vertex relocation, or the λ-Opt neighborhood proposed by Lin (1965). Neighborhoods N_1 and N_2 operate on one or two routes, whereas N_3 operates on a single route and N_4 on two routes. The inherent symmetry in neighborhoods N_2 and N_4 is removed by specifying that $r \le r'$ in N_2 and $r < r'$ in N_4. Neighborhoods N_1 and N_4 can create an additional route by considering one empty route, or remove a route by combining two routes. Initial experiments with a number of other traditional neighborhoods suggest that a combination of those described perform best. Interestingly, this corroborates the suggestions from our analysis of commonly-used neighborhoods in § 3.5.

If we fix s_1 and s_2, then neighborhoods N_1, N_2 and N_4 all have size $O(n^2)$, while N_3 has size $O(n)$. We explore the composite neighborhood $N = N_1 \cup N_2 \cup N_3 \cup N_4$ as follows. We define subsets of each neighborhood by each pair of vertices for $\sigma^r(i)$ and $\sigma^{r'}(i')$ in the definitions above, where we set $\sigma^r(i) = \sigma^{r'}(i)$ for N_3, and term these *sub-neighborhoods*. We search the composite neighborhood by considering pairs (i,j) in a random order, where i and j are customers or a depot for each route. For each pair (i,j), we evaluate the associated composite sub-neighborhood, and we accept the best neighbor of the first sub-neighborhood that contains an improving neighbor. Inspired by Nagata and Bräysy (2008) and Vidal et al. (2012), our initial experiments with this approach show an effective balance between computational speed and improvement to the objective value is achieved.

A computational bottleneck is created in NS by the significant number of moves that must be evaluated. This is especially significant for the tardiness objective that often leads to linear evaluation, or constant-time approximate evaluation (see, for example, Taillard et al. 1997).

To reduce the time complexity of move evaluation, we extend the evaluation by "concatenation" approach that is developed by Kindervater and Savelsbergh (1997), and more recently expounded by Vidal et al. (2015b). As an example, given routes r and r', let us assume that an insert move is applied, such that $(\sigma^r(i),\dots,\sigma^r(j))$ is inserted after customer $\sigma^{r'}(i')$. If we define $\oplus$ as the concatenation operator, then the two modified routes of this neighboring solution can be found as $(\sigma^r(0),\dots,\sigma^r(i-1))\oplus(\sigma^r(j+1),\dots,\sigma^r(n^r+1))$ and $(\sigma^{r'}(0),\dots,\sigma^{r'}(i'))\oplus(\sigma^r(i),\dots,\sigma^r(j))\oplus(\sigma^{r'}(i'+1),\dots,\sigma^{r'}(n^{r'}+1))$.

The following preprocessed data is used to evaluate much of the information about a route:

- $D(\sigma^r(i),\dots,\sigma^r(j))$ is the total distance traveled while visiting $(\sigma^r(i),\dots,\sigma^r(j))$ in order;
- $Q(\sigma^r(i),\dots,\sigma^r(j))$ is the total load for $(\sigma^r(i),\dots,\sigma^r(j))$;
- $L(\sigma^r(i),\dots,\sigma^r(j))$ is the duration of the route that visits $(\sigma^r(i),\dots,\sigma^r(j))$ in order;
- $R_{\max}(\sigma^r(i),\dots,\sigma^r(j))$ the latest release date for customers in $(\sigma^r(i),\dots,\sigma^r(j))$.

For a single customer $i\in V'$, we have $D(i)=0$, $Q(i)=q_i$, $L(i)=0$, $R_{\max}(i)=r_i$. Preprocessing this data or evaluating a concatenation of subsequences is then achieved by successively applying the following operators. If $\pi=(\sigma^r(i),\dots,\sigma^r(j))\oplus(\sigma^{r'}(u),\dots,\sigma^{r'}(v))$ for suitably defined i, j, u, v, r and r', then

$$D(\pi)=D(\sigma^r(i),\dots,\sigma^r(j))+c_{\sigma^r(j)\sigma^{r'}(u)}+D(\sigma^{r'}(u),\dots,\sigma^{r'}(v)), \tag{4}$$

$$Q(\pi)=Q(\sigma^r(i),\dots,\sigma^r(j))+Q(\sigma^{r'}(u),\dots,\sigma^{r'}(v)), \tag{5}$$

$$L(\pi)=L(\sigma^r(i),\dots,\sigma^r(j))+\tau_{\sigma^r(j)\sigma^{r'}(u)}+L(\sigma^{r'}(u),\dots,\sigma^{r'}(v)), \tag{6}$$

$$R_{\max}(\pi)=\max\{R_{\max}(\sigma^r(i),\dots,\sigma^r(j)),R_{\max}(\sigma^{r'}(u),\dots,\sigma^{r'}(v))\}. \tag{7}$$

Proposition 1. *Preprocessing using the operators* (4)-(7) *requires $O(n^2)$ time. For any neighborhood in which a neighboring solution is obtained by moving a constant number of subsequences to different positions, checking feasibility and evaluating its total distance cost require $O(1)$ time.*

Proof. There are $O(n^2)$ subsequences of consecutive customers. Using (4),

$$D((\sigma^r(i),\dots,\sigma^r(j))=D((\sigma^r(i),\dots,\sigma^r(j-1))\oplus D(\sigma^r(j)). \tag{8}$$

Thus, the preprocessed distance data is computed for each subsequence in $O(1)$ time, and for all subsequences in $O(n^2)$ time. A similar argument applies for the preprocessed load, duration and latest release date data.

For a neighboring solution involving the movement of a constant number of subsequences, equations (4) and (5) are applied to obtain the distance and load for each route in $O(1)$ time. Thus, the capacity constraint can be checked in $O(1)$ to ascertain whether the neighboring solution is feasible, and its total distance cost can also be evaluated in $O(1)$ time. $\square$

Note that for the additive preprocessing data, only $O(n)$ subsequences are needed to find the data on any subsequence. Considering distance, for example, $D(\sigma^r(1),\dots,\sigma^r(j))$ for $j=1,\dots,n^r$ and $r=1,\dots,m$ can be computed in $O(n)$ time. From these values, we can evaluate $D(\sigma^r(i),\dots,\sigma^r(j))$ for any i and j in $O(1)$ time using

$$D(\sigma^r(i),\dots,\sigma^r(j)) = D(\sigma^r(1),\dots,\sigma^r(j)) - D(\sigma^r(1),\dots,\sigma^r(i)). \tag{9}$$

We now consider how to evaluate the weighted tardiness efficiently. First, let $E_{\sigma^r(i')}$ be the earliness of customer $\sigma^r(i')$ in a subsequence of route r. For example, if the subsequence of r begins with customer $\sigma^r(i)$, where $1 \le i \le i' \le n^r$, then $E_{\sigma^r(i')} = \max\{0, d_{\sigma^r(i')} - L(\sigma^r(i),\dots,\sigma^r(i'))\}$. We also define the function $g^r_{i,j}(t)$ for $t \ge 0$ to be the total weighted tardiness for customers in the sequence $(\sigma^r(i),\dots,\sigma^r(j))$, where the service for customer $\sigma^r(i)$ starts at time t, for all i and j satisfying $1 \le i \le j \le n^r$ and $r=1,\dots,m$. We first focus on the computation of the values $g^r_{i,n^r}(t)$.

The functions $g^r_{i,j}(t)$ are non-decreasing, continuous and piecewise-linear, and we adapt the method of Ergun and Orlin (2006) for constructing the functions. We can represent $g^r_{i,n^r}(t)$ by a sequence $(b_0, g^r_{i,n^r}(b_0)), (b_1, g^r_{i,n^r}(b_1)),\dots,(b_k, g^r_{i,n^r}(b_k))$ of $(t, g^r_{i,n^r}(t))$ values at which the gradient of the function changes, where $k \le n^r - i + 1$. Although the values of k and $b_1,\dots,b_k$ depend on i and r, for notational conciseness we omit introducing subscripts i and superscripts r. Each pair $(b_h, g^r_{i,n^r}(b_h))$ for $h=0,1,\dots,k$ is referred to as a *breakpoint*, where $b_0 = 0$ is the first point in the domain of the function $g^r_{i,n^r}(t)$. Using this representation, the function can be expressed as

$$g^r_{i,n^r}(t) = \begin{cases} g^r_{i,n^r}(b_h) + \frac{g^r_{i,n^r}(b_h+1) - g^r_{i,n^r}(b_h)}{b_{h+1}-b_h}(t-b_h), & \text{if } b_h \le t < b_{h+1},\ h=0,\dots,k-1, \\ g^r_{i,n^r}(b_k) + \sum_{h=i}^{n^r} w_{\sigma^r(h)}(t-b_k), & \text{if } t \ge b_k. \end{cases} \tag{10}$$

For a given t, knowledge of the breakpoints allows the value of $g^r_{i,n^r}(t)$ to be computed in $O(\log n)$ time using bisection search to find the segment corresponding to the value of t.

Our procedure for computing breakpoints for $g^r_{i,n^r}(t)$ for $1 \le i \le n^r$ and $1 \le r \le m$ is given in Algorithm 2. This algorithm relies on the observation that for a subsequence $(\sigma^r(i),\dots,\sigma^r(n_r))$, the $b_1, b_2,\dots,b_k$ correspond to the distinct values among $E_{\sigma^r(i)},\dots,E_{\sigma^r(n_r)}$, and that the order of these earliness values does not change if the time that service starts at $\sigma^r(i)$ is changed. Our algorithm sets $\nu = (\nu(1),\dots,\nu(n_r - i + 1))$ to be a sequence of the customers of $E_{\sigma^r(i)},\dots,E_{\sigma^r(n^r)}$ ordered by non-decreasing earliness, and $\bar{\nu}$ stores the sequence ν used in the previous iteration.

Having used Algorithm 2 to compute the breakpoints, function $g^r_{i,n^r}(t)$ is specified by (10). The function $g^r_{i,j}(t)$ for $j=1,\dots,n^r$ is then defined by

$$g^r_{i,j}(t) = g^r_{i,n^r}(t) - g^r_{j+1,n^r}(t + L(\sigma^r(i),\dots,\sigma^r(j)) + \tau_{\sigma^r(j),\sigma^r(j+1)}). \tag{11}$$

This allows us to define $Z(\sigma^r(i),\dots,\sigma^r(j))$ as the total weighted tardiness for the customers $(\sigma^r(i),\dots,\sigma^r(j))$ visited in order, starting from the depot. That is computed using

$$Z(\sigma^r(i),\dots,\sigma^r(j)) = g^r_{i,j}(R_{\max}(\sigma^r(i),\dots,\sigma^r(j)) + \tau_{0,\sigma^r(i)}), \tag{12}$$

Algorithm 2 Procedure to create functions $g^r_{i,n^r}(t)$

```
1:  for all $1 \le r \le m$
2:      Set $i = n^r$, and $\nu = \emptyset$
3:      while $i \ge 1$
4:          Set $E_{\nu(h)} = \max\{0, E_{\nu(h)} - \tau_{\sigma^r(i)\sigma^r(i+1)}\}$ for $1 \le h < n^r - i$
5:          Set $E_{\sigma^r(i)} = \max\{0, d_{\sigma^r(i)} - L(\sigma^r(i), \dots, \sigma^r(n^r))\}$
6:          Form sequence $\bar{\nu} = (\bar{\nu}(1), \dots, \bar{\nu}(n^r - i + 1))$ by inserting $\sigma^r(i)$ in $\nu$ so that
            $E_{\bar{\nu}(1)} \le \cdots \le E_{\bar{\nu}(n^r-i+1)}$ and set $\nu = \bar{\nu}$
7:          Set $k = 0$, $b_k = 0$
8:          Choose the largest index $j \in \{1, \dots, n^r - i + 1\}$ such that $E_{\nu(j)} = E_{\nu(1)}$
9:          Compute $W = \sum_{h=1}^{j} w_{\nu(h)}$
10:         if $E_{\nu(j)} = 0$ then
11:             Compute $g^r_{i,n^r}(b_k) = \sum_{i'=i}^{n^r} w_{\sigma^r(i')} \max\{0, L(\sigma^r(i), \dots, \sigma^r(i')) - d_{\sigma^r(i')}\}$
12:         else
13:             Set $g^r_{i,n^r}(b_k) = 0$
14:         while $j < n^r - i + 1$
15:             Set $k = k + 1$ and $b_k = E_{\nu(j+1)}$, and compute $g^r_{i,n^r}(b_k) = g^r_{i,n^r}(b_{k-1}) + W(b_k - b_{k-1})$
16:             Choose the largest index $j' \in \{j + 1, \dots, n^r - i + 1\}$ such that $E_{\nu(j')} = E_{\nu(j+1)}$
17:             Set $W = W + \sum_{h=j+1}^{j'} w_{\nu(h)}$ and $j = j'$
18:         Set $i = i - 1$
```

where $R_{\max}(\sigma^r(i), \dots, \sigma^r(j))$ is the time that the vehicle departs from the depot. Analogous to equations (4)-(7) in which $\oplus$ is defined as a concatenation operator for subsequences, we can also evaluate the total weighted tardiness for a concatenation of subsequences using

$$Z(\pi) = g^r_{i,j}(R_{\max}(\pi) + \tau_{0,\sigma^r(i)}) + g^{r'}_{u,v}(R_{\max}(\pi) + \tau_{0,\sigma^r(i)} + L(\sigma^r(i), \dots, \sigma^r(j)) + \tau_{\sigma^r(j),\sigma^{r'}(u)}), \tag{13}$$

where $\pi = (\sigma^r(i), \dots, \sigma^r(j)) \oplus (\sigma^{r'}(u), \dots, \sigma^{r'}(v))$.

Recall that some neighbors under the insert, swap and inter-route 2-Opt neighborhoods reverse a subsequence of jobs. To evaluate such neighbors efficiently, we also compute values of $\bar{g}^r_{i,j}(t)$ for $t \ge 0$, which is the total weighted tardiness for customers in the sequence $(\sigma^r(j), \dots, \sigma^r(i))$, where the service for customer $\sigma^r(j)$ starts at time $t \ge 0$, in a similar way to our computation of $g^r_{i,j}(t)$.

Proposition 2. *Preprocessing using Algorithm 2 to determine the breakpoints $(b_h, g^r_{i,n^r}(b_h))$ for $h = 0, 1, \dots, k$, $i = 1, \dots, n^r$ and $r = 1, \dots, m$ requires $O(n^2)$ time. For any neighborhood where a neighboring solution is obtained by moving a constant number of subsequences to different positions, evaluating its total weighted tardiness after preprocessing requires $O(\log n)$ time.*

Proof. Steps 4-18 of Algorithm 2 require $O(n)$ for each of the n choices of r and i, which gives a time complexity for these steps of $O(n^2)$ overall. For Step 6, the sequence π is updated by finding the appropriate position into which $\sigma^r(i)$ is inserted by applying bisection search. Thus, Step 6 requires $O(\log n)$ time for each $\sigma^r(i)$, and $O(n \log n)$ time overall. This establishes that all breakpoints are computed by Algorithm 2 in $O(n^2)$ time.

Iterative application of equation (13) shows that evaluating a neighboring solution created by moving a constant number of subsequences to different positions requires a constant number of evaluations of functions $g^r_{i,j}(t)$ for various values of i, j, r and t. Each $g^r_{i,j}(t)$ function is obtained from (11) by evaluating two $g^r_{i,n^r}(t)$ functions. Lastly, each $g^r_{i,n^r}(t)$ is computed from (10) in $O(\log n)$ time, which is the time for requirement for applying bisection search to find the correct piecewise linear segment corresponding to t in (10). Combining these statements, we obtain an overall time complexity of $O(\log n)$ for evaluating the total weighted tardiness after moving a constant number of subsequences. $\square$

4.2. Fitness function

The diversity amongst solutions in the population is a key consideration in PRAs, and increasingly in EAs more widely. If the population diversity is too low, then the exploratory potential of the algorithm becomes weak. This is because less information is captured in the population, and the shorter relinking trajectories that are produced result in less of the search space being explored. If solutions in the population are more diverse, then we may expect more interesting and varied relinking trajectories, particularly if these solutions have good objective values. To address the dual objectives of improving solution quality but retaining a useful level of diversity, we follow the biased fitness approach of Vidal et al. (2012). This combines the rank of a solution in the population both for the objective value and for a measure of diversity.

To measure the diversity in the population, it is useful to define a measure of the distance between two solutions x and x'. No single choice of distance measure appears ideally suited for the problem. However several features of a solution may convey meaningful information, and different attributes/distance measures can be defined. Considering computational effort, we propose the use of arcs $(i,j) \in A$ as the attributes, with the distance between two solutions being the number of different arcs used in those solutions. Let $I_{ij}(x, x' : \xi)$ be a binary indicator for any pair of solutions x and x', and arc $(i,j) \in A$ used in x, which is defined by

$$I_{ij}(x, x' : \xi) = \begin{cases} 1, & \text{if } (i,j) \text{ is used in solution } x \text{ and not in } x', \\ 1-\xi, & \text{if } i \neq 0,\ j \neq 0,\ (i,j) \text{ is used in solution } x \text{ and } (j,i) \text{ in } x', \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

where ξ is a parameter in the interval $0 \le \xi \le 1$. Parameter ξ, for $\xi > 0$, reduces the distance if arcs are used in opposite directions in the two solutions. This definition of $I_{ij}(x, x' : \xi)$ is used to create the effect of using a balance of edge- and arc-based distance. If $\xi = 1$ then the distance is edge-based, and if $\xi = 0$ then the distance is arc-based. Observe that $\xi = 1$ considers the assignment of customer vertices to the same routes more explicitly, but can underestimate the distance between solutions when the schedules are important or the distances between customers are asymmetric, and vice versa for $\xi = 0$. Observe that in the second expression (14) the depot arcs are not considered.

Notice that $\sum_{(i,j)\in x} I_{ij}(x, x' : \xi) = \sum_{(i,j)\in x} I_{ij}(x', x : \xi)$ if and only if x and x' contain the same number non-empty routes. To adjust for solutions with varying numbers of non-empty routes, we therefore define $\rho(x)$ as the number of non-empty routes in a solution x. The distance between x and x' is then defined as

$$\text{dist}(x, x' : \xi) = \sum_{(i,j)\in x} I_{ij}(x, x' : \xi) + \max\{0, \rho(x) - \rho(x')\}, \tag{15}$$

where $\text{dist}(x, x' : \xi) = \text{dist}(x', x : \xi)$ and $0 \le \text{dist}(x, x' : \xi) \le n + m$ for any choice of x and x'. This measure of distance is also related to the Hamming distance for solutions represented by $n \times n$ binary matrices with the entry in row i and column j set equal to one if $(i, j) \in A$ is used in the solution.

Let $N_c(x)$ be a set containing n_c solutions x' from the population with smallest values of $\text{dist}(x, x' : \xi)$. We define the *diversity* $\mathcal{D}(x)$ of x as the average distance to solutions of this set, which is given by

$$\mathcal{D}(x) = \frac{1}{n_c} \sum_{x' \in N_c(x)} \text{dist}(x, x' : \xi). \tag{16}$$

Let N_e be a set containing n_e solutions from the population with lowest objective value. We also define $p(x)$ and $\delta(x)$ as the rank of a solution x in the population, ordered by non-decreasing value of the penalized objective (3) and non-increasing value of the diversity (16), respectively. We can then define the *fitness* of a solution x in the population (which is to be minimized) as

$$F(x) = p(x) + (1 - n_e/|P|)\,\delta(x). \tag{17}$$

If $|P| > n_e$, the coefficient of $\delta(x)$ prevents the solutions N_e from being given the worst fitness in the population. Even if one of these solution has the lowest diversity it will not be given the worst fitness (for a proof, see Vidal et al. (2012)).

The population can contain both feasible and infeasible solutions. However, if all of the solutions are infeasible, then it may be difficult to navigate to a feasible region of the search space. For this reason, and to retain a feasible solution if one is found, we modify the ranking mechanism by assigning the best feasible solution in the first position of the ranking defined by p. This will also encourage this solution to be selected more frequently in relinking, thereby ensuring further emphasis on exploring the feasibility boundary and achieving feasibility. This boundary is expected to be of particular interest for instances where the capacity constraints are tight.

4.3. Parent selection and relinking

Path-relinking is first suggested by Glover (1989) as a technique for exploring trajectories connecting solutions found during NS. The method is now considered more generally as a recombination approach, and compared to crossover in GAs relies less on randomization. First, two solutions are selected as the *initial solution* x_I, and the *guiding solution* x_G. Starting from x_I, a trajectory is constructed by iteratively applying neighborhood moves to introduce attributes from x_G. The number of different attributes describes a distance measure between solutions. Also, the trajectory contains offspring solutions with decreasing distance to x_G and increasing distance from x_I. Note that x_G can be defined as a set of solutions, although we choose x_G to be a single solution in our implementation.

We use binary tournament to select two parent solutions for relinking, which encourages some variety in the selection but prioritizes lower fitness. Once two different solutions are selected, these are randomly assigned to be either x_I or x_G. For the relinking procedure described and the problem considered, our initial experiments suggest taking x_I as the solution with highest fitness produces trajectories structurally different to the reverse, although neither dominates the other. Both feasible and infeasible solutions can be selected for x_I and x_G, and this encourages further exploration of the boundary of feasibility.

For any solution x we need to define its attributes, and for a pair of solutions x and x' we need to define the set of attributes that are in x' and not in x, and the cardinality of this set. As described in § 4.2, the arcs (i,j) of a solution x can be regarded as the attributes. Further, for solutions x and x', we define $\mathcal{A}(x,x') = \{(i,j) \in A\colon (i,j) \in x', (i,j) \notin x\}$ as the set of attributes in x' that are not in x, and $\text{dist}(x,x':0) = |\mathcal{A}(x,x')|$ as stated in (15), as the cardinality.

Let $\Delta = \text{dist}(x_I, x_G : 0)$, N_r be a chosen set of neighborhoods which introduce attributes, S be the set of solutions returned from relinking, ϕ be a parameter representing the average number of offspring returned from a relinking, $\phi' = \Delta/(\phi+1)$, $\bar{\phi}$ be a parameter the indicates the next iteration at which the current solution is copied to S, and the operator $|[a]|$ rounds a to the nearest integer. A general outline of the relinking procedure is given in Algorithm 3.

The current solution x_c is initially set equal to x_I, and at each iteration of relinking we seek to progressively reduce $\text{dist}(x_c, x_G : 0)$ by at least one. While $\text{dist}(x_c, x_G : 0) > 2$, we randomly order the neighborhoods N_r, and choose the move which improves the penalized objective (3) the most from the first neighborhood featuring a move that introduces an attribute from $\mathcal{A}(x_c, x_G)$ to x_c. Every $\bar{\phi}$ iterations, we copy x_c into the set S, and each time $\bar{\phi}$ is chosen as a random integer between $|[0.75\phi']|$ and $|[1.25\phi']|$.

To design neighborhoods for the set N_r, we consider introducing an arc $(i,j) \in \mathcal{A}(x_c, x_G)$. As suggested in Reghioui et al. (2007), we define a block as a subsequence of consecutive customers that are connected in both x_c and x_G, and the immediate predecessor and successor of the subsequence

Algorithm 3 Relinking procedure

```
1: Set the parameter value $\phi'$, initialize $S = \emptyset$ and $x_c = x_I$, and randomly pick an integer $\bar{\phi}$ from $\big[|[0.75\phi']|, |[1.25\phi']|\big]$
2: while dist$(x_c, x_G : 0) > 2$
3:     Randomly order neighborhoods $N_r$
4:     Set $N_c$ as the first neighborhood with a move introducing an attribute in $x_G$ to $x_c$
5:     Apply most improving move of $N_c$ that introduces an attribute of $\mathcal{A}(x_c, x_G)$ to $x_c$
6:     if Number of iterations since $\bar{\phi}$ updated $= \bar{\phi}$ then
7:         Copy $x_c$ into $S$ and randomly pick an integer $\bar{\phi}$ from $\big[|[0.75\phi']|, |[1.25\phi']|\big]$
8: return $S$
```

in x_c and x_G are different. Therefore, removing a block from its current position in x_c causes no increase in dist$(x_c, x_G : 0)$. Furthermore, inserting this block in a position that introduces the desired arc $(i, j) \in \mathcal{A}(x_c, x_G)$ into x_c decreases dist$(x_c, x_G : 0)$. If $i, j \in R^r$ for x_c, then there is a block-insert neighbor which introduces (i, j). This neighbor has the block $(\sigma^r(p), \cdots, \sigma^r(p+q))$, where $\sigma^r(p) = j$, relocated to immediately after vertex i. If $i \in R^r$ and $j \in R_{r'}$ for x_c, where $r \neq r'$, then there is a 2-Opt* neighbor which introduces the desired arc (i, j) into x_c and does not increase dist$(x_c, x_G : 0)$. This neighbor exchanges the subsequences $(\sigma^r(p), \cdots, \sigma^r(n^r))$ and $(\sigma^{r'}(q), \cdots, \sigma^{r'}(n^{r'}))$, where $\sigma^r(p-1) = i$ and $\sigma^{r'}(q) = j$. The 2-Opt* moves to introduce an arcs $(0, j)$ for $j \in V'$ in x_c, create a new route, and are considered at any iteration if $\rho_{x_c} < \rho_{x_G}$. For further clarity, an example relinking is illustrated in § B.

The penalized objective (3) is used in relinking to select neighborhood moves, which increases the number of offspring that can be explored through relinking. Notice that solutions that are infeasible with respect to the capacity are also possible in the trajectory for any value of β, due to the constraint on introducing at least one arc of $\mathcal{A}(x_c, x_G)$ to x_c. If no capacity feasible solutions can be reached under this criterion, then infeasible solutions will be visited. By definition, the trajectory is guaranteed to return to the level of feasibility of x_G by the end. A solution which is not neighboring either x_I or x_G can only be found if $\Delta > 4$. Otherwise, only one or two offspring solutions would be produced from the relinking of x_I and x_G, and both offspring neighbor x_I or x_G. If x_I and x_G are locally optimal for the neighborhoods N_r then this relinking is unnecessary, however this is not strictly true in our PRA because the value of β may since have changed in (3).

Relinking to different solutions in the population enables a comprehensive exploration of the locality of a solution in "promising" directions. Further, the use of the penalized objective (3), and varying between the inter- and intra-route neighborhood in N_r, compounds the potential for local exploration and also encourages a more comprehensive exploration of the search space between solutions.

PROPOSITION 3. *For given x_I and x_G, relinking can be performed in $O(\Delta^2 n)$ time.*

Proof. Finding Δ from (15) and identifying the set $\mathcal{A}(x_I, x_G)$ can be performed in $O(n)$ time, and $|\mathcal{A}(x_I, x_G)| = O(\Delta)$. The number of solutions in the trajectory and consequently the number of iterations of the relinking procedure is therefore $O(\Delta)$. At each iteration, the maximum number of possible moves is $O(\Delta)$. The 2-Opt* moves are identified in $O(1)$ time from $\mathcal{A}(x_c, x_G)$, but the block-insert moves are found in $O(n)$ time. Thus, the overall time complexity is $O(\Delta^2 n)$. $\square$

Note that an alternative implementation in which data is preprocessed in $O(n^2)$ time at each iteration requires $O(\Delta n^2)$ time. Thus, the more straightforward implementation that leads to the result in 3 is preferable.

The neighborhoods used in relinking are restricted and therefore solutions in the trajectory are not guaranteed to be locally optimal. We perform NS on each solution in S prior to considering introduction to the population. Defining S as the complete trajectory would require performing NS on each of these solutions, which represents a significant computational effort. Introducing a large number of solution to the population from a single relinking may also encourage lower diversity in the population.

Solutions closer to each other in the trajectory have a greater probability of sharing basins of attraction, and therefore leading to the same local optima. This fact is also exaggerated by the use of similar neighborhoods in the NS and relinking procedures. Various techniques have been proposed to select offspring solutions in the literature. For example, Hashimoto and Yagiura (2008) propose storing a fixed number of the best solutions that have better objective value than their immediate neighbors in the trajectory. A simpler policy is described by Reghioui et al. (2007), who store solutions a fixed number of iterations apart in the trajectory. In contrast, Sörensen and Schittekat (2013) return a single solution, which has the best objective value or is at the midpoint of the trajectory.

Preliminary experiments on the VRPJA suggest the PRA described is robust to the approach used to select S. We therefore decide to retain ϕ approximately equidistant solutions from the trajectory. This reduces the computational effort and obtains a spread of solutions with some additional randomization. The current solution x_c is periodically copied into S, and the number of iterations to wait for the next copying is selected as a random integer from the interval $\left[|[0.75\phi']|, |[1.25\phi']|\right]$. The relinking procedure therefore produces $|S| \sim U\left[|[0.75\phi]|, |[1.25\phi]|\right]$ solutions. Each solution is improved by NS and enters the population if not already present. Relinking can be performed between the same initial and guiding solutions during the algorithm, and the randomization ensures some variation in S.

4.4. Population management

The population is initialized with 1.5μ randomly generated solutions improved by NS. This is motivated by beginning the search with diverse solutions which have at least moderately low objective values. When the population size reaches $\mu+\lambda$, it is reduced to μ considering all solutions in the population. The population is reduced by iteratively removing the solution with the highest fitness and updating the fitness of the remaining solutions. As discussed in § 4.3, the set of solutions S returned from each relinking are improved by NS and are introduced to the population if not already present. Notice that solutions joining the population after relinking are immediately available to be selected as parent solutions.

Many population-based approaches refresh the population or remove a portion of the population and re-initialize with newly generated solutions. This is usually performed if the progress of the search, or the diversity of the population, appears to have stagnated. In the PRA, each time γ relinkings are performed and neither the best quality solution nor the best quality feasible solution are improved, then the population is refreshed. The top $\mu/3$ solutions in the population are retained and μ new solutions are generated identically to the population initialization. This attempts to utilize progress made during the search, whilst introducing a significant amount of diversity. Clearly, if $n_e \leq \mu/3$, then the feasible solution with best value and $\max\{0, n_e-1\}$ other solutions with lowest value for the penalized objective are retained.

4.5. Infeasibility penalty

Exploring infeasible regions of the search space in heuristic approaches is conjectured to facilitate transition between feasible regions, which may otherwise be distant or unreachable (e.g., see Glover and Hao 2011). In exact approaches, a number of constraint relaxation techniques have proven useful, such as Lagrangian, linear and state-space relaxation. We relax the route capacity constraints and penalize the level of infeasibility, as shown in (3). The penalty on the level of infeasibility is controlled by adapting β in (3) during the search.

To initially prevent infeasibility, $\beta=\beta_0$, where β_0 is a large value. After three solutions have been generated in the initial population and NS applied, β is set to be the average of the cost per unit of demand for each route. To encourage a useful trade-off between infeasibility and objective value throughout the search, β is adaptively updated each time κ offspring solutions are produced since the most recent update of the penalty or refreshing of the population. Let p_f be the target proportion of feasible solutions, ϵ the range of this proportion, $\bar{p}_f$ the proportion of the κ solutions that are feasible after NS, and δ the update factor. The value of β is then updated accordingly using

$$\beta = \begin{cases} \beta\delta, & \text{if } \bar{p}_f < p_f - \epsilon, \\ \beta/\delta, & \text{if } \bar{p}_f > p_f + \epsilon, \\ \beta, & \text{otherwise.} \end{cases} \tag{18}$$

Although the size of the search space is increased by allowing infeasibility, our preliminary experiments with NS (using random first-improvement) exhibit a faster progression through the search space. This effect is attributed to a greater proportion of improving neighbors for some of the solutions, which therefore creates many additional paths in the search space.

5. Computational experiments

In this section, we introduce a set of benchmark instances (§ 5.1), and describe the comparator ILS (§ 5.2). Following this, we present the results from the calibration of the algorithms (§ 5.3), a comparison of the performance of the proposed heuristics on the benchmark set (§ 5.4), and an analysis of the PRA results independently (§ 5.5). Finally, a sensitivity analysis of the PRA parameters is performed (§ 5.6).

The algorithms are implemented in C++ and complied using GCC 4.8.1. All experiments are performed using a single core of an Intel Xeon E5–2670 2.6GHz processor (running Linux Red Hat Enterprise Server release 6.3).

5.1. Benchmark instances

We propose three problem types that have different values for α in (1), and a set of 96 benchmark instances for the VRPJA. This enables us to perform computational experiments with the aim of assessing the performance of the proposed heuristics and obtaining insights into the problem.

Generating instances of scheduling problems is discussed by Hall and Posner (2001), who propose a number of principles and properties for generating adequately representative and unbiased data. Following these principles, we restrict the set of problem features varied to enable greater comparability and to facilitate analysis of the key features of the novel problem that we study. To reduce any possible bias in the instances, we encourage size- and scale-invariance in the problem features, and we also use the uniform distribution to generate many of the new features of the instances. We define $U_{\mathbb{Z}}[c,d]$ as the integer uniform distribution between integers c and d inclusive.

We extend four CVRP instances introduced by Christofides et al. (1979), where $n=50$ and $m=5$, $n=100$ and $m=10$, $n=150$ and $m=14$, and $n=199$ and $m=20$. The coordinates of customers are randomly distributed in the interval $[0,100]$, so that they lie inside a square with sides of length 100, and the depot is relatively close to the center. The instances are well documented and are available from the website `http://neo.lcc.uma.es`. The optimal CVRP solutions for the instances are found by Pecin (2014).

Let I be a CVRP instance with m_I vehicles, and $D_{\max}$ be the duration of the longest route in the best known solution of I. After generating the travel times, we introduce $E(\tau)$ as an estimate of the average travel time between vertices. Specifically

$$E(\tau)=\frac{\sum_{i\in V'}\sum_{j\in\mathcal{N}_i}\tau_{ij}}{|V'|\,|[n^{1/2}]|}, \tag{19}$$

where the operator $|[a]|$ is defined as previously and rounds a to the nearest integer, and $\mathcal{N}_i$ as the set of $|[n^{1/2}]|$ vertices that have smallest travel times to vertex i. We choose the $n^{1/2}$ vertices with shortest travel time because a number of studies, such as Beardwood et al. (1959) and Johnson et al. (1996), identify a relationship between the expected cycle costs of the TSP and $n^{1/2}$.

Various parameters are used to generate the instances: $e_m \geq 0$ defines the proportion of additional vehicles above m_I, so that the number of vehicles is $(1+e_m)m_I$, $b \geq 0$ is the spread of the release dates as a proportion of $D_{\max}$, $k \in \mathbb{Z}^+$ is the looseness of the due dates as a multiple of $E(\tau)$, and $0 \leq v \leq 1$ is half of the range that k can vary within as a proportion of k.

We adapt and extend I as follows:

1. Set $m = (1+e_m)m_I$, $w_i = 1$, for all $i \in V'$, and $\tau_{ij} = c_{ij}$, for all $i, j \in V$.
2. Generate $r_i = X$, for all $i \in V'$, where $X \sim U_{\mathbb{Z}}\big[0, |[bD_{\max}]|\big]$.
3. Calculate $E(\tau)$, and generate $d_i = r_i + |[(k+Y-kv)E(\tau)]|$, for all $i \in V'$, where $Y \sim U[0, 2kv]$.

For simplicity, unit weights and unit vehicle speeds are used. Notice that if the underlying CVRP instance has a feasible solution, then the resulting VRPJA instance also has a feasible solution because $e_m \geq 0$ and hence $m \geq m_I$.

In Table 2, the different values of the parameters used to generate the instances are presented. Introducing constraints and objectives related to the arrival times of customers to the CVRP affects the total distance traveled in optimal solutions. This is adjusted in the objective (1) by selecting different values of α, where we consider the objectives weighted equally and use weightings to increase the emphasis on each objective individually. In some instances, it may be beneficial to use more vehicles than in the optimal CVRP solution, and sometimes considerably more. Therefore, we consider two settings for e_m, resulting in instances with either an equal number of vehicles to the underlying CVRP instance, or 50% more vehicles. Preliminary experiments with 100% more vehicles suggest that more than 50% has little effect on the optimal solution for most of the instances. The release dates are a novel consideration in VRPs, and we use a range of spreads to enable a more detailed analysis. To decide on appropriate values for parameter k, we consider the range of the number of customers in routes in the best known solutions, and the average number of customers per route n/v, where v is the number of routes in the best known solution.

Table 2 **Instance generation parameters**

Instance parameter		Values
α	objective weight	$0.3, 0.5, 0.7$
e_m	proportion of extra vehicles	$0, 0.5$
b	spread of release dates	$0.25, 0.50, 0.75, 1.00$
k	looseness of due dates	$4, 6, 8$
v	range in looseness of due dates	0.25

Notice that if b and k are sufficiently low and high, respectively, then the distance traveled will tend to dominate the objective for any value of α. Alternatively, high b and low k result in the weighted tardiness dominating the objective for any value of α. If the weighted tardiness has high priority, then more vehicles will be used to enable a greater variety of vehicle departure times and earlier arrival times at customers; otherwise, additional vehicles often represent extra distance traveled. As a consequence, if there is a low number of vehicles available, then the priority of the weighted tardiness will increase. The considerations above demonstrate some of the complex interplay between the problem features, and emphasize the need to generate a set of varied instances to gain a better understanding of the impact on solutions of these key features.

5.2. Iterated local search

ILS is an elementary stochastic extension of NS, applying a kick to escape whenever a local optimum is reached. The simplicity and generality of the framework are key features, and it performs competitively when applied to numerous problems in combinatorial optimization. For an overview of the design, implementation and some applications of ILS, we refer to Lourenço et al. (2010).

In Algorithm 4, we describe the basic implementation of our comparator ILS algorithm for the VRPJA. This is primarily a comparison for the proposed PRA, and the NS of Section § 4.1 is used as the improvement procedure. In the initialization, the current solution is generated randomly and is then improved by NS. An alternate solution is produced by kicking the current solution, NS is then applied and the resulting solution replaces the current solution if it has lower penalized objective value (3).

Algorithm 4 Iterated local search

```
1: Initialize current solution $x$
2: Improve $x$ by applying NS
3: **while** *running time* $< T_{\max}$
4:     Apply $\eta$ random moves to $x$ producing $x'$ (apply a kick)
5:     Improve $x'$ by applying NS
6:     **if** $f(x') < f(x)$ **then** set $x$ as $x'$
7: **return** $x$
```

For the kick, we apply η random moves in the swap neighborhood with $s_1 = 1$ (see § 4.1). The kick can produce infeasible solutions with respect to vehicle capacity, and the level of infeasibility is restrictively penalized using $\beta = \beta_0$ in (3), as defined in § 4.5. As a result, when NS is applied to infeasible solutions, the objective will be dominated by initially achieving feasibility. The only parameters of this method are the kick size η, and a limit on computation time $T_{\max}$, which is used as a termination criterion.

5.3. Algorithm calibration

Common to metaheuristics and particularly EAs, the PRA presented relies on a variety of correlated parameters. The objective of parameter calibration is find a set of parameters to optimize some performance measures of the algorithm. Mercer and Sampson (1978) suggest using metaheuristics to calibrate metaheuristics, termed metacalibration, but this early work reports insufficient computational resources.

To calibrate the PRA presented, we apply the covariance matrix adaptation-evolutionary strategy (CMA-ES) metaheuristic proposed by Hansen and Ostermeier (2001). The CMA-ES has performed well in a variety of settings, and in particular EAs have been calibrated to achieve competitive performance (Smit and Eiben 2009, Vidal et al. 2012). To evaluate a set of parameters, the PRA is executed on a training-set of instances that are selected to represent the problem range. The calibration objective is to minimize the average percentage gap in the objective value across the training-set compared to the best found in all previous experiments.

Our preliminary experiments suggested a number of parameters are more robust, and these are fixed at values that appear to perform well. For the parameter limiting the size of the neighborhoods N_1 and N_2, $s_1 = 2$ is found to provide a good compromise between quality and efficiency. We set the parameter limiting the size of N_3 to $s_2 = 10$ because reversals of large sections of routes appear to increasingly disturb the schedules. To initially prevent infeasibility, the initial value of the penalty β is set to $\beta_0 = 1000$. For the parameters controlling the infeasibility penalty, the update rate is $\delta = 1.2$, the range of the target proportion of feasible offspring is $\epsilon = 0.05$, and the number of offspring between updates is $\kappa = 50$. Finally, the number of iterations considered in the condition on population refresh is set to $\gamma = 80$.

Similarly to Vidal et al. (2012), we tune the remaining parameters separately for the different problem types. This may reveal any dependency in the parameters on the value of α, and the results can also be used to infer good parameter settings. In Table 3, the best solutions of the parameter calibration are given for the three values of α, and the parameter values used in the main experiments that follow are given in the "Final values" column. Multiple settings for the final parameter values are given in order of increasing α. The boundaries on the parameter used in the metacalibration are also given in the "Range" column. Some of these are simply proportions, μ and λ are set through observations from the literature, and ϕ and p_c are estimated.

The results suggest that the best parameters for the values of α are generally similar, and the most diverse are λ, p_e and p_c. We found that averaging and rounding the other parameters across the problem types caused negligible degradation of the final objective values achieved. For $\alpha = 0.5$ and 0.7, averaging p_e and p_c also caused negligible degradation of the final objective values.

The problem type dependent parameters suggest that the objectives ideally require different exploration strategies in the PRA. When the weighted tardiness has a greater emphasis, then it

Table 3 **Results from meta-EA calibration of PRA parameters**

Parameter		Range	$\alpha = 0.3$	0.5	0.7	Final values
μ	population size	$[10, 40]$	10	14	13	**12**
ϕ	number of solutions from relinking	$[1, 30]$	6	5	5	**5**
λ	number of solutions in generation	$[\phi, 100]$	8	14	28	**10/15/30**
p_f	target proportion of feasible solutions	$[0, 1]$	0.38	0.5	0.53	**0.5**
p_e	proportion of elite solutions ($n_e = \mu p_e$)	$[1/\mu, 1]$	0.12	0.14	0.20	**0.1/0.2/0.2**
p_c	proportion of solutions in diversity ($n_c = \mu p_c$)	$[0, 0.33]$	0.12	0.27	0.21	**0.1/0.25/0.25**
ξ	edge reduction in dist′	$[0, 1]$	0.43	0.28	0.82	**0.5**

is preferable to more regularly manage the population. Additionally, only the closest solution in the population is considered in the diversity of solutions, and this leads to a maximally diverse population. Relinking and NS are therefore performed more frequently among solutions with low fitness, and the diversity is given almost equal weight in the fitness.

When the total distance traveled has a greater emphasis, then the population is managed less regularly. A further contrast to the case of weighted tardiness dominating is that the three closest solutions are considered in the diversity, and this allows clusters of solutions to be formed by a smaller penalization of solutions with lower distances to the two closest solutions. These differences encourage greater and more localized exploration in each generation before the population is reduced.

For the ILS, we used the same set of training instances and consider $\eta = 1, 2, 3, 4$. The results suggested that $\eta = 2$ achieves the best objective values in acceptable computation time for the values of α considered. Larger values of η are not considered due to the results, the greater disturbance to the solutions and the resulting infeasibility that often occurs.

5.4. Comparison of algorithm performance

To evaluate the relative performance of the PRA and ILS methods for the VRPJA, results for the benchmark set are compared in Table 4. For comparability, the total running time is set as $T_{\max} = 10$min for both PRA and ILS, and the results are based on 10 independent runs. The values PIA and PIB represent the average percentage improvement in the objective value achieved by the PRA compared to the ILS, for the average and best of the 10 runs respectively. Further, TBS$_P$ and TBS$_I$ represent the average time in minutes that the solution with best objective value of a run is first identified, for the PRA and ILS respectively.

Considering the results from these experiments. the PRA appears superior, and achieves better objective values in shorter computation time than ILS. Notably, both PIA and PIB are largest when the weighted tardiness has a higher objective weighting. This suggests that the contrasting features of the PRA, such as a more directed exploration of the search space and the information contained in a diverse population of solutions, are useful for small α. Equivalently, this may be attributable to the ILS finding a larger number of local optima, which therefore have a greater

Table 4 Comparison of the PRA and ILS results

n	$\alpha = 0.3$				$\alpha = 0.5$				$\alpha = 0.7$			
	PIA	PIB	TBS$_P$	TBS$_I$	PIA	PIB	TBS$_P$	TBS$_I$	PIA	PIB	TBS$_P$	TBS$_I$
50	2.64	0.39	1.11	3.90	1.84	0.10	0.84	3.54	1.13	0.17	0.78	2.99
100	2.09	0.48	2.96	5.26	1.36	0.36	2.33	4.73	0.83	0.14	1.84	4.78
150	3.77	1.72	4.38	6.86	2.27	0.87	3.80	6.52	1.61	0.55	3.87	6.87
199	3.79	2.11	5.85	7.67	2.75	1.58	5.40	7.44	2.13	1.34	5.84	7.35
Avg.	3.07	1.18	3.57	5.92	2.06	0.73	3.09	5.56	1.42	0.55	3.08	5.50

chance of being distant from the optimal solutions, and the reliance on only accepting solutions with improving objective value.

The PRA achieves good quality objective values more robustly than the ILS, as evidenced by the large PIA values. The ratio between the PIA and PIB values is similar for the different values of α, and this suggests that the relative robustness is consistent. Notably, both PIA and PIB increase with n, showing that the PRA achieves greater improvement in objective value over the ILS for the larger instances. These instances clearly have a larger search space and are expected to contain a greater number of local optima. These results support the previous conjectures relating to the contrasting features, and may also suggest that the effect of the kick in ILS is not large or varied enough for larger instances. The PIA is notably lower for $n = 100$, which suggests that the robustness of the methods are more similar. This characteristic may be attributable to these instances having the most sparsely distributed set of customers.

Considering TBS$_P$ and TBS$_I$, we observe that PRA finds the solution with best objective value faster on average than ILS. Again, this suggests that the contrasting features of the PRA are more effective and also efficient. When α is lower, then both TBS$_P$ and TBS$_I$ are larger, which suggests that a higher objective weighting for the weighted tardiness increases the time taken for the algorithms to converge and find the best solutions. Finally, note that the growth of TBS$_P$ with respect to n is larger than for TBS$_I$, but this is expected for a population-based method. This may be a consideration when solving larger instances, although the PIA and PIB are also increasing with n and we would expect longer computation times to achieve better objective values.

5.5. PRA results on the instances

To assess and validate the benchmark instances, we present a detailed analysis of the PRA results. We consider the effects on the objective values of the different parameters that define the problem features and the time taken to find the best solution.

In Table 5, we present the average results for the PRA across the 10 runs with $T_{\max}$ set at 10 minutes, where Av represents the average objective value, and TBS is the average time taken to find the best solution of the run. To analyze the effects of the problem attributes introduced, we average the results for all the underlying CVRP instances or equivalently for all values of n.

Table 5 Average results of PRA on benchmark instances

b	k	$\alpha = 0.3$				$\alpha = 0.5$				$\alpha = 0.7$			
		$e_m = 0$		$e_m = 0.5$		$e_m = 0$		$e_m = 0.5$		$e_m = 0$		$e_m = 0.5$	
		Av	TBS	Av	TBS	Av	TBS	Av	TBS	Av	TBS	Av	TBS
0.25	4	1052.6	3.7	735.9	1.5	1088.2	4.0	946.8	1.9	1102.9	2.9	1077.3	2.9
	6	458.0	3.9	407.6	3.1	650.0	3.2	632.8	1.9	817.0	2.7	815.4	2.1
	8	304.4	4.3	304.4	4.8	503.2	3.7	503.2	4.1	694.4	3.9	694.4	3.3
0.50	4	1477.9	3.6	953.9	2.6	1410.6	3.1	1130.5	1.3	1319.2	2.3	1253.3	1.6
	6	623.9	3.4	478.5	3.5	795.0	3.7	741.1	2.6	943.1	3.7	937.3	2.9
	8	364.5	3.9	360.3	4.4	585.4	4.0	584.3	4.4	778.8	4.5	778.5	3.7
0.75	4	1834.1	3.9	1175.7	2.1	1688.9	3.6	1308.0	2.3	1520.9	3.4	1396.4	2.9
	6	855.7	4.2	550.2	3.1	975.3	2.4	829.7	1.8	1074.0	2.9	1047.1	2.9
	8	429.9	4.5	396.8	3.6	661.4	3.0	646.6	3.2	865.6	3.3	862.0	3.8
1.00	4	2119.5	4.3	1335.9	2.5	1914.4	4.2	1428.8	1.8	1675.6	3.9	1494.5	2.0
	6	1064.7	3.9	627.6	2.0	1147.1	3.9	913.8	2.3	1202.0	3.5	1144.4	2.4
	8	523.7	5.5	438.4	3.5	748.6	3.8	709.3	3.7	943.4	3.5	938.5	2.9
Avg.		925.7	4.1	647.1	3.1	1014.0	3.6	864.6	2.6	1078.1	3.4	1036.6	2.8

The experiments show contrasting performance, and this suggests a varied set of instances. The instance parameters appear to have regular effects across the set, and respect the principles discussed in the instance generation (§ 5.1). Different results are produced using the values of α proposed, and this reveals the effect of different weightings in the objective.

A greater spread of release dates, which is represented by larger values for b, can cause the weighted tardiness to become larger. This increases the overall objective values found, particularly if the due dates are tight as indicated by low values of k. The value of TBS also increases, and this is presumably because more computation time is necessary to find good partitions and schedules of the customers.

The value of TBS is lower when the due dates are tighter, suggesting the instances can be solved more quickly when a greater proportion of solutions have high tardiness. Notice that this will cause a large number of feasible solutions (and infeasible solutions with respect to the vehicle capacity) to have high objective values, resulting in a steeper landscape. The value of TBS also decreases as the value α becomes higher. This may be attributable to the greater diversity of objective values amongst solutions when the total distance traveled becomes dominant, which enables the algorithm to navigate the search space more easily.

Increasing the number of vehicles by increasing e_m permits more flexibility in vehicle departure times and therefore customer arrival times. The greater flexibility can decrease the weighted tardiness for certain instances and values of α. If the change in total distance traveled due to an additional route is less than the reduction in weighted tardiness, then another vehicle clearly improves the objective value. Also, if greater numbers of routes are used in a solution, then the scheduling and capacity constraints will be less restrictive. This is most prominent when the scheduling is tight, i.e., when b is large or k is small.

5.6. Sensitivity analysis of PRA

We now present our results from experiments performed to analyze the individual importance of some of the main components in the PRA. We disable each component independently, and the results for each alternative are again taken from 10 independent runs with $T_{\max}$ set at 10 minutes. The variants of the PRA with a component disabled are: "No NS" that does not apply NS to offspring solutions; "No Infe." that does not adjust the infeasibility penalty β; and "No Div." that does not include diversity in the fitness function. The results from these variants are summarized in Table 6, where the PRA column refers to the version with no disabling. The value Gap represents the worsening of the average objective value compared to the full PRA as a percentage of the latter, and again TBS is the average time taken to find the best solution in a run.

Table 6 **Sensitivity analysis of the PRA**

α		No NS	No Infe.	No Div.	PRA
0.3	Gap	4.10	0.46	0.28	–
	TBS	5.80	4.98	3.62	3.57
0.5	Gap	2.72	0.22	0.17	–
	TBS	5.60	4.42	3.37	3.09
0.7	Gap	1.80	0.19	0.09	–
	TBS	5.55	4.44	3.36	3.08

The experiments confirm the importance of each component for solving the VRPJA with the proposed PRA. Considering the results apart from those where NS is disabled, we note that all variants of the method still have better average performance than the ILS. The most important component tested appears to be the NS, but allowing solutions that are infeasible with respect to capacity and incorporating a diversity measure also play an important role in the performance. Removing the different components appears to have a similar effect for the different values of α, but the results suggest these components are most critical to performance for lower values of α. Note that the effect of disabling the components may be exaggerated because the algorithm parameters are not changed accordingly.

Considering infeasible solutions and an adaptive penalty weight both improves the objective value and significantly decreases the TBS values. This supports the results from preliminary experiments, and the conjecture on the utility of infeasible solutions for heuristics, as described in § 4.5. The diversity measure used in the fitness function appears to contribute the least of the components tested, although when α is lower, the gap is almost equal for either disabling the infeasibility or diversity. This suggests both components are important for minimizing the weighted tardiness, but the diversity less so when the total distance becomes more dominant.

6. Conclusion

In this paper, we present a new vehicle routing problem that considers the time that customer orders become available for delivery at the depot. The objective is to minimize a convex combination of operational cost and customer service objectives, represented by the total distance traveled and total weighted tardiness, respectively. If a relationship between the operational costs and the perceived cost of tardy deliveries to customers can be established, then the problem can be applied at an operational level to find delivery routes for a single period. Another option is to evaluate a small number of values for the weight that balances the objective components, and potentially the individual customer tardiness weights, to find a solution which reaches a suitable compromise for the decision maker. This also suggests an alternative more strategic application of the problem to evaluate different possible scenarios and decisions. As one example, given a set of customers with representative release and due dates and weights, the effect of different numbers of vehicles and their capacity on the other features of the solution can be investigated.

We propose a path-relinking algorithm (PRA) to address the problem, and an iterated local search algorithm (ILS) as a comparator. The PRA builds on features of recently proposed heuristics for similar problems, although notably it is conceptually more simple than many. Some contributions include the efficient neighborhood search and evaluation procedures, the path-relinking procedure and the balanced edge- and arc-based solution distance used in measuring population diversity. Observing the results from these heuristics, there are significant effects associated with introducing release dates and weighted tardiness into a vehicle routing problem. Also, the performance of the PRA is shown to dominate the ILS for the benchmark set introduced, considering either the average time to find the best solution or the average or best objective values. Moreover, the results support the growing interest in more sophisticated and population-based heuristics for vehicle routing problems.

This introduction and initial investigation of the vehicle routing problem with job availability constraints should prove valuable in initiating and promoting research of the problem, and the integration of further aspects of machine scheduling into routing problems. This is particularly important considering the novelty of these types of problems, and the immediate practicality for managing the operations of supply chains. For practitioners, the results and discussion should present a clear analysis of some of the effects associated with explicitly considering release dates for customer orders, and introducing the customer service level objective. In addition, we reveal relationships and dependencies between the problem features that hint at the complex and varied interplay between these in different instances of the problem. Furthermore, the insights yielded from the analyses presented should prove helpful in the future for designing exact algorithms and heuristics for this and similar problems.

Acknowledgments

The authors thank Thibaut Vidal for a number of helpful comments and explanations about components of his hybrid genetic algorithms. We also acknowledge the use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton, in the completion of this work.

Appendix

A. MIP formulation

The VRPJA can be modeled as a mixed integer programming problem (MIP) using *three-index flow* variables. This approach is common for solving VRPs with complicating constraints on the routes. To linearize the constraints associated to arrival times at customer vertices, the big M method is used. Although, this method can lead to numerical instability and encourage fractional values for the variables.

The *three-index vehicle flow* formulation involves $O(n^2m)$ binary variables $x_{ijk} \in \{0,1\}$, which are equal to one if route k uses $(i,j) \in A$. To model the VRPJA, we also define $O(n)$ variables $u_i \in \mathbb{R}^+$ for the arrival time at $i \in V'$, $O(m)$ variables $v_k \in \mathbb{R}^+$ for the departure time of route k, and $O(n)$ variables $z_i \in \mathbb{R}^+$ for the tardiness to the delivery of $i \in V'$. We will denote this formulation VF, and it is defined as follows.

$$z(\text{VF}) = \min \left\{ \alpha \sum_{i,j\in V} c_{ij} \sum_{k\in K} x_{ij}^k + (1-\alpha) \sum_{i\in V'} w_i z_i \right\} \tag{20}$$

s.t.

$$\sum_{j\in V\setminus\{i\}} \sum_{k\in K} x_{ij}^k = 1, \qquad \forall\, i \in V', \tag{21}$$

$$\sum_{j\in V'} x_{0j}^k \le 1, \qquad \forall\, k \in K, \tag{22}$$

$$\sum_{j\in V\setminus\{i\}} x_{ij}^k - x_{ji}^k = 0, \qquad \forall\, i \in V, k \in K, \tag{23}$$

$$\sum_{i\in V'} q_i \sum_{j\in V\setminus\{i\}} x_{ij}^k \le Q, \qquad \forall\, k \in K, \tag{24}$$

$$r_i \sum_{j\in V\setminus\{i\}} x_{ij}^k \le v_k, \qquad \forall\, i \in V', k \in K, \tag{25}$$

$$v_k + \tau_{0i} - u_i \le M_{0i}(1 - x_{0i}^k), \qquad \forall\, i \in V', k \in K, \tag{26}$$

$$u_i + \tau_{ij} - u_j \le M_j \left(1 - \sum_{k\in K} x_{ij}^k \right), \qquad \forall\, i,j \in V', i \ne j, \tag{27}$$

$$u_i \le z_i + d_i \le M_i, \qquad \forall\, i \in V', \tag{28}$$

$$u_i \ge r_i + \sum_{j\in V\setminus\{i\}} \tau_{ji} \sum_{k\in K} x_{ji}^k, \qquad \forall\, i \in V', \tag{29}$$

$$z_i \ge 0, \qquad \forall\, i \in V', \tag{30}$$

$$x_{ij}^k \in \{0,1\}, \qquad \forall\, i,j \in V, i \ne j, k \in K. \tag{31}$$

Where the M_{0i} and M_i for $i \in V'$, are upper bounds on the arrival time at i directly from the depot and on the arrival time at i, respectively.

The objective function is identical to (1), and constraints (21)-(24) and (31) are the constraints of the three-index vehicle flow formulation of the CVRP. Constraints (25) and (26) ensure that each route departs at the latest release date of the assigned customers. Constraints (27) and (28) ensure that the correct time is spent traveling between customers and the tardiness to each customer delivery is calculated, respectively. Lastly, constraints (30) lower bound the z-variables and (29) lower bound the u-variables.

A tighter linear relaxation can be achieved from VF, if we require that there is at least one route r, and $v_r = \max_{i \in V'} r_i$. This is clearly valid, because any optimal solution must use at least one vehicle, and at least one vehicle departs at the latest release date of all the customers. To attempt to tighten the relaxation further, the u- and v-variables can also be disaggregated in a number of ways. We do not explore this or any valid inequalities for VF, because the symmetry present in the x-variables will create significant difficulty when solving any formulation of this type.

B. Illustration of an example of relinking

In Figure 2, an example of a relinking is given, where the vertical lines separating customers in x_I represent arcs not included in x_G. We assume the first neighborhood chosen is 2-Opt* and the neighbor selected introduces $(3,4)$, which reduces $\text{dist}(x_c, x_G : 0)$ by one. The following neighborhood is also 2-Opt* and $(1,3)$ is introduced, this also connects two routes and removes $(1,0)$ which is not present in x_G, and $\text{dist}(x_c, x_G : 0)$ is therefore reduced by two. The third neighborhood is block-insert, introducing $(9,8)$ and as a side-effect $(0,1)$ and $(8,0)$, therefore reducing $\text{dist}(x_c, x_G : 0)$ by three. This continues until the sixth move produces x_G.

$\Delta = 11$	1. 2-Opt$^*(8,4)$ $\text{dist}_2(x_c, x_G) = 10$	2. 2-Opt$^*(3,[3])$ $\text{dist}_2(x_c, x_G) = 8$	3. bl-ins$(8,8,[2])$ $\text{dist}_2(x_c, x_G) = 5$
x_I: $0 \mid 7\ 2 \mid 10\ 6 \mid 0$ $0 \mid 3 \mid 8 \mid 1 \mid 0$ $0 \mid 4 \mid 5 \mid 9 \mid 0$	$0 \mid 7\ 2 \mid 10\ 6 \mid 0$ $0 \mid 3\ 4 \mid 5 \mid 9 \mid 0$ $0 \mid 8 \mid 1 \mid 0$	$0 \mid 7\ 2 \mid 10\ 6 \mid 0$ $0 \mid 8 \mid 1\ 3\ 4 \mid 5 \mid 9 \mid 0$	$0 \mid 7\ 2 \mid 10\ 6 \mid 0$ $0\ 1\ 3\ 4 \mid 5 \mid 9\ 8\ 0$
	4. bl-ins$(7,2,[1])$ $\text{dist}_2(x_c, x_G) = 3$	5. 2-Opt$^*(7,9)$ $\text{dist}_2(x_c, x_G) = 2$	6. 2-Opt$^*(9,5)$ $\text{dist}_2(x_c, x_G) = 0$
x_G: $0\ 10\ 6\ 5\ 7\ 2\ 0$ $0\ 1\ 3\ 4\ 9\ 8\ 0$	$0\ 10\ 6 \mid 7\ 2\ 0$ $0\ 1\ 3\ 4 \mid 5 \mid 9\ 8\ 0$	$0\ 10\ 6 \mid 9\ 8\ 0$ $0\ 1\ 3\ 4 \mid 5\ 7\ 2\ 0$	$0\ 10\ 6\ 5\ 7\ 2\ 0$ $0\ 1\ 3\ 4\ 9\ 8\ 0$

Figure 2 An example of a relinking trajectory.
Note. Vertical lines represent arcs not in x_G.

References

Beardwood J, Halton JH , Hammersley JM (1959) The shortest path through many points. *Math. Proc. Cambridge Philos. Soc.* 55:299–327.

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

Boudia M, Louly MAO, Prins C (2007) A reactive GRASP and path relinking for a combined production-distribution problem. *Comput. Oper. Res.* 34:3402–3419.

Bräysy O, Dullaert W, Gendreau M (2004) Evolutionary algorithms for the vehicle routing problem with time windows. *J. Heuristics* 10:587–611.

Bräysy O, Gendreau M (2005a) Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transportation Sci.* 39:104–118.

Bräysy O, Gendreau M (2005b) Vehicle routing problem with time windows, part II: Metaheuristics. *Transportation Sci.* 39:119–139.

Cattaruzza D, Absi N, Feillet D, Guyon O, Libeaut X (2013) The multi-trip vehicle routing problem with time windows and release dates. 10^{th} Metaheuristics Internat. Conf., Singapore.

Chang Y-C, Lee C-Y (2004) Machine scheduling with job delivery coordination. *Eur. J. Oper. Res.* 158:470–487.

Chen Z-L (2010) Integrated production and outbound distribution scheduling: Review and extensions. *Oper. Res.* 58:130–148.

Chen Z-L, Vairaktarakis GL (2005) Integrated scheduling of production and distribution operations. *Management Sci.* 51:614–628.

Christofides N, Mingozzi A, Toth P (1979) The vehicle routing problem. Christofides N, Mingozzi A, Toth P, Sandi C, eds. *Combinatorial Optimization* (Wiley, Chichester, UK), 315–338.

Coelho LC, Cordeau J-F, Laporte G (2014) Thirty years of inventory routing. *Transportation Sci.* 48:1–19.

Cordeau J-F, Laporte G, Mercier A (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *J. Oper. Res. Soc.*52:928–936.

Cordeau J-F, Maischberger M (2012) A parallel iterated tabu search heuristic for vehicle routing problems. *Comput. Oper. Res.* 39:2033–2050.

Dantzig GB, Ramser JH (1959) The truck dispatching problem. *Management Sci.* 6:80–91.

Desaulniers G, Desrosiers J, Spoorendonk S (2010) The vehicle routing problem with time windows: State-of-the-art exact solution methods. JJ Cochran, ed., *Wiley Encyclopedia of Operations Research and Management Science*. Wiley, New York, 5742–5749.

Desaulniers G, Madsen OBG, Ropke S (2014) The vehicle routing problem with time windows. P Toth, D Vigo, eds., *Vehicle Routing: Problems, Methods and Applications*. SIAM, Philadelphia, US, 5742–5749.

Ergun Ö, Orlin JB (2006) Fast neighborhood search for the single machine total weighted tardiness problem. *Oper. Res. Lett.* 34:41–45.

Fu Z, Eglese R, Li LYO (2008) A unified tabu search algorithm for vehicle routing problems with soft time windows. *J. Oper. Res. Soc.* 59:663–673.

Garey MR, Johnson DS (1978) “Strong” NP-completness results: Motications, examples and implications. *J. ACM* 25:499–508.

Gendreau M, Tarantilis CD (2010) Solving large-scale vehicle routing problems with time windows: The state-of-the-art. Report CIRRELT-2010-04, Université de Montréal, Montreal, CA.

Glover F (1989) Tabu search - Part 1. *ORSA J. Comput.* 1:190–206.

Glover F, Hao J-K (2011) The case for strategic oscillation. *Ann. Oper. Res.* 183:163–173.

Glover F, Laguna M, Martí R (2000) Fundamentals of scatter search and path relinking. *Control Cybernet.* 39:653–684.

Hall NG, Posner ME (2001) Generating experimental data for computational testing with machine scheduling applications. *Oper. Res.* 49:854–865.

Hall NG, Potts CN (2003) Supply chain scheduling: Batching and delivery. *Oper. Res.* 51:566–584.

Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.* 9:159–95.

Hashimoto H, Yagiura M (2008) A path relinking approach with an adaptive mechanism to control parameters for the vehicle routing problem with time windows. J. Hemert, C. Cotta, eds. *Evolutionary Computation in Combinatorial Optimization* (Springer, New York), 254–265.

Hendel Y, Sourd F (2006) Efficient neighborhood search for the one-machine earliness & tardiness scheduling problem. *Eur. J. Oper. Res.* 173:108–119.

Hoff A, Andersson H, Christiansen M, Hasle G, Løkketangen A (2010) Industrial aspects and literature survey: Fleet composition and routing. *Comput. Oper. Res.* 37:2041–2061.

Ibaraki T, Imahori S,Kubo M, Masuda T, Uno T, Yagiura M (2005) Effective Local Search Algorithms for Routing and Scheduling Problems with General Time-Window Constraints. *Transportation Sci.* 39:206–232.

Ibaraki T, Imaho S, Nonobe K, Sobue K, Uno T, Yagiura M (2008) An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. *Discrete Appl. Math.* 156:2050–2069.

Johar F (2014) PhD Progress Report. University of Southampton, UK.

Johnson DS, McGeoch LA, Rothberg EE (1996) Asymptotic experimental analysis for the Held-Karp traveling salesman bound. *Proc. 7th ACM-SIAM Sympos. Discrete Algorithms*, SIAM, 341–350.

Kindervater G, Savelsbergh M (1997) Vehicle routing: Handling edge exchanges. E Aarts, JK Lenstra, eds. *Local Search in Combinatorial Optimisation* (Wiley, Chichester, UK), 337–360.

Lenstra J, Rinnooy Kan AHG, Brucker P (1977) Complexity of machine scheduling problems. *Ann. Discrete Math.* 1:343–362.

Liberatore F, Righini G, Salani M (2010) A column generation algorithm for the vehicle routing problem with soft time windows. *4OR* 9:49–82.

Lin S (1965) Computer solutions of the traveling salesman problem. *Bell System Tech. J.* 44:2245–2269.

Lourenço HR, Martin OC, Stützle, T (2010) Iterated local search: Framework and applications. M Gendreau, J-Y Potvin, eds. *Handbook of Metaheuristics* (Springer, New York), 363–397.

Mercer RE, Sampson JR (1978) Adaptive Search Using a Reproductive Meta-Plan. *Kybernetes* 7:215–228.

Nagata Y, Bräysy, O (2008) Efficient local search limitation strategies for vehicle routing problems. *Evol. Comput. Combin. Optim.* 4972:48–60.

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

Nagy G, Salhi S (2007) Location-routing: Issues, models and methods. *Eur. J. Oper. Res.* 177:649–672.

Pecin D (2014) Exact algorithms for the capacitated vehicle routing problem. PhD Thesis, Pontifícia Universidade Católica do Rio de Janiero, Rio de Janiero, Brazil.

Potvin J-Y (2009) State-of-the art review: Evolutionary algorithms for vehicle routing. *INFORMS J. Comput.* 21:518–548.

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

Reghioui M, Prins C, Labadi N (2007) GRASP with path relinking for the capacitated arc routing problem with time windows. M Giacobini, ed., *Appl. of Evol. Comput.*. Springer, 722–731.

Resende MGC, Ribeiro CC, Glover F, Martí R (2010) Scatter search and path-relinking: Fundamentals, advances, and applications. M Gendreau, J-Y Potvin, eds. *Handbook of Metaheuristics* (Springer, New York), 87–107.

Smit SK, Eiben AE (2009) Comparing parameter tuning methods for evolutionary algorithms. *Proc.* 11^{th} *Cong. Evol. Comput.* (IEEE, Picastaway, NJ), 399–406.

Solomon, MM (1997) Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* 35:254–265.

Sörensen K, Schittekat P (2013) Statistical analysis of distance-based path relinking for the capacitated vehicle routing problem. *Comput. Oper. Res.* 40:3197–3205.

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

Toth P, Vigo D (2002) *The vehicle routing problem.* SIAM, Philadelphia, US.

Ullrich, CA (2013) Integrated machine scheduling and vehicle routing with time windows. *Eur. J. Oper. Res.* 227:152–165.

Vidal T, Crainic TG, Gendreau M, Lahrichi N, Rei W (2012) A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Oper. Res.* 60:611–624.

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

Vidal T, Crainic TG, Gendreau M, Prins C (2013b) 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.

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

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

Vidal T, Crainic TG, Gendreau M, Prins C (2015b) Timing problems and algorithms: Time decisions for sequences of activities. *Networks* 65:102–128.