CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

A Hybrid Genetic Algorithm for the Periodic Vehicle Routing Problem with Time Windows

Phuong Khanh Nguyen
Teodor Gabriel Crainic
Michel Toulouse

April 2011

CIRRELT-2011-25

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

A Hybrid Genetic Algorithm for the Periodic Vehicle Routing Problem with Time Windows

Phuong Khanh Nguyen[1,2], Teodor Gabriel Crainic[1,3,*], Michel Toulouse[1,4]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)
[2] Department of Computer Science and Operations Research, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7
[3] Department of Management and Technology, Université du Québec à Montréal, C.P. 8888, succursale Centre-ville, Montréal, Canada H3C 3P8
[4] Department of Computer Science, State University, 700 North Greenwood Avenue, North Hall 328, Tulsa, OK 74106-0700, USA

Abstract. We propose a new population-based hybrid meta-heuristic for the periodic vehicle routing problem with time windows. Two neighborhood-based meta-heuristics are used to educate the offspring generated by a new crossover operator to enhance the solution quality. This hybridization provides the means to combine the exploration capabilities of population-based methods and the systematic, sometimes aggressive search capabilities of neighborhood-based methods, as well as their proficiency to explore the infeasible part of the search space to both repair infeasible solutions and tunnel toward, hopefully, improved ones. Extensive numerical experiments and comparisons with all methods proposed in the literature show that the proposed methodology yields very high quality solutions, improving those currently published.

Keywords. Periodic Vehicle Routing Problem (PVRP), time windows, hybrid genetic algorithm, meta-heuristics, tabu search, variable neighborhood search.

Acknowledgements. Partial funding for this project has been provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grants programs.

---

* Corresponding author: TeodorGabriel.Crainic@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2011

1 Introduction

The Vehicle Routing Problem (VRP) is one of the most extensively studied problems in operations research due to its methodological interest and practical relevance to many fields, including transportation, logistics, telecommunications, and production; see, e.g., a number of recent surveys (Bräysy and Gendreau, 2005a,b; Cordeau et al., 2002a,b, 2007; El-mihoub et al., 2006; Gendreau et al., 2002; Golden et al., 2002; Laporte and Semet, 2002; Laporte et al., 2000) and books (Golden et al., 2008; Toth and Vigo, 2002). Many of these contributions targeted basic problem settings such as the capacitated VRP and the Vehicle Routing Problem with Time Windows (VRPTW). More recent and significantly less studied are richer problem settings (Hartl et al., 2006) aiming at more refined representations of actual applications and combining several "complicating" requirements and restrictions, such as customers that require multiple visits, heterogeneous vehicle fleets, limits on route duration or length, etc.

We focus on such a rich, relatively little studied VRP setting, namely the *Periodic Vehicle Routing Problem with Time Windows* (*PVRPTW*). Addressing the PVRPTW requires the generation of a limited number of routes for each day of a given planning horizon, to minimize the total travel cost while satisfying the constraints on vehicle capacity, route duration, customer service time windows, and customer visit requirements. The PVRPTW generalizes the VRPTW by extending the planning horizon to several days where customers generally do not require delivery on every day in this period, but rather according to one of a limited number of possible combinations of visit days (the so-called *patterns*). This generalization extends the scope of applications to many commercial distribution activities such as waste collection, street sweeping, grocery distribution, mail delivery, etc. It also raises new resolution challenges due to the requirement of balancing aggregate daily workloads in order to achieve efficient feasible solutions. The PVRPTW is actually NP-Hard as it includes the Periodic Vehicle Routing Problem (PVRP), known to be NP-hard, while the single-period case corresponds to the NP-hard VRPTW (Lenstra and Rinnooy Kan, 1981).

In this paper, we introduce the first genetic algorithm (*GA*) for the PVRPTW. It is a population-based hybrid meta-heuristic in which a set of neighborhood-based meta-heuristics cooperate with the GA population evolution mechanism to enhance the solution quality. This hybridization provides the means to combine the exploration capabilities of population-based methods and the systematic, sometimes aggressive search capabilities of neighborhood-based methods, as well as their proficiency to explore the infeasible part of the search space to both repair infeasible solutions and tunnel toward, hopefully, improved ones.

The crossover operators we propose for the PVRPTW constitute one of the main contributions of our work and aim to keep the balance between exploration and exploitation. The first operator perturbs parents to uncover new points in the search space,

thus favoring exploration. The second combines the two selected parents to exploit the good features and the search history information they contain. The second contribution is the hybridization, which uses local search procedures and neighborhood-based meta-heuristics as education strategies to repair and enhance the fitness of the GA population. Studies have been published where GAs are hybridized with local-search methods in order to improve their exploitation capability (e.g., El-mihoub et al., 2006; Knowles and Corne, 2000; Ishibuchi and Narukawa, 2004). Local search may reduce the diversity of the population (Merz and Katayama, 2004), however, and therefore limit the exploring capability of the GA. A few studies have indicated that a more aggressive search with a neighborhood-based meta-heuristic starting from the crossover-generated offspring may improve the diversity makeup of educated offspring and the global performance of the GA in terms of solution quality (Crainic and Gendreau, 1999; Lü et al., 2010). Our concept of GA hybridization builds on these insights. In the hybrid algorithm we propose, offspring are thus first educated using neighborhood-based meta-heuristics, which extend the search along routes and patterns beyond the offspring's immediate local optimum. Then, more intensification-oriented local search procedures optimize the educated offspring relative to its patterns and routes. A logical byproduct of this more extensive exploration of the offspring neighborhoods by meta-heuristics is a graceful restoration of the feasibility of a very large percentage of the infeasible offspring produced by GA crossover operators.

We tested the algorithm we propose on all previously published benchmark instances and compare our results to all currently published results. The proposed *Hybrid Genetic Algorithm* (*HGA*) produces 19 new best-known solutions and finds 1 best-known solution on the set of 20 PVRPTW instances of Cordeau et al. (2001), improving the solution quality by 0.80% on average in term of best solution cost. We also improve all the 45 instances of Pirkwieser and Raidl (2008), improving the average solution cost by 0.90%. The proposed HGA improves all the currently published results in the literature, and outperforms the other methods even when they are given the same computational time. We hope these encouraging results will contribute to stimulate more investigations into developing hybrid meta-heuristics for solving heavily constrained optimization problems.

The remainder of the paper is organized as follows. We formulate the problem in Section 2 and give a brief literature review in Section 3. The new HGA and its components are introduced and discussed in Section 4. Section 5 is dedicated to the experimental results. Finally, Section 6 concludes the paper.

2 Problem formulation

The PVRPTW is defined on a complete undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{0, 1, \ldots, n\}$ is the vertex set and $\mathcal{E} = \{(i, j) : i, j \in \mathcal{V}, i \neq j\}$ is the edge set. A

distance (or travel time) c_{ij} is associated with every edge $(i,j) \in \mathcal{E}$. The depot vertex is indexed by 0. $\mathcal{V}_c = \mathcal{V}\backslash\{0\}$ is the set of customer vertices. Each vertex $i \in \mathcal{V}_c$ has a demand $q_i \geq 0$ on each day of the planning horizon of $\mathcal{T}$ days, a service time $s_i \geq 0$, a time window $[e_i, l_i]$, where e_i is the earliest time service may begin and l_i is the latest time, and requires a fixed number of visits f_i to be performed according to one of the allowable visit-day patterns in the list $\mathcal{R}_i$. The time window specifying the interval vehicles leave and return to the depot is given by $[e_0, l_0]$. A fleet of m vehicles, each with capacity Q_k is based at the depot. Vehicles are grouped into set $\mathcal{K}$. Vehicle routes are restricted to a maximum duration of D_k, $k = 1, \ldots, m$.

In this paper, we address the case with a homogeneous vehicle fleet with $Q_k = Q$ and a common duration restriction $D_k = D, \forall k = 1, \ldots, m$. The PVRPTW can then be seen as the problem of generating (at most) m vehicle routes for each day of the planning horizon, to minimize the total cost over the entire planning horizon, such as 1) each vertex i is visited the required number of times, f_i, corresponding to a single pattern of visit days chosen from R_i, and is serviced within its time window; these are soft, i.e., a vehicle may arrive before e_i and wait to begin service; 2) each route starts from the depot, visits the vertices selected for that day, with a total demand not exceeding Q, and returns to the depot after a duration (travel time) not exceeding D.

Let a_{rt} be 1 if day $t \in \mathcal{T}$ belongs to pattern r, and 0 otherwise. Route-selection, pattern-selection, and continuous timing decision variables are used in the formulation:

- $x^t_{ijk} = \begin{cases} 1 & \text{if vehicle } k \in \mathcal{K} \text{ traverses edge } (i,j) \in \mathcal{E} \text{ on day } t \in \mathcal{T}; \\ 0 & \text{otherwise;} \end{cases}$
- $y_{ir} = \begin{cases} 1 & \text{if pattern } r \in \mathcal{R}_i \text{ is assigned to customer } i \in \mathcal{V}_c; \\ 0 & \text{otherwise;} \end{cases}$
- w^t_{ik} indicates the service starting time for vehicle $k \in \mathcal{K}$ at customer $i \in \mathcal{V}_c$ on day $t \in \mathcal{T}$.

Let M be an arbitrary large constant. The PVRPTW can then be formulated as

$$\text{Minimize} \quad \sum_{t\in\mathcal{T}} \sum_{(i,j)\in\mathcal{E}} \sum_{k\in\mathcal{K}} c_{ij} x^t_{ijk} \tag{1}$$

$$\text{S.t.} \quad \sum_{r\in\mathcal{R}_i} y_{ir} = 1, \quad \forall i \in \mathcal{V}_c, \tag{2}$$

$$\sum_{j\in\mathcal{V}} x^t_{ijk} = \sum_{j\in\mathcal{V}} x^t_{jik}, \quad \forall i \in \mathcal{V}, k \in \mathcal{K}, t \in \mathcal{T}, \tag{3}$$

$$\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{V}} x_{ijk}^t = \sum_{r\in\mathcal{R}_i} y_{ir}a_{rt} \quad \forall i \in \mathcal{V}_c, t \in \mathcal{T}, \tag{4}$$

$$\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{V}} x_{0jk}^t \leq m, \quad \forall t \in \mathcal{T}, \tag{5}$$

$$\sum_{i,j\in\mathcal{S}} x_{ijk}^t \leq |\mathcal{S}| - 1, \quad \forall S \subseteq \mathcal{V}_c, k \in \mathcal{K}, t \in \mathcal{T}, \tag{6}$$

$$\sum_{j\in\mathcal{V}_c} x_{0jk}^t \leq 1, \quad \forall k \in \mathcal{K}, t \in \mathcal{T}, \tag{7}$$

$$\sum_{i\in\mathcal{V}_c} q_i \sum_{j\in\mathcal{V}} x_{ijk}^t \leq Q, \quad \forall k \in \mathcal{K}, t \in \mathcal{T}, \tag{8}$$

$$w_{ik}^t + s_i + c_{ij} - M(1 - x_{ijk}^t) \leq w_{jk}^t, \quad \forall (i,j) \in \mathcal{E}, k \in \mathcal{K}, t \in \mathcal{T}, \tag{9}$$

$$e_i \sum_{j\in\mathcal{V}} x_{ijk}^t \leq w_{ik}^t \leq l_i \sum_{j\in\mathcal{V}} x_{ijk}^t, \quad \forall i \in \mathcal{V}_c; k \in \mathcal{K}; t \in \mathcal{T}, \tag{10}$$

$$w_{ik}^t + s_i + c_{i0} - M(1 - x_{i0k}^t) \leq D, \quad \forall i \in \mathcal{V}_c; k \in \mathcal{K}; t \in \mathcal{T}, \tag{11}$$

$$x_{ijk}^t \in \{0,1\}, \quad \forall (i,j) \in \mathcal{E}, k \in \mathcal{K}, t \in \mathcal{T}, \tag{12}$$

$$y_{ir} \in \{0,1\}, \quad \forall i \in \mathcal{V}_c; r \in \mathcal{R}_i, \tag{13}$$

$$w_{ik}^t \geq 0, \quad \forall i \in \mathcal{V}_c, k \in \mathcal{K}, t \in \mathcal{T}. \tag{14}$$

The objective function (1) minimizes the total travel cost. Constraints (2) ensure that a feasible pattern is assigned to each customer. Constraints (3) enforce flow conservation ensuring that a vehicle arriving at a customer on a given day, leaves that customer on the same day. Constraints (4) guarantee that each customer is visited on the days corresponding to the assigned pattern, while Constraints (5) make sure that the number of vehicles used on each day does not exceed m. Relations (6) are subtour elimination constraints. Constraints (7) ensure that each vehicle is used at most once a day, while (8) guarantee that the load charged on a vehicle does not exceed its capacity. Constraints (9) enforce time feasibility, i.e., vehicle k cannot start servicing j before completing service at the previous customer i and traveling from i to j, i.e., not before $w_{ik}^t + s_i + c_{ij}$. Constraints (10) ensure that customer time window restrictions are respected, while (11) constrain the route length. Constraints (12), (13), and (14) define the sets of decision variables.

3 Literature review

The complexity of the PVRPTW calls for heuristic solution approaches when realistically-sized instances are contemplated. Cordeau et al. (2001) introduced the problem setting and pioneered the application of heuristics by proposing a tabu search algorithm, which

allows infeasible solutions together with associated penalty terms in the objective function for violations of time windows, route duration, and vehicle capacity constraints. Moves either relocate a customer to a different route in the same day or change its pattern, which provides two ways to improve solutions, by modifying the routing and the visit pattern assignments to customers. The authors also introduced a set of 20 benchmark PVRPTW instances.

Pirkwieser and Raidl (2008) proposed a Variable Neighborhood Search (VNS) heuristic for the PVRPTW, with the particularity that it accepts worsening solutions based on a Metropolis criterion. Pirkwieser and Raidl (2009a) later introduced a hybrid scheme between this VNS heuristic and an ILP-based column generation procedure addressing a set-covering formulation. In this hybrid, the VNS is the sole provider of columns for the set-covering, which is solved via a generic ILP solver. If the latter improves on the current best solution, this new solution is transfered to the VNS for further enhancement. Since ILP solvers cannot tackle large instances, validation relied on a new set of smaller instances derived from the basic Solomon VRPTW - 100 customers instances. The authors then proposed a hybrid between an evolutionary algorithm and the column generation approach (Pirkwieser and Raidl, 2010), as well as a rigid synchronous cooperative multi-search approach, named multiple VNS (mVNS) (Pirkwieser and Raidl, 2009b). In the latter setting, several VNS meta-heuristics run independently, synchronize after a given number of iterations to determine the best solution, the worst VNS thread being then restarted from this best solution, while the others continue their own search. The mVNS was also hybridized with the column generation approach as per Pirkwieser and Raidl (2009a). At a synchronization point, the best mVNS solution is passed to the IPL solver. If the resulting solution improves the mVNS best solution, the worst VNS search is initialized with it and the mVNS is restarted. This mVNS-ILP combination is repeated a fixed number of times. The mVNS-ILP hybrid generally produced better results than mVNS, without dominating over the entire instance set.

Overall, the current best published results on the 20 instances of Cordeau et al. (2001) are reported by Pirkwieser and Raidl (2008). Only the latter authors published results for the set of smaller instances they introduced Pirkwieser and Raidl (2009b), but reported average solution costs instead of best solution costs. With respect to this criteria, the current best solutions are provided by Pirkwieser and Raidl (2009b) for instances with a planning horizon of four and six days, and by (Pirkwieser and Raidl, 2010) for instances with planning horizon of eight days.

4 The Proposed Hybrid Meta-heuristic

This section is dedicated to introducing the Hybrid Genetic Algorithm we propose. We start by describing the individual representation and evaluation procedure (Sections 4.1

and 4.2, respectively). The main phases of the HGA, illustrated in Figure 1, are typical of generational genetic algorithms and are introduced next. The algorithm uses one population only, which may contain both feasible and infeasible individuals. The initial population is created using three greedy heuristics (Section 4.3). A new population is generated from the current one through selection, crossover, mutation, education, and replacement operators (Sections 4.4 to 4.7, respectively). The population is always ordered in increasing order of fitness. Our contributions are both in adapting GA operators to the particular requirements of the PVRPTW and in designing the overall organization of the hybrid algorithm to answer the challenges of this problem setting.

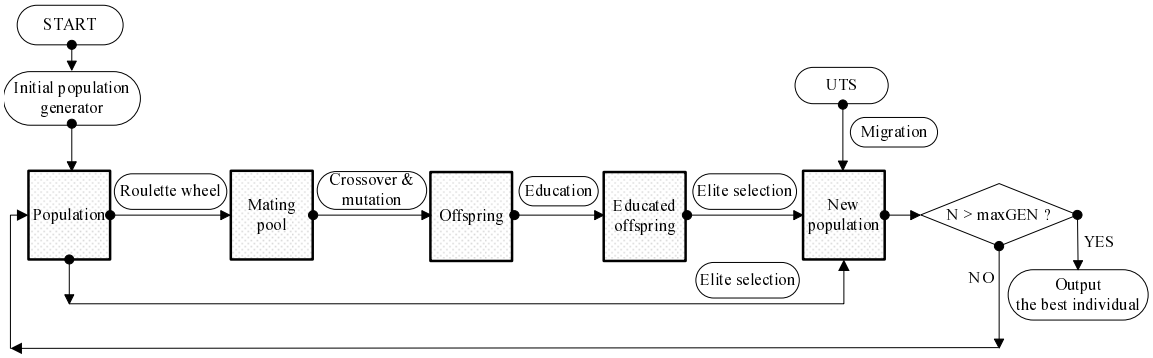

Figure 1: The Hybrid Genetic Algorithm Structure

4.1 Individual representation

An individual for the HGA we propose corresponds to a feasible or infeasible solution to the PVRPTW, which specifies the pattern assigned to each customer, the number of routes (that is, vehicles), and the delivery order within each route. Each individual is then represented by two chromosomes, the first addressing the pattern-to-customer assignments and the second corresponding to the routes performed on each day of the planning horizon.

The **Pattern** chromosome is associated with the n customers. Each entry i of this chromosome is a positive integer k that describes the pattern assigned to customer i. The binary representation of k stands for the days the associated customer receives a visit. Figure 2 illustrates the representation of an individual corresponding to a solution of an instance of 10 customers and 3 days. In this illustration of the Pattern chromosome (Figure 2a), the first customer is serviced according to Pattern[1] = 3 = {011}, i.e., on day 2 and 3 (days are numbered from left to right).

For each day in the planning horizon, a group of routes services customers on that day. The **Route** chromosome corresponds to the combination of this set of vectors, each representing the ordered sequence of customers for one route of a day. Figure 2b

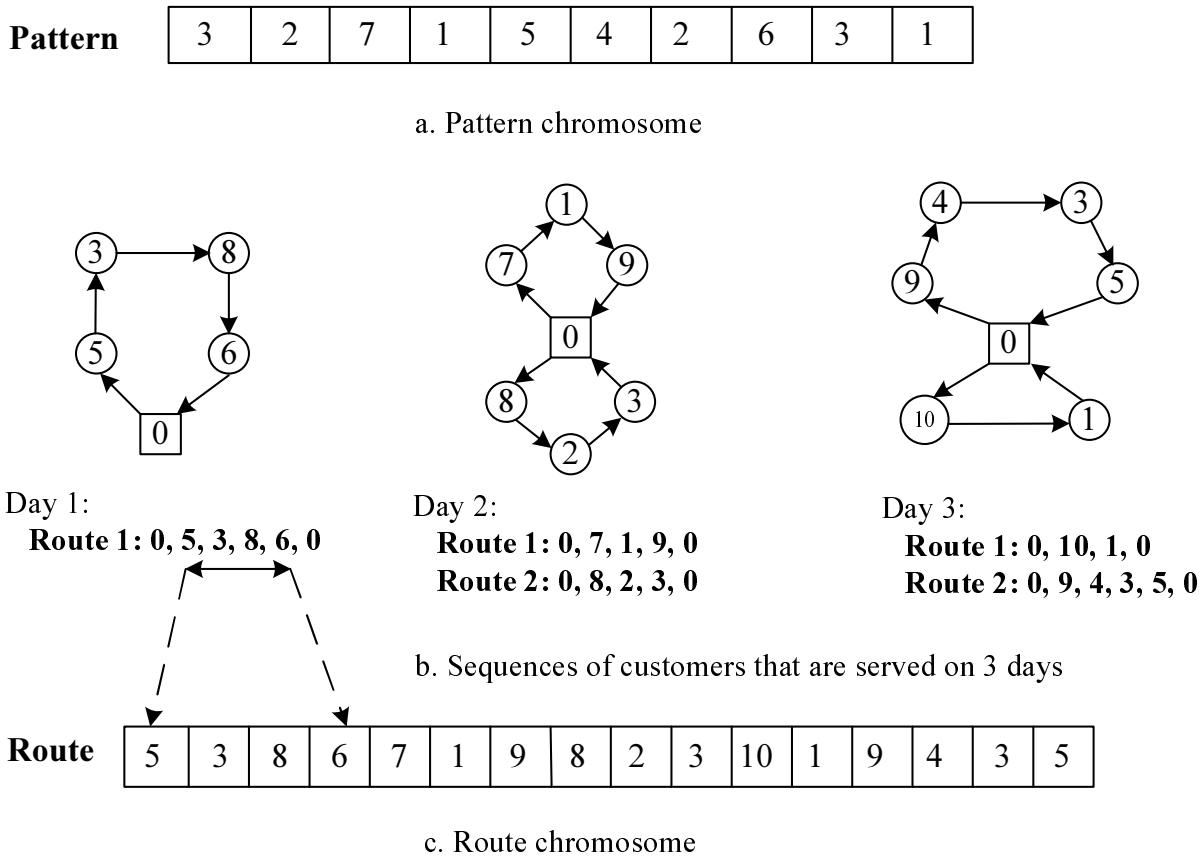

Figure 2: Representation of an individual.

illustrates the routes for the 3 days, while Figure 2c displays the corresponding Route chromosome (the only route of the first day is emphasized).

4.2 Search space and individual evaluation

Allowing meta-heuristics to consider infeasible solutions often yields a better search able to reach higher-quality solutions more efficiently (e.g., Cordeau et al., 2001). We follow this trend and explicitly allow infeasible solutions during the search process by relaxing constraints on the maximum vehicle load, route duration, and customer service time windows.

Given a solution s, let $c(s)$ denote the total travel cost of its routes, and let $q(s)$, $d(s)$, and $w(s)$ denote the total violation of capacity, duration, and time window restrictions, respectively. The values of $q(s)$ and $d(s)$ are computed on a route basis with respect to the Q and D values, whereas $w(s) = \sum_{i=1}^{n} max\{(a_i - l_i), 0\}$, where a_i is the arrival time at customer i. Solutions are then evaluated according to the weighted *fitness function*

$f(s) = c(s) + \alpha q(s) + \beta d(s) + \gamma w(s)$, where α, β, and γ are penalty parameters adjusted dynamically during the search.

Several techniques are available to adjust the penalty parameters. Thus, Cordeau et al. (2001) based their update on the current solution. We prefer to follow Barbosa and Lemonge (2002), which makes use of information related to the complete population. Let $\overline{q}$, $\overline{d}$, and $\overline{w}$ stand for the violation of vehicle capacity, route duration, and customer service time window constraints, respectively, averaged over the current population. Let

$$h = \begin{cases} c(s_{worst}) & \text{if there is no feasible solution in the population;} \\ c(s_{bestfeasible}) & \text{otherwise.} \end{cases}$$

The penalty parameters are then computed by the following rules:

$$\alpha = h\frac{\overline{q}}{\overline{q}^2 + \overline{d}^2 + \overline{w}^2}\ , \quad \beta = h\frac{\overline{d}}{\overline{q}^2 + \overline{d}^2 + \overline{w}^2}\ , \quad \gamma = h\frac{\overline{w}}{\overline{q}^2 + \overline{d}^2 + \overline{w}^2}\ .$$

Every time the current best feasible solution is improved, h is redefined, all fitness values are recomputed using the updated penalty coefficients, and the population is sorted accordingly. This adaptive scheme automatically determines the penalty parameter corresponding to each group of constraints during the evolutionary process so that the constraints that are more difficult to satisfy receive a relatively higher penalty coefficient.

4.3 Initial population

Solutions in the initial population are generated by, first, assigning randomly an allowable pattern of visit days to each customer and, second, by solving a VRPTW for each day using three greedy heuristics: 1) the Time-Oriented, Nearest-Neighbor heuristic of Solomon (1987); 2) the parallel route building heuristic of Potvin and Rousseau (1993); 3) our own route construction method. We use the methods of Solomon (1987) and of Potvin and Rousseau (1993) because these heuristics are very fast, and they appear to be complementary. Indeed, comparing the two, the former seems to perform better for clustered problem instances, while the opposite is true for the other problem settings (Bräysy and Gendreau, 2005a). Moreover, applying three different heuristics helps create diversity within the initial population.

Our own route construction method is quite flexible with regard to problems with a fixed number of vehicles. It follows the cluster first - route second scheme. During clustering, customers are first sorted in increasing order of the angle they make with the depot. Next, a customer j is chosen randomly and the sequence of n customers $j, j+1, ..., n, 1, ..., j-1$ is divided into m clusters of size $\lceil n/m \rceil$ (the last one may be

smaller), one for each available vehicle. Clustering is performed by a sweep starting from j and proceeding counter-clockwise through the customers.

Routing is performed iteratively for each cluster using Solomon's insertion heuristic of type I1. For added flexibility in routing, the procedure considers not only the customers in the cluster, but also a small number of customers not yet being serviced by a route, selected from the two immediate neighboring clusters. Each route is initialized with the customer not yet assigned to a route, which displays the lowest ending time for service. The remaining not-yet-assigned customers are then added sequentially to the route until it is full, with respect to vehicle-capacity and route-duration constraints. The customers not yet assigned left once the m routes are created, if any, are then inserted into the existing routes to minimize the increase in the total travel distance. The algorithm stops when all customers are serviced.

4.4 Mating selection

The selection operator chooses high-fitness individuals within the population for mating purposes. A *mating pool* of $nPop$ individuals is thus formed by using a Roulette-wheel procedure that provides individuals with high fitness values higher probabilities of being selected. An individual may be selected more than once. Then, each time offspring are required during the course of the HGA, two individuals in the mating pool are selected randomly and passed to the crossover operators. These two individuals are then deleted from the mating pool.

4.5 Crossover and mutation operators

Exploration and exploitation are important issues for search algorithms. We therefore propose two crossover operators for the PVRPTW, one aiming at exploring the search space, while the other seeks to exploit the information present within the population. Combined, these two crossover operators help to keep the balance between exploration and exploitation, thus improving the search efficiency of the algorithm. In the following presentation, P_1 and P_2 denote the parents involved in a crossover operation.

The first operator favors exploration by perturbing parents to uncover new points in the search space. It creates offspring by transferring a partial set of routes from one parent, along with pattern assignments from both. *The exploration crossover operator* proceeds in two steps:

STEP 1. Assign a pattern to each customer

(a) *Inherit pattern assignments from parent* P_1. Randomly select two cutting points

in the sequence of customers of the Route chromosome of parent P_1. The visit days of customers between these cutting points are copied into offspring C.

(b) *Inherit pattern assignments from parent* P_2. Let $day(i)$ denote the set of visit days currently assigned to customer i in C. Scan the Pattern chromosome of parent P_2 from left to right: for each day t and customer i with frequency not satisfied yet and not already in C, copy customer i on day t into C if there exists a pattern of visit days in R_i including $day(i) \cup \{t\}$.

(c) *Complete pattern assignments*. Assign a random pattern $r \in R_i$, such that r includes $day(i)$, to each customer i whose frequency is not satisfied.

STEP 2. Assign customers to routes

(a) *Copy routes from parent* P_1. The customers between the two cutting points determined in Step 1a are routed as in the P_1 parent and the corresponding sequences are copied into C.

(b) *Assign remaining customers to routes*. The customers in C not yet routed are assigned to routes using the cost insertion procedure of Potvin and Rousseau (1993).

Parents selected for mating are instances of good individuals in the current population. The second crossover operator is designed to exploit the good genetic material and the search history information they contain through route combination. *The exploitation crossover operator* treats each day t in the planning horizon, and randomly selects one parent among $\{P_1, P_2\}$. All routes of the selected parent in day t are copied into the offspring C. Then, customers are removed/inserted from/into days such that the pattern of visit days in C are satisfied for all customers. If there is more than one possibility to delete a customer, the one that gives the maximum gain is deleted. Customers are inserted using the cost insertion of Potvin and Rousseau (1993).

The mutation operator is applied to each offspring yielded by the crossover operators. Mutation consists in changing the pattern assignment of a few customers, which are selected through a low probability P_m. A new pattern is then assigned to each selected customer i: Either an eligible pattern (i.e., in R_i) not assigned to i in any of the individuals of the current population, if such a pattern exists, or one not assigned to i in the current offspring.

4.6 Education

Crossover and mutation operators yield offspring, which may be feasible or infeasible, but is "sent to school" in all cases. The goal of the *Education* procedure is to improve the quality of the offspring, as well as restore as much feasibility as possible (when needed).

Two issues must be addressed in this context. First, while GAs proved their worth in exploring broad and complex search spaces, they also appear less well suited for fine-tuning solutions which are near local optima (Gendreau and Potvin, 2005; García-Martínez and Lozano, 2008). Hybridization with local-search methods has been proposed to address this issue, but this strategy often degrades the diversity of the population (Merz and Katayama, 2004), and therefore limits the capability of the GA to find successions of improvements through its iterative process ("exploration"). The second issue concerns the fact that crossover and mutation operators often yield offspring that violates some of the problem requirements and a repair phase is required. Local search has also been proposed to address this issue, but it might not be sufficient for heavily constrained optimization problems, like PVRPTW, as our initial experiments have shown.

We therefore propose an education procedure based on a different principle, one which embeds neighborhood-based meta-heuristics into the GA to both maintain its exploring ability and restore the feasibility of offspring before its insertion in the population. This principle follows the insight of our previous work on cooperative search methods (Crainic and Gendreau, 1999) and has been recently reinforced by the results of Lü et al. (2010). Compared with local-search procedures, a higher proportion of offspring have their feasibility restored by meta-heuristics and the educated offspring have a higher average fitness.

The proposed education procedure integrates two meta-heuristics, the *Unified Tabu Search* (*UTS*) of Cordeau et al. (2001) and the *Random Variable Neighborhood Search* (*RVNS*) of Pirkwieser and Raidl (2008). We selected these meta-heuristics for several reasons. On the one hand, they were applied to the PVRPTW and, thus, one obtains not only a basis for comparisons, but also known behavior and performance for the problem of interest. On the other hand, while both can contribute to routing and pattern-assignment improvements by changing the patterns assigned to customers and the routes of each day, the way moves are selected, evaluated, and performed is particular to each meta-heuristic. Thus, e.g., UTS selects between a routing or a pattern move based on the maximization of the cost improvement, while RVNS selects randomly at each iteration whether to execute a pattern or a route move. Combining the two yields a hybrid HGA, which produces more diversified individuals across generations.

The pseudo code of Algorithm 1, where CNG stands for the current number of generations, gives the general structure of the education procedure: Offspring is first educated through the neighborhood-based meta-heuristics and, then, intensification-oriented local search further enhance the educated offspring in their pattern assignment and routing dimensions. To save computation time and improve the diversify of the GA, UTS and RVNS are applied alternately every generation. A rather small number of iterations is allowed to each meta-heuristic. This may impair the performance of RVNS by "clashing" with its random move-selection characteristic, which helps to explore new points in the search space but yields poor solutions if the meta-heuristic is interrupted too soon.

Consequently, a local-search *Pattern-Improvement* procedure is applied to further improve the solutions obtained by RVNS. Finally, a *Route-Improvement* procedure locally re-optimizes the routes for each day separately.

Algorithm 1 EducationProcedure(solution s, CNG)

```
1: if CNG is even then
2:    UTS(s)
3: else
4:    RVNS(s)
5:    Pattern_improvement(s)
6: end if
7: Route_improvement(s)
8: return s
```

More in detail, the pattern-improvement procedure proceeds by assigning a new pattern to each customer and keeping those that actually improve the solution. Customers are handled in random order. Then, for each customer i and each of its unassigned patterns $r' \in R_i$ (if any), one removes i from the previous visit days and inserts it in the corresponding days of the new pattern r' so as to minimize the increase of the fitness function. A 2-opt heuristic is then applied to each route changed by this reassignment. If the new solution improves over the current one, a 2-opt* heuristic is applied for further improvement of the new current solution. One then proceeds to the next pattern or, if all have been tried out, to the next customer.

Route-improvement is the last education activity and is performed by applying a number of well-known local-search route improvement techniques. Two are intra-route operators, the 2-opt of Lin (1965) and the Or-opt of Or (1976). The others are inter-route operators, the λ-interchange Osman (1993), the 2-opt* of Potvin and Rousseau (1995), and the CROSS-exchange of Taillard et al. (1997). For the λ-interchange, we only consider the cases where $\lambda = 1$ and $\lambda = 2$ corresponding to the (1,0), (1,1), (2,0), (2,1), and (2,2)-interchange operators.

The procedure starts by applying in random order the five λ-interchange and the 2-opt* and CROSS-exchange inter-route operators. Each neighborhood is searched on all possible pairs of routes (in random order) of the same day and stopped on the first improvement. The solution is then modified and the process is repeated until no further improvement can be found. The search is then continued by locally improving each route of the current day in turn. The intra-route 2-opt and Or-opt neighborhoods are sequentially and repeatedly applied until no more improvement is found.

4.7 Generation replacement and HGA general structure

Once the mating pool is emptied, the generational change takes place. The goal is to produce a new generation that conserves the best characteristics of the individuals encountered so far and that displays a good variety of genetic material. An elitist approach is thus used, keeping some of the parents and all children, while also welcoming a few new individuals generated outside the evolutionary mechanism.

HGA evolves a population of $nPop+nKeep+nOFF$ individuals. Recall (Section 4.4) that $nPop$ parents are selected from the current population to build the mating pool (the same individual may appear more than once according to its fitness) and $nPop$ offspring are created. The next generation is then composed of these $nPop$ new individuals plus the best $nKeep$ individuals in the current population ($nKeep < nPop$). This elitism not only preserves the best individuals from one generation to the next, but also guarantee that the best individual at generation $t + 1$ is always better than or equal to the best individual at generation t. A small number $nOFF$ of new individuals is then created by using UTS to add new genetic material to the population. $nOFF$ is about 5% of $nPop$. Algorithm 2 summarizes the hybrid meta-heuristic we propose for the PVRPTW.

Algorithm 2 Hybrid Genetic Algorithm

```
1: Randomly generate an initial population of nPop individuals
2: repeat
3:    Evaluate the fitness for each population individual
4:    Create the mating pool with the nPop size using Roulette wheel
5:    while the mating pool is not empty do
6:       Select two parents at random from the mating pool
7:       Apply crossover operators to produce two offspring
8:       Apply mutation operator to each offspring
9:       Apply education procedure to the offspring
10:      Delete two selected parents from the mating pool
11:   end while
12:   Generation replacement
13: until stopping condition satisfied
14: Print the current best solution
```

5 Computational Analyzes

The objective of the numerical experimentations is twofold. First, to study a number of variants of main algorithmic components and strategies and thus develop insights into addressing the challenges of designing hybrid meta-heuristics for tightly constrained combinatorial optimization problems (Section 5.1). The second objective consists in

evaluating the performance of the proposed HGA through comparisons with currently published results (Section 5.2).

HGA is implemented in C++. Experiments were run on an Intel Core 2 Duo CPU workstation with 2.4GHz, 2GB RAM. Two sets of instances were used throughout the experiments. The first, introduced by Cordeau et al. (2001), is made up of two sets (identified as *a* and *b*) of ten Euclidean instances each, ranging from 48 to 288 customers, 3 to 20 homogeneous vehicles, and with a planning horizon of 4 or 6 days. Instances *a* and *b* have narrow and large time windows, respectively. The depot has a [0,1000] time window in all instances. The second set of instances was generated by Pirkwieser and Raidl (2009a) and is made up of three sets of fifteen instances each. These Euclidean instances were created based on the Solomon VRPTW 100-customer instance set with a planning horizon of four, six, and eight days, denoted *p4*, *p6*, and *p8*, respectively.

5.1 Analysis of design decisions

A hybrid meta-heuristic like HGA is always the result of a number of decisions on the structure, components, and parameter values of the method. Two main design components are studied through their impact on the behavior of the proposed HGA: the education process (Section 5.1.1), and the selection and replacement strategies (Section 5.1.2). The calibration of the search parameters is presented in Section 5.1.3.

5.1.1 Variants of the education procedure

The first experiments aimed to measure the impact of each neighborhood-based meta-heuristic on the GA performance and determine an appropriate hybridization scheme.

We implemented the UTS and RVNS meta-heuristics following the description given in Cordeau et al. (2001) and Pirkwieser and Raidl (2008), respectively (Annex A details the algorithms), and set up four hybrid algorithms following the general structure of Algorithm 2. The first two, *HGA-UTS* and *HGA-VNS*, embed one method only, UTS or RVNS being called at each iteration, respectively. The third, *HGA-UTS/VNS*, combines the two, UTS and RVNS being used alternately at every generation. The fourth, *HGA*, enhances the HGA-UTS/VNS structure by performing the pattern improvement procedure following the RVNS execution.

To level the comparison field, the number of iterations of UTS (1000) and RVNS (3000) was set to yield comparable computing times. Similarly, the number of iterations of RVNS was reduced for HGA such that the total computing effort for RVNS and pattern improvement be approximately the same as that of RVNS in HGA-UTS/VNS.

Experiments were contucted on the Cordeau et al. (2001) instances.

Table 1: Aggregated performance comparison between education schemes (Cordeau et al., 2001) instances

	HGA-UTS	*HGA-VNS*	*HGA-UTS/VNS*	*HGA*
Time (hours)	19.65	20.18	11.44	10.49
GAP	+0.50%	+0.94%	+0.03%	-0.80%

Table 1 displays the aggregated results of this experiment, as the average CPU time and gap to the best known solution (BKS) of each hybrid strategy. As expected, the experiments showed that hybridization with UTS and RVNS impacts the method differently, the aggregate performances being too close to discriminate (slight advantage to HGA-UTS). In the main, this is explained by how move types are selected by each meta-heuristic, either routing or pattern modification based on cost improvement, or random selection, provided it satisfies the Metropolis selection criteria, respectively. The results also showed that both hybrids improved in solution quality upon UTS but not relative to RVNS, and the situation did not change with increased education effort (e.g., doubling the number of UTS and RVNS iterations in the hybrids). Combining the two neighborhood-based meta-heuristics into the same hybrid algorithm produced the desired result, HGA-UTS/VNS outperforming HGA-UTS and HGA-VNS (even when the education effort was reduced, e.g., running 200 and 800 iterations of UTS and VNS, respectively). The hybrid also improved the BKS for small instances. The best performance, both in solution quality and computation time, is offered by the HGA hybrid, however, which adds the pattern-improvement post-optimization procedure to RVNS.

5.1.2 Variants of selection and replacement schemes

In the next set of experiments, we analyzed the impact of the selection operator and replacement policies on the diversity of the mating pool and population. Two types of selection operators, roulette wheel and binary tournament, were investigated in conjunction with two replacement strategies:

- Strategy 1. The population of generation $i + 1$ is made up of the $nKeep$ best solutions of generation i plus the $(nPop - nKeep)$ best offspring. The population and mating pool sizes, as well as the number of offspring are all equal to $nPop$.
- Strategy 2. The population of generation $i + 1$ is made up of the $nKeep$ best solutions of generation i plus all $nPop$ offspring. The size of mating pool and the number of offspring are equal to $nPop$, while the size of the population is $(nKeep + nPop)$.

Table 2: Proportion of individuals in the mating pool from $nKeep$ best solutions of previous generation

Strategy	$nKeep$	Binary Tournament	Roulette Wheel
1	15	35% - 70%	20% - 65%
	35	65 % - 95%	60% - 85%
2	15	30% - 60%	10% - 40%
	35	50% - 80%	30% - 65%

The experiment was run for 1000 generations with $nPop = 50$ and different values of $nKeep$. Table 2 reports the proportion of solutions in the mating pool that comes from the $nKeep$ best solutions of the previous generation, the percentage values in each row of the last two columns representing the lower and upper bounds on the number of solutions from $nKeep$, respectively. Results show that binary tournament selects more often solutions from $nKeep$ and, thus, it is more strongly biased toward previous elite solutions than roulette wheel. This favors the exploitation of accumulated information, but more education work might be needed to explore new regions of the solution space. We actually empirically observed a significant difference between the impact of the two replacement strategies on the performance of the HGA. For both selection operators, better solutions were obtained by Strategy 2 than Strategy 1. Consequently, roulette wheel selection and the second replacement strategy were used in all subsequent experiments.

5.1.3 Calibration of search parameters

The main parameters of the proposed HGA requiring calibration are the population size, the cardinality of the elite group, and the number of iterations for UTS and RVNS. The parameter values were selected considering a number of criteria, solution quality, improvement of the best solution through generations, and the computational cost for HGA. The calibration was performed using a small set of instances with diverse characteristics, such as small and large time windows, number of customers, and number of periods, which impact the ease of addressing instances. Thus, larger time windows imply more feasible places to insert customers into routes, thereby increasing the number of feasible solutions with respect to this constraint. Similarly, more customers or longer planning horizons imply more feasible pattern combinations.

Experiments showed that a somewhat larger population size and longer education explorations (number of iterations of UTS and RVNS meta-heuristics) yield a more extensive sampling of the solution space and favor identifying good solutions based on correct selections of patterns for customers together with good routings. The experiments also showed that appropriate parameter values change with the size of the instance. The

interval examined and the final parameter settings appear in Table 3, where small and large instances consist of less and more than 100 customers, respectively. Additionally, for all instances, the UTS procedure uses a tabu length of $\theta = 1.5log_{10}(n)$ (smaller than the $7.5log_{10}(n)$ in Cordeau et al., 2001).

Table 3: Calibration of main HGA parameters

Parameters	Interval explored	Final values	
		Small instances	Large instances
Population size ($nPop$)	[50, 400]	100	300
Elite set ccardinality ($nKeep$)	[25, 300]	60	200
Number of UTS iterations	[50, 150]	60	100
Number of RVNS iterations	[100, 800]	400	800
Maximum number of generations	[500, 5000]	750	2000, 3000

The last calibration step examined the stopping criterion, which is based on the total number of HGA iterations. The best solution and corresponding computation time were measured for each instance from generation 500 to generation 5000, by steps of 250. The results indicate that, for small instances, the best solutions can not be improved after 1000 generations, while the average improvement of the best solution between generations 750 and 1000 is less than 0.001%. Consequently, the number of generations was set to 750 for small instances. For large instances, the number of generations was set to 2000 and 3000 for instances with less and more than 200 customers, respectively.

5.2 Numerical Results

The performance of the proposed HGA is evaluated by comparing its performance with published results on the instances provided by Cordeau et al. (2001) and Pirkwieser and Raidl (2009a). More specifically, comparisons are performed with the UTS of Cordeau et al. (2001) and the VNS variants of Pirkwieser and Raidl (2008). For concision sake, only aggregated results are provided in this section. Details may be found in Annex B.

Table 4: Comparative performance on Cordeau et al. (2001) instances

Algorithms	Runs	Average GAP to BKS		Time (min)
		Best	Average	
UTS	11	2.48%	-	652
VNS	30	0.48%	1.97%	1.40
RVNS	30	0.33%	1.70%	1.53
VNS with 2-opt*	30	0.32%	1.70%	1.42
RVNS with 2-opt*	30	0.18%	1.50%	1.54
HGA	30	-0.80%	-0.44%	654

Table 4 displays comparison results on the Cordeau et al. (2001) instances. We report the number of repetitions (second column) performed and the corresponding average gap to the previous best known solutions (BKS) of the best and average solutions (the latter measure is not available for UTS). Average reported computing times are displayed in the last column.

HGA produces high quality solutions, with an average error gap of -0.44% to the previous BKS, compared to more than 1.50% for the other algorithms. HGA and VNS yield better solutions than UTS on all instances, while HGA produces better solutions than VNS on 19 out of 20 instances (with a maximum cost reduction of 1.60%), both algorithms identifying the same solution on instance 1a. The best solutions in the initial population are 28.38% greater than the best solutions obtained by HGA on average, illustrating the significant effect of hybrid strategy.

The next round of experimentations focused on the instances proposed by Pirkwieser and Raidl and used by them in Pirkwieser and Raidl (2009a,b, 2010). It is noteworthy that the authors truncated the calculated travel costs to one digit, which reduced the total cost. More importantly, for instances with tight time windows, truncation can produce many more feasible solutions compared to the no-truncation case, hence resulting in a quite huge reduction of total cost. We therefore give the performance results of HGA with and without truncation. One also notes that, best solutions were not reported, and only average solutions and standard deviations computed over 30 runs for each instance are available. We therefore selected the best solutions over all their algorithms, noted by *Previous Best Average*, and compare the results of HGA to this measure.

Table 5 sums up the comparison of average results from 30 runs of HGA, without truncation, with the reported results of the algorithms in Pirkwieser and Raidl (2009a,b, 2010): VNS and VNS-ILP (Pirkwieser and Raidl, 2009a); mVNS and mVNS-ILP (Pirkwieser and Raidl, 2009b); Evolutionary Algorithm (EA), combination of column generation and evolutionary algorithm (CG-EA), and CG-ILP (Pirkwieser and Raidl, 2010). We report for each algorithm, the averages of percentage of deviation from the Previous Best Average (Column *GAP*) and computation time (Column *Time*). Several settings were given for VNS-ILP, mVNS, and mVNS-ILP, without a clear dominating one. In this case we selected the best solution over all settings (5 or 7, each with 30 repetitions) for each instance set, together with the corresponding computation time. One observes that HGA performs again very well, obtaining the smallest average gap for all 45 instances, and improving the average cost by 0.33%, 0.68%, and 1.70% on p4, p6, and p8 instances, respectively.

Table 6 displays the performance of HGA for the two cases, with and without travel-cost truncation. As expected, HGA displays enhanced performance in the former case compared to the latter. Experiments also showed that the best solutions in the initial population were, on average, 20.50% and 17.61% greater than the best solution obtained

Table 5: Comparative performances on Pirkwieser and Raidl instances

Algorithms	p4			p6			p8		
	Runs	GAP	Time (min)	Runs	GAP	Time (min)	Runs	GAP	Time (min)
VNS	90	0.92%	0.81	90	0.44%	0.93	30	0.19%	0.51
VNS-ILP	150	0.72%	0.57	150	0.28%	0.81	30	0.06%	0.58
mVNS	150	0.55%	0.43	210	0.19%	0.80	-	-	-
mVNS-ILP	150	0.008%	0.70	210	0.04%	0.82	-	-	-
EA	30	3.14%	0.48	30	3.45%	0.61	30	4.26%	0.73
CG-EA	30	2.06%	0.60	30	2.62%	0.80	30	3.14%	1.00
CG-ILP	30	3.11%	0.56	30	8.77%	0.80	30	12.64%	1.00
HGA	30	-0.33%	70.54	30	-0.68%	86.39	30	-1.70%	97.51

in the cases without and with truncation, respectively, illustrating again the significant effect of the hybrid meta-heuristic.

Table 6: HGA and the travel-cost truncation issue; Pirkwieser and Raidl instances

Result		p4	p6	p8
Previous Best Average	Avg cost	3293.80	4414.98	5494.13
HGA with truncation	Avg cost	3282.82	4385.30	5402.09
	GAP	-0.33%	-0.68%	-1.70%
HGA without truncation	Avg cost	3300.51	4410.77	5422.00
	GAP	0.19%	-0.07%	-1.34%

We conclude this section with a few remarks on the computational effort of HGA. In its current implementation, this efforts appears indeed high compared to the methods in the literature, the meta-heuristic education procedure compounding the issue. Yet, the issue is not really significant. On the one hand, the proposed methodology may be greatly accelerated by using the classical strategy of performing the crossover and education procedures in parallel (Crainic and Toulouse, 2010). This strategy is particularly adapted to the present case for two main reasons. First, in the generational GA paradigm, all mating and offspring generation and education is performed before a new generation is considered. Decomposing this component of the work clearly then yields independent tasks. Second, each such task is still computation intensive, involving at least a sequence of mating and offspring generation and education (this case corresponds to the finer-grain decomposition assigning each pair of parents to a particular processor). Quasi-linear speedup factors may consequently be expected.

On the other hand, the proposed hybrid GA meta-heuristic is actually using the computation effort to provide higher-quality solutions. To support this claim, we let the two neighborhood-based meta-heuristics search address each instance for a computing

Table 7: Performance comparison for fixed computing effort; Cordeau et al. (2001) instances

Instances	Time (hours)	UTS	VNS	RVNS	VNS-2opt*	RVNS-2opt*	HGA
1a	0.53	-0.47	-1.09	-0.75	-0.48	-0.69	2989.58
2a	0.84	-3.67	-2.21	-2.61	-2.06	-2.28	5107.51
3a	3.73	-3.04	-2.85	-2.79	-2.62	-3.12	7158.77
4a	7.68	-4.66	-2.28	-2.85	-2.61	-2.58	7981.85
5a	13.52	-4.05	-4.07	-4.19	-3.75	-3.88	8584.35
6a	17.76	-5.34	-3.33	-3.56	-3.04	-2.87	10935.60
7a	0.99	-1.46	-2.72	-1.59	-1.92	-1.73	6892.71
8a	6.50	-3.07	-4.58	-4.12	-4.54	-3.99	9730.63
9a	17.09	-3.85	-3.40	-2.88	-3.42	-3.01	13707.30
10a	31.67	-4.14	-4.65	-5.01	-3.94	-3.88	17754.20
1b	0.45	-0.73	-0.35	-0.45	-0.22	-0.18	2284.83
2b	1.13	-2.96	-2.02	-1.48	-1.58	-1.73	4141.15
3b	4.45	-2.12	-2.43	-2.49	-2.73	-2.89	5567.15
4b	14.46	-4.43	-3.37	-2.17	-2.9	-2.24	6471.74
5b	14.43	-1.63	-2.32	-2.02	-2.15	-2.16	6963.11
6b	23.81	-3.06	-1.61	-1.61	-1.65	-1.80	8855.97
7b	1.07	-1.60	-1.75	-1.96	-1.67	-1.42	5509.08
8b	7.40	-3.72	-2.98	-2.62	-3.02	-2.55	7677.68
9b	18.47	-1.81	-2.50	-2.64	-2.27	-2.10	10874.80
10b	31.05	-2.22	-2.73	-2.64	-2.57	-2.77	13851.40
Average	10.85	-2.90	-2.66	-2.52	-2.45	-2.39	8151.97

time equivalent to that of the HGA for the same instance. The results are summed up in Table 7, which displays the gaps to the solution yielded by HGA of the solutions obtained by the meta-heuristics in the same computing time as HGA. The negative values observed for all instances and procedures indicate the computing effort of HGA is well spent, the proposed hybrid meta-heuristic outperforming the other methods present in the literature.

6 Conclusion

We introduced HGA, a population-based hybrid meta-heuristic for the periodic vehicle routing problem with time windows. In this first GA algorithm for the PVRPTW, two neighborhood-based meta-heuristics are used to "educate" the offspring generated by a new crossover operator to enhance the solution quality. This hybridization provides the means to combine the exploration capabilities of population-based methods and the systematic, sometimes aggressive search capabilities of neighborhood-based methods, as

well as their proficiency to explore the infeasible part of the search space to both repair infeasible solutions and tunnel toward, hopefully, improved ones.

Extensive numerical experiments and comparisons with all methods proposed in the literature showed the proposed methodology to yield very high quality solutions, improving those currently published. The proposed HGA actually outperforms the other methods, even when these are given the same computational time.

We plan to follow on these encouraging results and explore the potential of hybrid meta-heuristics to address heavily constrained optimization problems, as well as the parallelization strategies that make these meta-heuristics computationally attractive.

Acknowledgments

While working on this project, the second author was the NSERC Industrial Research Chair in Logistics Management, ESG UQAM, and Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal, and the Department of Economics and Business Administration, Molde University College, Norway. Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grants programs.

References

H. Barbosa and A. Lemonge. An adaptive penalty scheme in genetic algorithms for constrained optimization problems. In *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA, USA*, pages 287–294. Morgan Kaufmann Publishers Inc., 2002. ISBN 1-55860-878-8.

O. Bräysy and M. Gendreau. Vehicle Routing Problem with time windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science*, 39(1):104–118, 2005a.

O. Bräysy and M. Gendreau. Vehicle Routing Problem with time windows, Part II: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005b.

J. F. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for Vehicle Routing Problems with time windows. *Journal of the Operational Research Society*, 52:928–936, 2001.

J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis. The VRP with Time Windows. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 129–154. SIAM, Philadelphia, PA, 2002a.

J.-F. Cordeau, M. Gendreau, G. Laporte, J.-Y. Potvin, and F. Semet. A Guide to Vehicle Routing Heuristics. *Journal of the Operational Research Society*, 53:512–522, 2002b.

J.-F. Cordeau, G. Laporte, M.W.F. Savelsbergh, and D. Vigo. Vehicle Routing. In Barnhart, C. and Laporte, G., editors, *Transportation*, Handbooks in Operations Research and Management Science, pages 367–428. North-Holland, Amsterdam, 2007.

T. G. Crainic and M. Gendreau. Towards an Evolutionary Method - Cooperating Multi-Thread Parallel Tabu Search Hybrid. In S. Voß, S. Martello, C. Roucairol, and Osman, I.H., editors, *Meta-Heuristics 98: Theory & Applications*, pages 331–344. Kluwer Academic Publishers, Norwell, MA, 1999.

T.G. Crainic and M. Toulouse. Parallel Meta-Heuristics. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of Metaheuristics*, pages 497–541. Springer, 2010.

T. A. El-mihoub, A. A. Hopgood, L. Nolle, and A. Battersby. Hybrid genetic algorithms: A review. *Engineering Letters*, 13(2):124–137, 2006.

C. García-Martínez and M. Lozano. Local search based on genetic algorithms. In P. Siarry and Z. Michalewicz, editors, *Advances in Metaheuristics for Hard Optimization*, Natural Computing Series, pages 199–221. Springer, Berlin Heidelberg, 2008.

M. Gendreau and J.-Y. Potvin. Metaheuristics in combinatorial optimization. *Annals of Operations Research*, 140(1):189–213, 2005.

M. Gendreau, G. Laporte, and J.-Y. Potvin. Metaheuristics for the Vehicle Routing Problem. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 129–154. SIAM, Philadelphia, PA, 2002.

B. L. Golden, A. A. Assad, and E. A. Wasil. Routing vehicles in the real world: Applications in the solid waste, beverage, food, dairy, and newspaper industries. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, pages 245–286. SIAM, Philadelphia, PA, 2002.

B. L. Golden, S. Raghavan, and E. A. Wasil. *The vehicle routing problem: latest advances and new challenges*. Springer, 2008.

R. Hartl, G. Hasle, and G. E. Jansens. Special Issue on Rich Vehicle Routing Problems. *Central European Journal of Operations Research*, 13(2), 2006.

H. Ishibuchi and K. Narukawa. Some issues on the implementation of local search in evolutionary multiobjective optimization. In *Proceedings of Genetic and Evolutionary Computation Conference LNCS*, pages 1246–1258, 2004.

J. Knowles and D. Corne. Memetic algorithms for multiobjective optimization: Issues, methods and prospects. In N. Krasnogor, J. E. Smith, and W. E. Hart, editors, *Recent advances in Memetic algorithms*, pages 325–332. Springer, 2000.

G. Laporte and F. Semet. Classical Heuristics for the Vehicle Routing Problem. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 109–128. SIAM, Philadelphia, PA, 2002.

G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet. Classical and Modern Heuristics for the Vehicle Routing Problem. *International Transactions in Operational Research*, 7(4/5):285–300, 2000.

J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44:2245–2269, 1965.

Z. Lü, F. Glover, and J.-K. Hao. A hybrid metaheuristic approach to solving the UBQP problem. *European Journal of Operational Research*, 207(3):1254 – 1262, 2010. ISSN 0377-2217.

P. Merz and K. Katayama. Memetic algorithms for the unconstrained binary quadratic programming problem. *BioSystems*, 2004:99–118, 2004.

I. Or. *Traveling Salesman-type Combinatorial Problems and their relation to the Logistics of Blood Banking*. PhD thesis, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL, 1976.

I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the Vehicle Routing Problems. *Annals of Operations Research*, 41:421–452, 1993.

S. Pirkwieser and G. R. Raidl. Boosting a variable neighborhood search for the periodic vehicle routing problem with time windows by ilp techniques. In *Proceedings of the 8th Metaheuristic International Conference (MIC 2009)*, Hamburg, Germany, 2009a.

S. Pirkwieser and G. R. Raidl. Metaheuristics for the periodic vehicle routing problem with time windows. In *Proceedings of the 3rd International Workshop on model-based metaheuristics (Metaheuristics 2010)*, Vienna, Austria, 2010.

S. Pirkwieser and G. R. Raidl. Multiple variable neighborhood search enriched with ILP techniques for the periodic vehicle routing problem with time windows. In *Proceedings of Hybrid Metaheuristics - Sixth International Workshop*, volume 5818 of LNCS, pages 45–59, Udine, Italy, 2009b. Springer.

S. Pirkwieser and G. R. Raidl. A variable neighborhood search for the periodic Vehicle Routing Problem with time windows. In *Proceedings of the 9th EU/MEeting on Metaheuristics for Logistics and Vehicle Routing*, Troyes, France, 2008.

J.-Y. Potvin and J. M. Rousseau. An exchange heuristic for routing problems with time windows. *Journal of the Operational Reasearch Society*, 46:1433–1446, 1995.

J.-Y. Potvin and J.-M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340, May 1993. URL `http://ideas.repec.org/a/eee/ejores/v66y1993i3p331-340.html`.

M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265, 1987.

E. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin. A tabu search heuristic for the Vehicle Routing Problem with soft windows. *Transportation Science*, 31:170–186, 1997.

P. Toth and D. Vigo. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2002.

A The UTS and RVNS Meta-heuristics

This Annex briefly presents the two meta-heuristics that have been implemented for the offspring education procedure.

A.1 Unified Tabu Search

UTS (Cordeau et al., 2001) uses its own penalty coefficients, α_1, α_2, and α_3, for violations of vehicle capacity, route duration and customer service time window constraints, respectively. The cost function of a solution s then becomes $fit(s) = c(s) + \alpha_1 q(s) + \alpha_2 d(s) + \alpha_3 w(s)$.

An attribute set $B(s) = \{(i,k,l) : \text{customer } i \text{ is visited by vehicle } k \text{ on day } l\}$ is associated to each solution s. Let b_{rl} be a binary constant equal to 1 if and only if day l belongs to pattern r, for each pattern $r \in R$ and every day $l \in \mathcal{T}$ of solution s. The neighborhood $N(s)$ of a solution s is then defined by two transformations to (1) relocate a customer within a day (routing modification), and (2) replace the pattern of a customer (pattern modification):

1. Remove customer i from route k on day l and insert it into another route k'.
2. Replace pattern r currently assigned to customer i with another pattern $r' \in R_i$; Then, for $l = 1, \ldots, t$ do
   - If $b_{rl} = 1$ and $b_{r'l} = 0$, remove customer i from its route of day l;
   - If $b_{rl} = 0$ and $b_{r'l} = 1$, insert customer i into the route of day l minimizing the increase in fitness $f(s)$.

UTS starts from a given offspring s and chooses, at each iteration, the best non-tabu solution in $N(s)$. After each iteration, the values of the parameters α_1, α_2, and α_3 are modified by a factor $1+\delta$; multiplied by the factor is the solution is feasible with respect to the respective constraint, divided, otherwise. We set $\delta = 0.5$, the best value reported by the authors (Cordeau et al., 2001).

To diversify the search, any solution $\overline{s} \in N(s)$ such that $fit(\overline{s}) \geq fit(s)$ is penalized by a factor $p(\overline{s})$ proportional to the addition frequency of its attributes, $p(\overline{s}) = \lambda c(\overline{s})\sqrt{nm}\sum_{(i,k,l)\in B(\overline{s})} \rho_{ikl}$, where ρ_{ikl} is the number of times attribute (i,k,l) has been added to the solution during the search process and $\lambda = 0.015$.

The tabu length was adjusted to the smaller number of iterations performed and set to $\theta = 1.5log_{10}(n)$. The post-optimization heuristic is not implemented in the education

Algorithm 3 UTS(s)

$\alpha_1 = 1$, $\alpha_2 = 1$, $\alpha_3 = 1$
$s_2 \leftarrow s$, $f(s_2) = f(s)$
if s is feasible **then**
  $s_1 \leftarrow s$
  $c(s_1) = c(s)$
else
  $c(s_1) = \infty$
end if
$flag = 0$
for $it = 1, ..., UTS_{it}$ **do**
  Choose a solution $\overline{s} \in N(s)$ that minimizes $fit(\overline{s}) + p(\overline{s})$ and is not tabu or satisfies the aspiration criteria.
  if solution $\overline{s}$ is feasible and $c(\overline{s}) < c(s_1)$ **then**
    $s_1 \leftarrow \overline{s}$
    $c(s_1) = c(\overline{s})$
    $flag = 1$
  else if $f(\overline{s}) < f(s_2)$ **then**
    $s_2 \leftarrow \overline{s}$
    $f(s_2) = f(\overline{s})$
  end if
  Compute $q(\overline{s})$, $d(\overline{s})$ and $w(\overline{s})$ and update α_1, and α_2, α_3 accordingly
  $s \leftarrow \overline{s}$
end for
if $flag$ **then**
  return s_1
else
  return s_2
end if

procedure. The UTS procedure returns the best feasible solution s_1 if it exists, the best infeasible solution s_2, otherwise. The procedure is summarized in Algorithm 3.

A.2 Random Variable Neighborhood Search (RVNS)

RVNS (Pirkwieser and Raidl, 2008) uses three different neighborhood structures. For each of these structures, the authors defined six moves, hence resulting in a total of 18 neighborhoods: (1) randomly changing patterns of customers, (2) moving a random segment of customers of a route to another route on the same day, and (3) exchanging two random segments of customers between two routes on the same day. In the latter two cases, the segments are reversed with a small probability, $p_{rev} = 0.1$. The neighborhoods are summarized in table A8.

Table A8: Neighborhood structures of RVNS

Neighborhood structure	Value of δ	Neighborhood
Change pattern up to	δ = [1,6] times	N_1,...,N_6
Move segment of maximal length	$\delta = [1, 5]$	$N_7, ..., N_{11}$
	bounded by the route size	N_{12}
Exchange segment of maximal length	δ = [1,5]	$N_{13}, ..., N_{17}$
	bounded by corresponding route size	N_{18}

RVNS starts with a random neighborhood ordering and generates a new ordering each time a full VNS iteration is completed. For intensification, RVNS applies 2-opt in a best-improvement fashion. Additionally, each new incumbent solution is subject to a 2-opt*. RVNS accepts solutions which degrade the objective value under the Metropolis criterion, like in simulated annealing. Thus, an inferior solution s' is accepted with probability $e^{-(f(s')-f(s))/T}$, depending on the cost difference to the current solution s relative to the temperature T. A linear cooling scheme is applied, T being decreased every τ iterations by an amount of $(T * \tau)/\tau_{max}$, where τ_{max} denoted the maximal VNS iterations. We set τ=100, the initial temperature value $T_0 = 10$, and $\tau_{max} = [100, 800]$. The detailed description of the implementation is given in Algorithms 4 and 5.

Algorithm 4 Shaking(solution s, int k, double p_{rev})

1: **if** $(1 \leq k \leq 6)$ **then**
2: $s' \leftarrow$ ShakingPattern(s, k) (i.e., neighborhood N_k)
3: **else if** $(7 \leq k \leq 12)$ **then**
4: $s' \leftarrow$ ShakingMoveSegment(s, k-6, p_{rev}) (neighborhood N_{k-6})
5: **else if** $(13 \leq k \leq 18)$ **then**
6: $s' \leftarrow$ ShakingExchangeSegments(s, k-12, p_{rev}) (neighborhood N_{k-12})
7: **end if**
8: return s'

Algorithm 5 RVNS(solution s, int max_{iter})

1: $p_{rev} = 0.1$; $T = 10$ {initial temperature}
2: $s^* \leftarrow s$
3: $numNeighbor = 18$ {number of neighborhoods}
4: $curIter = 1$ {current iteration}
5: **repeat**
6: RandPermutation(ar, $numNeighbor$) {random permunation of neighborhood sequence}
7: $curNeighbor = 1$ {index of current neighborhood}
8: **while** ($curNeighbor \leq numNeighbor$ and $curIter \leq max_{iter}$) **do**
9: $k =$ ar[$curNeighbor$]
10: $s' =$ Shaking(s, k, p_{rev})
11: 2-opt(s')
12: **if** (s' better than s) **then**
13: 2-opt*(s')
14: $s \leftarrow s'$
15: **if** (s better than s^*) **then**
16: $s^* \leftarrow s$
17: **end if**
18: $curNeighbor = 1$
19: RandPermutation(ar, $numNeighbor$)
20: **else**
21: {
22: $prob_{accept} = e^{-(f(s')-f(s))/T}$
23: **if** (accept s' with probability $prob_{accept}$) {accept worse solution} **then**
24: $s \leftarrow s'$
25: $curNeighbor = 1$
26: RandPermutation(ar, $numNeighbor$)
27: **else**
28: $curNeighbor+ = 1$
29: **end if**
30: }
31: **end if**
32: $curIter+ = 1$
33: Update temperature T if needed
34: **end while**
35: **until** ($curIter > max_{iter}$) {exceed maximum number of iterations}
36: return s^*

B Detailed Results

Table A9 compares the average results of HGA with all other existing algorithms on Cordeau et al. (2001) instances. The first four columns indicate, respectively, the instance name, the number of customers n, the number of days T, and the number of vehicles m. The next group of columns compare the average cost of HGA to the average cost of UTS (Cordeau et al., 2001) and the VNS variants of Pirkwieser and Raidl (2009a,b, 2010): VNS, RVNS, VNS with 2-opt* (VNS-2opt*), and RVNS with 2-opt* (RVNS-2opt*). The last two rows provide average measures over all instances: the average gap to the previous BKS, and computation time for each algorithm. Boldface indicates instances for which HGA improves previous BKS. The best solution cost in the initial population for HGA is also included in *Best(Initial)* column, for each problem, to confirm that HGA itself has a significant effect. On average, the best solutions in the initial population are 28.38% larger than the best solutions obtained by HGA. Tables A10 and A11 report the mean and standard deviation for computation time and solution cost, respectively.

Table A9: Best performance comparison among PVRPTW algorithms; Cordeau et al. (2001) instances

Instances				UTS	VNS	RVNS	VNS-2opt*	RVNS-2opt*	HGA		prev BKS	HGA
No	n	T	m	(11 runs)	(30 runs)	(30 runs)	(30 runs)	(30 runs)	(30 runs)		-	(all exp.)
				Best	Average	Average	Average	Average	Average	Best (Initial)		
1a	48	4	3	3007.84	3021.22	3017.18	3014.66	3011.41	3000.56	3864.13	2989.58	2989.58
2a	96	4	6	5328.33	5269.02	5264.30	5262.48	5240.45	5129.45	6501.59	5127.98	**5107.51**
3a	144	4	9	7397.10	7440.17	7395.76	7414.25	7391.74	7193.20	9544.20	7260.37	**7158.77**
4a	192	4	12	8376.95	8242.71	8218.47	8205.00	8211.57	8017.42	10786.50	8089.15	**7981.85**
5a	240	4	15	8967.90	8895.04	8897.47	8870.79	8840.93	8613.09	11762.70	8723.63	**8584.35**
6a	288	4	18	11686.91	11323.50	11313.85	11281.67	11255.80	10971.81	14243.80	11063.00	**10935.60**
7a	72	6	5	6991.54	7026.89	7013.80	7033.89	7009.86	6913.35	8321.46	6917.71	**6892.71**
8a	144	6	10	10045.05	10067.34	10038.22	10055.57	9998.75	9782.38	12726.10	9854.36	**9751.66**
9a	216	6	15	14294.97	14238.18	14159.34	14105.26	14129.01	13738.84	18232.80	13891.03	**13707.30**
10a	288	6	20	18609.72	18484.32	18407.11	18369.76	18362.83	17795.88	23738.30	18023.62	**17754.20**
1b	48	4	3	2318.37	2289.92	2289.54	2291.29	2290.02	2284.83	2644.54	2289.17	**2284.83**
2b	96	4	6	4276.13	4239.47	4213.42	4238.65	4221.31	4167.17	4980.38	4149.96	**4141.15**
3b	144	4	9	5702.07	5721.65	5711.27	5705.25	5704.53	5598.37	6964.89	5608.67	**5567.15**
4b	192	4	12	6789.73	6658.90	6649.30	6656.73	6633.67	6492.64	8232.62	6534.12	**6471.74**
5b	240	4	15	7102.36	7143.51	7119.15	7146.39	7111.50	6984.43	8924.21	6995.87	**6963.11**
6b	288	4	18	9180.15	9019.76	9041.25	8999.79	9024.72	8892.68	11042.10	8895.31	**8855.97**
7b	72	6	5	5606.08	5595.78	5587.57	5576.99	5579.25	5532.97	6635.49	5517.71	**5509.08**
8b	144	6	10	7987.64	7921.17	7880.96	7887.61	7882.10	7711.76	9861.57	7712.40	**7677.68**
9b	216	6	15	11089.91	11194.49	11125.45	11136.13	11085.97	10893.22	13486.60	10944.59	**10874.80**
10b	288	6	20	14207.64	14362.18	14310.99	14330.41	14293.03	13896.22	17036.10	14065.16	**13851.40**
Avg GAP to prev BKS				2.48%	1.97%	1.70%	1.70%	1.50%	-0.44%			
Avg Time (min)				652	1.39	1.52	1.41	1.54	654.09			

We run our algorithm for both cases, with and without travel cost truncation. Table A12 summarizes the HGA results for the Pirkwieser and Raidl (2009a) instances with and without travel cost truncation. The algorithms was run 0 times per instance. Best results, average results, and standard deviations are reported

Table A13 compares the mean and standard deviation for computation times of HGA and VNS and VNS-ILP (Pirkwieser and Raidl, 2009a); mVNS and mVNS/ILP (Pirkwieser and Raidl, 2009b); Evolutionary Algorithm (EA), combination of column gener-

Table A10: Computation time of HGA; Cordeau et al. (2001) instances

Instances	HGA	
	Avg Time (hour)	Std
1a	0.53	0.14
2a	0.84	0.44
3a	3.73	5.05
4a	7.68	2.64
5a	13.52	2.42
6a	17.76	6.16
7a	0.99	0.27
8a	6.5	3.94
9a	17.09	5.47
10a	31.67	5.6
1b	0.45	0.11
2b	1.13	0.3
3b	4.45	5.56
4b	14.46	8.85
5b	14.43	14.64
6b	23.81	8.69
7b	1.07	0.33
8b	7.4	8.65
9b	18.47	7.26
10b	32.05	8.21

ation and evolutionary algorithm (CG-EA), and CG-ILP (Pirkwieser and Raidl, 2010). Using the same format, Tables A14, A15, and A16 compare the mean and standard deviation for solution cost for four, six, and eight-day planning horizon, respectively.

Table A11: Average cost and sd. deviation comparison; Cordeau et al. (2001) instances

Instances	VNS		RVNS		VNS-2opt*		RVNS-2opt*		HGA	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
1a	3021.22	23.90	3017.18	13.59	3014.66	14.48	3011.41	10.53	3000.56	6.9
2a	5269.02	81.33	5264.30	57.77	5262.48	72.61	5240.45	59.99	5129.45	19.26
3a	7440.17	62.66	7395.76	70.35	7414.25	73.30	7391.74	58.16	7193.2	16.20
4a	8242.71	65.99	8218.47	63.12	8205.00	70.97	8211.57	47.10	8017.42	19.24
5a	8895.04	73.42	8897.47	106.92	8870.79	67.32	8840.93	68.52	8613.092	16.52
6a	11323.5	70.66	11313.85	111.59	11281.67	87.08	11255.80	76.96	10971.81	24.05
7a	7026.89	42.88	7013.80	53.16	7033.89	56.08	7009.86	51.16	6913.35	11.20
8a	10067.34	70.00	10038.22	68.20	10055.57	68.83	9998.75	53.58	9782.38	15.40
9a	14238.18	123.64	14159.34	101.71	14105.26	108.49	14129.01	71.15	13738.84	7.75
10a	18484.32	110.50	18407.11	134.36	18369.76	116.09	18362.83	110.74	17795.88	23.92
1b	2289.92	1.73	2289.54	1.13	2291.29	5.73	2290.02	1.66	2284.83	0.00
2b	4239.47	32.79	4213.42	43.10	4238.65	35.48	4221.31	39.67	4167.17	14.81
3b	5721.65	37.38	5711.27	55.32	5705.25	41.98	5704.53	61.81	5598.37	13.20
4b	6658.90	49.35	6649.30	42.44	6656.73	42.52	6633.67	54.37	6492.64	15.74
5b	7143.51	63.16	7119.15	55.09	7146.39	65.42	7111.50	61.57	6984.43	13.93
6b	9019.76	44.14	9041.25	52.28	8999.79	44.26	9024.72	54.87	8892.68	16.65
7b	5595.78	34.44	5587.57	34.42	5576.99	24.15	5579.25	27.29	5532.97	16.27
8b	7921.17	65.48	7880.96	54.95	7887.61	83.12	7882.10	50.62	7711.76	35.00
9b	11194.49	78.77	11125.45	62.99	11136.13	72.75	11085.97	70.57	10893.22	20.77
10b	14362.18	102.60	14310.99	106.21	14330.41	96.90	14293.03	121.86	13896.22	29.11
Avg	8407.76		8382.72		8379.19		8363.92		8180.51	
GAP to BKS	1.97%		1.70%		1.70%		1.50%		-0.44%	

Table A12: HGA results on Pirkwieser and Raidl (2009a) instances with and without travel cost truncation

Instances	Previous Best Average	HGA with truncation				HGA without truncation			
No	Avg	Best (Initial)	Best	Avg	Std	Best (Initial)	Best	Avg	Std
p4r101	4090.09	4590.6	4082.6	4085.94	2.41	4795.34	4142.35	4163.43	5.6
p4r102	3732.34	4011.6	3725.2	3730.56	2.42	4027.44	3739.34	3744.30	3.24
p4r103	3165.72	3675.6	3153.1	3160.81	5.02	3685.05	3165.62	3168.57	2.81
p4r104	2595.44	3010.0	2570.8	2581.53	6.92	3099.08	2582.67	2592.07	8.53
p4r105	3675.09	4092.1	3638.9	3650.45	8.92	4153.81	3664.14	3678.06	11.09
p4c101	2909.39	3203.6	2907.4	2907.49	0.24	3233.38	2913.81	2913.83	0.13
p4c102	2905.16	3694.0	2883.3	2890.98	5.84	3908.80	2888.31	2893.86	7.51
p4c103	2759.78	3528.2	2735.8	2746.23	9.59	4388.63	2742.17	2763.43	8.99
p4c104	2454.69	2804.8	2424.3	2450.91	11.23	2853.89	2446.85	2468.79	14.58
p4c105	2906.69	3326.1	2884.1	2895.33	6.90	3995.58	2893.99	2907.47	9.39
p4rc101	3974.09	4785.3	3955.9	3963.02	5.84	4785.22	3975.39	3977.81	4.31
p4rc102	3764.99	4331.4	3755.8	3761.92	6.27	4381.65	3765.03	3777.56	8.11
p4rc103	3466.99	4232.4	3450.1	3454.60	4.82	4239.32	3472.07	3479.30	5.75
p4rc104	3031.49	3419.0	2996.5	3008.34	12.42	3504.05	3004.59	3019.73	14.52
p4rc105	3970.49	4505.4	3942.6	3954.16	8.05	4511.63	3953.91	3959.46	9.08
p6r101	5385.03	5741.9	5377.5	5379.73	3.98	5835.14	5394.13	5398.65	4.14
p6r102	5237.75	5612.0	5206.4	5215.61	4.62	5648.08	5295.50	5302.56	6.37
p6r103	3991.46	4586.5	3946.9	3968.69	10.04	4792.17	3961.67	3980.51	14.02
p6r104	3370.82	3820.2	3352.9	3362.09	8.41	3930.49	3361.71	3375.91	8.02
p6r105	4328.45	5005.4	4291.0	4302.94	10.93	5036.95	4308.19	4321.17	10.06
p6c101	4049.95	4624.1	3984.3	3995.69	5.99	4694.64	3992.66	4015.34	14.12
p6c102	3861.62	4572.0	3841.7	3853.83	5.25	4614.18	3850.02	3858.76	7.20
p6c103	3574.98	4486.7	3529.6	3555.71	17.63	4523.24	3535.06	3575.18	23.18
p6c104	3278.92	4486.9	3236.5	3248.35	6.94	4994.83	3244.48	3259.09	11.99
p6c105	4104.31	4733.6	4052.1	4060.14	4.31	5310.62	4059.07	4076.46	11.42
p6rc101	5815.8	6262.8	5791.9	5801.08	7.37	6566.15	5799.67	5812.68	10.06
p6rc102	5440.55	6222.9	5352.6	5373.82	17.14	6263.21	5387.76	5402.64	21.45
p6rc103	4344.02	4957.6	4288.1	4298.33	11.76	5084.31	4316.78	4339.45	17.24
p6rc104	4122.25	4764.1	4092.5	4100.14	10.26	4786.14	4109.99	4152.33	22.73
p6rc105	5318.69	6232.8	5253.0	5263.35	10.76	6231.85	5280.32	5290.84	9.56
p8r101	6557.78	6927.4	6472.7	6482.12	8.39	6949.77	6492.00	6506.12	6.96
p8r102	6193.62	6357.3	6102.0	6111.74	11.99	6685.53	6155.07	6166.42	8.51
p8r103	4806.69	5834.8	4698.8	4712.29	8.36	5887.78	4717.23	4750.36	12.49
p8r104	4477.2	5610.0	4394.5	4413.17	6.58	5794.33	4411.70	4431.75	14.42
p8r105	5585.25	6639.3	5473.2	5496.06	13.15	6877.9	5491.93	5506.08	17.51
p8c101	4781.05	5203.6	4695.2	4746.40	14.95	5559.17	4704.07	4741.37	12.56
p8c102	5169.88	6557.1	4933.3	4981.59	23.16	6543.74	4976.19	5003.14	16.36
p8c103	4788.86	6536.4	4672.6	4705.75	9.28	7555.88	4700.94	4714.46	5.13
p8c104	4845.02	5496.9	4638.9	4653.45	8.39	5583.63	4650.28	4669.24	11.43
p8c105	5237.81	6297.9	5143.4	5215.40	19.21	6489.18	5164.94	5230.06	17.76
p8rc101	7035.37	7823.1	6891.7	6924.30	21.76	7897.33	6922.45	6952.49	12.05
p8rc102	5951.36	6326.6	5787.6	5823.83	13.69	6477.04	5805.43	5827.16	10.69
p8rc103	5552.43	6989.7	5448.7	5458.14	10.15	6858.05	5463.97	5486.54	14.16
p8rc104	5071.39	6119.8	5003.5	5046.43	19.31	6101.96	5013.45	5054.10	17.68
p8rc105	6358.24	7769.6	6232.0	6260.65	12.89	7494.72	6265.06	6290.74	10.88
Average	4400.87	5106.87	4339.81	4356.74		5258.46	4359.60	4377.76	

Table A13: Computation time comparisons; Pirkwieser and Raidl (2009a) instances

Instances	VNS	VNS/ILP	mVNS	mVNS/ILP	EA	CG-EA	CG-ILP	HGA	
	Average Time (min)							Avg Time (min)	Std
p4r101	0.74	2.70	1.15	1.31	0.47	0.52	0.47	51.45	0.08
p4r102	0.77	2.90	1.20	1.36	0.45	0.52	0.50	73.62	0.17
p4r103	0.40	2.98	1.23	1.46	0.44	0.56	0.55	71.83	0.19
p4r104	0.90	3.40	1.39	2.12	0.47	0.67	0.67	75.98	0.21
p4r105	0.79	3.84	1.22	2.08	0.44	0.49	0.48	65.65	0.12
p4c101	0.73	0.38	1.12	1.18	0.51	0.58	0.19	68.52	0.12
p4c102	0.81	2.90	1.25	1.50	0.50	0.74	0.68	71.34	0.17
p4c103	0.91	0.50	1.41	1.98	0.51	0.73	0.72	88.64	0.13
p4c104	0.87	3.31	1.35	1.98	0.48	0.77	0.77	93.87	0.14
p4c105	0.80	2.78	1.24	1.56	0.49	0.63	0.62	69.34	0.13
p4rc101	0.84	0.43	1.30	1.78	0.44	0.52	0.51	60.86	0.13
p4rc102	0.84	0.43	1.29	1.72	0.47	0.53	0.52	66.11	0.33
p4rc103	0.91	0.48	1.40	1.57	0.47	0.58	0.57	64.92	0.14
p4rc104	0.92	3.46	1.46	2.35	0.49	0.65	0.64	70.22	0.14
p4rc105	0.81	0.43	1.26	1.67	0.47	0.51	0.50	65.88	0.08
Average	0.81	2.06	1.29	1.71	0.48	0.60	0.56	70.54	
p6r101	0.85	3.25	3.08	3.91	0.66	0.69	0.68	77.52	0.14
p6r102	0.89	4.43	3.21	5.70	0.61	0.71	0.71	75.02	0.06
p6r103	0.96	4.05	3.42	5.45	0.61	0.80	0.80	89.49	0.07
p6r104	0.98	0.68	3.50	5.67	0.61	0.88	0.87	95.14	0.07
p6r105	0.47	3.80	3.37	5.51	0.59	0.73	0.72	77.45	0.05
p6c101	0.65	3.47	3.57	5.32	0.61	0.83	0.82	75.87	0.06
p6c102	0.98	3.63	3.55	5.50	0.64	0.92	0.91	88.61	0.06
p6c103	1.11	4.16	4.03	6.36	0.62	0.92	0.91	104.54	0.06
p6c104	1.07	4.62	3.89	6.16	0.60	0.90	0.90	105.61	0.05
p6c105	0.98	3.52	3.54	5.56	0.61	0.88	0.88	85.56	0.18
p6rc101	0.90	0.51	3.27	5.73	0.57	0.65	0.65	75.95	0.20
p6rc102	0.93	3.50	3.38	5.70	0.60	0.70	0.69	83.86	0.13
p6rc103	1.06	3.91	3.82	6.08	0.59	0.84	0.83	83.95	0.06
p6rc104	1.05	4.35	3.71	6.55	0.60	0.84	0.83	97.06	0.15
p6rc105	0.96	3.50	3.44	5.79	0.59	0.70	0.69	80.23	0.13
Average	0.93	3.42	3.52	5.67	0.61	0.80	0.80	86.39	
p8r101	0.43	0.49			0.73	0.89	0.88	88.19	0.14
p8r102	0.47	0.51			0.74	1.01	1.00	90.47	0.34
p8r103	0.53	0.60			0.73	1.02	1.01	102.05	0.10
p8r104	0.54	0.71			0.73	1.04	1.03	104.63	0.09
p8r105	0.47	0.59			0.71	0.97	0.96	90.61	0.11
p8c101	0.50	0.54			0.73	1.03	1.02	99.21	0.40
p8c102	0.63	0.65			0.74	1.08	1.07	102.43	0.10
p8c103	0.59	0.67			0.73	1.04	1.03	122.56	0.08
p8c104	0.44	0.62			0.69	1.00	0.99	92.66	0.13
p8c105	0.51	0.55			0.76	1.02	1.02	95.24	0.07
p8rc101	0.47	0.52			0.72	0.91	0.91	82.47	0.08
p8rc102	0.49	0.53			0.69	1.00	1.00	96.19	0.42
p8rc103	0.55	0.57			0.74	1.02	1.02	95.06	0.07
p8rc104	0.53	0.63			0.72	1.00	0.99	106	0.07
p8rc105	0.47	0.54			0.71	0.97	0.96	94.85	0.11
Average	0.51	0.58			0.73	1.00	1.00	97.51	

Table A14: Average cost and st. deviation for 4-day planning horizon; Pirkwieser and Raidl (2009a) instances

Instances	VNS		VNS/ILP		mVNS		mVNS/ILP	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
p4r101	4134.47	18.81	4112.11	18.7	4118.6	12.07	4090.09	5.29
p4r102	3756.77	15.45	3742.11	9.45	3741.79	5.01	3732.34	1.94
p4r103	3191.59	13.56	3182.65	11.98	3184.83	9.37	3165.72	6.95
p4r104	2604.74	12.94	2599.43	13.06	2602.15	11.10	2595.44	8.20
p4r105	3692.96	14.74	3675.09	16.22	3687.94	8.92	3679.66	11.93
p4c101	2910.53	0.37	2910.13	0.38	2909.91	0.72	2909.39	0.76
p4c102	2960.70	37.37	2951.60	35.15	2921.32	20.02	2905.16	14.38
p4c103	2793.80	30.37	2801.59	37.26	2777.30	20.18	2759.78	13.82
p4c104	2476.27	19.95	2473.77	19.26	2465.70	11.65	2454.69	12.32
p4c105	2973.57	42.71	2990.91	59.02	2942.26	29.05	2906.69	15.55
p4rc101	4001.34	14.66	3980.51	12.04	3988.69	11.61	3974.09	6.40
p4rc102	3798.00	14.47	3795.87	15.96	3801.01	14.74	3764.99	6.76
p4rc103	3494.06	23.22	3485.33	26.11	3494.62	18.55	3466.99	12.01
p4rc104	3058.48	18.83	3045.37	18.22	3042.86	11.52	3031.49	17.15
p4rc105	4001.89	20.06	3988.68	24.21	3995.56	13.49	3970.49	5.67
Average	3323.28		3315.68		3311.64		3293.80	
GAP	0.90%		0.67%		0.55%		0.009%	

Instances	EA		CG-EA		CG-ILP		HGA	
p4r101	4199.14	45.00	4162.54	35.92	4119.54	15.56	4085.94	2.41
p4r102	3784.31	33.14	3780.5	24.52	3777.63	29.37	3730.56	2.42
p4r103	3248.05	31.50	3217.31	24.84	3258.92	44.13	3160.81	5.02
p4r104	2691.66	36.48	2673.09	29.85	2780.10	64.70	2581.53	6.92
p4r105	3777.90	34.10	3745.00	28.82	3801.99	33.80	3650.45	8.92
p4c101	2918.47	12.02	2921.08	22.11	2917.91	6.32	2907.49	0.24
p4c102	3032.23	49.41	2963.28	42.32	2925.01	49.33	2890.98	5.84
p4c103	2874.99	54.80	2825.01	42.33	2973.94	89.65	2746.23	9.59
p4c104	2542.46	24.39	2518.90	32.90	2479.80	24.76	2450.91	11.23
p4c105	3072.79	86.04	2977.45	54.82	2991.24	77.13	2895.33	6.90
p4rc101	4081.77	44.36	4047.87	32.44	4087.80	43.80	3963.02	5.84
p4rc102	3904.33	56.09	3869.21	53.28	3870.02	35.30	3761.92	6.27
p4rc103	3596.08	45.32	3549.13	33.54	3670.73	60.17	3454.60	4.82
p4rc104	3142.79	37.99	3114.51	36.46	3185.14	42.46	3008.34	12.42
p4rc105	4052.78	42.09	4040.32	22.11	4047.39	40.29	3954.16	8.05
Average	3394.65		3360.35		3392.48		3282.82	
GAP	3.07%		2.03%		3.01%		-0.33%	

Table A15: Average cost and st. deviation for 6-day planning horizon; Pirkwieser and Raidl (2009a) instances

Instances	VNS		VNS/ILP		mVNS		mVNS/ILP	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
p6r101	5417.67	20.27	5395.27	11.92	5400.68	7.1	5385.03	3.33
p6r102	5261.35	17.65	5237.75	9.99	5250.62	13.72	5244	13.18
p6r103	4014.16	21.37	4001.86	20.65	4004.61	15.35	3991.46	12.83
p6r104	3380.17	13.35	3371.90	16.1	3376.35	12.41	3370.82	11.53
p6r105	4355.02	27.43	4334.60	26.79	4340.76	15.94	4328.45	16.34
p6c101	4072.01	38.11	4070.44	40.33	4049.95	24.47	4050.81	26.17
p6c102	3876.88	17.27	3883.75	16.49	3861.62	10.81	3861.86	8.69
p6c103	3583.02	33.04	3594.89	43.15	3579.45	19.34	3574.98	24.81
p6c104	3291.93	16.76	3280.58	17.62	3278.92	18.69	3280.17	14.42
p6c105	4139.66	53.95	4158.06	51.36	4110.27	26.88	4104.31	23.19
p6rc101	5833.84	26.41	5826.98	19.20	5830.37	13.51	5815.80	16.78
p6rc102	5473.67	24.34	5463.50	26.65	5449.68	26.63	5440.55	27.78
p6rc103	4360.17	21.22	4344.02	19.31	4365.63	19.79	4351.50	18.34
p6rc104	4127.46	23.06	4122.25	21.93	4130.70	17.69	4132.90	21.11
p6rc105	5330.60	19.67	5319.48	27.39	5326.71	17.63	5318.69	16.99
Average	4434.51		4427.02		4423.76		4416.76	
GAP	0.44%		0.27%		0.20%		0.04%	

Instances	EA		CG-EA		CG-ILP		HGA	
p6r101	5471.23	33.24	5453.07	32.6	5505.08	42.9	5379.73	3.98
p6r102	5315.03	31.43	5318.87	25.76	5445.35	40.39	5215.61	4.62
p6r103	4149.57	41.18	4120.37	34.46	4254.40	67.58	3968.69	10.04
p6r104	3465.46	28.20	3441.55	22.04	3665.01	62.43	3362.09	8.41
p6r105	4514.95	46.59	4457.93	48.46	4647.59	112.43	4302.94	10.93
p6c101	4192.24	77.09	4162.92	68.33	4592.38	194.23	3995.69	5.99
p6c102	3960.89	56.36	3950.54	65.92	4414.48	208.85	3853.83	5.25
p6c103	3788.68	63.95	3719.95	82.20	4191.75	170.11	3555.71	17.63
p6c104	3450.31	54.19	3422.22	56.05	3766.94	92.55	3248.35	6.94
p6c105	4285.79	84.27	4181.5	56.15	4551.39	187.19	4060.14	4.31
p6rc101	5932.49	46.38	5909.63	40.87	6128.02	67.00	5801.08	7.37
p6rc102	5577.50	54.72	5553.47	52.54	5756.83	83.19	5373.82	17.14
p6rc103	4521.57	50.34	4476.44	45.03	4699.27	67.30	4298.33	11.76
p6rc104	4306.30	52.83	4267.67	41.26	4436.69	85.22	4100.14	10.26
p6rc105	5467.39	58.06	5450.10	44.81	5582.21	70.66	5263.35	10.76
Average	4559.96		4525.74		4775.83		4385.30	
GAP	3.28%		2.51%		8.17%		-0.68%	

Table A16: Average cost and st. deviation for 8-day planning horizon; Pirkwieser and Raidl (2009a) instances

Instances	VNS		VNS/ILP		EA		CG-EA		CG-ILP		HGA	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
p8r101	6574.92	36.55	6557.78	37.26	6711.50	46.38	6696.89	75.06	6820.33	68.96	6482.12	8.39
p8r102	6193.62	99.24	6205.41	45.66	6300.33	49.19	6313.65	70.20	6508.59	125.34	6111.74	11.99
p8r103	4809.73	26.53	4806.69	34.24	4999.87	67.08	4930.83	43.14	5250.39	114.26	4712.29	8.36
p8r104	4495.36	28.29	4477.20	28.89	4667.39	52.74	4598.77	64.23	5181.33	117.11	4413.17	6.58
p8r105	5600.71	39.34	5585.25	37.59	5817.43	69.13	5744.09	53.36	6013.38	105.65	5496.06	13.15
p8c101	4781.05	41.91	4786.88	39.11	4991.15	119.77	4900.84	75.41	5185.67	131.66	4746.40	14.95
p8c102	5169.88	71.00	5188.8	76.68	5410.25	121.16	5308.69	114.27	6442.4	0.00	4981.59	23.16
p8c103	4794.7	50.18	4788.86	36.49	5029.64	105.63	4965.95	69.92	6428.76	272.02	4705.75	9.28
p8c104	4845.02	71.06	4853.57	65.88	5234.18	79.30	5202.05	91.42	5592.48	254.28	4653.45	8.39
p8c105	5261.79	58.32	5237.81	42.58	5434.17	91.13	5384.95	95.36	6293.36	254.71	5215.40	19.21
p8rc101	7075.80	75.20	7035.37	64.98	7225.04	98.40	7134.84	80.09	7432.93	152.95	6924.30	21.76
p8rc102	5951.36	47.72	5954.42	67.27	6249.95	102.21	6163.02	76.37	6392.33	149.02	5823.83	13.69
p8rc103	5560.42	34.89	5552.43	33.69	5847.79	95.54	5778.00	73.80	6126.24	128.82	5458.14	10.15
p8rc104	5080.84	35.82	5071.39	38.96	5301.08	52.53	5277.23	46.59	5873.2	144.27	5046.43	19.31
p8rc105	6383.60	43.49	6358.24	30.28	6606.78	79.69	6530.36	62.94	6727.78	120.43	6260.65	12.89
Average	5505.25		5497.34		5721.77		5662.01		6151.28		5402.09	
GAP	0.20%		0.06%		4.14%		3.06%		11.96%		-1.70%	