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 Generational Genetic Algorithm for the Periodic Vehicle Routing Problem with Time Windows

Phuong Nguyen Khanh
Teodor Gabriel Crainic
Michel Toulouse

November 2012

CIRRELT-2012-66

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 Generational Genetic Algorithm for the Periodic Vehicle Routing Problem with Time Windows

Phuong Nguyen Khanh[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, P.O. Box 6128, Station Centre-Ville, Montréal, Canada H3C 3J7

[3] Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

[4] Department of Computer Science, Oklahoma State University, 700 North Greenwood Avenue, North Hall 328, Tulsa, OK 74106-0700, USA

Abstract. We propose the first generational genetic algorithm for the periodic vehicle routing problem with time windows. The algorithm takes the form of a hybrid meta-heuristic in which a set of neighborhood-based meta-heuristics cooperate with the population evolution mechanism to enhance the quality of the solutions obtained by a powerful crossover operator. 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 is highly competitive, providing new best solutions for several large instances.

Keywords. Periodic vehicle routing problem, time windows, hybrid generational genetic algorithm, meta-heuristics, tabu search, variable neighborhood search.

Acknowledgements. While working on this project, T.G. Crainic was the Natural Sciences and Engineering Research Council of Canada (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 Research Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grant programs. We also gratefully acknowledge the support of Fonds de recherche du Québec - Nature et technologies (FRQNT) through their infrastructure grants and of Calcul Québec and Compute Canada through access to their high-performance computing infrastructure.

---

* Corresponding author: TeodorGabriel.Crainic@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2012

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 generational 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 individuals and enhance their fitness as well as to promote diversity 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; Vidal et al., 2013). Local search may reduce the diversity of the population (Merz and Katayama, 2004), however, and therefore limit the exploration 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. This more extensive exploration of the offspring neighborhoods by meta-heuristics yields two additional benefits for the search: the first is the restoration of the feasibility of a very large percentage of the infeasible offspring produced by GA crossover operators; the second is the discovery of solutions with new customer-day assignments that once added to the current population increase substantially the GA capacity to uncover new combinations of customer-day assignments through crossover operations.

We tested the algorithm we propose on all previously published benchmark instances and compare our results to all currently published results. The proposed *Hybrid Generational Genetic Algorithm* (*HGGA*) produces 9 new best-known solutions and finds 5 best-known solutions on the set of 20 PVRPTW instances of Cordeau et al. (2004), improving the solution quality by 0.05% on average in terms of best solution cost. We also improve all the 45 instances of Pirkwieser and Raidl (2009a), improving the average solution cost by 0.27%. 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 HGGA 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 its visit days over 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 partly-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:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 $w^t_{ik}+s_i+c_{ij}$. 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, 2004) 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 ILP 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.

Cordeau and Maischberger (2012) proposed a parallel iterated tabu search heuristic which belongs to pC/KS/MPDS category introduced by Crainic et al. (2005). This heuristic embeds a tabu search within the framework of iterated local search as the improvement phase, yielding an 'iterated tabu search'. Each 'iterated tabu search' thread starts from different solution using different parameters, and these threads communicate through a central memory for exchanging their working solutions. The authors used up to 64 threads in the parallel variant.

Vidal et al. (2013) proposed a hybrid genetic search which combines the exploration capabilities of genetic algorithms with the intensification via local search-based improvement procedures and diversity management mechanisms. To further improve solution quality, they also applied a decomposition phase in which pattern assigned to each customer is fixed, then resulting VRPTW subproblem for each period is solved separately by their hybrid genetic algorithm.

Overall, the current best published results on the 20 instances of Cordeau et al. (2004) are reported by Vidal et al. (2013). Only the latter authors and Pirkwieser and Raidl published results for the set of smaller instances that introduced in Pirkwieser and Raidl (2009b). Pirkwieser and Raidl (2009a,b, 2010) reported only average solution costs and no best solution costs.

4 The Proposed Hybrid Meta-heuristic

This section is dedicated to introducing the Hybrid Generational 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 HGGA, 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.

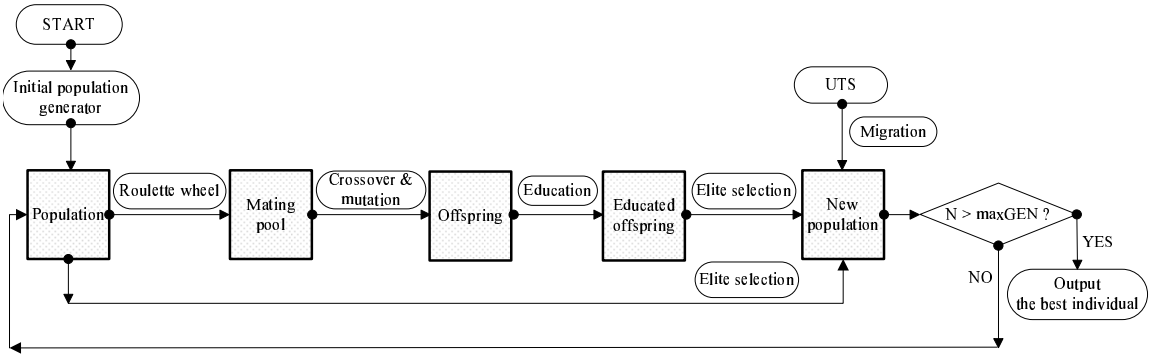

Figure 1: The Hybrid Generational Genetic Algorithm Structure

4.1 Individual representation

An individual for the HGGA we propose corresponds to a feasible or infeasible solution to the PVRPTW specifying 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 with 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).

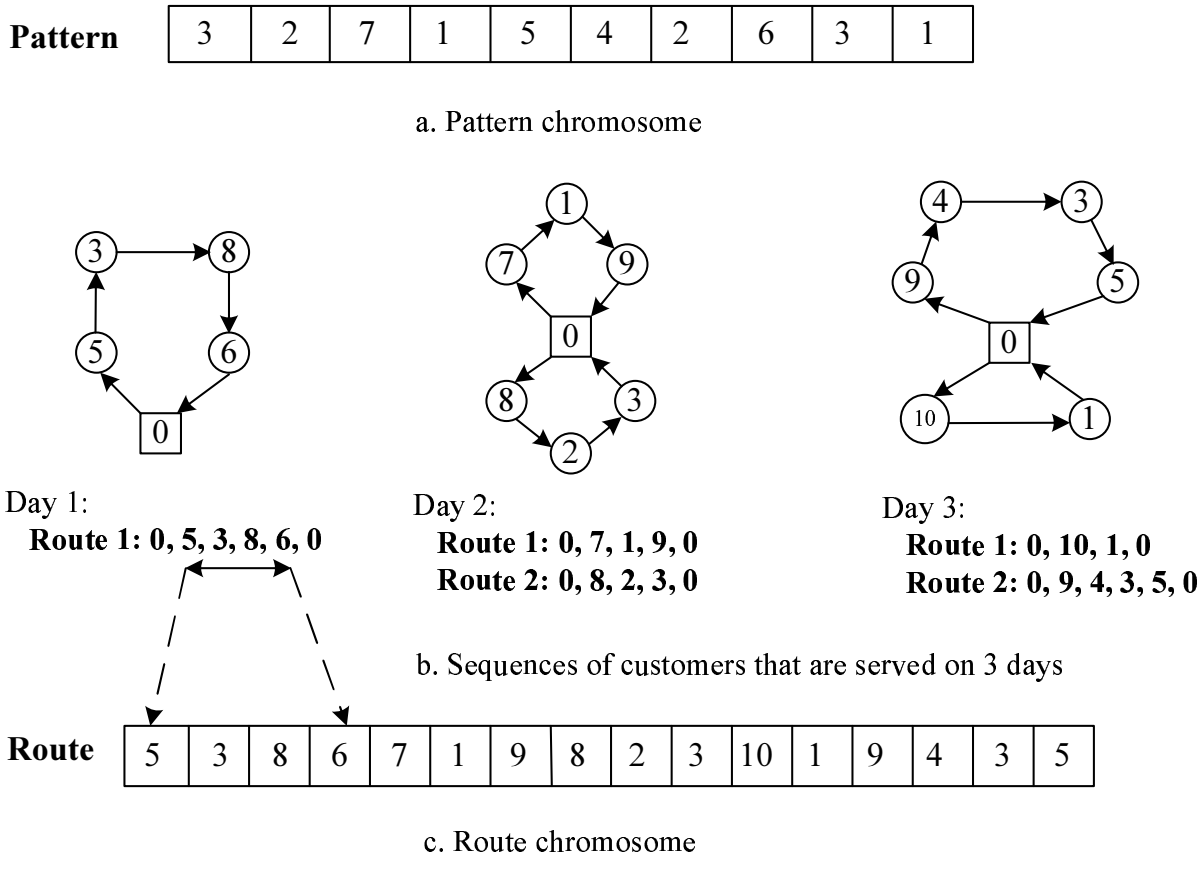

Figure 2: Representation of an individual.

For each day in the planning horizon, a group of routes services customers on that

day. The **Route** chromosome corresponds to the concatenation of this set of vectors, each representing the ordered sequence of customers for one route of a day. Figure 2b 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., 2004). 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. (2004) 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 displaying 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 HGGA, 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 to restore the feasibility as much 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 that 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 generational process. 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. (2004) 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 HGGA, 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 enhances 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. 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 details, 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), i is removed from its current routes and the cheapest fitness insertion is performed to insert i into routes of corresponding days of the new pattern r'. 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 of 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 HGGA 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 which have different pattern assignments, while also welcoming individuals with unused pattern assignments generated by the education procedure.

HGGA evolves a population in which each individual has a different pattern assignment. Recall (Section 4.4) that the mating pool is populated with $nPop$ individuals selected from the current population (the same individual may appear more than once according to its fitness) and that $nPop$ offspring are created through crossover, mutation and education. The next generation is then composed of these $nPop$ new individuals, the best $nKeep$ individuals in the current population ($nKeep < nPop$), plus feasible solutions found by the education operator that have a pattern assignment not yet being seen in the population. Finally, for those individuals that share a same pattern assignment, only the best one is kept in the next generation. This elitism not only preserves the best individuals from one generation to the next, but also guarantees that all new features created during the previous generation are kept, thus hopefully contributing to uncover new points of the search space in the next generation. Algorithm 2 summarizes the hybrid meta-heuristic we propose for the PVRPTW.

Algorithm 2 Hybrid Generational Genetic Algorithm

```
1: Randomly generate an initial population of nPop individuals
2: repeat
3:    Evaluate the fitness for each individual in the population
4:    Create the mating pool of 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 each offspring
10:      Delete both selected parents from the mating pool
11:   end while
12:   Generation replacement
13: until maximum number of generations > maxGEN
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 the 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 HGGA through comparisons with currently published results (Section 5.2).

HGGA is implemented in C++. Experiments were run on an Intel Xeon 2.8 GHz, 16 GB RAM. Two sets of instances were used throughout the experiments. The first, introduced by Cordeau et al. (2004), 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 HGGA is always the result of a number of decisions on the structure, components, and parameter values of the method. Two main design com-

ponents are studied through their impact on the behavior of the proposed HGGA: 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 in order to determine an appropriate hybridization scheme.

We implemented the UTS and RVNS meta-heuristics following the descriptions given in Cordeau et al. (2004) 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, *HGGA-UTS* and *HGGA-VNS*, embed one method only, UTS or RVNS being called at each iteration, respectively. The third, *HGGA-UTS/VNS*, combines the two, UTS and RVNS being used alternately at every generation. The fourth, *HGGA*, enhances the HGGA-UTS/VNS structure by performing the pattern improvement procedure following the RVNS execution.

To level the comparison field, the number of iterations of UTS (200) and RVNS (1200) was set to yield comparable computing times. Similarly, the number of iterations of RVNS was reduced for HGGA such that the total computing effort for RVNS and pattern improvement be approximately the same as that of RVNS in HGGA-UTS/VNS. Experiments were conducted on the Cordeau et al. (2004) instances.

Table 1: Aggregated performance comparison between education schemes on Cordeau et al. (2004) instances

	HGGA-UTS	*HGGA-VNS*	*HGGA-UTS/VNS*	*HGGA*
Time (hours)	5.22	5.77	5.20	5.18
GAP	+1.07%	+1.48%	+0.34%	-0.05%

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 HGGA-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, HGGA-UTS/VNS outperforming HGGA-UTS and HGGA-VNS (even when the education effort was reduced, e.g., running 100 and 1000 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 HGGA 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 solution quality. 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 all $nPop$ offspring.
- Strategy 2. The population of generation $i + 1$ is made up of the $nKeep$ best solutions of generation i, all $nPop$ offspring, plus feasible solutions discovered during the education procedure displaying pattern assignments not yet being used by $nKeep$ best solutions and $nPop$ offspring.

Note that in both strategies, each individual in the population has a different pattern assignment as only the best one among those using the same pattern assignment is kept.

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

Strategy	$nKeep$	Binary Tournament		Roulette Wheel	
		$\%nKeep$	GAP	$\%nKeep$	GAP
1	40	6% - 8%	2.54%	26% - 37%	0.38%
	70	13 % - 16%	2.09%	48% - 53%	0.05%
2	40	4% - 7%	2.2%	24% - 34%	0.07%
	70	10% - 15%	2.06%	37% - 50%	0%

The experiment was run for 2000 generations with $nPop = 100$ and different values of $nKeep$. The experiment was performed using a small set of instances {1a,11a,3a,3b,9a,9b} from Cordeau et al. (2004) 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. We report the average results over 10 runs in Table 2. The $\%nKeep$ column 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 this column representing the lower and upper bounds on the number of solutions from $nKeep$, respectively. Over all, roulette wheel selection and the second replacement strategy yield better solutions than the others. Thus, we report in the GAP column the gaps for the average values obtained by each selection and replacement strategy with respect to that of the roulette wheel selection and the second replacement strategy.

We observe a significant difference in Table 2 between the two selection operators in terms of the impact they have on the performance of the HGGA. Further, for both selection operators, Strategy 2 yields better solutions 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 HGGA 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 HGGA. The calibration was performed using the same small set of instances used in Section 5.1.2.

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 or equal to 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., 2004), and the RVNS procedure lowers the temperature every $\tau = 10$ iterations (smaller than the 100 in Pirkwieser and Raidl, 2008).

The last calibration step examined the stopping criterion, which is based on the total number of HGGA iterations. The best solution and corresponding computation time were measured for each instance from generation 250 to generation 5000, by steps of 250. The results indicate that, for small instances, the best solutions can not be improved after 500 generations, while the average improvement of the best solution between generations

Table 3: Calibration of main HGGA parameters

Parameters	Interval explored	Final values	
		Small instances	Large instances
Population size ($nPop$)	[50, 400]	70	100
Elite set cardinality ($nKeep$)	[25, 300]	40	70
Number of UTS iterations	[50, 150]	50	100
Number of RVNS iterations	[100, 800]	400	800
Maximum number of generations ($maxGEN$)	[250, 5000]	500	1500, 2000

500 and 750 is less than 0.001%. Consequently, the number of generations was set to 500 for small instances. For large instances, the number of generations was set to 1500 and 2000 for instances with less and more than or equal to 200 customers, respectively.

5.2 Numerical Results

The performance of the proposed HGGA is evaluated through comparison with published results on the instances provided by Cordeau et al. (2004) and Pirkwieser and Raidl (2009a). For concision sake, only aggregated results are provided in this section. Details may be found in Annex B.

Table 4: Best performance comparison among PVRPTW algorithms; Cordeau et al. (2004) instances

Instances				Prev BKS	CLM04	PR08	CM12	VCGP13	HGGA			
No	n	m	T	Cost	Best 10	Best X	Best 30	Best 10	Avg 10	Best 10 Cost	GAP	T (hours)
1a	48	3	4	2909.02	0.07	0	0	0	0	2909.02	0	0.03
2a	96	6	4	5026.57	0.57	0.19	0	0	0.04	5026.57	0	0.27
3a	144	9	4	7023.9	2.93	1.63	0.54	0.38	0.48	7024.96	0.02	2.15
4a	192	12	4	7755.77	2.54	1.63	0.66	0.47	0.25	**7738.25**	-0.23	3.66
5a	240	15	4	8311.17	3.39	2.18	0.58	0.37	0.56	8319.89	0.11	5.46
6a	288	18	4	10473.24	4.34	2.3	0.66	0.04	0.76	10478.80	0.05	7.81
7a	72	5	6	6782.68	0.62	0.07	0	0.01	0.1	6782.68	0	0.17
8a	144	10	6	9574.8	1.81	1.53	0.3	0.19	0.32	9574.86	0.0006	4.21
9a	216	15	6	13201.06	3.13	1.99	0.75	0.35	0.64	**13188.40**	-0.1	8.71
10a	288	20	6	16920.96	4.81	4.31	2.01	0.47	0.11	**16906.80**	-0.08	15.05
1b	48	3	4	2277.44	0.73	0	0	0	0	2277.44	0	0.03
2b	96	6	4	4121.5	3.3	0.39	0.08	0.01	0.39	4122.03	0.01	0.43
3b	144	9	4	5489.33	2.9	1.57	0.01	0.59	0.41	5489.77	0.01	2.34
4b	192	12	4	6347.77	3.89	2.03	0.56	0.08	0.73	**6345.65**	-0.03	4.14
5b	240	15	4	6777.54	4.09	2.84	0.34	0.19	0.6	**6775.39**	-0.03	7.43
6b	288	18	4	8582.72	4.03	2.76	0.89	0.14	0.19	**8580.06**	-0.03	8.41
7b	72	5	6	5481.61	0.43	0.42	0	0	0.26	5481.61	0	0.26
8b	144	10	6	7599.01	3.64	1.71	0.75	0.28	0.31	**7595.75**	-0.04	4.78
9b	216	15	6	10532.51	3.39	3.36	0.45	0.54	0.42	**10508.80**	-0.23	10.17
10b	288	20	6	13406.89	4.28	4	0.63	0.27	-0.1	**13363.00**	-0.33	18
Avg Gap to BKS (%)					2.74	1.75	0.46	0.22	0.32		-0.05	
Avg Time (min)					160	N/A	11.32	32.74		310.46		
Processor					P4-2G	Opt-2.2G	Xe-2.93G	Xe-2.93G		Xe-2.8G		

Table 4 compares the gaps for the HGGA relative to previous BKS and those of the other algorithms on the Cordeau et al. (2004) instances:

- CLM04: UTS of Cordeau et al. (2004)
- PR08: RVNS of Pirkwieser and Raidl (2008)
- CM12: Parallel iterated tabu search of Cordeau and Maischberger (2012)
- VCGP13: Hybrid genetic algorithm of Vidal et al. (2013)

The first four columns indicate the instance name, the number of customers n, the number of vehicles m, and the number of days T, respectively. The fifth column displays the previous BKS reported in Vidal et al. (2013). All gaps are reported as % value w.r.t. the previous BKS. The next four columns provide, respectively, the gaps for the values reported by Cordeau et al. (2004), Pirkwieser and Raidl (2008), Cordeau and Maischberger (2012), Vidal et al. (2013). The last four columns report the gaps for the average values obtained by our algorithm over 10 runs, the cost of the best solution found during these runs, the corresponding gap and the computation time. The last three rows provide average measures over all instances: the average gap to the previous BKS, the computation time in minutes, and the type of processor used by each algorithm. Boldface indicates instances for which HGGA improves previous BKS.

HGGA produces high quality solutions, with an average error gap of -0.05% to the previous BKS, compared to more than 0.22% for the other algorithms. HGGA produces 9 new best-known solutions and finds 5 best-known solutions over 20 instances. Experiments also showed that the best solutions in the initial population were 27.59% greater than the best solutions obtained by HGGA on average, illustrating the significant effet of hybrid strategy.

The next round of experimentations focused on the instances proposed by Pirkwieser and Raidl, and used in Pirkwieser and Raidl (2009a,b, 2010) and Vidal et al. (2013). 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 HGGA with and without truncation. One also notes that best solutions were only reported by Vidal et al. (2013), and just for the truncation case. Pirkwieser and Raidl (2009a,b, 2010) only reported average costs and standard deviations computed over 30 runs for each instance, while average costs over 10 runs for each instance set p4, p6, p8 are available in Vidal et al. (2013). We therefore selected the best average costs of each instance set over all previous algorithms, noted by *Previous Best Average*, and compared the results of HGGA to this measure. Notice that all *Previous Best Average* are produced by Vidal et al. (2013). Detailed results may be found in Annex B.

Table 5 sums up the comparison of average results from 10 runs of HGGA, with 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); and Hybrid Genetic Algorithm (VCGP13) of (Vidal et al., 2013). We report the averages of the percentages of deviation from the Previous Best Average (Column *GAP*) and computation time (Column *Time*) for each algorithm. 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. Pirkwieser and Raidl (2009a) used an Opteron 2.2 GHz, Pirkwieser and Raidl (2009b, 2010) used a Core2 Quad 2.83 GHz, and Vidal et al. (2013) used an Intel Xeon 2.93 GHz. One observes that HGGA performs again very well, obtaining the smallest average gap for all 45 instances, and improving the average cost by 0.09%, 0.25%, and 0.48% on p4, p6, and p8 instances, respectively.

Table 5: Comparative performances on Pirkwieser and Raidl (2009a) instances

Algorithms	Processor	p4			p6			p8		
		Runs	GAP	Time (min)	Runs	GAP	Time (min)	Runs	GAP	Time (min)
VNS	Opt-2.2G	90	1.32%	0.81	90	1.21%	0.93	30	2.18%	0.51
VNS-ILP		150	1.12%	0.57	150	1.05%	0.81	30	2.05%	0.58
mVNS		150	0.95%	0.43	210	0.96%	0.80	-	-	-
mVNS-ILP		150	0.41%	0.70	210	0.81%	0.82	-	-	-
EA	Qd-2.83G	30	3.55%	0.48	30	4.25%	0.61	30	6.33%	0.73
CG-EA		30	2.47%	0.60	30	3.41%	0.80	30	5.19%	1.00
CG-ILP		30	3.52%	0.56	30	9.61%	0.80	30	14.88%	1.00
VCGP13	Xe-2.93G	10	0%	3.21	10	0%	4.44	10	0%	5.06
HGGA	Xe-2.8G	10	-0.09%	33.88	10	-0.25%	43.20	10	-0.48%	48.75

Table 6 displays the performance of HGGA for the two cases, with and without travel-cost truncation. As expected, HGGA 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, 23.17% and 21.86% greater than the best solution obtained in the cases without and with truncation, respectively, illustrating again the significant effect of the hybrid meta-heuristic.

We conclude this section with a few remarks on the computational effort of HGGA. In its current implementation, this effort 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

Table 6: HGGA and the travel-cost truncation issue; Pirkwieser and Raidl (2009a) instances

Result		p4	p6	p8
Previous Best Average	Avg cost	3280.30	4381.19	5387.09
HGGA with truncation	Avg cost	3277.23	4370.43	5361.22
	GAP	-0.09%	-0.25%	-0.48%
HGGA without truncation	Avg cost	3297.31	4393.97	5394.08
	GAP	0.55%	0.29%	0.13%

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-grained 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 time equivalent to that of the HGGA for the same instance. All algorithms were run 10 times and the best results were taken for comparison. The results are summed up in Table 7, which displays the gaps to the solution yielded by HGGA of the solutions obtained by the meta-heuristics for the same computing time as HGGA. The negative values observed for all instances and procedures indicate the computing effort of HGGA is well spent, the proposed hybrid meta-heuristic is outperforming the two meta-heuristics, UTS of Cordeau et al. (2004) and RVNS of Pirkwieser and Raidl (2008), which are used in the education procedure.

6 Conclusion

We introduced HGGA, a population-based hybrid meta-heuristic for the periodic vehicle routing problem with time windows. In this first generational genetic 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 the methods proposed in the

Table 7: Performance comparison for fixed computing effort; Cordeau et al. (2004) instances

Instances	Time (hours)	CLM04	PR08	HGGA
1a	0.03	-0.04	0	2909.02
2a	0.27	-0.51	-0.15	5026.57
3a	2.15	-2.69	-1.71	7024.96
4a	3.66	-2.11	-1.79	7738.25
5a	5.46	-2.91	-1.32	8321.15
6a	7.81	-3.88	-2.11	10478.80
7a	0.17	-0.28	-0.1	6782.68
8a	4.21	-1.7	-2.05	9574.86
9a	8.71	-3.04	-1.65	13188.40
10a	15.05	-4.52	-3.12	16906.80
1b	0.03	-0.51	0	2277.44
2b	0.43	-2.67	-0.8	4122.03
3b	2.34	-2.49	-1.69	5489.77
4b	4.14	-3.53	-2.05	6345.65
5b	7.43	-2.08	-3.1	6775.39
6b	8.41	-2.67	-3.09	8580.06
7b	0.26	-0.36	-0.33	5481.61
8b	4.78	-3.54	-1.31	7595.75
9b	10.17	-3.33	-3.12	10508.80
10b	18	-2.75	-4.14	13363.00
Average	5.18	-2.28	-1.68	7924.49

literature show that the proposed methodology is highly competitive, providing new best solutions for several large instances.

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, Collaborative Research and Development and Discovery Grants programs. We also gratefully acknowledge the support of Fonds de recherche du Québec through their infrastructure grants and of Calcul Québec and Compute Canada through access to their high-performance computing infrastructure.

References

H. Barbosa and A. Lemonge. An adaptive penalty scheme in genetic algorithms for constrained optimization problems. In *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 and M. Maischberger. A parallel iterated tabu search heuristic for vehicle routing problems. *Computers and Operations Research*, 39(9):2033–2050, Sept. 2012. ISSN 0305-0548.

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, and A. Mercier. An improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *Journal of the Operational Research Society*, 55:542–546, 2004.

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 I.H. Osman, editors, *Meta-Heuristics 98: Theory & Applications*, pages 331–344. Kluwer Academic Publishers, Norwell, MA, 1999.

T. G. Crainic, M. Gendreau, and J.-Y. Potvin. *Parallel Tabu Search*, pages 289–313. John Wiley & Sons, Inc., 2005.

T.G. Crainic and M. Toulouse. Parallel Meta-Heuristics. In M. Gendreau and J.-Y. Potvin, 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 Problem. *Annals of Operations Research*, 41:421–452, 1993.

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.

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. 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. Methaheuristics 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.

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, 1993.

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.

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 time 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.

T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers and Operations Research*, 40(1):475–489, 2013.

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., 2004) 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 $f(s) = c(s) + \alpha_1 q(s) + \alpha_2 d(s) + \alpha_3 w(s)$.

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 if 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., 2004).

To diversify the search, any solution $\overline{s} \in N(s)$ such that $f(\overline{s}) \geq f(s)$ 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 a 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 $f(\overline{s}) + p(\overline{s})$ 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. The value of τ was adjusted to the smaller number of iterations performed and set to τ=10, 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.

B Detailed Results

Table A9 summarizes the HGGA results for the Pirkwieser and Raidl (2009a) instances with and without travel cost truncation. The algorithm was run 10 times per instance. Best results, average results, standard deviations, and computation time are reported.

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

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

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

$p_{rev} = 0.1$; $T = 10$ {initial temperature}
$s^* \leftarrow s$
$numNeighbor = 18$ {number of neighborhoods}
$curIter = 1$ {current iteration}
repeat
  RandPermutation(ar, $numNeighbor$) {random permutation of neighborhood sequence}
  $curNeighbor = 1$ {index of current neighborhood}
  while ($curNeighbor \leq numNeighbor$ and $curIter \leq max_{iter}$) **do**
    $k =$ ar[$curNeighbor$]
    $s' =$ Shaking(s, k, p_{rev})
    2-opt(s')
    if (s' better than s) **then**
      2-opt*(s')
      $s \leftarrow s'$
      if (s better than s^*) **then**
        $s^* \leftarrow s$
      end if
    else
      {
      $prob_{accept} = e^{-(f(s')-f(s))/T}$
      if (accept s' with probability $prob_{accept}$) {accept worse solution} **then**
        $s \leftarrow s'$
      else
        $curNeighbor += 1$
      end if
      }
    end if
    $curIter += 1$
    Update temperature T if needed
  end while
until ($curIter > max_{iter}$) {exceed maximum number of iterations}
return s^*

VCGP13 of (Vidal et al., 2013) is the only one to provide the best solutions, and just for the truncation case. Therefore, we also include their best results of the truncation case in the last column for the comparison purpose.

Table A9: HGGA results on Pirkwieser and Raidl (2009a) instances with and without travel cost truncation

Instances	Without truncation			With truncation				
	HGGA			HGGA				VCGP13
No	Best	Avg	Std	Best	Avg	Std	Time (min)	Best
p4r101	4162.90	4169.59	7.44	4082.6	4084.59	2.63	25.73	4082.0
p4r102	3738.16	3742.76	4.29	3724.5	3728.98	3.52	32.81	3724.3
p4r103	3166.02	3172	9.1	3153.1	3158.86	5.8	31.92	3153.1
p4r104	2578.06	2585.99	8.75	2566.3	2576.77	7.82	33.99	2566.0
p4r105	3659.17	3667.67	8.82	3638.8	3647.34	10.37	32.83	3638.9
p4c101	2913.81	2913.81	0	2907.4	2907.4	0	34.26	2907.4
p4c102	2888.31	2894.39	7.7	2882.9	2887.48	4.77	35.67	2882.9
p4c103	2735.20	2752.55	12.68	2734.5	2745	7.34	41.02	2734.5
p4c104	2425.30	2437.63	15.15	2419.0	2425.96	11.08	43.33	2419.0
p4c105	2888.30	2902.44	10.94	2884.1	2886.62	9.17	34.67	2884.1
p4rc101	3975.53	3980.43	9.89	3955.3	3960.53	11.58	30.43	3956.4
p4rc102	3765.02	3777.95	11.5	3755.7	3755.72	0.04	31.06	3755.7
p4rc103	3469.81	3482.64	14.63	3450.1	3461.38	16.07	32.46	3449.9
p4rc104	2999.94	3013.01	11.52	2991.6	2995.49	5.02	35.11	2991.5
p4rc105	3952.03	3966.77	15.49	3933.1	3936.27	2.92	32.94	3932.6
Average	3287.84	3297.31	9.86	3271.93	3277.23	6.54	33.88	3271.89
p6r101	5393.83	5398.83	6.6	5376.4	5380.55	6.65	38.76	5376.1
p6r102	5298.10	5303.3	10.37	5201.3	5211.37	7.3	37.51	5201.6
p6r103	3955.42	3975.68	14.42	3940.6	3956.29	13.71	44.75	3940.5
p6r104	3341.60	3375.2	14.72	3335.9	3344.35	10.5	47.57	3335.8
p6r105	4288.44	4300.96	10.64	4273.0	4285.02	13.45	38.73	4272.9
p6c101	3984.30	3998.94	9.97	3981.2	3991.09	9.12	37.93	3981.2
p6c102	3843.20	3856.08	8.76	3841.7	3841.7	0	44.31	3841.7
p6c103	3523.30	3542.28	18.48	3523.3	3529.66	8.14	52.27	3523.6
p6c104	3210.58	3230.69	13.43	3217.4	3225.104	8.92	52.81	3206.3
p6c105	4052.10	4072.3	11.9	4052.1	4052.1	0	42.82	4052.1
p6rc101	5792.86	5806.72	9.57	5780.6	5787.24	7.75	37.97	5781.5
p6rc102	5353.53	5380.13	18.28	5333.2	5342.72	11.01	41.93	5333.3
p6rc103	4289.15	4310.79	20.2	4273.3	4288.97	13.9	41.98	4273.1
p6rc104	4076.84	4092.66	18.11	4062.1	4078.36	15.29	48.53	4062.0
p6rc105	5241.73	5264.98	24.12	5227.2	5241.86	11.71	40.11	5227.1
Average	4376.33	4393.97	13.97	4361.29	4370.43	9.16	43.19	4360.59
p8r101	6492.00	6505.19	11.8	6469.6	6483.93	9.83	44.09	6471.3
p8r102	6158.62	6163.74	7.88	6098.1	6108	11.55	45.23	6097.9
p8r103	4717.23	4760.88	18.7	4687.1	4696.72	13.91	51.02	4687.0
p8r104	4402.72	4430.98	15.54	4353.1	4366.52	14.87	52.31	4355.8
p8r105	5491.93	5505.27	19.38	5476.9	5484.21	14.65	45.31	5476.5
p8c101	4687.93	4713.2	16.1	4679.1	4693.37	10.42	49.6	4679.1
p8c102	4942.33	4966.82	15	4933.3	4953.35	16.28	51.22	4933.3
p8c103	4673.79	4689.97	16.94	4664.3	4676.4	15.47	61.28	4664.0
p8c104	4606.24	4626.79	20.81	4591.7	4603.21	14.57	46.33	4591.6
p8c105	5146.48	5169.32	21.27	5134.2	5134.2	0	47.62	5134.2
p8rc101	6883.01	6898.79	19.19	6846.9	6858.81	12.82	41.24	6847.2
p8rc102	5778.46	5799.03	20.91	5763.4	5769.08	7.53	48.09	5763.3
p8rc103	5447.04	5464.45	20.28	5425.1	5433.06	8.19	47.53	5424.9
p8rc104	4942.07	4960.9	16.39	4929.6	4941.31	7.77	53	4929.5
p8rc105	6224.69	6255.93	29.08	6203.6	6216.2	12.31	47.43	6203.4
Average	5372.97	5394.08	17.95	5350.4	5361.22	11.34	48.75	5350.6