Citation for published version:
Vidal, T, Battarra, M, Subramanian, A & Erdogan, G 2015, 'Hybrid metaheuristics for the clustered vehicle routing problem', Computers and Operations Research, vol. 58, pp. 87-99.
https://doi.org/10.1016/j.cor.2014.10.019

DOI:
10.1016/j.cor.2014.10.019

Publication date:
2015

Document Version
Early version, also known as pre-print

Link to publication

Hybrid Metaheuristics for the Clustered Vehicle Routing Problem

Thibaut Vidal[a], Maria Battarra[b,*], Anand Subramanian[c], Güneş Erdoğan[d]

[a]*Massachusetts Institute of Technology Laboratory for Information and Decision Systems 77 Massachusetts Avenue, Room 32-D566, Cambridge, MA 02139, U.S*
[b]*University of Southampton, School of Mathematics, Southampton, SO17 1BJ, UK*
[c]*Universidade Federal da Paraíba - Departamento de Engenharia de Produção, Centro de Tecnologia, Campus I - Bloco G, Cidade Universitária, João Pessoa-PB, 58051-970, Brazil*
[d]*University of Southampton, School of Management, Southampton, SO17 1BJ, UK*

Abstract

The Clustered Vehicle Routing Problem (CluVRP) is a variant of the Capacitated Vehicle Routing Problem in which customers are grouped into clusters. Each cluster has to be visited once, and a vehicle entering a cluster cannot leave it until all customers have been visited. This article presents two alternative hybrid metaheuristic algorithms for the CluVRP. The first algorithm is based on an Iterated Local Search algorithm, in which only feasible solutions are explored and problem-specific local search moves are utilized. The second algorithm is a Hybrid Genetic Search, for which the shortest Hamiltonian path between each pair of vertices within each cluster should be precomputed. Using this information, a sequence of clusters can be used as a solution representation and large neighborhoods can be efficiently explored, by means of bi-directional dynamic programming, sequence concatenation, and appropriate data structures. Extensive computational experiments are performed on benchmark instances from the literature, as well as new large scale instances. Recommendations on the choice of algorithm are provided, based on average cluster size.

Keywords: Clustered Vehicle Routing, Iterated Local Search, Hybrid Genetic algorithm, Large Neighborhoods, Shortest Path

---

1. Introduction

This paper addresses the *Clustered Vehicle Routing Problem* (CluVRP), which has been introduced by Sevaux and Sörensen (2008). The CluVRP is defined over an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertex 0 is the *depot* and any other vertex $i \in \mathcal{V} \setminus \{0\}$ is a customer with demand $q_i > 0$. A fleet of m vehicles, each with capacity Q, is stationed at the depot. The

*Corresponding author: Tel. +44 (0)23 8059 3863
Email addresses: vidal@mit.edu (Thibaut Vidal), m.battarra@soton.ac.uk (Maria Battarra), anand@ct.ufpb.br (Anand Subramanian), g.erdogan@soton.ac.uk (Güneş Erdoğan)

set of customers is partitioned into N disjoint and nonempty subsets called *clusters*, such that $\mathcal{V} = V_1 \cup \cdots \cup V_N$. The customers in each cluster have to be visited consecutively, that is, the vehicle visiting a customer in the cluster cannot leave the cluster until all the other customers in the cluster have been visited. Each edge $(i, j) \in \mathcal{E}$ is associated with a travel cost c_{ij}, and the objective is to minimize the total travel cost. The CluVRP is a generalization of the Capacitated Vehicle Routing Problem (CVRP, c.f. book of Toth and Vigo 2002), obtained when each cluster contains a single vertex, and of the Clustered Traveling Salesman Problem (CluTSP, Chisman, 1975), obtained when $m = 1$. The CVRP and the CluTSP are both $\mathcal{NP}$-Hard, and so is the CluVRP.

Sevaux and Sörensen (2008) introduced the CluVRP in the context of a real-world application where containers are employed to carry goods. The customers expecting parcels in the same container form a cluster, because the courier has to deliver the content of a whole container before handling another container. Clusters also arise in applications involving passenger transportation, where passengers prefer to travel with friends or neighbors (as in the transportation of elderly to recreation centres). Gated communities (residential or industrial areas enclosed in walled enclaves for safety and protection reasons) provide another natural example of clusters. The customers within a gated community are likely to be visited by a single vehicle in a sequence, otherwise the vehicles have to spend additional time for the security controls at the gates.

Clusters can thus be imposed by the geography, the nature of the application, as well as by practitioners aiming to achieve *compact* and easy-to-implement routing solutions. Clustered routes allow drivers to be assigned to areas (i.e., certain streets or postcodes) and allow the development of familiarity, which makes their task easier. In addition, clustered routes have significantly less overlaps. In several cases, the additional routing costs due to cluster constraints are compensated by the ease of implementation and the enhanced driver familiarity.

The literature on the CluVRP is quite limited as of the time of this writing. Sörensen et al. (2008) and Sevaux and Sörensen (2008) presented an integer programming formulation capable of finding the best Hamiltonian path between each pair of vertices in each cluster. Barthélemy et al. (2010) suggested to adapt CVRP algorithms to the CluVRP by including a large positive term M to the cost of the edges between clusters and a cluster and the depot. The CluVRP is solved as a CVRP by means of the algorithm of Clarke and Wright (1964) followed by 2-OPT moves and Simulated Annealing (SA). The authors also suggested to dynamically set the penalty M, but observed that the M term interferes with the Boltzmann acceptance criterion of the SA and leads to erratic performance. Computational results were not reported in this initial paper.

Pop et al. (2012) described the directed CluVRP as an extension of the *Generalized Vehicle Routing Problem* (GVRP, Ghiani and Improta, 2000). The authors adapted two polynomial-sized formulations for the GVRP to the directed CluVRP, but again no computational results were reported. Recently, Battarra et al. (2014) proposed exact algorithms for the CluVRP and provided a set of benchmark instances with up to 481 vertices. The best performing algorithm relies on a preprocessing scheme, in which the best Hamiltonian path is precomputed for each pair of endpoints in each cluster. This allows for selecting a pair of endpoints in each cluster rather than the whole path, relegating some of the problem complexity in the preprocessing scheme. The resulting minimum cost Hamiltonian path problems are reduced to instances of the Traveling Salesman Problem (TSP) and optimally solved with Concorde (Applegate et al., 2001). CluVRP instances of much larger size than the corresponding CVRP instances were optimally solved, thus highlighting the advantage of acknowledging the presence of clusters.

In this paper, we introduce hybrid adaptations of state-of-the-art CVRP metaheuristics for the CluVRP. Rather than rediscovering well-known metaheuristic concepts, we exploit the current knowledge on iterated local search and hybrid genetic algorithms (Subramanian, 2012; Vidal et al., 2014a) and focus our attention on developing efficient problem-tailored neighborhood searches and effectively embedding them into these metaheuristic frameworks. The proposed neighborhood searches aim at 1) better exploiting clustering constraints by means of pruning techniques, 2) exploring larger neighborhoods by means of dynamic programming, 3) reducing the computational time by means of re-optimization, bi-directional search, and data structures. Finally, these experiments lead to further insights on which type of metaheuristic to use for different instance sizes and cluster characteristics.

The remainder of the paper is organized as follows. Section 2 introduces the challenges related to the CluVRP. Sections 3 and 4 describe the proposed metaheuristics and efficient neighborhood-search strategies, whereas Section 5 discusses our computational results. Conclusions are drawn in Section 6, and further avenues of research are discussed.

2. Motivation

Battarra et al. (2014) showed that exact algorithms are capable of solving relatively large CluVRP instances. However, the CPU times remain prohibitively long for large-scale or real time applications. In this paper, we exploit the properties of the CluVRP to develop specialized hybrid metaheuristics that take advantage of cluster constraints. Solution quality is assessed by a comparison with exact solutions whenever possible, and among metaheuristics when it is not.

Two recent and successful metaheuristic frameworks are used in this work. The Iterated Local Search (ILS) algorithm of Subramanian (2012) is simple and flexible, combining the intensification strength of Local Search (LS) operators and effective diversification through perturbation operators. It proved to be remarkably efficient for many variants of the Vehicle Routing Problem (VRP), including the VRP with Simultaneous Pickup and Delivery (Subramanian et al., 2010), the Heterogeneous VRP (Penna et al., 2013), the Minimum Latency Problem (Silva et al., 2012), and the TSP with Mixed Pickup and Delivery (Subramanian and Battarra, 2013). The success of ILS is due to an intelligent design of intensification and diversification neighbourhoods, as well as their random exploration. This latter component allows for extra diversity, and leads to high quality solutions, even when applied to other problems such as scheduling (Subramanian et al., 2014).

ILS explores only feasible solutions, and allows for testing the M approach suggested by Barthélemy et al. (2010) without possible interferences between M and penalties applied to infeasible solutions. As mentioned in the introduction, the M approach consists of including a large positive term to all those edges that are connecting clusters and connecting the depot to the clusters. Any CVRP algorithm in which the M is chosen to be large enough returns a CluVRP solution in which the number of penalized edges is minimized, therefore a solution in which the cluster constraint is satisfied. Note that the number of edges connecting clusters or connecting the depot to a cluster is $m + N$ and their penalization can be easily deducted from the solution cost.

One drawback of this transformation is that most VRP neighborhoods consider moves of one or two vertices. These neighborhoods can often not relocate complete clusters, and thus many moves appear largely deteriorating due to M penalties, significantly inhibiting the progress towards higher quality solutions. As shown in this paper, ILS can partly overcome this issue by means of perturbation moves. However, as demonstrated by our computational results, a more clever application of the framework specific to the CluVRP considering relocating and exchanges of whole clusters and intra-cluster improvements produces solutions of comparable quality in considerably less CPU time. In the next section, we describe the ILS and these hybrid algorithms in more details.

The Unified Hybrid Genetic Search (UHGS) currently obtains the best known solutions for more than 30 variants of the CVRP and represents the state-of-the-art among hybrid metaheuristics for VRPs. More precisely, the algorithm succesfully solves problems with diverse attributes, such as multiple depots and periods (Vidal et al., 2012), time windows and vehicle-site dependencies (Vidal et al., 2013a), hours-of-service-regulations for various countries (Goel and Vidal, 2013), soft, multiple, and general time windows, backhauls, asymmetric, cumulative and load-dependent costs, simultaneous pickup and delivery, fleet mix, time dependency and service site choice (Vidal et al., 2014a), and prize-collecting problems (Vidal et al., 2014c), among others. It has been recently demonstrated that several combinatorial decisions, such as customer selections or depot placement, can be relegated directly at the level of cost and route evaluations, allowing to always rely on the same metaheuristic and local search framework while exploring large neighborhoods in polynomial or pseudo-polynomial time (Vidal et al., 2014b,c).

Our UHGS implementation is based on the assumption that the costs of the optimal Hamiltonian paths among vertices in the same cluster can be efficiently precomputed as in Battarra et al. (2014). Once these paths and their costs are known, an effective route representation as an ordered sequence of clusters can be adopted, and a fast shortest path-based algorithm then transforms this solution representation into the corresponding optimal sequence of customers, which will be explained in detail in Section 4. This approach drastically reduces the size of the search space of the UHGS method, which optimizes the assignment and sequencing of $\mathcal{O}(N)$ clusters instead of $\mathcal{O}(n)$ customers.

Our computational experiments allow to quantify the trade-off between adopting the preprocessing scheme to compute the Hamiltonian paths, which requires the solution of $\sum_{i=1,\dots,N} |V_i| \times (|V_i| - 1)$ TSP instances and searching in the space of clusters with UHGS, or working in the space of vertices with a well-designed ILS. As long as the average size of the clusters is not high, the computational burden of the preprocessing is not prohibitive, but is observed to become significant when the cluster size increases. On the other hand, UHGS is much faster (ignoring the preprocessing time) and obtains higher quality solutions. Through our computational experiments, we aim at identifying a critical cluster size that makes an approach with cluster-based solution representation more desirable than an approach using vertex-based representation.

3. The ILS metaheuristic

ILS is a well-known metaheuristic framework that iteratively alternate stages of local search (intensification) and perturbation moves (diversification). The interested reader is referred to Lourenço et al. (2003) for a detailed description of the metholodogy, whereas the structure of a typical ILS algorithm is presented in Algorithm 1.

The algorithm starts by generating an initial solution s_0; this solution is then improved by means of local search ($LocalSearch(s_0)$) and a local optimal solution s^* is obtained. Perturbation moves are applied to s^*, generating a new solution s', which in turn is improved by means of local search, generating the solution $s^{*\prime}$. The algorithm updates s^* if an acceptance criterion is met. A typical stopping criterion is to interrupt the execution of the algorithm if no improvement is obtained after n_I consecutive iterations.

Algorithm 1 Iterated Local Search

```
Procedure ILS:
s_0 ← GenerateInitialSolution;
s* ← LocalSearch(s_0);
While Stopping criterion is not met
    s' ← Perturb(s*, history);
    s*' ← LocalSearch(s');
    s* ← AcceptanceCriterion(s*, s*', history);
end ILS;
```

The ILS of Subramanian (2012) is a multi-start heuristic which returns the best solution after n_R restarts (i.e., the Algorithm 1 is executed n_R times). The initial solution is generated using a *parallel cheapest insertion heuristic*. Classical VRP/TSP neighborhoods are explored during the local search phase and the perturbation operator consists of multiple shift and swaps based moves selected at random. The neighborhood structures adopted in Subramanian (2012) are inter-route moves (RELOCATE, RELOCATE2, SWAP, SWAP(2,1), SWAP(2,2), 2-OPT*), as well as intra-route moves (RELOCATE, OR-OPT2, OR-OPT3, 2-OPT and SWAP). Detailed descriptions of these families of neighborhoods can be found in Subramanian (2012) and Vidal et al. (2013b). Inter-route LS neighborhoods are considered one by one in random order and, whenever an improving solution is found, intra-route LS operators are applied, also in random order, to this solution.

As previously mentioned, the algorithm of Subramanian (2012) can be used for solving the CluVRP by applying suitable penalties to edges between clusters and between clusters and the depot. Although simple, this straightforward adaptation has two main drawbacks: (i) most of the local search moves violate the cluster constraint, leading to high penalties, and consuming a large part of the CPU time; and (ii) many promising moves that relocate full clusters are not included in the neighborhoods, thus reducing the intensification capabilities of the LS. ILS was therefore adapted to better take advantage of clusters. In what follows, we denote this adaptation as ILS-Clu.

The ILS-Clu is a hybrid algorithm built upon the structure of ILS. Large neighborhoods proved to be very effective in solving VRP variants, however, to identify promising moves can be a difficult task that is usually left for a large part to randomization (e.g., in Adaptive Large Neighborhood Search, Pisinger and Ropke, 2007). In contrast, the CluVRP structure enables to apply moves to relevant sets of customers. Thus, the LS phase of ILS-Clu explores moves on different levels: among clusters, among edges connecting clusters or clusters with the depot, and within each cluster. This mechanism enables to explore a sufficiently large variety of moves while significantly reducing CPU time. As in ILS, the initial solution is generated using a *parallel cheapest insertion heuristic*. Iteratively, a randomly selected customer is inserted with minimum cost, either between customers from the same cluster, or between two clusters.

Four types of LS procedures are used in ILS-Clu: "InterRouteSearch$_C$", "IntraRouteSearch$_C$", "IntraClusterSearch" and "IntraClusterRestrictedSearch". The former two modify the sequence of clusters in the routes without changing the Hamiltonian paths and their endpoints in each cluster, whereas the latter two optimize the sequence of customers within each cluster. The set of neighborhoods used during "InterRouteSearch$_C$" contains the same inter-route neighborhoods as ILS – previously described – but moves are applied on clusters instead of single deliveries. The intra-route neighborhoods of "IntraRouteSearch$_C$" follow the same rationale.

The LS procedures "IntraClusterSearch" and "IntraClusterRestrictedSearch" rely on the RELOCATE, 2-OPT, and SWAP neighborhoods. "IntraClusterRestrictedSearch" explores only a linear number of LS-moves, more precisely those involving at least one endpoint customer in the cluster, whereas "IntraClusterSearch" explores all moves within a cluster. Algorithm 2 displays the main structure of ILS-Clu, and highlights the differences with ILS.

LEFT TO DO

Both ILS and ILS-Clu apply a perturbation mechanism after each LS stage, which consists

Algorithm 2 Local search of ILS and ILS-Clu

Local Search of ILS:

NL $\leftarrow$ set of InterRouteSearch neighborhoods;

While NL $\neq \emptyset$
  Choose randomly Neighborhood $\in$ NL;
  Find best s' of $s \in$ Neighborhood;
  If $f(s') < f(s)$ **then**
    $s \leftarrow s'$;
    $s \leftarrow$ IntraRouteSearch(s);
    Update NL;
  else
    Remove Neighborhood from NL;

return s;
end.

Local Search of ILS-Clu:

$NL_C \leftarrow$ set of InterRouteSearch$_C$ neighborhoods;

While $NL_C \neq \emptyset$
  Choose randomly Neighborhood $\in NL_C$;
  Find best s' of $s \in$ Neighborhood;
  If $f(s') < f(s)$ **then**
    $s' \leftarrow$ IntraClusterRestrictedSearch(s');
    $\bar{s} \leftarrow$ IntraRouteSearch$_C(s')$;
    If $f(\bar{s}) < f(s')$ **then**
      $s \leftarrow$ IntraClusterRestrictedSearch$(\bar{s})$;
    else
      $s \leftarrow \bar{s}$;
    Update NL_C;
  else
    Remove Neighborhood from NL_C;
$s \leftarrow$ IntraClusterSearch(s);

return s;
end.

of one or two randomly selected SHIFT(1,1) or SWAP moves. In ILS, SHIFT(1,1) relocates a random customer from its route r to a random position in another route r', and simultaneously relocates a random customer from r' to a random position in r. The same process is applied in ILS-Clu but considering clusters instead of single customers. Moreover, in ILS, SWAP exchanges two customers from different routes, whereas in ILS-Clu the exchange involves two clusters of the same route.LEFT TO DO

4. The UHGS metaheuristic

UHGS is a successful framework capable of producing high quality solutions for many VRP variants. It is a hybrid algorithm, where the diversification strength of a Genetic Algorithm (GA) is combined with the improvement capabilities of local search. One main challenge in the design of a hybrid genetic algorithm is to achieve a good balance between intensification and diversification while controlling the use of computationally intensive local search procedures. This balance is usually achieved by selecting a suitable initial population, crossover operators, mutation, and selection mechanisms. The variety of design choices and the tuning of a multitude of parameters often inhibit the flexibility of the GAs. In fact, most of the previous attempts in the literature focused on the design of problem-specific operators, failing to lead to general algorithms and frequently resulting in a large number of parameters to be tuned. UHGS (Vidal et al., 2014a) managed to overcome most of these drawbacks by adopting the following strategies.

4.1. General UHGS methodology

UHGS evolves a population of individuals representing problem solutions, by means of selection, crossover and education operators. Note that the operator *education* involves a complete local-search procedure aimed at improving the solutions rather than a randomized mutation. The population is managed to contain between μ^{MIN} and $\mu^{\text{MIN}} + \mu^{\text{GEN}}$ individuals, by pruning μ^{GEN} individuals whenever the maximum size is attained. The method is run until It_{max} individuals have been successively created without improvement of the best solution.

UHGS achieves a fine balance between intensification and diversification by means of a bi-criteria evaluation of solutions. The first criterion is the contribution of a solution to the population diversity, which is measured as the Hamming distance of the solution to the closest solutions in the population. The second criterion is the objective value. Solutions are ranked with respect to both criteria, and the sum of the ranks provides a "biased fitness" (Vidal et al., 2014a), used for both parents selection and survivors selection when the maximum population size is attained. To deal with tightly constrained problems, linearly penalized route-constraint violations – capacity or distance – are included in the objective. Penalty coefficients are dynamically adjusted to ensure a target ratio of feasible solutions during the search; infeasible solutions are managed in a secondary population.

During crossover, the whole solution is represented as a giant tour visiting all customers once, without intermediate depot trips. As such, a simple ordered crossover that works on permutations can be used. The optimal splitting of the giant tour into separate routes is performed optimally in polynomial time as a shortest path subproblem on an auxiliary graph (Prins, 2004). This process is known to be widely applicable in a unified manner to many vehicle routing variants as long as it is possible to perform separate efficient route evaluations to compute the cost of edges in the auxiliary graph (Vidal, 2013). Finally, UHGS relies on local search to improve every new offspring solution generated during the search. The LS operators used in UHGS are 2-OPT, 2-OPT*, CROSS and I-CROSS (Vidal et al., 2014a). The last two neighborhoods are limited to exchanges, with possible inversions, of up to two customers.

Local search is usually the bottleneck of most advanced metaheuristics for vehicle routing variants, and thus efficient evaluations of routes generated by the neighborhoods are critical for the overall algorithm's performance. When additional attributes (constraints, objectives or decisions) are considered, these route evaluations may be time consuming if implemented in a straightforward manner. To improve this process, UHGS relies on auxiliary data structures that collect partial information on any sub-sequence of consecutive customers in the incumbent solution. This information is then used for efficiently evaluating the cost and feasibility of new routes generated by local search moves since any such move can be seen as a recombination of subsequences of consecutive customers from the incumbent solution.

A simple illustration of this concept is now given. Consider a CVRP solution with two routes: $r_1 = (0, 1, 2, 3, 4, 5, 0)$ and $r_2 = (0, 6, 7, 8, 9, 0)$. To efficiently evaluate the capacity constraints, the partial load $Q(\pi)$ for any sub-sequence π of the incumbent solution is preprocessed prior to move evaluations. An inter-route 2-OPT* move breaking the edges $(3,4)$ and $(7,8)$ leads to two new routes $r'_1 = (0,1,2,3,8,9,0)$ and $r'_2 = (0,6,7,4,5,0)$ requiring cost and load feasibility evaluations. Loads $Q(\pi_{0\to3})$, $Q(\pi_{4\to0})$, $Q(\pi_{0\to7})$ and $Q(\pi_{8\to0})$ are known for sequences $(0,1,2,3)$, $(4,5,0)$, $(0,6,7)$ and $(8,9,0)$, respectively. Denoting $\oplus$ as the concatenation operator, we have $Q(r'_1) = Q(\pi_{0\to3} \oplus \pi_{8\to0}) = Q(\pi_{0\to3}) + Q(\pi_{8\to0})$, and in the same way $Q(r'_2) = Q(\pi_{0\to7}) + Q(\pi_{4\to0})$. Load constraints can thus be checked in $\mathcal{O}(1)$ operations, independently of the number of customers in the route. Otherwise, a straightforward approach sweeping through the new route and cumulating the demands would take a number of operations proportional to $\mathcal{O}(n)$. This type of route evaluation is referred to as move evaluation *by concatenation* in Vidal et al. (2014a), and can be applied to a wide range of resources (load, distance, time), constraints and objectives.

4.2. Application to the CluVRP

Our application of UHGS to the CluVRP relies on two contributions: a route representation based on an ordered sequence of clusters to reduce the search space, and efficient route evaluation procedures using concatenations to evaluate the cost of a route assimilated to a sequence of clusters. Thus, UHGS is applied on sequences of clusters, and the optimal path within the clusters is determined implicitly during move and route evaluations. This leads to an implicit structural problem decomposition, considering only a VRP of a size proportional to the number of clusters $N < n$, and relegating difficult combinatorial decisions on path selections within clusters at the level of route evaluations. These methodological elements can be easily integrated into the UHGS framework, and it was possible to use the original UHGS implementation with the sole addition of a new route-evaluation operator.

The method relies on the fact that in any cluster V_k, the cost $\hat{c}_{ij}$ of the best Hamiltonian path between customer $i \in V_k$ and customer $j \in V_k$ that services all other customers in $V_k \setminus \{i,j\}$ has been preprocessed (Battarra et al., 2014). Using this information, it is possible to obtain from a route represented as a sequence of clusters the best sequence of visits to customers in polynomial time by finding a shortest path in the auxiliary graph $\mathcal{G}'$ illustrated in Figure 1.

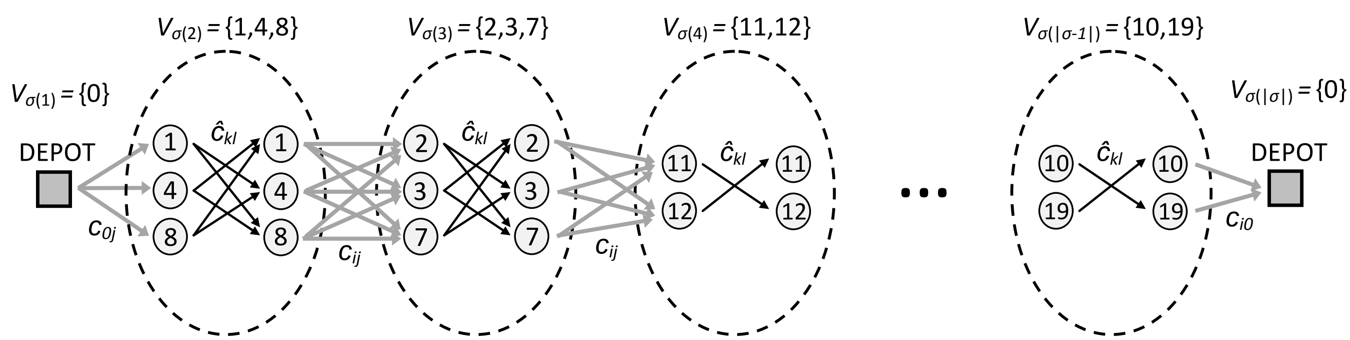

Figure 1: Route representation in UHGS

In Figure 1, black lines correspond to precomputed Hamiltonian paths within clusters. For each cluster in the route, a set containing two copies of each node is generated. Pairs of node copies are connected by an arc (k, l). The cost of this arc is set to the cost of the shortest Hamiltonian path $\hat{c}_{kl}$ in the cluster between k and l. The depot is then connected to the first copy of each node in the first cluster, and the second copies of the nodes are connected to the first node copies of the next cluster, and so on. The cost associated to these arcs (in gray in the figure) is the travel distance between the endpoints. A similar shortest path subproblem was previously used for the GVRP by Pop et al. (2013) and Vidal (2013).

A straightforward application of this technique leads to route evaluations in $\mathcal{O}(NB^2)$ operations, where B is the maximum number of customers in a cluster. These evaluations are computationally expensive. Another contribution of this work is to show that efficient procedures based on preprocessing and concatenations allow for performing each move evaluation in amortized $O(B^2)$ operations, thus only depending on the square of the cluster size. These more efficient evaluation procedures are now described.

For ease of notation, define $\lambda_i = |V_i|$ for any cluster V_i. In addition to the Hamiltonian paths within clusters, which are pre-processed a single time before starting the method, UHGS now pre-processes some information on each subsequence σ of consecutive clusters in each current solution. This information is updated whenever a solution change, e.g. a local search move, is applied. For any sequence $\sigma = (\sigma(1), \ldots, \sigma(|\sigma|))$ of clusters, the method computes for any i^{th} customer of the cluster $\sigma(1)$, and any j^{th} customer of the cluster $\sigma(|\sigma|)$, the shortest path $S(\sigma)[i, j]$ inside the sequence of clusters connecting i and j. These values can be computed as follows.

First, let us assume a subsequence $\bar{\sigma} = (V_k)$ containing a single cluster. If the cluster is restricted to a single customer, i.e., $V_k = \{v_i\}$, then $S(\bar{\sigma})[i, i] = 0$. Otherwise, $S(\bar{\sigma})[i, j]$ is given in Equation (1), in which $\hat{c}_{ij}$ represents the distance of the best Hamiltonian path connecting i and j in the cluster V_k.

$$S(\bar{\sigma})[i, j] = \begin{cases} +\infty & \text{if } i = j \\ \hat{c}_{ij} & \text{if } i \neq j \end{cases} \quad \text{for } i \in \{1, \ldots, \lambda_k\} \text{ and } j \in \{1, \ldots, \lambda_k\}, \tag{1}$$

Any longer subsequence can be viewed as a concatenation of shorter subsequences. Equation (2) enables to evaluate $S(\sigma)$ on a new sub-sequence resulting of the concatenation of a pair of subsequences σ_1 and σ_2, by induction on the concatenation operation $\oplus$. It is a direct application of the Floyd-Warshall algorithm.

$$\begin{aligned} S(\sigma_1 \oplus \sigma_2)[i, j] = \min_{1 \leq x \leq \lambda_{\sigma_1(|\sigma_1|)}, 1 \leq y \leq \lambda_{\sigma_2(1)}} & S(\sigma_1)[i, x] + c_{xy} + S(\sigma_2)[y, j], \\ & \text{for } i \in \{1, \ldots, \lambda_{\sigma_1(1)}\} \text{ and } j \in \{1, \ldots, \lambda_{\sigma_2(|\sigma_2|)}\} \end{aligned} \tag{2}$$

Equation (2) can be used to perform preprocessing on all subsequences of clusters in the current solution by iteratively appending a sequence made of a single cluster at the end. The same equation is then applied during neighborhood exploration to compute the cost of routes resulting from local moves. These routes can be viewed as a concatenation of a bounded number of subsequences from the current solution (Vidal, 2013). Figure 2 illustrates this pre-processing and move-evaluations mechanism. We consider the relocation of a subsequence of clusters σ_2 into a route originally made of $\sigma_1 \oplus \sigma_3$. The new route is thus a concatenation of the three sequences $\sigma_1 \oplus \sigma_2 \oplus \sigma_3$. The original shortest path problem is depicted on the top of the figure. Since pre-processing has been done on subsequences from the incumbent solution, the sets of shortest paths $S(\sigma_1)[i,j]$, $S(\sigma_2)[i,j]$ and $S(\sigma_3)[i,j]$ are known. They describe the shortest paths between any origin node i in the first cluster of the sequence, and any destination node j in the last cluster of the sequence. As a consequence, these known paths can be substituted in the graph, leading to an equivalent shortest path problem over a smaller graph, illustrated in the bottom of the figure. The number of nodes and arcs in this new graph does not depend anymore on the number of clusters in the route, but only on the number of concatenations, thus resulting in a reduced computational complexity.

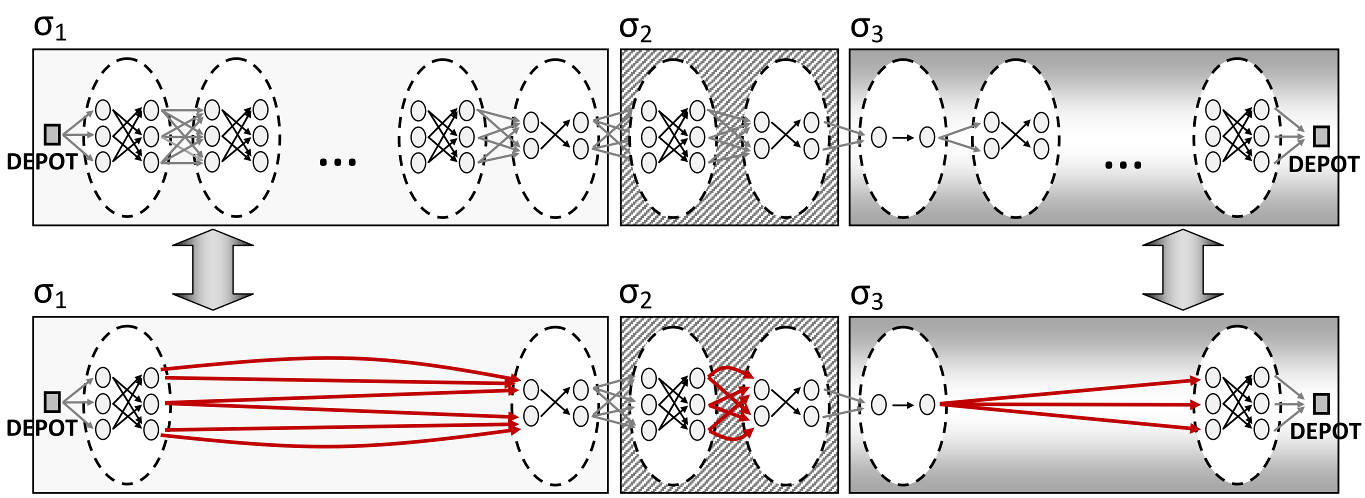

Figure 2: Using preprocessed information on subsequences

Proposition 1. *Using the proposed preprocessing, the amortized complexity of move evaluations, for classic VRP neighborhoods such as* Relocate, Swap, 2-Opt, 2-Opt*, *is* $\mathcal{O}(B^2)$.

Proof. First, from the current incumbent solution, the preprocessing phase requires computing the shortest paths between each pair of nodes, for each route. For each route, the graph $\mathcal{G}'$ is directed and acyclic. Equation (2) is applied iteratively, in lexicographic order starting from any cluster σ_i, $i \in \{1, \dots, |\sigma|\}$ and iteratively applied to σ_j for $j \in \{i+1, \dots, |\sigma|\}$ to produce all shortest paths. This equation is thus used $\mathcal{O}(N^2)$ times to perform a complete preprocessing on all routes. Each evaluation of this expression requires $\mathcal{O}(B^2)$ time. The total effort for the preprocessing phase is $\mathcal{O}(N^2B^2)$.

After preprocessing, a local search using classic VRP neighborhoods is performed. Any move based on less than k edge exchanges can be assimilated to a recombination of up to $k+1$ subsequences of consecutive clusters. This is the case for the mentioned neighborhoods with $k \leq 4$. Thus, each move evaluation is performed with a bounded number of calls to Equation (2), in $\mathcal{O}(B^2)$ elementary operations. The size S of each neighborhood is quadratic in the number of clusters (e.g. swapping any cluster i with cluster j leads to $S = \Theta(N^2)$ possible moves), such that a complete neighborhood exploration takes $\mathcal{O}(B^2N^2)$ time. The amortized complexity per move evaluation, considering both preprocessing and effective evaluation, is thus $\mathcal{O}(\frac{N^2B^2}{S}) = \mathcal{O}(B^2)$. □

5. Computational results

Computational experiments have been conducted on multiple benchmark instance sets. The first sets have been recently presented in Battarra et al. (2014). The authors considered instances proposed in the GVRP literature by Bektaş et al. (2011) (instance sets GVRP2 and GVRP3) and then generated larger instances by adapting the CVRP instances proposed by Golden et al. (1998) (instance set Golden) using the method reported in Bektaş et al. (2011). Among the instances proposed in Battarra et al. (2014), we have included in our benchmark set all Golden instances (200 up to 484 customers) and the most challenging ones among the GVRP2 and GVRP3 sets (the instances denoted as G and C in the GVRP literature, with 101 up to 262 customers). We have also generated an additional instance set with even larger problems (called hereafter Li), by adapting the method of Bektaş et al. (2011) on the instances originally proposed by Li et al. (2005). The latter set contains instances with up to 1200 customers. We decided to have clusters with average cardinality $\overline{\lambda} = 5$, leading to larger instances with 113 up to 241 clusters. A summary of the characteristics of our benchmark set is provided in Table 1. All sets of instances are available upon request and detailed result tables are displayed in the appendix (where the column *Exact* denotes the best upper bounds found in Battarra et al., 2014).

Table 1: Summary of benchmark set characteristics

Instance Set	Source	# Inst.	n	N
C	Bektaş et al. (2011)	2	101-200	34-100
G	Bektaş et al. (2011)	8	262-262	88-131
Golden	Battarra et al. (2014)	220	201-481	17-97
Li	New	12	560-1200	113 -241

An extensive calibration effort was spent in previous studies to find good and robust parameters for UHGS (Vidal et al., 2012) and ILS (Subramanian, 2012). We relied on this knowledge to obtain an initial parameter setting, and then scaled the parameters controlling algorithm termination to generate solutions for large-scale instances in comparable CPU time. As such, the population-size parameters of UHGS are set to $(\mu^{\text{MIN}}, \mu^{\text{GEN}}) = (8, 8)$ and the termination criterion is $It_{max} = 400$. For ILS, the number of restarts has been set to $n_R = 50$ and the number of perturbations is $n_I = n + 5m$ as in Subramanian (2012). The choice of $n_I = 1000$ was adopted for ILS-Clu. All experiments have been conducted on a Xeon CPU with 3.07 GHz and 16 GB of RAM, running under Oracle Linux Server 6.4. Each algorithm was executed 10 times for each instance using a different random seed.

Table 2 summarizes the results obtained by the proposed hybrid metaheuristics. For each benchmark set, the number of instances "$|Inst.|$" is given, as well as the number of times "# BKS" the best known solution is found by ILS, ILS-Clu and UHGS, respectively. Columns 6-9 provide the average CPU time per instance in seconds. $UHGS_p$ also includes the CPU time dedicated to computing the cost of all intra-cluster Hamiltonian paths with Concorde. Columns 10-12 report the average percentage deviation from the best known solutions "Avg. % Dev.". Note that the percentage deviation for a solution of value z from the best known solution value z_{BKS} is computed as $\frac{z - z_{BKS}}{z_{BKS}} \times 100$. The last row reports the overall number of best known solutions found by each method, the average CPU time and percentage average deviation.

From the experiments, it appears that UHGS is capable of finding most of the best known solutions (234 out of 242). In most cases, the average percentage gaps among the three methods is still small: ILS-Clu has an average deviation of 0.19% from the best known solutions and ILS has an average deviation of 0.13%. ILS is remarkably slower than the two other algorithms. The average CPU time for the large instances in the Li data set is 9548.6 seconds, versus 535.8, 345.3, 660.0 seconds for ILS-Clu, UHGS, and UHGS$_p$, respectively. Note that ILS performs about 0.06% better than ILS-Clu, but ILS-Clu is 15 times faster on average.

UHGS$_p$ is faster on average than ILS, even with the exhaustive search of all intra-cluster Hamiltonian paths using Concorde. This preprocessing phase is fast when the average cluster size is limited, but requires large CPU time when the cluster size increases, as in the case of the Golden instances. A heuristic evaluation of the cost of intra-cluster Hamiltonian paths could be a viable alternative. This is left as a research perspective. Finally, UHGS is faster than ILS-Clu for very large instances. ILS-Clu is on average faster on the G and C data sets, but slower on average on the Li set.

Table 2: Summary of results for the G, C, Golden and Li instances

		# BKS			Avg. Time (s)				Avg. % Dev.		
Instance Set	$\|Inst.\|$	ILS	ILS-Clu	UHGS	ILS	ILS-Clu	UHGS	UHGS$_p$	ILS	ILS-Clu	UHGS
G	2	0	1	2	127.6	53.5	150.2	165.2	0.64	0.22	0.00
C	8	6	8	7	26.0	17.8	27.1	35.1	0.19	0.04	0.05
Golden	220	127	87	213	698.8	53.9	53.7	854.9	0.11	0.19	0.01
Li	12	1	0	12	9548.6	535.8	345.3	660.0	0.34	0.21	0.00
Tot:	242	134	96	234	1110.7	76.6	68.1	812.4	0.13	0.19	0.01

Table 3: Summary of results for the Golden instance set grouped by instance size

	# Opt.			Avg. Time (s)				Avg. % Dev.		
n	ILS	ILS-Clu	UHGS	ILS	ILS-Clu	UHGS	UHGS $_p$	ILS	ILS-Clu	UHGS
201	11	11	11	81.18	19.92	14.57	2866.57	0	0	0
241	11	9	11	141.92	23.95	22.4	172.94	0	0.03	0
241	11	11	11	159.43	27.2	20.92	176.87	0	0	0
253	8	5	11	145.67	21.86	23.33	164.69	0.05	0.12	0
256	10	9	10	148.03	21.57	22.09	135.45	0.03	0.06	0.03
281	10	8	11	308.71	47.15	34.14	3848.32	0	0.03	0
301	10	11	11	309.59	37.9	30.75	191.26	0.02	0	0
321	3	2	11	442.25	47.27	47.57	243.39	0.08	0.07	0
321	7	2	11	333.56	34.29	38.12	152.78	0.12	0.25	0
324	6	4	11	336.87	31.55	43.82	175.73	0.19	0.31	0
361	3	1	11	816.36	73.97	66.94	2220.3	0.09	0.13	0
361	10	7	11	523.32	52.43	38.65	276.27	0.01	0.04	0
397	1	0	9	713.35	52.3	65.99	279.15	0.33	0.48	0.03
400	3	0	11	658.46	46.9	59.62	198	0.3	0.56	0
401	5	2	11	1115.99	85.16	82.91	1384.18	0.12	0.14	0
421	7	1	10	886.08	77.59	49.59	351.74	0.11	0.22	0.09
441	4	0	10	1573.62	101.69	97.09	1017.63	0.08	0.16	0
481	1	0	9	2336.64	130.52	137.95	2608.13	0.12	0.13	0.01
481	3	3	11	1344.21	74.34	84.3	246.31	0.15	0.39	0
484	3	1	11	1420.29	72.22	94.05	389.16	0.48	0.73	0
Tot:	127	87	213							

A more detailed comparison of the algorithms is displayed in Table 3 for the Golden instances. The large number of instances in this set allows for an analysis of the algorithms' performances by varying the number of customers and cluster size. The table reports aggregated results, obtained by averaging over instances with the same number of customers.

A correlation between the size of the instance and the performance of ILS can be observed; larger instances lead to larger gaps and higher CPU time. On the other hand, the performance of ILS-Clu is less dependent on instance size. For example, instances of group 12 with 484 customers are the most challenging for ILS-Clu with a 0.73% average deviation, but the deviation for instances of group 4, with size $n = 481$, is only 0.13% in average. A similar observation stands for UHGS, the most challenging instance groups being 4, 9, 14, and 20 with 481, 256, 397 and 421 customers, respectively.

Aggregating the Golden instances by average cluster size λ, as done in Table 4, leads to a further level of understanding of algorithms performance. All algorithms find solutions close to the best known when the average cluster size is large and therefore less clusters are present. The average CPU time of ILS does not depend on the average cluster size, whereas UHGS is consistently faster when large and few clusters are present. ILS-Clu attains its minimum CPU

Table 4: Summary of results for the Golden instances grouped by average cluster size

	# Opt.			Avg. Time (s)				Avg. Dev.		
λ	ILS	ILS-Clu	UHGS	ILS	ILS-Clu	UHGS	UHGS$_p$	ILS	ILS-Clu	UHGS
5	17	8	20	670.40	55.60	36.29	140.32	0.02	0.11	0.00
6	18	12	20	663.38	53.84	37.22	155.89	0.02	0.08	0.00
7	17	9	20	670.84	52.68	39.55	173.19	0.01	0.11	0.00
8	11	9	20	688.00	50.83	43.15	251.55	0.08	0.12	0.00
9	12	10	20	689.86	48.98	45.71	307.15	0.08	0.18	0.00
10	12	7	20	691.00	49.16	49.64	553.37	0.11	0.18	0.00
11	10	9	20	709.81	48.85	50.82	417.97	0.13	0.18	0.00
12	9	9	20	695.70	50.12	54.77	1025.66	0.13	0.18	0.00
13	8	5	20	725.73	53.12	69.04	916.16	0.18	0.29	0.00
14	7	5	17	699.89	59.99	73.52	2327.81	0.24	0.37	0.06
15	6	4	16	682.94	70.72	91.43	3135.31	0.23	0.31	0.03
Avg:	11.55	7.91	19.36	689.78	53.99	53.74	854.94	0.11	0.19	0.01

time when the average cluster size is approximately 9 customers. This is due to the fact that ILS-Clu performs both intra and inter-cluster LS moves; a balanced instance in terms of number and size of the clusters is a good compromise in terms of CPU time. Finally UHGS was capable of improving the best known solutions for five instances from Battarra et al. (2014). The values of these solutions are listed in Table 7.

6. Conclusions

This paper focused on the CluVRP, a generalization of the CVRP where customers are grouped into clusters. Three metaheuristics have been proposed, two of which are based on iterated local search, while the third is a hybrid genetic algorithm with a cluster-based solution representation. Efficient large neighborhood search procedures based on re-optimization techniques have been developed and integrated with the hybrid genetic search. The resulting three methods produce high quality solutions, and algorithms taking advantage of the cluster structure and large neighborhoods are remarkably faster. The hybrid genetic algorithm and large neighborhood search leads to solutions of higher quality that the two ILS based algorithms, but its pre-processing phase may become time consuming for instances with large clusters. Future work should consider heuristic preprocessing techniques to enhance CPU time, and other large neighborhoods strategies taking advantage of clusters.

Acknowledgements: This work was partially supported by CORMSIS, Centre of Operational Research, Management Sciences and Information Systems, and by CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico (grant 471158/2012-7).

References

Applegate, D., Bixby, R., Chvàtal, V., Cook, W., 2001. Concorde TSP solver. URL http://www.tsp.gatech.edu/concorde/index.html

Barthélemy, T., Rossi, A., Sevaux, M., Sörensen, K., June 2010. Metaheuristic approach for the clustered VRP. In: EU/ME 2010 - 10th anniversary of the metaheuristic community. Lorient, France.

Battarra, M., Erdoğan, G., Vigo, D., 2014. The clustered vehicle routing problem. Operations Research 62, 58–71.

Bektaş, T., Erdoğan, G., Ropke, S., 2011. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. Transportation Science 45, 299–316.

Chisman, J., 1975. The clustered traveling salesman problem. Computers & Operations Research 2, 115–119.

Clarke, G., Wright, J. W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12, 568–581.

Ghiani, G., Improta, G., 2000. An efficient transformation of the generalized vehicle routing problem. European Journal of Operational Research 122, 11–17.

Goel, A., Vidal, T., 2013. Hours of service regulations in road freight transport: an optimization-based international assessment. Transportation Science, Articles in Advance. URL http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2057556

Golden, B., Wasil, E., Kelly, J. P., Chao, I.-M., 1998. Metaheuristics in vehicle routing. In: Fleet Management and Logistics. Kluwer, Boston, pp. 33–56.

Li, F., Golden, G., Wasil, E., 2005. Very large-scale vehicle routing: New test problems, algorithms, and results. Computers & Operations Research 32, 1165–1179.

Lourenço, H. R., Martin, O. C., Stützle, T., 2003. Iterated local search. In: Glover, F., Kochenberger, G. (Eds.), Handbook of Metaheuristics. Vol. 57 of International Series in Operations Research & Management Science. Springer US, pp. 320–353.

Penna, P., Subramanian, A., Ochi, L., 2013. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. Journal of Heuristics 19, 201–232.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Computers & Operations Research 34, 2403–2435.

Pop, P., Kara, I., Horvat-Marc, A., 2012. New mathematical models of the generalized vehicle routing problem and extensions. Applied Mathematical Modelling 36, 97–107.

Pop, P., Matei, O., Pop Sitar, C., 2013. An improved hybrid algorithm for solving the generalized vehicle routing problem. Neurocomputing 109, 76–83.

Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research 31, 1985–2002.

Sevaux, M., Sörensen, K., October 2008. Hamiltonian paths in large clustered routing problems. In: Proceedings of the EU/MEeting 2008 workshop on Metaheuristics for Logistics and Vehicle Routing. Troyes, France.

Silva, M., Subramanian, A., Vidal, T., Ochi, L., 2012. A simple and effective metaheuristic for the Minimum Latency Problem. European Journal of Operational Research 221, 513–520.

Sörensen, K., Van den Bergh, J., Cattrysse, D., Sevaux, M., June 2008. A multiobjective distributionproblem for parcel delivery at TNT. In: Invited Talk at the International Workshop on Vehicle Routing in Practice, VIP'08. Oslo, Norway.

Subramanian, A., 2012. Heuristic, exact and hybrid approaches for vehicle routing problems. Ph.D. thesis, Universitade Federal Fluminense.

Subramanian, A., Battarra, M., 2013. An iterated local search algorithm for the traveling salesman problem with pickups and deliveries. Journal of the Operational Research Society 64, 402–409.

Subramanian, A., Battarra, M., Potts, C., 2014. An iterated local search heuristic for the single machine total weighted tardiness problem with sequence-dependent setup times. International Journal of Production Research 52, 2729–2742.

Subramanian, A., Drummond, L., Bentes, C., Ochi, L., Farias, R., 2010. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. Computers & Operations Research 37, 1899–1911.

Toth, P., Vigo, D. (Eds.), 2002. The vehicle routing problem. Society for Industrial and Applied Mathematics, Philadelphia.

Vidal, T., 2013. Approches générales de résolution pour les problémes multi-attributs de tournés de véhicules et confection d'horaires. Ph.D. thesis, Université de Montréal & Université de Technologie de Troyes.

Vidal, T., Crainic, T., Gendreau, M., Lahrichi, N., Rei, W., 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research 60, 611–624.

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

Vidal, T., Crainic, T., Gendreau, M., Prins, C., 2013b. Heuristics for multi-attribute vehicle routing problems: a survey and synthesis. European Journal of Operational Research 231, 1–21.

Vidal, T., Crainic, T., Gendreau, M., Prins, C., 2014a. A unified solution framework for multi-attribute vehicle routing problems. European Journal of Operational Research 234, 658–673.

Vidal, T., Crainic, T., Gendreau, M., Prins, C., 2014b. Implicit depot assignments and rotations in vehicle routing heuristics. European Journal of Operational Research 237, 15–28.

Vidal, T., Maculan, N., Ochi, L., Penna, P., 2014c. Large neighborhoods with implicit customer selection for vehicle routing problems with profits. Tech. rep., CIRRELT, Montréal.

7. Detailed results

Table 5: Detailed results, Golden 1–4.

Inst.	n	N	m	Exact	ILS Best	ILS Avg.	ILS Avg. Time(s)	ILS-Clu Best	ILS-Clu Avg.	ILS-Clu Avg. Time(s)	UHGS Best	UHGS Avg.	UHGS Avg. Time(s)	UHGS Preproc. (s)	UHGS Total Time(s)
Golden 1	241	17	4	**4831**	**4831**	4831.4	129.6	**4831**	4831.9	23.9	**4831**	**4831**	14.2	541	555.2
Golden 1	241	18	4	**4847**	**4847**	4849.6	134.3	**4847**	4848.4	21.8	**4847**	**4847**	17.1	151	168.1
Golden 1	241	19	4	**4872**	**4872**	4877.8	137	**4872**	4877.1	22	**4872**	**4872**	17.5	151	168.5
Golden 1	241	21	4	**4889**	**4889**	4896.2	135.8	**4889**	4891.2	21.2	**4889**	**4889**	17.6	147	164.6
Golden 1	241	22	4	**4908**	**4908**	4908.7	147	**4908**	4912.3	22.2	**4908**	**4908**	17.6	151	168.6
Golden 1	241	25	4	**4899**	**4899**	4905.9	134.4	**4899**	4899.9	24	**4899**	**4899**	20	144	164
Golden 1	241	27	4	**4934**	**4934**	4943	143.1	**4934**	4935.3	22.4	**4934**	**4934**	20.6	115	135.6
Golden 1	241	31	4	**5050**	**5050**	5051.5	150	**5050**	5050.2	23.6	**5050**	**5050**	21.6	90	111.6
Golden 1	241	35	4	**5102**	**5102**	5113.3	152.6	5108	5118	24.3	**5102**	**5102**	28.5	65	93.5
Golden 1	241	41	4	**5097**	**5097**	5108.4	149.1	5107	5113.3	27.8	**5097**	**5097**	31.9	55	86.9
Golden 1	241	49	3	**5000**	**5000**	5011.2	148.3	**5000**	5013.8	30.5	**5000**	**5000**	39.7	46	85.7
Golden 2	321	22	4	**7716**	**7716**	7719.5	461.8	7718	7722.2	51.6	**7716**	**7716**	31	368	399
Golden 2	321	23	4	**7693**	**7693**	7698.7	444.5	**7693**	7700.2	51.4	**7693**	**7693**	31.6	332	363.6
Golden 2	321	25	4	**7668**	**7668**	7675	441.9	7669	7677	49.3	**7668**	**7668**	32.8	305	337.8
Golden 2	321	27	4	**7638**	7644	7649.6	460.3	**7638**	7649.4	45.2	**7638**	7638.2	34.3	262	296.3
Golden 2	321	30	4	**7617**	7619	7635.3	447.4	7629	7644	38.8	**7617**	**7617**	37.6	172	209.6
Golden 2	321	33	4	**7640**	7649	7660.3	452.8	7646	7658	38.8	**7640**	7641.5	48.3	152	200.3
Golden 2	321	36	4	**7643**	7650	7656.5	439.9	7653	7660.4	39.4	**7643**	7649.7	49.1	167	216.1
Golden 2	321	41	4	**7738**	7759	7771.5	410.5	7742	7756.7	43.5	**7738**	**7738**	47.6	125	172.6
Golden 2	321	46	4	**7861**	7874	7888	431.8	7871	7884.7	45.3	**7861**	7863.8	55.6	99	154.6
Golden 2	321	54	4	**7920**	7925	7937.9	427.5	7926	7931.6	53.4	**7920**	7920.4	73	96	169
Golden 2	321	65	4	**7892**	7898	7910.8	446.4	7900	7906.3	63.2	**7892**	7893.4	82.4	76	158.4
Golden 3	401	27	4	**10540**	**10540**	10547.6	1077.6	10547	10572.5	88.3	**10540**	**10540**	51.2	9678	9729.2
Golden 3	401	29	4	**10504**	**10504**	10513.5	1020.5	10509	10520.9	83.7	**10504**	**10504**	58.2	2696	2754.2
Golden 3	401	31	4	**10486**	**10486**	10497.5	1103.9	10500	10511.5	78.4	**10486**	**10486**	56	326	382
Golden 3	401	34	4	**10465**	**10465**	10492.7	1093	10483	10497.6	79.1	**10465**	**10465**	59.6	372	431.6
Golden 3	401	37	4	**10482**	**10482**	10504.1	1096.4	**10482**	10515.2	78.1	**10482**	**10482**	64.6	346	410.6
Golden 3	401	41	4	**10501**	10518	10533	1194.8	10523	10544.2	79.5	**10501**	**10501**	76.1	290	366.1
Golden 3	401	45	4	**10485**	10502	10526.1	1078	10495	10525.1	76.4	**10485**	**10485**	69.3	97	166.3
Golden 3	401	51	4	**10583**	10605	10637.3	1172.1	**10583**	10639.6	78.8	**10583**	**10583**	83.6	153	236.6
Golden 3	401	58	4	**10776**	10789	10810.7	1157.1	10797	10821.5	84.1	**10776**	10777.9	124.8	124	248.8
Golden 3	401	67	4	**10797**	10824	10843.2	1110.9	10821	10844.8	96.4	**10797**	10801.2	121.5	123	244.5
Golden 3	401	81	4	**10614**	10654	10669.7	1171.6	10662	10682	113.8	**10614**	10621.6	147.1	109	256.1
Golden 4	481	33	4	**13598**	13606	13614.3	2397.9	13602	13628.5	130.6	**13598**	**13598**	87.4	7460	7547.4
Golden 4	481	35	4	**13643**	**13643**	13688.4	2362.8	13678	13706.4	127.2	**13643**	**13643**	88.5	7434	7522.5
Golden 4	481	37	4	**13520**	13525	13557.9	2207.7	13535	13569.3	113.5	**13520**	**13520**	89.1	1161	1250.1
Golden 4	481	41	4	**13460**	13466	13494.9	2116.6	13468	13492.7	113.4	**13460**	**13460**	95.9	5438	5533.9
Golden 4	481	44	4	**13568**	13588	13604.9	2113.7	13591	13601.5	112.8	**13568**	13570.5	121.1	1789	1910.1
Golden 4	481	49	4	**13758**	13784	13812.8	2262.5	13767	13794.8	121.8	**13758**	**13758**	117.9	1659	1776.9
Golden 4	481	54	4	**13760**	13772	13829.5	2368	13779	13816.6	123.7	**13760**	**13760**	121.2	113	234.2
Golden 4	481	61	4	**13791**	13810	13857.6	2294.6	13800	13846.6	123.5	**13791**	13791.9	133.3	1635	1768.3
Golden 4	481	69	4	**13966**	14000	14034.8	2584.6	13979	14022.8	136.1	**13966**	13967.2	195.6	198	393.6
Golden 4	481	81	4	**13975**	13987	14031.5	2710.1	14010	14031.6	149.3	13977	13980.2	174	152	326
Golden 4	481	97	4	**13775**	13821	13857.5	2284.7	13808	13849	183.8	13783	13792.2	293.5	133	426.5

Table 6: Detailed results, Golden 5–8.

Inst.	n	N	m	Exact	ILS Best	ILS Avg.	ILS Avg. Time(s)	ILS-Clu Best	ILS-Clu Avg.	ILS-Clu Avg. Time(s)	UHGS Best	UHGS Avg.	UHGS Avg. Time(s)	UHGS Preproc. (s)	UHGS Total Time(s)
Golden 5	201	14	4	**7622**	**7622**	7622.5	74.5	**7622**	7640.8	22.5	**7622**	**7622**	8.3	7651	7659.3
Golden 5	201	15	3	**7424**	**7424**	7443.6	94.2	**7424**	7427.6	23.9	**7424**	**7424**	12.2	7568	7580.2
Golden 5	201	16	3	**7491**	**7491**	**7491**	94.3	**7491**	**7491**	25.1	**7491**	**7491**	13.3	7758	7771.3
Golden 5	201	17	3	**7434**	**7434**	7445.8	90.6	**7434**	7439.9	21.5	**7434**	**7434**	13.3	6893	6906.3
Golden 5	201	19	4	**7576**	**7576**	**7576**	71.2	**7576**	**7576**	16.9	**7576**	**7576**	12.8	498	510.8
Golden 5	201	21	4	**7596**	**7596**	7596.3	76.7	**7596**	7596.3	16	**7596**	**7596**	12.8	209	221.8
Golden 5	201	23	4	**7643**	**7643**	**7643**	78.7	**7643**	**7643**	16.9	**7643**	**7643**	14.6	151	165.6
Golden 5	201	26	4	**7560**	**7560**	**7560**	78.3	**7560**	**7560**	18	**7560**	**7560**	14.9	221	235.9
Golden 5	201	29	4	**7410**	**7410**	**7410**	75.3	**7410**	**7410**	17.1	**7410**	**7410**	16.5	219	235.5
Golden 5	201	34	4	**7429**	**7429**	**7429**	78.6	**7429**	7430.6	20.1	**7429**	**7429**	19.4	164	183.4
Golden 5	201	41	4	**7241**	**7241**	7243	80.5	**7241**	**7241**	21	**7241**	**7241**	22.1	40	62.1
Golden 6	281	19	3	**8624**	**8624**	8629.2	353.4	**8624**	**8624**	61.5	**8624**	**8624**	29.1	18971	19000.1
Golden 6	281	21	3	**8628**	**8628**	8629.7	371.1	**8628**	8632.4	59.8	**8628**	**8628**	27.7	20337	20364.7
Golden 6	281	22	3	**8646**	**8646**	8648	378.7	**8646**	8650.1	59.6	**8646**	**8646**	29.9	789	818.9
Golden 6	281	24	4	**8853**	**8853**	8868.5	281	**8853**	8869.8	41	**8853**	**8853**	26	525	551
Golden 6	281	26	4	**8910**	**8910**	8916.6	271.9	**8910**	8921.1	39.2	**8910**	**8910**	28	465	493
Golden 6	281	29	4	**8936**	**8936**	8962.8	302.7	8949	8963.7	40.5	**8936**	**8936**	34.3	257	291.3
Golden 6	281	32	4	**8891**	**8891**	8905.8	282.7	**8891**	8899.9	39	**8891**	**8891**	30.6	92	122.6
Golden 6	281	36	4	**8969**	**8969**	8970.4	284.7	**8969**	8971.6	38.8	**8969**	**8969**	32.3	160	192.3
Golden 6	281	41	4	**9028**	**9028**	9039.8	284.6	9039	9043.2	40.4	**9028**	**9028**	43	150	193
Golden 6	281	47	4	**8923**	**8923**	8924.2	282.8	**8923**	**8923**	46.3	**8923**	**8923**	39.6	134	173.6
Golden 6	281	57	4	**9028**	9031	9052.6	302.2	9031	9052.1	52.6	**9028**	9028.7	54.9	76	130.9
Golden 7	361	25	3	**9904**	9909	9931	1038.6	9922	9947.9	90.2	**9904**	**9904**	57.7	12354	12411.7
Golden 7	361	26	3	**9888**	9907	9914.5	951.4	9905	9924.4	84.4	**9888**	**9888**	57.4	3017	3074.4
Golden 7	361	28	3	**9917**	**9917**	9944	946.7	9922	9949.9	80	**9917**	**9917**	61.7	3092	3153.7
Golden 7	361	31	4	**10021**	10035	10048.9	737.3	10025	10054.7	66.1	**10021**	**10021**	48.5	2626	2674.5
Golden 7	361	33	4	**10029**	10040	10057.2	707.6	10042	10059.8	62.6	**10029**	**10029**	51.8	980	1031.8
Golden 7	361	37	4	**10131**	10141	10150	723.5	10134	10148.1	65.7	**10131**	**10131**	62	1005	1067
Golden 7	361	41	4	**10052**	**10052**	10071.3	736.5	**10052**	10067.6	67.1	**10052**	**10052**	58.1	116	174.1
Golden 7	361	46	4	**10080**	**10080**	10100	744.7	10094	10113.1	64.9	**10080**	**10080**	62.5	201	263.5
Golden 7	361	52	4	**10095**	10096	10117.6	804.7	10116	10130.6	65.3	**10095**	**10095**	83	108	191
Golden 7	361	61	4	**10096**	10115	10159.8	823.9	10128	10139.5	77.4	**10096**	**10096**	84.5	93	177.5
Golden 7	361	73	4	**10014**	10038	10051.9	765.2	10034	10057	89.9	**10014**	10014.9	109.1	95	204.1
Golden 8	441	30	4	**10866**	**10866**	10885.5	1650.7	10883	10897.3	102.7	**10866**	**10866**	67.1	1004	1071.1
Golden 8	441	32	4	**10831**	**10831**	10845.8	1700.1	10845	10866.9	104.2	**10831**	**10831**	67.3	1003	1070.3
Golden 8	441	34	4	**10847**	10849	10868.8	1565.7	10852	10892	104	**10847**	**10847**	65.9	970	1035.9
Golden 8	441	37	4	**10859**	10871	10885.1	1530.1	10883	10893.7	100.2	**10859**	**10859**	76.2	829	905.2
Golden 8	441	41	4	**10934**	**10934**	10964.9	1660.8	10945	10959	94.5	**10934**	**10934**	81.8	792	873.8
Golden 8	441	45	4	**10960**	**10960**	10996.5	1503.3	10969	10987.8	88.9	**10960**	**10960**	82.4	4836	4918.4
Golden 8	441	49	4	**11042**	11064	11089.5	1568.9	11060	11083.4	83.6	**11042**	11043.3	95.8	127	222.8
Golden 8	441	56	4	**11194**	11211	11244.6	1678.1	11210	11251.9	87.7	**11194**	11196.7	92.6	166	258.6
Golden 8	441	63	4	**11252**	11267	11312.3	1563.2	11267	11309	100.9	**11252**	11263.6	127.4	155	282.4
Golden 8	441	74	4	**11321**	11347	11372	1394.3	11357	11371	114.4	**11321**	11326.8	162.7	132	294.7
Golden 8	441	89	4	**11209**	11215	11258	1494.7	11243	11271.8	137.7	11211	11212.2	148.8	112	260.8

Table 7: Detailed results, Golden 9–12.

Inst.	n	N	m	Exact	ILS Best	ILS Avg.	ILS Avg. Time(s)	ILS-Clu Best	ILS-Clu Avg.	ILS-Clu Avg. Time(s)	UHGS Best	UHGS Avg.	UHGS Avg. Time(s)	UHGS Preproc. (s)	UHGS Total Time(s)
Golden 9	256	18	4	**300**	**300**	**300**	142.3	**300**	**300**	22.7	**300**	**300**	15	164	179
Golden 9	256	19	4	**299**	**299**	299.3	135.3	**299**	299.4	22	**299**	**299**	16.1	157	173.1
Golden 9	256	20	4	**296**	**296**	**296**	144.7	**296**	296.3	21.9	**296**	**296**	17.6	133	150.6
Golden 9	256	22	4	**290**	**290**	291	141.9	**290**	291.1	21.3	**290**	**290**	18.2	138	156.2
Golden 9	256	24	4	**290**	**290**	290.9	146.8	**290**	291.6	20.6	**290**	**290**	19.3	113	132.3
Golden 9	256	26	4	**288**	**288**	288.5	149.7	**288**	290.2	18.9	**288**	**288**	21.5	92	113.5
Golden 9	256	29	4	**292**	**292**	293.7	146.4	**292**	294.8	19.2	**292**	**292**	23.6	211	234.6
Golden 9	256	32	4	**297**	**297**	298	148.9	**297**	298.3	20.9	**297**	**297**	22.5	79	101.5
Golden 9	256	37	4	**294**	**294**	294.7	149.5	**294**	294.5	21.1	**294**	**294**	25.9	68	93.9
Golden 9	256	43	4	**295**	**295**	296.2	156.8	296	296.9	22.8	**295**	295.8	31.6	59	90.6
Golden 9	256	52	4	**296**	297	298.6	166	297	298.2	26	297	297	31.6	33	64.6
Golden 10	324	22	4	**367**	**367**	368.6	331.3	369	369.9	31.2	**367**	**367**	30.3	230	260.3
Golden 10	324	24	4	**361**	**361**	362.5	319.4	**361**	362.2	28.7	**361**	**361**	31.6	175	206.6
Golden 10	324	25	4	**359**	**359**	360.4	322.4	**359**	360	29.3	**359**	**359**	31.9	173	204.9
Golden 10	324	27	4	**361**	**361**	362.1	329.9	**361**	362.6	29.9	**361**	**361**	33.4	168	201.4
Golden 10	324	30	4	**367**	**367**	368.8	350	368	369	30.6	**367**	367.1	40.6	145	185.6
Golden 10	324	33	4	**373**	375	376.6	352.8	376	377.2	31.1	**373**	374.2	37.7	125	162.7
Golden 10	324	36	4	**385**	387	387.7	338.6	387	388.9	29.3	**385**	385.2	42.1	144	186.1
Golden 10	324	41	4	**400**	401	402.4	330	**400**	401.9	29.7	**400**	400.2	51.8	83	134.8
Golden 10	324	47	4	**398**	399	400.1	339.3	399	400.1	31.6	**398**	398.1	52.7	74	126.7
Golden 10	324	54	4	**393**	395	396.9	336.3	394	395.3	34.9	**393**	393.6	54.5	72	126.5
Golden 10	324	65	4	**387**	**387**	391.7	355.7	390	392	40.9	**387**	387.9	75.5	62	137.5
Golden 11	400	27	5	**457**	**457**	458.1	596.1	458	459	42.4	**457**	**457**	38.3	238	313.5
Golden 11	400	29	5	**455**	**455**	457.8	586.6	456	458.8	42.7	**455**	**455**	44.5	222	222
Golden 11	400	31	5	**455**	**455**	457.1	615.1	457	459.1	42.2	**455**	455.4	49.8	217	217
Golden 11	400	34	5	**455**	456	457.4	648.9	457	458.7	40.8	**455**	455.2	49.6	188	188
Golden 11	400	37	5	**459**	460	461.8	693.7	461	462.6	41	**459**	**459**	49.9	163	163
Golden 11	400	40	5	**461**	462	463.8	677.7	463	463.5	41.3	**461**	**461**	48.4	142	180.3
Golden 11	400	45	5	**462**	464	465.3	636.4	465	465.7	43.5	**462**	462.1	52.7	270	314.5
Golden 11	400	50	5	**458**	459	461.4	677.3	460	461.2	45.9	**458**	458.5	57.8	123	172.8
Golden 11	400	58	5	**456**	458	459.9	701.1	460	462	49.6	**456**	456.4	80.3	105	154.6
Golden 11	400	67	5	**454**	457	459.2	733.8	458	460.5	59.7	**454**	454.6	85.7	91	140.9
Golden 11	400	80	5	**451**	455	457.2	676.3	456	458.6	67	**451**	451.7	98.8	63	111.4
Golden 12	484	33	5	**535**	**535**	538.1	1338.3	537	538.4	63	**535**	**535**	80.3	270	322.7
Golden 12	484	35	5	**537**	**537**	538.4	1367.3	**537**	539.7	60.4	**537**	**537**	62.6	255	312.8
Golden 12	484	38	5	**535**	**535**	540	1327.9	537	540.5	61.9	**535**	**535**	70.3	231	311.3
Golden 12	484	41	5	**537**	539	542.5	1516.8	538	542.7	63.9	**537**	**537**	70.9	200	285.7
Golden 12	484	44	5	**535**	537	540.3	1406.3	540	542.2	62.4	**535**	535.8	87.4	194	292.8
Golden 12	484	49	5	**533**	537	538.3	1443	538	540.9	68.9	**533**	533.4	104	192	272.3
Golden 12	484	54	5	**535**	537	540.2	1679.6	539	542.9	69.5	**535**	535.3	92.1	1656	1718.6
Golden 12	484	61	5	538	538	541.3	1453.1	539	544.8	72.7	<u>**535**</u>	535.2	89.4	152	222.3
Golden 12	484	70	5	546	538	540	1519.4	540	543.6	80.5	<u>**533**</u>	533.5	104.4	131	201.9
Golden 12	484	81	5	546	541	543.4	1296.2	544	547.5	85.4	<u>**535**</u>	536	133.7	93	180.4
Golden 12	484	97	5	560	548	553	1275.5	548	554.5	105.9	<u>**544**</u>	544.7	139.6	56	160

Table 8: Detailed results, Golden 13–16.

Inst.	n	N	m	Exact	ILS Best	ILS Avg.	ILS Avg. Time(s)	ILS-Clu Best	ILS-Clu Avg.	ILS-Clu Avg. Time(s)	UHGS Best	UHGS Avg.	UHGS Avg. Time(s)	UHGS Preproc. (s)	UHGS Total Time(s)
Golden 13	253	17	4	**552**	**552**	**552**	144.7	**552**	553.8	21	**552**	**552**	15.8	159	251.1
Golden 13	253	19	4	**549**	**549**	549.1	138.4	551	551.3	20.7	**549**	**549**	16.4	128	217.4
Golden 13	253	20	4	**548**	**548**	548.3	140.9	**548**	549.5	19.7	**548**	**548**	17.9	127	231.4
Golden 13	253	22	4	**548**	**548**	548.4	141.2	549	549.5	20.5	**548**	**548**	20.3	110	243.7
Golden 13	253	23	4	**548**	**548**	548.5	138.5	**548**	549.1	20.1	**548**	**548**	19.6	107	246.6
Golden 13	253	26	4	**542**	**542**	542.1	146.6	**542**	542.6	20.2	**542**	**542**	20.4	94	109.8
Golden 13	253	29	4	**540**	**540**	540.3	147.7	**540**	540.9	21.3	**540**	**540**	21.8	218	234.4
Golden 13	253	32	4	**543**	**543**	543.7	145.7	544	544.9	21.3	**543**	**543**	23.7	72	89.9
Golden 13	253	37	4	**545**	546	547.9	147.4	546	548.8	21.7	**545**	545.2	33.7	54	74.3
Golden 13	253	43	4	**553**	554	555.1	146.4	554	555.6	25.3	**553**	**553**	29.2	44	63.6
Golden 13	253	51	4	**560**	561	562.1	164.9	561	563	28.6	**560**	560.4	37.9	29	49.4
Golden 14	321	22	4	**692**	**692**	692.8	320.7	693	695	35.4	**692**	**692**	25.6	214	235.8
Golden 14	321	23	4	**688**	**688**	688.3	330.6	**688**	689.7	32.2	**688**	**688**	27.3	181	204.7
Golden 14	321	25	4	**678**	**678**	679.1	317.3	679	680	31.2	**678**	**678**	29	169	202.7
Golden 14	321	27	4	**676**	**676**	677.6	317.3	**676**	678.6	31.1	**676**	**676**	29.3	147	176.2
Golden 14	321	30	4	**678**	680	680.5	319.3	682	682.3	30.5	**678**	**678**	32.2	128	165.9
Golden 14	321	33	4	**682**	**682**	683.6	341.5	684	685.3	31	**682**	**682**	32.9	118	143.6
Golden 14	321	36	4	**687**	**687**	688.3	349.3	688	689.5	33	**687**	**687**	33.7	163	190.3
Golden 14	321	41	4	**690**	691	692.4	339.7	691	692.9	33.7	**690**	690.1	50.2	83	112
Golden 14	321	46	4	**694**	697	698	352	697	699	34.1	**694**	695.7	48.5	65	94.3
Golden 14	321	54	4	**699**	701	703	339	703	704.2	38.2	**699**	700.1	52.7	53	85.2
Golden 14	321	65	4	**703**	**703**	706.3	342.7	704	705.9	47	**703**	**703**	58	37	69.9
Golden 15	397	27	4	**842**	844	844.2	728.5	844	846.3	54.9	**842**	**842**	47.5	244	277.7
Golden 15	397	29	4	**843**	844	846.4	736.6	845	849.2	52.2	**843**	843.4	52.5	218	268.2
Golden 15	397	31	4	**837**	839	840.9	742.7	841	843.3	48.5	**837**	837.1	49.3	203	251.5
Golden 15	397	34	4	**838**	842	844.2	755.2	844	846.6	48.2	**838**	838.5	72.4	169	221.7
Golden 15	397	37	4	**845**	848	850.1	739.6	848	853.5	48.5	**845**	845.2	51.6	152	210
Golden 15	397	40	4	**849**	**849**	851.9	744.5	850	853	50	**849**	849.2	65.3	141	188.5
Golden 15	397	45	5	**853**	856	857.8	655.4	855	858.4	45.5	**853**	853.1	58.5	1033	1085.5
Golden 15	397	50	5	**851**	854	855.7	710.7	857	859	48.8	**851**	851.8	68.1	111	160.3
Golden 15	397	57	5	**850**	854	856.4	700.3	856	858.3	50.6	**850**	850.4	83.5	94	166.4
Golden 15	397	67	5	**855**	857	862.1	685.4	861	862.9	57.2	857	857.3	82.9	73	124.6
Golden 15	397	80	5	**857**	863	864.3	648.1	862	864.2	70.9	858	859.6	94.3	51	116.3
Golden 16	481	33	5	**1030**	**1030**	**1030**	1268.8	**1030**	1031.2	67.7	**1030**	**1030**	54.7	296	354.5
Golden 16	481	35	5	**1028**	**1028**	1029.4	1238	**1028**	1030.6	65.5	**1028**	**1028**	59.5	251	319.1
Golden 16	481	37	5	**1028**	**1028**	1029	1288.6	**1028**	1029.7	63.1	**1028**	**1028**	60.9	238	321.5
Golden 16	481	41	5	**1032**	1033	1034.3	1329.9	1033	1034.6	64.8	**1032**	**1032**	60.1	199	281.9
Golden 16	481	44	5	**1028**	1029	1031.2	1297	1032	1033.3	64.6	**1028**	**1028**	63.8	179	273.3
Golden 16	481	49	5	**1031**	1033	1034.6	1264.7	1034	1036	65.4	**1031**	**1031**	71.2	161	215.7
Golden 16	481	54	5	**1022**	1024	1026	1505.4	1027	1028.8	69.2	**1022**	**1022**	83.9	214	273.5
Golden 16	481	61	5	**1013**	1015	1018.7	1498.7	1022	1023.1	74	**1013**	1013.8	94.7	143	203.9
Golden 16	481	69	5	**1012**	1015	1017.7	1525.2	1020	1020.8	79.4	**1012**	1012.3	114.5	120	180.1
Golden 16	481	81	5	**1018**	1019	1023.7	1312	1023	1026.5	93.2	**1018**	**1018**	105.3	94	157.8
Golden 16	481	97	5	**1018**	1023	1027.1	1258.1	1027	1029.9	110.9	**1018**	1020	158.8	57	128.2

Table 9: Detailed results, Golden 17–20.

Inst.	n	N	m	Exact	ILS Best	ILS Avg.	ILS Avg. Time(s)	ILS-Clu Best	ILS-Clu Avg.	ILS-Clu Avg. Time(s)	UHGS Best	UHGS Avg.	UHGS Avg. Time(s)	UHGS Preproc. (s)	UHGS Total Time(s)
Golden 17	241	17	3	**418**	**418**	**418**	177.8	**418**	418.1	32.3	**418**	**418**	15.4	209	292.9
Golden 17	241	18	3	**419**	**419**	**419**	176.2	**419**	**419**	30.9	**419**	**419**	17.2	192	286.7
Golden 17	241	19	3	**422**	**422**	**422**	169.4	**422**	**422**	31	**422**	**422**	17.8	172	286.5
Golden 17	241	21	3	**425**	**425**	**425**	171.1	**425**	**425**	30.2	**425**	**425**	20	162	267.3
Golden 17	241	22	3	**424**	**424**	**424**	179.5	**424**	424.1	31.3	**424**	**424**	20.1	155	313.8
Golden 17	241	25	3	**418**	**418**	**418**	173.3	**418**	418.4	25.9	**418**	**418**	21.9	111	126.4
Golden 17	241	27	3	**414**	**414**	**414**	165.3	**414**	**414**	25	**414**	**414**	22.9	81	98.2
Golden 17	241	31	4	**421**	**421**	421.1	132	**421**	421.3	20.9	**421**	**421**	21	73	90.8
Golden 17	241	35	4	**417**	**417**	417.1	135.9	**417**	417.4	20.8	**417**	**417**	22	53	73
Golden 17	241	41	4	**412**	**412**	412.1	134.7	**412**	**412**	23.8	**412**	**412**	24.6	36	56.1
Golden 17	241	49	4	**414**	**414**	414.1	138.5	**414**	414.7	27	**414**	**414**	27.3	32	53.9
Golden 18	301	21	4	**592**	**592**	**592**	304.1	**592**	593.6	41.1	**592**	**592**	22	329	351.9
Golden 18	301	22	4	**594**	**594**	**594**	318.3	**594**	595.6	39	**594**	**594**	22.5	300	321
Golden 18	301	24	4	**592**	**592**	592.1	323.4	**592**	593.7	41.2	**592**	**592**	23	294	316
Golden 18	301	26	4	**590**	**590**	**590**	319.6	**590**	590.9	36	**590**	**590**	24.5	229	253.6
Golden 18	301	28	4	**577**	**577**	**577**	317.9	**577**	577.4	35.6	**577**	**577**	26.3	164	191.3
Golden 18	301	31	4	**578**	**578**	**578**	302.2	**578**	578.7	35.3	**578**	**578**	28.8	136	158
Golden 18	301	34	4	**582**	**582**	582.1	305.7	**582**	582.1	34.2	**582**	**582**	29.6	112	134.5
Golden 18	301	38	4	**586**	587	587.3	301.8	**586**	587	35.6	**586**	**586**	34.1	100	123
Golden 18	301	43	4	**594**	**594**	594.8	303.5	**594**	594.5	36	**594**	**594**	35.5	77	101.5
Golden 18	301	51	4	**601**	**601**	601.9	309.5	**601**	601.9	39.4	**601**	**601**	43.6	51	77.3
Golden 18	301	61	4	**599**	**599**	599.6	299.5	**599**	600.1	43.8	**599**	**599**	48.4	47	75.8
Golden 19	361	25	10	**925**	**925**	**925**	342.5	926	926.4	55.1	**925**	**925**	13.8	607	636.6
Golden 19	361	26	10	**924**	**924**	**924**	335.9	**924**	925.2	52.2	**924**	**924**	13.9	519	553.1
Golden 19	361	28	4	**808**	**808**	808.6	658.1	809	810.2	63.3	**808**	**808**	35.1	389	424.5
Golden 19	361	31	4	**811**	**811**	812	597.7	812	813	56.7	**811**	811.2	48.6	288	331.6
Golden 19	361	33	4	**797**	**797**	798.3	609.5	**797**	798.3	51.4	**797**	**797**	43.3	201	249.4
Golden 19	361	37	5	**799**	**799**	799.9	539.1	**799**	800.4	46.3	**799**	**799**	38.1	147	160.8
Golden 19	361	41	5	**789**	**789**	**789**	532.6	**789**	**789**	45.6	**789**	**789**	36.1	123	136.9
Golden 19	361	46	5	**788**	**788**	**788**	529.3	**788**	**788**	44.3	**788**	**788**	37.7	116	151.1
Golden 19	361	52	5	**800**	**800**	**800**	536.6	**800**	800.2	47.5	**800**	**800**	42.9	109	157.6
Golden 19	361	61	5	**807**	**807**	808.6	521.5	**807**	807.8	51.6	**807**	**807**	48.4	85	128.3
Golden 19	361	73	5	**810**	811	812.5	553.6	811	812.1	62.7	**810**	810.1	67.2	71	109.1
Golden 20	421	29	11	**1220**	**1220**	**1220**	529	1221	1226.1	74	**1220**	**1220**	21	821	857.1
Golden 20	421	31	12	**1232**	**1232**	1232.1	506.2	1235	1237.2	73.8	**1232**	**1232**	20.7	536	573.7
Golden 20	421	33	12	**1208**	**1208**	1208.2	490.5	1212	1212.9	68.3	**1208**	**1208**	22.2	444	486.9
Golden 20	421	36	5	**1059**	1060	1062	1045.9	1060	1064.8	85.5	**1059**	**1059**	44.1	394	442.4
Golden 20	421	39	5	**1052**	**1052**	1053.2	1083	**1052**	1054.5	78.3	**1052**	**1052**	45	260	327.2
Golden 20	421	43	5	**1052**	**1052**	1053.8	1034.2	1053	1055.6	73.7	**1052**	**1052**	48.8	209	230
Golden 20	421	47	5	**1053**	1054	1055.2	1038.1	1055	1056	73.4	**1053**	**1053**	60.1	78	98.7
Golden 20	421	53	5	**1058**	1058	1059.8	834	1060	1061.1	75.6	**1058**	**1058**	55.9	207	229.2
Golden 20	421	61	5	**1058**	1058	1059.7	1050.4	1060	1061.1	76.1	**1058**	**1058**	62.4	203	247.1
Golden 20	421	71	5	**1049**	1059	1060.4	1049.1	1059	1061.1	83.5	1059	1059	71.5	186	231
Golden 20	421	85	5	**1049**	1050	1052	1086.5	1050	1051.5	91.3	**1049**	**1049**	93.9	97	145.8

Table 10: Detailed results, GVRP2

Inst.	n	N	m	Exact	ILS			ILS-Clu			UHGS				
					Best	Avg.	Avg. Time(s)	Best	Avg.	Avg. Time(s)	Best	Avg.	Avg. Time(s)	Preproc. (s)	Total Time(s)
G	262	131	12	–	3736	3749.6	127.1	3709	3718	66.8	**3693**	3711	230.9	9	239.9
C	101	51	5	**642**	**642**	**642**	5.1	**642**	**642**	8.5	**642**	642	9.2	3	12.2
C	121	61	4	**807**	**807**	809.2	15.7	**807**	**807**	16.3	**807**	808.3	22.8	4	26.8
C	151	76	6	**816**	**816**	817.5	22.2	**816**	**816**	21.8	**816**	817.6	27.4	4	31.4
C	200	100	8	994	965	971.6	54.3	**955**	961.4	35.6	960	964.2	89.8	8	97.8

Table 11: Detailed results, GVRP3

Inst.	n	N	m	Exact	ILS			ILS-Clu			UHGS				
					Best	Avg.	Avg. Time(s)	Best	Avg.	Avg. Time(s)	Best	Avg.	Avg. Time(s)	Preproc. (s)	Total Time(s)
G	262	88	9	3596	3294	3305.9	128	**3290**	3291.8	40.2	**3290**	3293.2	69.6	21	90.6
C	101	34	4	**607**	**607**	**607**	7.4	**607**	**607**	7.1	**607**	**607**	7.3	8	15.3
C	121	41	3	**691**	694	695.4	14.3	**691**	**691**	12.6	**691**	**691**	17.8	9	26.8
C	151	51	4	**804**	**804**	**804**	29.5	**804**	**804**	16.2	**804**	**804**	18.4	12	30.4
C	200	67	6	**908**	**908**	**908**	59.5	**908**	**908**	24.4	**908**	**908**	24.1	16	40.1

Table 12: Detailed results, Li

Inst.	n	N	m	ILS			ILS-Clu			UHGS				
				Best	Avg.	Avg. Time(s)	Best	Avg.	Avg. Time(s)	Best	Avg.	Avg. Time(s)	Preproc. (s)	Total Time(s)
Li	560	113	39	**27962**	28079.7	714.6	27970	27997.1	122.7	**27962**	27963	83.9	200	283.9
Li	600	121	62	29087	29111.8	783.6	29063	29077.6	123	**29051**	29055.6	94.8	196	290.8
Li	640	129	10	21368	21515.2	2930.2	21365	21435.3	240.8	**21243**	21330.9	254.6	265	519.6
Li	720	145	11	24777	24834.4	4444.1	24578	24664.5	330	**24486**	24557.4	315.5	327	642.5
Li	760	153	78	35190	35206.7	1669.5	35175	35182.9	242.6	**35166**	35169.6	143.2	284	427.2
Li	800	161	11	27350	27482.7	7422.5	27284	27362.8	457.8	**27238**	27307.7	363.8	263	626.8
Li	840	169	86	37878	37895	1881.6	37860	37871.3	298.3	**37859**	37871.4	222.8	302	524.8
Li	880	177	11	30637	30766.5	10824.4	30568	30665.7	571.3	**30483**	30580.2	444.7	402	846.7
Li	960	193	11	32873	32958	14578.7	32763	32888.3	742.6	**32656**	32717.1	515.6	290	805.6
Li	1040	209	11	35991	36083	20634.3	35976	36044.6	843.5	**35885**	35919.1	554.6	440	994.6
Li	1120	225	11	38690	38749	22831.5	38727	38785.8	1187.2	**38669**	38687.8	566.9	397	963.9
Li	1200	241	11	41534	41621.2	25868.3	41563	41622.9	1269.8	**41461**	41490.7	582.8	411	993.8