Journal of Universal Computer Science, vol. 21, no. 9 (2015), 1143-1167
submitted: 24/3/15, accepted: 10/8/15, appeared: 1/9/15

An Adaptive Metaheuristic for Vehicle Routing Problems with Time Windows and Multiple Service Workers

Gerald Senarclens de Grancy
(University of Graz, Austria
research@senarclens.eu)

Abstract: Distribution planning in urban areas faces a lack of available parking space at customer sites. One approach to mitigate the issue is to cluster nearby customers around known parking locations. Deliveries from each parking location to its assigned customers occur by a second mode of transport (for example by foot). These lead to long service times at each of the clusters. However, long service times in conjunction with time windows can lead to inefficient routes as nearby customer clusters with overlapping service times may not be connected. As a consequence, assigning additional service workers to each vehicle is a strategy to reduce service times. The additional workers can do the last mile deliveries in parallel to reduce the service time of a cluster and hence permit more efficient routing. The trade-off between paying additional workers to reduce costs for vehicles and driving creates a new decision problem called the vehicle routing problem with time windows and multiple service workers (VRPTWMS).

The present work introduces a stochastic cluster first, route second algorithm. The clustering takes care of assigning and scheduling customers to parking locations. Its goal is to allow the routing algorithm to obtain high quality results. These two stages are linked together with a feedback loop based on the well established ant colony optimization metaheuristic. This allows learning from prior results and leads to vastly improved solution quality. For each of the used benchmark instances new best known solutions were generated. Furthermore, it is shown that applying the concept of bi-modal transportation potentially reduces both cost and environmental impact in regular vehicle routing problems with time windows.
Key Words: vehicle routing, clustering customers, time windows, metaheuristic, ant colony optimization
Category: I.2.6, I.2.8, J.7

1 Introduction

Given the importance of transportation, vehicle routing problems (VRP) and vehicle routing problems with time windows (VRPTW) have been studied for about half a century. Consequently, the literature proposes sophisticated solution methods including a series of local search operators and metaheuristics [Bräysy and Gendreau, 2005a, Bräysy and Gendreau, 2005b].

Like with most other related problems, the published articles rely on a strong assumption when proposing solution algorithms. They assume the availability of parking spaces directly at customer sites large enough to host any delivery truck. However, in urban areas of many modern cities this assumption doesn't hold true. Space is scarce and housing as well as smaller stores cannot provide any designated parking spaces for delivery vehicles. Even though such a lack of parking space is evidently an issue in distribution planning, comparably little academic work has dealt with it.

A good way to adopt to this lack of parking spaces is to drop the assumption that trucks can park directly at customers sites. Instead, it can be presumed that a delivery vehicle can stop close enough to a customer site that the goods can be transported between the parking location and the customer in another mode. Such a mode could be a hand trolley, delivery bike or any other transportation device small enough to be transported by the main truck. Note that the usual assumption about the availability of parking locations is not entirely dropped. It is only loosened by assuming that designated parking space is available at given locations close enough to the customers instead of directly at the customer sites. This is closer to reality because many large cities already provide designated loading zones for trucks.

If the geographic density of the customers is big enough, it pays off to deliver goods to more than one customer from a single parking space. However, clustering multiple customers together with known parking locations leads to long service times. These can be reduced by employing additional personnel on the delivery trucks to allow parallelizing the deliveries between parking sites and customers. This leads to a series of questions which have to be answered properly in order to allow efficient delivery:

- How should the customers be clustered together around each of the known parking locations?
- In which order should each cluster's customers be served?
- Which customer cluster should be served by which delivery truck?
- How many workers should be added to any given truck?
- In which order should each truck visit its assigned customer clusters?

The novel decision problem trying to answer all of these questions is called the vehicle routing problem with time windows and multiple service workers (VRPTWMS). Figure 1 illustrates a small instance with an optimal solution[1].

The VRPTWMS was originally motivated by the South-American soft drink industry trying to deliver beverage crates to small stores in Brazilian megacities [Pureza et al., 2011]. Transportation cost is particularly important to the soft drink industry as the distribution accounts for up to 70% of the value added costs [Golden and Wasil, 1987]. However, related problems occur for any good being small and light enough to be transported in an alternative mode. This particularly includes most packages sent by electronic commerce companies. Also,

[1] All illustrations were generated with matplotlib 1.3.1 [Hunter, 2007].

the customers in this problem are not limited to small stores but may be individuals awaiting postal deliveries.

The VRPTWMS occurs in urban areas around the world including a number of European cities. The affected places share some common patterns: congested streets, slow average driving speed and a lack of parking space. For example, the most recent London streets performance report at the time of this writing announced that the average driving speed across central London dropped to 14.16 km/h (8.8 mph) [Transport for London, 2014]. Another real-life application of the VRPTWMS relates to extensive pedestrian areas prohibiting direct vehicle delivery.

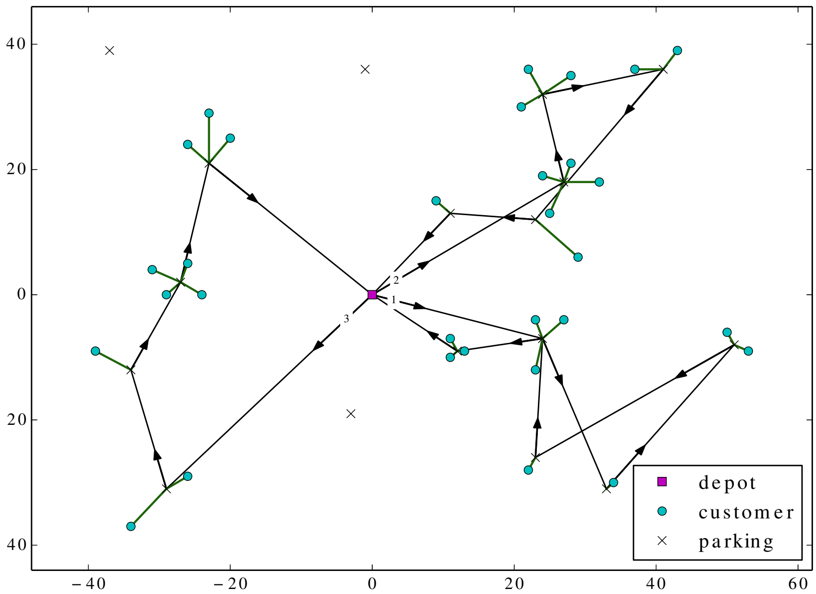


Figure 1: An example VRPTWMS instance solved to optimality. The numbers next to the depot designate the number of workers used on each truck. [Senarclens de Grancy and Reimann, 2015]

Given the rising importance of the VRPTWMS, this paper introduces the first metaheuristic based on stochastic extensions to existing heuristics for solving this decision problem. The stochastic elements help to prevent myopic decisions made by deterministic greedy heuristics. A heuristic approach was preferred since exact solutions, even when limited to the routing part, were not possible within a reasonable computation time for most instances. For example, [Pureza et al., 2011] report that only one out of 12 of their instances could be solved to optimality using CPLEX 11 on an Intel i7@2.67GHz with 12 GB RAM given a time limit of 10 hours. Their instances are derived from the classic VRPTW problems introduced by [Solomon, 1987]. The key difference is that the nodes have been adjusted to represent customer clusters instead of single customers.

The paper at hand makes multiple contributions concerning VRPTWMS. The introduced metaheuristic allows to obtain high quality solutions for the VRPTWMS and to better understand this novel problem. So far, it is the only metaheuristic ever proposed for the complete VRPTWMS. The key contribution is an adaptive learning mechanism that guides the cluster creation with information from routing solutions. Hence, the lack of an explicit objective function for clustering customers around known parking locations is alleviated. Furthermore, the introduced feedback loop also guides the routing process despite the fact that the routing problem is different in each iteration. Consequently, the proposed algorithms considerably improve the solution quality of each benchmark instance by allowing vast cost reductions. In addition, the applicability of the VRPTWMS and the introduced algorithm is demonstrated for more generic problems. The most important contribution is that this paper shows that allowing bi-modal transportation has the potential to improve reduce cost and environmental impact at the same time. Finally, the a research gap with regard to the customer clustering is addressed. This is done by suggesting that the average clusters size should be much larger than proposed by the literature.

The paper is organized as follows. Related literature is described in Section 2. Section 3 presents the details of the proposed algorithm. Experimental results on the benchmark instances from the literature are given in Section 4. Finally, conclusions and directions for future research are discussed in Section 5.

2 Problem Description and Literature Review

The VRPTWMS consists of a single depot, a set of customers and a set of available parking locations including the depot site. Each of the customers has to be served within their respective time window. Any of the parking spaces may be used more than once. A solution to the VRPTWMS constitutes a set of customer clusters and a set of truck routes visiting these clusters. A cluster consists of exactly one parking location and at least one customer. Each customer must be assigned to exactly one cluster and each cluster must be visited by exactly one vehicle. Deliveries may not be split.

The volume and weight of the deliveries is assumed to be transportable without a truck directly between a cluster's parking location and its customers. At the same time they are considered too heavy / bulky to service more than one customer without returning to the vehicle in between. The fleet is homogeneous and each vehicle can transport one (the driver) to γ service workers. The vehicle capacity v is higher than any customer's demand but less than the accumulated demand of all customers. The quality of a solution is determined by the number of vehicles employed, the required total number of service workers and the total distance driven by the vehicles. Note that neither the number of customer clusters nor the distance used by the second transportation mode directly affects the solution quality. The reason is that the workers performing the second mode of transportation are paid by shift without variable costs for gas or road taxes. This leads to a function for evaluating the quality of obtained solutions.

t required number of trucks

s required total number of service workers

d total distance driven by all trucks

c_t fixed cost per truck

c_s fixed cost per service worker

c_d cost per driven distance unit

$$C(t, s, d) = t \cdot c_t + s \cdot c_s + d \cdot c_d \tag{1}$$

The cost c_t per truck includes all fixed cost for the truck including possible tolls for entering a city center on a given day. The variable truck cost c_d includes distance based road tolls. One way to obtain feasible solutions is a cluster first, route second approach. In this case, the clusters are presented as black boxes to the routing heuristic. During cluster creation it is not known how many workers will serve the cluster. Obviously, this has nothing to do with cluster first, route second approaches to the VRP that arrange customers into subsets and solve a TSP for each of these subsets. In the VRPTWMS a cluster is a combination of a single parking space and one or more customers. Internally, each cluster requires a schedule for each possible number of workers. The sole paper to describe this situation is [Senarclens de Grancy and Reimann, 2015]. To allow efficient routing, each cluster has to provide information about its service times for each possible number of workers. The cluster service times increase with the distance between the parking location and the customers, the customer service times and the number of customers in the cluster. If the customer time windows inside a cluster are not overlapping, potential waiting times are also represented in the cluster service times. Finally, the speed of the second mode also affects the cluster service time. Additional workers generally reduce a cluster's service times by splitting up the customers in the cluster. Consequently, a cluster with a single customer will have the same service time no matter how many workers are used. For further details please refer to [Senarclens de Grancy and Reimann, 2015].

The first paper attempting to formulate and solve a simplified version of the VRPTWMS only investigated the routing stage [Pureza et al., 2011]. The authors assumed the clusters to be part of the input and presume the cluster service time to be a simple linear function of the combined demand and the number of service workers. The authors showed that this simplified problem is NP-hard. They created a mixed integer programming model. In order to obtain high quality results for the all instances, the authors implemented a tabu search and an ant colony optimization (ACO) metaheuristic. Another paper investigating the same subproblem implemented both a greedy randomized adaptive search procedure (GRASP) and an ACO [Senarclens de Grancy and Reimann, 2014]. In the paper, the two metaheuristics' performance was systematically compared. Doing so, the authors were able to provide new best results for all of the prior benchmark instances that had not been solved to optimality.

The first and besides the paper at hand sole publication allowing to solve the complete VRPTWMS is [Senarclens de Grancy and Reimann, 2015]. It introduces problem-specific clustering heuristics and made available a new set of designated benchmark instances for this problem. The authors defined characteristics of customer clusters and provide feasible solutions for all of the new benchmark instances. Their objective function's parameter setting ($c_t = 1$, $c_s = 0.1$, $c_d = 0.0001$) is in-line with prior literature [Pureza et al., 2011]. In order to make the results presented in this paper comparable, the same parameters will be used. However, the presented algorithms also work well for other settings.

Even though few papers have been dedicated to the VRPTWMS, similar problems have been studied more in depth. The problem at hand resembles the two-echelon VRP (2E-VRP) [Hemmelmayr et al., 2012] which also occurs in large cities and considers two levels. The main differences to the VRPTWMS are that the 2E-VRP does not consider time windows and solves sub-routing problems on the second level. Other problems resembling the VRPTWMS include the multi depot VRP (MDVRP) [Salhi and Nagy, 1999], location routing problem [Escobar et al., 2013], location-arc routing problem (also known as “park and loop” problem) [Doulabi and Seifi, 2013] as well as the truck and trailer routing problem (TTRP) [Villegas et al., 2011]. Among these problems, particularly the 2E-VRP and the park and loop problem also combine two modes of transportation. However, besides one exception [Lin et al., 2011], the author of this paper is not aware of any publications with regard to related problems that also deal with time windows. Finally, city logistics also addresses problems related to city planning with regard to a scarcity of large parking spaces, pedestrian areas and reduced maneuverability. An overview of related models can be found in [Crainic et al., 2009].

The metaheuristic described in the next section is based on the ACO method [Colorni et al., 1991, Dorigo and Stützle, 2004]. This method is inspired from the natural behavior of real ants laying out a pheromone trail to find shortest paths to good food sources. The metaheuristic relies on a problem-specific stochastic algorithm to repeatedly calculate solutions. The best prior solutions are then used to guide the stochastic component in order to find solutions with lower overall cost. ACO has been used successfully to provide high quality solutions to a wide range of NP-hard problems. It is known to work very well in all kinds of routing applications including non-traditional applications like routing of data packages [Junior et al., 2012].

3 Solution Method

Due to the problem's complexity, exact approaches suffer from limitations of computer memory as well as prohibitive computation time for all but the smallest instances. For proper operation, improvement-based metaheuristics require a sizable number of local search operators. Given the novelty of the VRPTWMS, no such operators exist for combined clustering and routing. Hence, a multi-start metaheuristic that allows learning from the best prior results is the method of choice. Multi-start metaheuristics include ACO and GRASP [Feo and Resende, 1995] - both of which have successfully been applied to the routing-only VRPTWMS. Among these two, only ACO has a native pheromone memory structure that facilitates learning. Consequently, the presented solution method is based on the ACO metaheuristic.

However, there are two issues with designing an ACO for the problem at hand. First, cluster creation cannot be supported directly with ACO's underlying concept of shortest paths. Second, the key idea of ACO is to solve the same problem multiple times and learn from prior results. Unfortunately, different clusters are created in each iteration and the routing algorithm has to deal with a different problem every time. Both issues have been resolved by abstracting ACO's long term pheromone memory. These modifications are detailed in the corresponding Subsections 3.1 and 3.2. The subsequent explanations are based on the notation outlined in table 1.

The basic mode of operation of the combined ACO algorithm is detailed in Listing 1 below. First, all pheromone matrices are initialized. As long as no stopping criterion (runtime limit or maximum number of generated solutions) is met, the solver continues. For a configurable number of ants, a complete solution is generated by first solving the clustering problem and then solving the routing based on the clustering solution. An improvement procedure allows removing superfluous service workers. If a new best solution is found that solution is stored for future reference. Once each ant has generated a solution, the pheromone

Table 1: Used notation

l_0	depot
N	set of customers $n \in N$
P	set of available parking locations $p \in P$
L	set of locations $l \in L$ $(N \cup P)$
δ_{ij}	distance between location i and j
z	cluster (set of assigned customers $n \in z$)
$\|z\|$	cardinality (number of customers in z)
z^p	cluster (set of assigned customers including parking location p)
$w_z^{\min}$	minimum number of workers required to serve z
n_z	customer n is a candidate for addition to cluster z
$t_{z,w}$	time required to service cluster z with w workers
t_{np}^{cu}	accumulated service time of servicing customer n from parking location p
t_z^{cu}	accumulated service time of cluster z
$t_{n \in z}^{cu}$	accumulated service time of cluster z after adding customer n
ζ_{ij}	pheromone link between location i and j
τ_{n_z}	total pheromone between candidate n and cluster z
$\tau_{z_i z_j}$	total pheromone between cluster z_i and z_j
$\tau_{\mathrm{l}_0\, z}$ / $\tau_{z\, \mathrm{l}_0}$	total pheromone between cluster z and the opening / closing depot
κ_{n_p}	attractiveness of creating a cluster with client n and parking location p
κ_{n_z}	attractiveness of adding client n to cluster z
$\kappa_{\mathrm{l}_0\, z_i\, \mathrm{l}_0}$	attractiveness of selecting cluster z_i as seed
$\kappa_{z_i z_j z_k}$	attractiveness of inserting cluster z_j between clusters z_i and z_k
γ	maximum number of workers on a single vehicle
υ	vehicle capacity

memory is updated with the best incumbent solution. Then, while no stopping criterion is met, each ant generates a new solution before the pheromone is updated again and so on. However, the newly created solutions will already be influenced by the pheromone memory. This means that decisions taken in the best solution known to the algorithm will have a higher attractiveness in future solutions. Hence, they are more likely to be taken again. This leads to focusing on exploring the neighborhood of good solutions as well as combining parts of the best recent solutions.

Listing 1: Overview of the ACO metaheuristic

```
def solve_aco(problem):
    pheromone = initialize_pheromone()
    while proceed():  # stopping criterion not met
        for ant in ants:  # each ant calculates a solution using pheromone
            clustering_solution = solve_clustering(problem, pheromone)
            routing_solution = solve_routing(problem, pheromone,
                                             clustering_solution)
            solution = improve_solution(routing_solution)
            if solution.total_cost() < problem.best_solution.total_cost():
                problem.best_solution = solution
        pheromone.update(problem.best_solution)
```

3.1 Clustering

The customers are clustered around the known parking locations via a stochastic version of sequential insertion heuristic. The deterministic version of the heuristic was introduced by [Senarclens de Grancy and Reimann, 2015]. It creates one cluster after the other. Once the heuristic decides to create a new cluster, all prior clusters are not considered for inserting further clients. In each iteration, the heuristic decides whether to create a new cluster or to insert a customer into the current cluster.

The stochastic modification is guided via a pheromone trail from the best incumbent solution. It is used for selecting which combination of a customer and a parking space should be used to start a new cluster. Subsequently, it is used for selecting which customer to add to the current cluster or whether to create a new one. The stochastic element is based on a weighted roulette wheel mechanism. Each action is assigned an attractiveness based on a value the heuristic calculates as well as the pheromone trail. The higher this attractiveness, the higher the chance for an action to be executed. Any impossible action like combining a customer with a parking location that is too far away results in an attractiveness value of 0.0 and will not be considered.

For selecting which customer n - parking space p pair is used for a new cluster, each customer is combined with its closest parking location. The attractiveness κ_{n_p} of the combination is defined by the product of the inverse total service time and the pheromone trail between the two locations:

$$\kappa_{n_p} = \frac{1}{t_{np}^{cu}} \cdot \zeta_{np} \tag{2}$$

In the above equation, t_{np} is the sum of the service time for 1 to γ workers. A cluster's service time is the same for any number of workers if the cluster only contains a single customer. Hence, the total service time is γ times the sum of the time required to walk back and forth between the parking space and the customer plus the customer's service time. ζ_{np} is high if parking location p and customer n were part of the same cluster in the best known incumbent solution. It requires a single lookup in the clustering pheromone data structure (see Subsection 3.3). This attractiveness calculation is summarized using the name `pick_seed(.)` in Listing 2.

The attractiveness κ_{n_z} of adding customer n to the existing cluster z which already has one or more customers is defined by

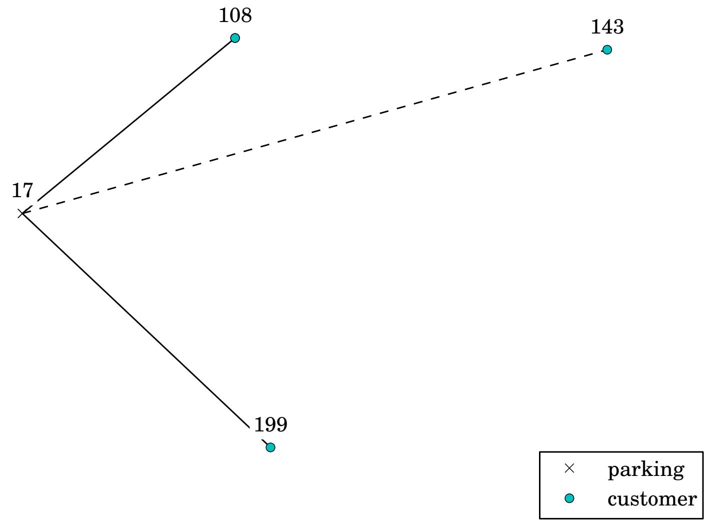
$$\kappa_{n_z} = \frac{1}{t^{cu}_{n\in z} - t^{cu}_{z}} \cdot \tau_{n_z} \tag{3}$$

For each number of service workers that can feasibly serve cluster z, the service time is added to t^{cu}_z $(= \sum_{w=w^{\min}_z}^{\gamma} t_{z,w})$. The same is done after adding n to z $(t^{cu}_{n\in z} = \sum_{w=w^{\min}_{n\in z}}^{\gamma} t_{n\in z,w})$. If adding n to z requires an additional service worker ($w^{\min}_z \neq w^{\min}_{n\in z}$), it is not possible to calculate the new cluster's service time with the prior minimum number of workers. To accommodate for this, a proxy value is required. In order to relate the missing value to the cluster, the original service time value before adding n is taken and multiplied by a penalty factor. The choice of proxy was tested during the setup phase of the computational experiments and delivered the best results.

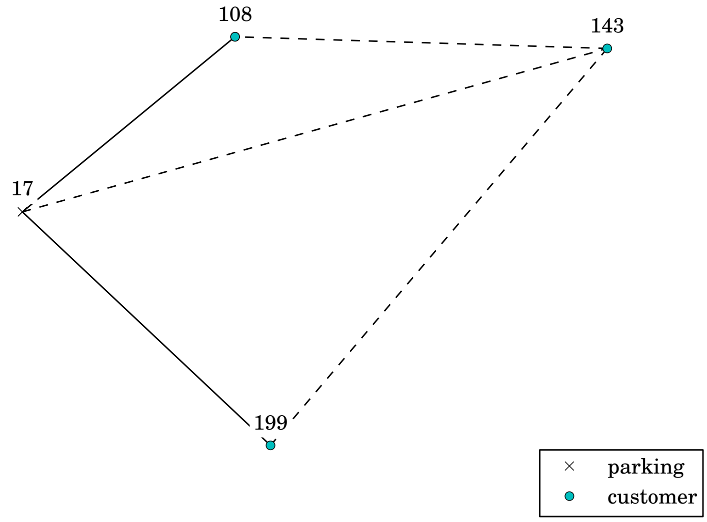
This allows to calculate a delta of the accumulated service times. The higher this delta, the lower is the likelihood that a customer will be added to a cluster. Therefore, the inverse of the accumulated delta increases the attractiveness of the addition.

Should the accumulated delta be 0 or close to 0, $\frac{1}{t_{n\in z} - t_z}$ is set to a predefined maximum value. This avoids divisions by zero and preserves a reasonable probability of selecting other feasible insertions with small attractiveness values. The latter is important because it allows the algorithm to explore the entire solution space. Typical additions have values for the inverse delta between 0.0 and 0.1. Occasionally these values reach up to about 0.4. Very small deltas should be favored while allowing other decisions to be made. Hence, setting the maximum attractiveness value to 0.5 is a good choice which was confirmed in the computational study.

The attractiveness κ is then altered to honor the long-term pheromone memory. τ_{n_z} is directly proportional to the attractiveness and describes the pheromone between the members of cluster z and the candidate n. τ_{n_z} is the normalized sum of all links between the candidate n and all elements of z^p (see Figure 2). Even though only the direct link between n and the cluster's parking location p is actually walked, it is important to consider all other elements of the cluster as well. This is mainly because a parking space may be used by more than one cluster. Ignoring the other members would falsely result in high pheromone values for adding n to other clusters using the same parking space. Normalizing (dividing by the number of elements in z^p) the value is required to avoid a pheromone bias for big clusters. Consequently, the pheromone for adding n to z



(a) Trail based on actually walked path



(b) Trail based on all virtual links

Figure 2: Illustration of two strategies of calculating a pheromone trail between a new customer (143) and an existing cluster.

is defined as

$$\tau_{n_z} = \frac{\sum_{l \in z^p} \zeta_{nl}}{|z^p|} \tag{4}$$

This value requires $|z^p|$ lookups in the clustering pheromone data structure. In Listing 2 the attractiveness calculation implicitly takes place in the procedure `pick_customer(.)`. This procedure either returns the selected customer or a value that evaluates to `False`. In the latter case no additional customer should be added and a new cluster started. This is in-line with the sequential clustering heuristic defined in [Senarclens de Grancy and Reimann, 2015].

Listing 2: Clustering heuristic

```
def solve_clustering(problem, pheromone):
    solution = ClusteringSolution()  # create new solution
    unassigned_customers = problem.all_customers()
    while unassigned_customers:  # as long as there are unassigned customers
        seed = pick_seed(problem, pheromone, unassigned_customers)
        cluster = Cluster(seed, seed.closest_parking)  # create new cluster
        unassigned_customers.remove(seed)
        while unassigned_customers:
            new_customer = pick_customer(problem, pheromone,
                                         cluster, unassigned_customers)
            if (new_customer):  # the stochastic has selected a customer
                cluster.add(new_customer)
                unassigned_customers.remove(new_customer)
            else:  # no additional customers should be added to the cluster
                break
        solution.clusters.append(cluster)
    return solution
```

3.2 Routing

A routing heuristic for the VRPTWMS with constant clusters as input was developed by [Senarclens de Grancy and Reimann, 2014]. It is based on Solomon's I1 heuristic [Solomon, 1987]. However, in each iteration, the proposed long-term memory relied on the same clusters. Hence, in order to use this routing heuristic, the pheromone memory needs to be re-designed. In each iteration it must correctly cope with different customer / parking space associations. Furthermore, a different subset of the available parking locations P may be used. Finally, the number of times each parking location is employed can also change.

As a consequence, it is necessary to take advantage of the structure of the problem. In order to allow a long-term memory to work, static elements have to be identified. These are all the customers to be served as well as the available parking spaces. The pheromone memory must be based on these locations instead of the higher-level clusters. This requires designing new pheromone lookup and update procedures. Other than that, the routing pheromone matrices described in [Senarclens de Grancy and Reimann, 2014] can be applied accordingly.

The depot is static over all the clustering solutions and does not pose new problems. For calculating the pheromone for a routing seed, it suffices to check if any of the customers or the parking space in a given cluster were served succeeding or preceding the depot. However, to ensure that the considered parking space and customers are inserted into the correct route, the depot needs to be abstracted to be different for each route. That implies that a pheromone matrix with a row for each potential route as well as a column for each parking location and each customer is required. Furthermore, it must be possible to distinguish if a cluster should be inserted after the "opening" or before the "closing" depot. Hence, two separate matrices are required to represent the depot adjacency pheromone. Note that there is only a single depot. To emphasize that the algorithm must distinguish whether a cluster is inserted at the beginning or the end of a route, the depot is referred differently for both situations. It is called opening depot at the beginning of a route and closing depot at the end.

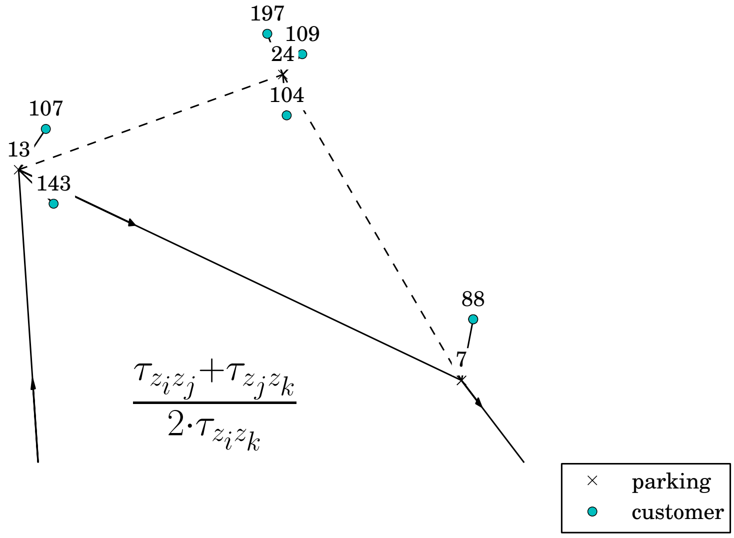
Both the insertion and seed attractiveness values are calculated based on the Solomon I1 heuristic attractiveness multiplied by a trail. The trail consists of a number of pheromone values indicating if two clusters were adjacent in the best incumbent solution. In this respect it is identical to the attractiveness used by [Senarclens de Grancy and Reimann, 2014]. However, the pheromone between two clusters cannot be stored directly as the clusters change in each iteration. Hence it is represented by the sum of all the pheromone links between the clusters' locations. This sum is normalized to be neutral to the cluster size to avoid a bias for large clusters. The normalization is performed by dividing the sum of the pheromone links by the number of links occurring between the evaluated clusters. The described concept is shown in Figure 3.

The pheromone trail influencing the likelihood of a seed cluster to be selected is the sum of the opening and closing depot pheromone. The opening depot pheromone is the normalized sum of all the pheromone links between the members of the currently investigated cluster and the opening depot. The pheromone between the closing depot and the investigated cluster is calculated in a corresponding fashion. The attractiveness of cluster z_i to be selected as a seed depends on its parking location's distance from the depot and its pheromone trail.

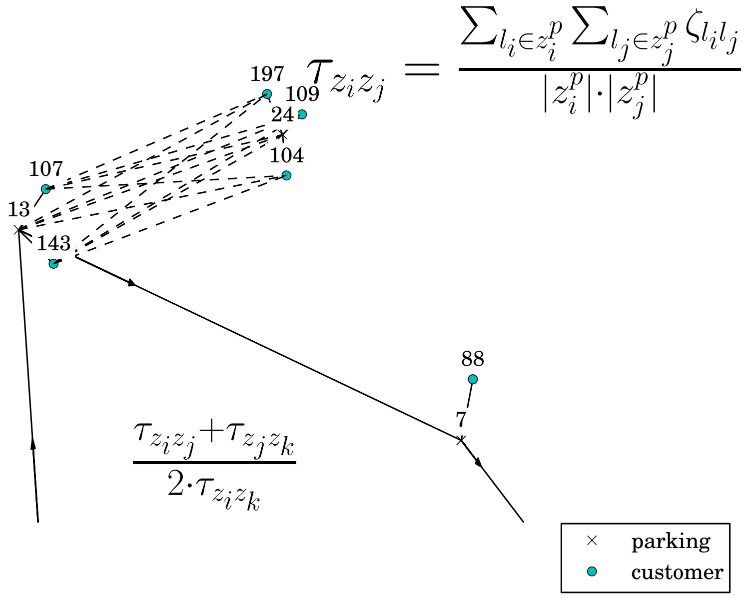
$$\kappa_{Dz^pD} = \delta_{Dp} * (\tau_{Dz} + \tau_{zD}) \tag{5}$$

The procedure using this κ_{Dz^pD} is called `pick_routing_seed(.)` in Listing 3 summarizing the routing algorithm. In the above equation, the pheromone values consist of the average of $|z^p|$ matrix look-ups to avoid the bias for large clusters. They are defined as

$$\tau_{Dz} = \frac{\sum_{l \in z^p} \zeta_{Dl}}{|z^p|} \quad \text{and} \quad \tau_{zD} = \frac{\sum_{l \in z^p} \zeta_{lD}}{|z^p|} \tag{6}$$



(a) Pheromone trail supports learning



(b) Pheromone lookup using static locations

Figure 3: Application of ACO on varying clusters.

Clusters are inserted into existing routes only at the position with the highest attractiveness. That leaves the selection of the cluster to insert as a stochastic decision. The attractiveness depends on the driving distance saved by the insertion as well as the pheromone trail. To insert cluster z_j between clusters z_i and z_k, the saved distance is

$$2 \cdot \delta_{Dz_j} - \delta_{z_i z_j} - \delta_{z_j z_k} + \delta_{z_i z_k} \tag{7}$$

The first part means that it is no longer necessary to serve z_j directly from the depot. The pheromone trail influencing the insertion is increased by the pheromone on the newly created route section. It is indirectly proportional to the replaced section (see Figure 3a).

$$\frac{\tau_{z_i z_j} + \tau_{z_j z_k}}{2 \cdot \tau_{z_i z_k}} \tag{8}$$

Therefore, the attractiveness of an insertion is defined by

$$\kappa_{z_i z_j z_k} = (2 \cdot \delta_{Dz_j} - \delta_{z_i z_j} - \delta_{z_j z_k} + \delta_{z_i z_k}) \cdot \frac{\tau_{z_i z_j} + \tau_{z_j z_k}}{2 \cdot \tau_{z_i z_k}} \tag{9}$$

The pheromone between any pair of clusters z_i and z_j is defined as the normalized sum of all links between the clusters. This concept is depicted in Figure 3b.

$$\tau_{z_i z_j} = \frac{\sum_{l_i \in z_i^p} \sum_{l_j \in z_j^p} \zeta_{l_i l_j}}{\left|z_i^p\right| \cdot \left|z_j^p\right|} \tag{10}$$

Should a cluster be inserted adjacent to the depot, one of the formulas from Equation 6 has to be used instead. The procedure using the above attractiveness values is called `pick_cluster(.)` in the routing algorithm's code listing below.

Listing 3: Routing heuristic

```
def solve_routing(problem, pheromone, clustering_solution):
    solution = RoutingSolution()  # create new solution
    unrouted_clusters = set(clustering_solution.clusters)
    while unrouted_clusters:  # while there are unrouted clusters
        route = Route(pick_routing_seed(problem, pheromone, unrouted_clusters))
        while unrouted_clusters:  # fill the current route
            insertion = pick_cluster(problem, pheromone, unrouted_clusters)
            if (not insertion.is_feasible):
                break
            route.insert(insertion)
            unrouted_clusters.remove(insertion.cluster)
        solution.add_route(r)
    return solution
```

3.3 Pheromone Data Structure and Update

The long term memory is divided into two parts - one for the clustering and one for the routing. Concerning the creation of clusters, high pheromone values indicate that locations were clustered together in the best recent solutions. That means that any customer can be linked to a number of other customers and a parking space. The most straight-forward way to store such information is a $|N| \times |L|$ matrix. That means that for each customer, linkage information to each other customer as well as each parking space can be stored. As no cluster can have more than one parking space, it is not required to store linkage data between parking spaces for the clustering heuristic.

The routing pheromone requires more data. The goal is to indicate whether pairs of clusters were immediate successors in the best incumbent solution. In this implementation, three matrices suffice to store the required information. The first is a $|L| \times |L|$ square matrix which stores high values if two locations were members of directly succeeding clusters on the same route. Although the depot requires separate treatment it is also needed in this matrix as it may serve as a parking space for a cluster in the middle of a route.

Even though all routes share the same depot to start and end their routes, the depot's pheromone requires special attention. Modelling the depot as a regular element of a route would mislead the pheromone guidance. It would increase the chance of a cluster to be inserted between the depot and one of its currently adjacent clusters if the cluster to be inserted was adjacing to the depot in the best incumbent solution. However, on each route one cluster succeeds and one preceeds the depot. Hence, different clusters would benefit from a higher likelyhood to be inserted before or after the depot, even though they were on a different route in the best incumbent solution. To avoid this problem, each route gets two distinct virtual depots – one for the opening and one for the closing depot. As detailed in [Senarclens de Grancy and Reimann, 2014], this allows the algorithm to take advantage of the knowledge on which route a cluster is to be inserted. In the worst case, every customer is in a separate cluster and every cluster on a separate route. Therefore, up to $|N|$ routes are theoretically possible. Because the clusters change in each iteration, it is required to store the depot adjacency information for each possible location. A separate matrix is required for the starting and ending depot, hence two additional $|N| \times |L|$ matrices are needed.

In total, four matrices are required to store all information for the long-term memory. The asymptotic memory requirement is $\mathrm{O}(|N| \cdot |L|)$ for the clustering and $2 \cdot \mathrm{O}(|N| \cdot |L|) + \mathrm{O}\left(|L|^2\right)$ for the routing. Since $\mathrm{O}(|N| \cdot |L|) < \mathrm{O}\left(|L|^2\right)$ and $a \cdot \mathrm{O}(b) = \mathrm{O}(b)$ if a is a constant, the memory requirement can be summarized as $3 \cdot \mathrm{O}(|N| \cdot |L|) + \mathrm{O}\left(|L|^2\right) = \mathrm{O}\left(|L|^2\right)$ (see [Hetland, 2010, pp 13-16] or [Weiss, 2006, pp 43-45] for details on the O notation).

All values in these matrices are initialized to 1.0. After each ant has calculated a solution (compare listing 1), the evaporation is simulated by multiplying all values by a constant $\rho < 1$. This constant is called the pheromone persistence factor. To avoid divisions by zero, no value in a matrix is allowed to be smaller than a predefined minimum > 0. In the reference implementation for the introduced algorithm, this value is called `MIN_DELTA` and is set to 0.01.

Thereafter, the best solution available to the algorithm is used to lay out new pheromone. This is done by adding $1 - \rho$ to all links occurring in the best solution. First, for each of the clusters the links between the cluster parking location and its customers as well as between the customers are reinforced (clustering pheromone matrix). Then, for each of the routes the links between the opening / closing depot and the members of the corresponding first / last clusters are intensified. This is done in the designated matrices for opening / closing depot pheromone. Finally, for each of the routes, the links between the adjoining clusters are strengthened in the $|L| \times |L|$ matrix.

4 Computational Study

The goal of the computational experiments was to find good settings for the required parameters and to test whether the introduced learning mechanism is capable of reliably improving the overall solution quality.

All experiments in this paper have been performed using the C++11 programming language[2]. Nevertheless, the presented code examples in this paper are based on the Python programming language version 3.4. It offers a much easier to read and to understand syntax than C++. The key advantage of using Python instead of pure pseudo-code is the elimination of syntactic ambiguities (e.g. undefined operators or precedence rules). While the code examples in this paper intend to make the introduced concepts easy to understand, possibly remaining questions about implementation details are addressed as well. In order to facilitate re-implementation and further work based on the presented results, the source code created for this article is available on the author's website[3] using an open source software license.

The computational experiments were performed on a single core of an Intel Core i5-4570 CPU @ 3.20GHz. The stopping criterion for the metaheuristic was a runtime limit of 900 seconds. The algorithm's implementation did not exceed 8752 kilobytes of memory[4] for any of the tested instances.

[2] The GNU C++ compiler version 4.8.2 was used on Kubuntu 14.04.
[3] `http://senarclens.eu/~gerald/research/`
[4] The maximum resident set size was measured using `/usr/bin/time -v`.

4.1 Parameter Settings

The key parameters when it comes to ACO are the number of ants and the pheromone persistence ρ. The first of them should correlate with the size of the used input. Experimental executions generated the best average results with the number of ants set to the number of available parking spaces (instead of the number of locations). The second parameter allows to control how fast the metaheuristic converges towards good solutions. If set too low, convergence is fast but not enough of the solution space is examined. This could lead to premature saturation of the pheromone leading to the algorithm getting stuck in a local optimum. If set too high, convergence is too slow to finish before the runtime is over. The selection of a persistence value depends on the CPU time required to generate a single solution, the total allowed runtime as well as the number of ants (responsible for the number of pheromone updates). Given a rather low number of 30 to 60 solutions per second, ρ has been set to 0.9 in the performed computational experiments. 30 to 60 solutions per second and a pheromone update occurring every 50 to 150 solutions lead to the update occurring roughly every 1-5 seconds. This means it occurs approximately 180 to 900 times during the whole runtime. Setting ρ to 0.9 requires 29 pheromone updates to reduce the initial values below 5% or 44 updates to reduce them below 1%. Under the above circumstances, an intensification of the search process around or between promising incumbent solutions can occur up to about 30 times which allows a good trade-off between exploration and intensification.

4.2 Results

In-line with prior papers on the subject all instances have been solved five times [Pureza et al., 2011, Senarclens de Grancy and Reimann, 2014]. To show the robustness of the metaheuristic, the best, average and worst result values are shown in Table 2. Given the small number of samples, the range denoted by the best and worst results provides a better measure of dispersion than the standard deviation. Notably, in all but two cases even the worst generated result is better then the prior best known result.

For all benchmark instances, the introduced feedback loop for learning from past results generated new best known solutions. Considering solely the best out of 5 runs, the aggregated total cost was reduced by 13.76%. The average solution quality still improves the prior best known results by reducing cost by 11.18%. Even the worst results outperform the best prior solutions on average by a cost reduction of 8.14%.

Table 3 shows details about each instance's currently best known solution. These values are published on the author's website[5] and will be regularly updated

[5] `http://senarclens.eu/~gerald/research/`

Instance	Prior BKS	ACO		
		Best (%)	Average (%)	Worst (%)
c200.050.n.1	47.53	40.91 (13.92)	42.01 (11.60)	43.33 (08.83)
c200.050.n.2	33.72	28.02 (16.90)	29.41 (12.77)	31.13 (07.68)
c200.050.n.3	26.56	21.59 (18.71)	21.95 (17.37)	22.99 (13.44)
c200.050.t.1	54.98	48.78 (11.27)	49.70 (09.60)	50.06 (08.94)
c200.050.t.2	42.17	35.80 (15.10)	36.25 (14.04)	37.68 (10.64)
c200.050.t.3	35.33	27.11 (23.25)	28.67 (18.84)	29.85 (15.53)
c200.050.w.1	44.55	38.70 (13.14)	40.45 (09.20)	42.11 (05.47)
c200.050.w.2	29.44	25.49 (13.40)	26.56 (09.77)	27.84 (05.42)
c200.050.w.3	24.26	20.16 (16.88)	20.59 (15.13)	21.57 (11.08)
c200.100.n.1	45.87	40.15 (12.47)	40.34 (12.05)	40.55 (11.61)
c200.100.n.2	32.13	26.93 (16.18)	27.84 (13.36)	28.84 (10.24)
c200.100.n.3	27.19	22.60 (16.88)	22.93 (15.68)	23.71 (12.79)
c200.100.t.1	55.27	44.36 (19.75)	45.60 (17.49)	48.39 (12.45)
c200.100.t.2	40.69	31.95 (21.49)	33.27 (18.23)	35.17 (13.58)
c200.100.t.3	35.23	25.98 (26.25)	26.94 (23.54)	27.65 (21.52)
c200.100.w.1	39.65	34.48 (13.05)	35.27 (11.06)	35.68 (10.02)
c200.100.w.2	28.78	23.08 (19.79)	23.77 (17.41)	24.08 (16.35)
c200.100.w.3	23.84	20.19 (15.32)	20.35 (14.64)	20.47 (14.13)
c200.150.n.1	42.21	36.22 (14.19)	36.43 (13.70)	36.83 (12.74)
c200.150.n.2	30.95	23.92 (22.72)	24.72 (20.12)	25.33 (18.16)
c200.150.n.3	24.70	20.47 (17.12)	21.06 (14.74)	21.49 (13.01)
c200.150.t.1	49.80	40.66 (18.35)	42.02 (15.61)	44.43 (10.79)
c200.150.t.2	37.00	30.30 (18.10)	31.09 (15.97)	31.77 (14.13)
c200.150.t.3	32.89	26.54 (19.32)	27.22 (17.24)	27.87 (15.27)
c200.150.w.1	36.17	32.42 (10.36)	33.04 (08.66)	33.83 (06.47)
c200.150.w.2	25.30	22.82 (09.79)	23.52 (07.03)	24.12 (04.66)
c200.150.w.3	21.90	20.17 (07.88)	20.62 (05.82)	21.51 (01.79)
r200.050.n.1	65.58	56.86 (13.30)	58.60 (10.64)	60.77 (07.33)
r200.050.n.2	44.83	38.85 (13.33)	40.84 (08.91)	42.85 (04.41)
r200.050.n.3	34.47	32.10 (06.88)	33.52 (02.76)	35.12 (-1.89)
r200.050.t.1	70.90	64.48 (09.06)	65.37 (07.80)	66.60 (06.07)
r200.050.t.2	54.01	48.61 (10.00)	50.56 (06.39)	52.43 (02.93)
r200.050.t.3	45.63	38.75 (15.08)	40.86 (10.46)	44.37 (02.77)
r200.050.w.1	62.29	56.65 (09.05)	57.56 (07.60)	58.87 (05.50)
r200.050.w.2	40.48	37.13 (08.29)	37.30 (07.85)	37.54 (07.26)
r200.050.w.3	31.85	28.20 (11.45)	30.11 (05.48)	30.81 (03.27)
r200.100.n.1	59.60	52.15 (12.50)	52.99 (11.09)	54.28 (08.93)
r200.100.n.2	40.92	35.35 (13.61)	36.33 (11.22)	37.68 (07.91)
r200.100.n.3	33.67	29.22 (13.22)	29.71 (11.77)	30.44 (09.58)
r200.100.t.1	67.58	58.70 (13.14)	60.23 (10.88)	63.12 (06.60)
r200.100.t.2	50.98	44.55 (12.62)	45.85 (10.06)	46.31 (09.17)
r200.100.t.3	40.93	36.67 (10.41)	38.49 (05.95)	39.95 (02.38)
r200.100.w.1	53.17	48.15 (09.45)	48.98 (07.88)	50.26 (05.46)
r200.100.w.2	36.17	30.72 (15.06)	32.18 (11.04)	33.44 (07.53)
r200.100.w.3	28.46	27.02 (05.06)	27.77 (02.41)	28.10 (01.25)
r200.150.n.1	58.11	50.66 (12.81)	53.12 (08.59)	55.11 (05.16)
r200.150.n.2	40.96	35.57 (13.16)	37.19 (09.20)	39.01 (04.77)
r200.150.n.3	34.24	30.06 (12.20)	31.03 (09.37)	32.05 (06.40)
r200.150.t.1	65.09	55.72 (14.40)	56.44 (13.29)	57.53 (11.61)
r200.150.t.2	46.48	39.30 (15.44)	40.84 (12.13)	42.33 (08.93)
r200.150.t.3	39.06	34.05 (12.83)	35.32 (09.57)	36.16 (07.43)
r200.150.w.1	50.19	45.64 (09.07)	47.43 (05.50)	49.48 (01.42)
r200.150.w.2	34.70	32.67 (05.85)	33.09 (04.65)	33.88 (02.36)
r200.150.w.3	28.76	27.54 (04.24)	28.58 (00.63)	29.23 (-1.64)
average	41.24	35.65 (13.76)	36.70 (11.18)	37.89 (08.33)

Table 2: Objective function values by instance (% deviations to prior BKS)

with fellow scientist's results. It can be noted that only a small number of clusters is created in relation to the number of trucks. That means that most routes only have very few clusters. However, these clusters include many customers and usually require the maximum number of service workers possible on the vehicles. This is due to the low cost of workers in relation to vehicles which was proposed by prior literature on the VRPTWMS [Pureza et al., 2011,Senarclens de Grancy and Reimann, 2015].

The values from Table 3 are aggregated by spacial distribution ((c)lustered vs. (r)andomly distributed locations), number of parking spaces (050 / 100 / 150), velocity ratio (10% / 20% / 30%) and time window width ((t)ight / (n)ormal / (w)ide) in Table 4. The results shown in both tables are as anticipated. If the customers and parking locations are closely spaced in separate districts, the overall cost is lower than for randomly distributed locations in the same gross area. The higher the velocity of the second mode of transportation in relation to the truck velocity, the lower the total cost gets. To a smaller extent, the same holds true for an increase in the number of available parking locations. Finally, the total cost is reduced the wider the time windows are (tight $\rightarrow$ normal $\rightarrow$ wide). All of these results are in-line with [Senarclens de Grancy and Reimann, 2015]. The strong improvements in solution quality as well as having the solution quality follow anticipated patterns strongly indicates that learning from past results pays off in the two-stage VRPTWMS. Finally, the results suggest that prior clustering heuristics created too many small clusters. The prior best known solutions used 89.46 clusters on average. The average size was hence 2.24 customers per cluster. The improved results obtained via the proposed feedback loop only use 47.92 clusters on average. This means that an average number of 4.66 customers per cluster (more than twice as many compared with the prior best known results) allowed for better routing results.

4.3 Comparison with Classic VRPTW Results

While the last subsection demonstrates that the proposed algorithm is clearly capable of providing vastly improved solutions, this subsection points out the relevance of allowing bi-modal transportation. The VRPTW has been thoroughly researched since more than 30 years. Publications propose highly efficient heuristics for creating solutions and local search operators for improving solutions. These are exploited by many different metaheuristics including simulated annealing, tabu search, ant colony optimization, adaptive large neighborhood search and variable neighborhood search to create top quality solutions. Publications concerning the VRPTW are summarized by [Bräysy and Gendreau, 2005a] and [Bräysy and Gendreau, 2005b].

The VRPTWMS is a recent extension of the VRPTW and received very little attention in comparison to the general VRPTW. Nevertheless, using the

Inst	Clusters	Vehicles	Workers	Distance	Cost
c200.050.n.1	58	32	86	3136.81	40.91
c200.050.n.2	30	21	59	2078.78	27.11
c200.050.n.3	28	17	45	1801.40	21.68
c200.050.t.1	59	38	104	3822.65	48.78
c200.050.t.2	43	27	74	2675.85	34.67
c200.050.t.3	34	21	59	2143.91	27.11
c200.050.w.1	48	29	85	3020.36	37.80
c200.050.w.2	35	20	53	1942.47	25.49
c200.050.w.3	33	16	40	1643.25	20.16
c200.100.n.1	57	30	81	3139.47	38.41
c200.100.n.2	45	21	57	2314.87	26.93
c200.100.n.3	44	17	42	1929.03	21.39
c200.100.t.1	57	34	93	3355.20	43.64
c200.100.t.2	42	25	66	2530.89	31.85
c200.100.t.3	30	20	58	1819.26	25.98
c200.100.w.1	60	27	71	2757.90	34.38
c200.100.w.2	41	18	46	1958.80	22.80
c200.100.w.3	26	15	39	1556.33	19.06
c200.150.n.1	64	28	75	3204.16	35.82
c200.150.n.2	43	19	47	2074.49	23.91
c200.150.n.3	25	15	40	1522.02	19.15
c200.150.t.1	77	32	80	3616.64	40.36
c200.150.t.2	65	24	60	3021.12	30.30
c200.150.t.3	45	19	51	2366.18	24.34
c200.150.w.1	80	26	61	3233.30	32.42
c200.150.w.2	43	17	46	2046.33	21.80
c200.150.w.3	27	15	39	1593.42	19.06
r200.050.n.1	56	44	125	3567.92	56.86
r200.050.n.2	42	30	86	2549.09	38.85
r200.050.n.3	28	24	67	1939.77	30.89
r200.050.t.1	56	49	140	3795.99	63.38
r200.050.t.2	53	38	103	3088.01	48.61
r200.050.t.3	36	29	86	2266.99	37.83
r200.050.w.1	53	43	121	3474.33	55.45
r200.050.w.2	41	28	76	2319.62	35.83
r200.050.w.3	34	22	60	2021.50	28.20
r200.100.n.1	67	41	108	3518.51	52.15
r200.100.n.2	43	27	76	2440.26	34.84
r200.100.n.3	39	22	58	2104.25	28.01
r200.100.t.1	66	44	119	3924.19	56.29
r200.100.t.2	49	33	93	2974.59	42.60
r200.100.t.3	42	28	75	2526.42	35.75
r200.100.w.1	62	37	99	3328.91	47.23
r200.100.w.2	46	24	65	2237.45	30.72
r200.100.w.3	35	21	54	1889.16	26.59
r200.150.n.1	68	39	104	3591.61	49.76
r200.150.n.2	52	26	71	2496.04	33.35
r200.150.n.3	52	22	55	2296.52	27.73
r200.150.t.1	75	43	110	3903.34	54.39
r200.150.t.2	62	31	79	3166.87	39.22
r200.150.t.3	46	24	66	2348.68	30.83
r200.150.w.1	60	35	96	3162.99	44.92
r200.150.w.2	46	25	65	2543.99	31.75
r200.150.w.3	40	20	52	1964.85	25.40

Table 3: Composition of best known solutions

Inst	Clusters	Vehicles	Workers	Distance	Cost
average	47.93	27.26	73.44	2624.94	34.87
average c	45.89	23.07	61.37	2455.74	29.46
average r	49.96	31.44	85.52	2794.14	40.28
average 050p	42.61	29.33	81.61	2627.15	37.76
average 100p	48.00	27.24	73.29	2612.73	34.83
average 150p	53.89	25.56	66.50	2675.14	32.47
average t	52.06	31.06	84.22	2963.71	39.77
average n	46.72	26.39	71.22	2539.17	33.77
average w	45.00	24.33	64.89	2371.94	31.06
average 10%	62.39	36.17	97.67	3419.68	46.28
average 20%	45.61	25.22	67.89	2469.97	32.26
average 30%	35.78	20.39	54.78	1985.16	26.07

Table 4: New best known results aggregated by the instances' properties

proposed algorithm to solve the classic 100 customer R1 instances proposed by [Solomon, 1987] shows the potential of bi-modal transportation. To do so, the R1 instances were used as input without any modification. Hence, every customer has a dedicated on-site parking location. However, the performed experiment allowed to stop a vehicle at a customer's parking location while visiting other customers via the second mode of transportation. In this experiment, the velocity for the second mode was set to 30% of the vehicle speed. This means that major congestion is assumed.

Despite a lack of local search operators in the proposed algorithm, three out of twelve R1 instances were clearly improved using the cost function in Equation 1. For the classic VRPTW instances, the best known solution's[6] number of trucks and required distance can be used directly in this equation. The number of workers is set to 1 for each instance since besides the driver, no additional workers are used. Considering the 1:10 cost ratio between a truck and a service worker proposed by [Pureza et al., 2011], three instances could be improved. Given this arguably extreme ratio, reducing a single truck allows for adding 10 workers without changing the objective function value. However, looking at the results of R101-R103, a much lower cost ratios still improve the objective function value. In R101, a cost ratio of 1:3 between a vehicle and additional workers still improves the objective function value. Reducing the number of trucks and drivers by 1 while adding 3 additional workers moves along the pareto front at a 1:3 ratio. However, reduced milage leads to an improved objective function value. Based upon the same reasoning, R102 has an improved objective function value up to a cost ratio of 1:2.5. In the case of R103 the cost can be considered to

[6] The values for the best known VRPTW solutions are taken from `http://www.sintef.no/Projectweb/TOP/VRPTW/Solomon-benchmark/100-customers/` at the time of this writing.

Instance	VRPTW		VRPTWMS			
	Vehicles	Distance	Clusters	Vehicles	Workers	Distance
R101	19	1650.80	82	18	22	1624.03
R102	17	1486.12	77	15	22	1453.92
R103	13	1292.68	84	12	15	1306.84

Table 5: Comparison to three classic VRPTW instances

be improved up to a ratio of almost 1:2 (given the slightly worse milage of the VRPTWMS solution). The result values for the improved instances are listed in Table 5.

Even though the cost ratio of 1:10 defined in the first publication about the VRPTWMS is limited to low-wage countries, the presented results encourage applicability in higher wage countries as well. This particularly holds since the fixed cost of a truck also the includes wage difference between an eligible driver and an unskilled worker.

5 Conclusions and Future Work

This paper makes a series of important contributions. These are the proposition of the first metaheuristic for the VRPTWMS that guides a clustering heuristic with good routing results. Furthermore, it is shown how to learn from past routes even though the topology of the routing problem changes in every iteration. However, the most important contribution is to show that given the right circumstances (speed and price ratio between the two modes of transportation) there is a strong potential in reducing cost and environmental impact at the same time. This paper not only suggests that it is possible to reduce the environmental impact of goods transportation in congested urban areas. It also demonstrates a strong incentive for carriers to do so since cost reductions are in their own economic interest.

Given both the novelty and the relevance of the VRPTWMS, a lot remains to be done. The more important areas to be researched include a series of local search operators for the clustering. For example, the current cluster construction heuristic fixes a cluster's parking space at the time the seed is selected. However, when customers are added, it is usually beneficial to change a cluster's parking location to the one closest to the center of the customers. Furthermore, the scheduling algorithm inside the clusters offers a potential for improvement. Another very interesting area for future research is to drop the assumption that loading zones are always available upon truck arrival and create a simulation-driven approach to account for this problem.

Finally, the vehicle routing community ought to spend more time researching multi-modal transportation in general. The research paper at hand already clearly demonstrates the potential of this approach in providing both financial savings as well as ecological benefits. However, besides lacking local search operations, the presented algorithm does not offer the decades of insight that were put into pure VRPTW algorithms. Still, even in this early stage of research it was possible to demonstrate gains by applying a multi-modal approach. This is why other alternative models should be investigated as well - for example with sub-tours inside the clusters instead of direct delivery between the vehicle and the customers. Furthermore, the option of dropping off workers and picking them up later on should be investigated. This could even be done at a different parking location. The idea behind the said research paths is to increase the number of real-world applications of multi-modal transportation.

References

[Bräysy and Gendreau, 2005a] Bräysy, O. and Gendreau, M. (2005a). Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, 39:104–118.

[Bräysy and Gendreau, 2005b] Bräysy, O. and Gendreau, M. (2005b). Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39:119–139.

[Colorni et al., 1991] Colorni, A., Dorigo, M., and Maniezzo, V. (1991). Distributed optimization by ant colonies. In *Actes de la première conférence européenne sur la vie artificielle*, pages 134–142. Elsevier Publishing.

[Crainic et al., 2009] Crainic, T. G., Ricciardi, N., and Storchi, G. (2009). Models for Evaluating and Planning City Logistics Systems. *Transportation Science*, 43(4):432–454.

[Dorigo and Stützle, 2004] Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization.* MIT Press, Cambridge, MA.

[Doulabi and Seifi, 2013] Doulabi, S. H. H. and Seifi, A. (2013). Lower and upper bounds for location-arc routing problems with vehicle capacity constraints. *European Journal of Operational Research*, 224(1):189–208.

[Escobar et al., 2013] Escobar, J. W., Linfati, R., and Toth, P. (2013). A Two-phase Hybrid Heuristic Algorithm for the Capacitated Location-routing Problem. *Comput. Oper. Res.*, 40(1):70–79.

[Feo and Resende, 1995] Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.

[Golden and Wasil, 1987] Golden, B. L. and Wasil, E. A. (1987). Computerized vehicle routing in the soft drink industry. *Operations Research*, 35(1):6.

[Hemmelmayr et al., 2012] Hemmelmayr, V. C., Cordeau, J.-F., and Crainic, T. G. (2012). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, 39(12):3215 – 3228.

[Hetland, 2010] Hetland, M. L. (2010). *Python Algorithms - Mastering Basic Algorithms in the Python Language.* Apress.

[Hunter, 2007] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

[Junior et al., 2012] Junior, L. S., Nedjah, N., and de Macedo Mourelle, L. (2012). Aco-based algorithms for search and optimization of routes in noc platform. *Journal of Universal Computer Science*, 18(7):917–936.

[Lin et al., 2011] Lin, S.-W., Yu, V. F., and Lu, C.-C. (2011). A simulated annealing heuristic for the truck and trailer routing problem with time windows. *Expert Syst. Appl.*, 38(12):15244–15252.

[Pureza et al., 2011] Pureza, V., Morabito, R., and Reimann, M. (2011). Vehicle routing with multiple deliverymen: Modeling and heuristic approaches for the VRPTW. *European Journal of Operational Research*, 218(3):636–647.

[Salhi and Nagy, 1999] Salhi, S. and Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, 50(10):1034–1042.

[Senarclens de Grancy and Reimann, 2014] Senarclens de Grancy, G. and Reimann, M. (2014). Vehicle routing problems with time windows and multiple service workers: a systematic comparison between ACO and GRASP. *Central European Journal of Operations Research*. To appear.

[Senarclens de Grancy and Reimann, 2015] Senarclens de Grancy, G. and Reimann, M. (2015). Evaluating two new heuristics for constructing customer clusters in a VRPTW with multiple service workers. *Central European Journal of Operations Research*, 23(2):479–500.

[Solomon, 1987] Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.

[Transport for London, 2014] Transport for London (2014). London streets performance report: quarter 3 2013/14. Technical report, Transport for London. Available online at `https://www.tfl.gov.uk/cdn/static/cms/documents/street-performance-report-quarter3-2013-2014.pdf` (accessed 2014-12-15).

[Villegas et al., 2011] Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., and Velasco, N. (2011). A GRASP with evolutionary path relinking for the truck and trailer routing problem. *Computers & OR*, 38(9):1319–1334.

[Weiss, 2006] Weiss, M. A. (2006). *Data structures and algorithm analysis in C++*. Pearson Education.