Published in *Journal of the Operational Research Society*, 2009, vol. 60, no. 7, pp. 934-943, which should be cited to refer to this work.
DOI: 10.1057/palgrave.jors.2602633

POPMUSIC for a Real World Large Scale Vehicle Routing Problem with Time Windows

Alexander Ostertag†, Karl F. Doerner†,
Richard F. Hartl†, Éric D. Taillard‡,
Philippe Waelti‡

† Department of Business Administration, University of Vienna
Bruenner Strasse 72, 1210 Vienna, Austria
{Alexander.Ostertag,Karl.Doerner,Richard.Hartl}@univie.ac.at

‡ SiM-TIC Institute, University of Applied Science,
Rue Galilée 15, CH-1400 Yverdon-Les-Bains, Switzerland
{Eric.Taillard,Philippe.Waelti}@heig-vd.ch

Abstract

This paper presents a heuristic approach based on the POPMUSIC framework for a large scale Multi Depot Vehicle Routing Problem with Time Windows (MDVRPTW) derived from real world data. Popmusic is a very powerful tool for tackling large problem instances. A Memetic Algorithm (MA) is used as an optimiser in the Popmusic framework. It is shown that a population based search combined with decomposition strategies is a very efficient and flexible tool to tackle real world problems with regards to solution quality as well as runtime.

Keywords: Vehicle Routing, Heuristics, Problem Decomposition

Introduction

Modern carrier fleet operators face the challenge of optimising their vehicle fleets and routing operations to stay competitive. Furthermore in real world situations more than one depot is usually used, from where drivers can start operating. This leads to combinatorial problems for which good solutions are very hard to find and practically impossible to solve exactly. Metaheuristics are used to face

this kind of problems in order to find high quality solutions with a reasonable computational effort. Even though developed metaheuristics scale well with the problem size, a borderline may exist when problems get too large to be solved efficiently. The practical solving limit of many algorithms is few hundreds of customers that have to be serviced with a fleet of few dozens of vehicles. So, real-life problem instances which can involve several thousands of customers clearly cross this borderline. The recent work of Mester & Bräysy (2007, 2005) who use active guided evolution strategies as well as the Variable Neighbourhood Search (VNS) approach by Kytöjoki et al. (2007) shows that even large instances may be solved in an efficient way. The 2-phase hybrid metaheuristic developed by Homberger & Gehring (2005) also proved to successfully solve problems from small sizes up to 1000 customers. Another possible approach is to use decomposition strategies like the POPMUSIC framework by Taillard & Voss (2001) that try to overcome size restrictions, by intelligently splitting the problem into sub problems and solving them separately. Decomposition strategies were recently also successfully applied to large scale real world problems by Flaberg et al. (2006). They considered the problem of newspaper delivery in the city of Oslo.

The present paper proposes such a strategy based on POPMUSIC that uses a Memetic Algorithm (MA) as optimiser for sub problems. These are small instances of multi depot Vehicle routing problem with time windows (MDVRPTW). Very few papers tackle the MDVRPTW. Let us quote the work of Cordeau et al. (2001) which uses a Tabu Search heuristic to solve MDVRPTW instances as well as a wide variety of related problems. A more recent work is the VNS by Polacek et al. (2004). Hybrid genetic algorithms have already been implemented in the context of VRPTW, for instance by Prins (2004), Berger & Barkaoui (2004). Memetic algorithms such as the one implemented as optimiser in our POPMUSIC framework are special cases of evolutionary algorithms often called genetic hybrids. So, it is interesting to provide some insights to generalisation of the VRPTW to the Multi Depots case.

The remainder of the paper is organised as follows. The first section describes the problem in detail. The next section describes each part of the Memetic Algorithm implemented as basic optimiser for the POPMUSIC framework. The general POPMUSIC framework as well as its adaptation for the MDVRPTW are presented in the next but one section. The results for three different strategies to solve the problem are then presented and discussed in subsequent section. A final conclusion and possible directions in which further research on POPMUSIC algorithms in the field of large scale MDVRPTW are given in the last section.

Problem Description

The real world problem considered in this paper was transformed so that it can be tackled as a MMDVRPTW. Compared to the well known Vehicle Routing Problem with Time Windows (VRPTW), the MDVRPTW is extended by having more than one depot with different locations and associated vehicle fleets. The MDVRPTW is defined on a complete graph $G = (V, A)$ where $V = \{v_1, ..., v_m, v_{m+1}, ..., v_{m+n}\}$ is the vertex set and $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ is the arc set. The n customers are represented by vertices v_{m+1} to v_{m+n}, while v_1 to v_m stand for the m depots. Several weights are associated to each vertex $v_i \in V, i = m+1, ..., m+n$. These weights represent the demands d_i, the service times s_i, as well as the time windows $[e_i, l_i]$ which are defined by the earliest e_i and latest l_i possible start times for the service. These time windows also apply to the depots ($i = 1, ..., m$) where they define the opening hours of the depots. Each arc (v_i, v_j) is associated with a non-negative travel time or cost. A vehicle fleet consisting of a total of K vehicles is globally assigned to the m depots. The fleet is homogeneous and each vehicle is characterised by a non-negative capacity D and a non-negative maximum route duration T. Finally, the distribution of the vehicles over the depots is defined as input data. The aim is to build K vehicle routes, each route starting from a depot and returning back to the same depot, so that each customer i belongs to one route exactly and is serviced during its corresponding time window $[e_i, l_i]$. Each vehicle route has to satisfy the additional constraints of the maximal allowed tour length T and vehicle capacity D. The objective considered in this paper is to minimise the total distance travelled by all vehicles.

The resulting MDVRPTW problem is a generalisation of the VRPTW that is known to be NP-hard. State-of-the art exact algorithms cannot solve problem instances with more than few dozens of customers. So, the only practical issue is to heuristically solve real-life problem instances. An overview about the VRPTW in the fields of heuristics and more sophisticated metaheuristics is presented in (Bräysy & Gendreau, 2005a,b).

Memetic Algorithm

General description

A typical MA is based on the structure of a Genetic Algorithm, but is further enriched in a way that it can exploit all available knowledge about the prob-

lem that is under consideration (for an overview see Moscato & Cotta (2003)). A Memetic Algorithm was developed following the basic principles of Genetic Algorithms (Reeves, 2003, Goldberg, 1989). A population of solutions is initially generated and maintained, while a reproductive process is applied. The offsprings are generated in a way that they incorporate desired features and are used to update the existing population. The idea of MAs is incorporated so that individual offsprings as well as parts of the population can be improved through a stochastic local search. The basic concept of the MA is shown in Figure 1.

Initialisation

The initial population is created through a modified I1 insertion heuristic (Solomon, 1987). In order to generate an initial population of size *popsize* with very different initial solutions, the I1 heuristic is modified with a stochastic insertion criterion. The modified I1 heuristic is composed of two stages. In a first clustering stage, all customers are assigned to their geographically closest depot. We are aware that there exist more sophisticated methods for assigning customers to depots (Salhi & Sari, 1997), however we opted for a simpler approach because we didn't want to focus on construction algorithms. In a second routing stage, K empty routes are generated consisting of about K/m routes for each depot. Each customer is then tentatively inserted into a route, and the resulting insertion costs are saved in a sorted list. After all customers have been tentatively inserted, the customer to be inserted is selected randomly from the first three entries of this list with probability 0.5 for the first entry and 0.25 for the remaining two entries. A route is considered complete when no more customers can be feasibly inserted. The customers eventually not assigned are inserted into the routes at the places where they generate the smallest violations. If all routes have been completed or all customers have been assigned, the heuristic stops. Afterwards, the 3-opt operator is applied to each route.

Selection

The fitness evaluation function of a solution S follows the implementation of Cordeau et al. (2001) and Polacek et al. (2004). The total travel time of the routes is denoted by $c(S)$. The values $q(S)$, $t(S)$ and $w(S)$ respectively denotes the total violation of load, duration and time window constraints. The arrival time a_i at each customer i is calculated and an arrival after the end of the time window $a_i > l_i$ is penalized while an arrival before the start of the time window $a_i < e_i$ is allowed but generates a waiting time. Each route is then

checked for violations with respect to D and T as well as the total violation of the time window constraints $\sum_{i=1}^{n} max(0, a_i - l_i)$. The fitness function is defined by $f(S) = c(S) + \alpha q(S) + \beta t(S) + \gamma w(S)$ where α, β and γ are positive weights.

The selection procedure follows the idea of binary tournament, where two solutions S_1 and S_2 are randomly selected from the population *pop* and are evaluated by the fitness function. Only the individual with the lower value is chosen for recombination. Then the selection procedure restarts for the selection of the second recombination partner.

Recombination

Recombination operators are used in evolutionary algorithms to simulate the reproductive process. This usually results in selecting two solutions and recombining them so that the offsprings hopefully inherit the good attributes of both parents. Sophisticated cross-over operators like those presented in (Prins, 2004) are difficult and time expensive to implement due to the large problem size as well as the extensions like time windows and multiple depots.

So, we apply a simple route based two-point crossover operator (see Bräysy & Gendreau, 2005b) that is not computationally expensive. This operator creates two offsprings O_1 and O_2 by combining, one at a time, b pair of routes, R_1 of parent solution S_1 with R_2 of parent solution S_2. Only the best offspring is kept. b is randomly drawn between one and the maximum possible combination of routes, with a bias towards small values. The probabilities for the amount of routes selected are 0.99 for one pair of routes, 0.0075 for two pairs and the remaining probability is equally distributed between three and the maximum number of pairs. After the pairs of routes are chosen, one route is randomly cut into three sequences. The length of the middle sequence is at most the length of the smallest route diminished by the position of the first customer of the middle sequence. The same sequence length is used for R_1 and R_2. Also the same starting positions are used so that time window violations can be anticipatively minimised. After the exchange of the middle sequences, the solution is checked for missing or duplicated customers. The latter are erased out of the routes where they appeared before recombination, and missing customers are inserted at the cheapest possible position using a I1 insertion heuristic proposed by Solomon (1987).

The crossover operator is illustrated in the following example:

$$R_1 \ (1\ 2 \mid 3\ 4\ 5 \mid 6\ 7)$$

$$R_2\ (5\ 2 \mid 3\ 1\ 4 \mid 9\ 6\ 7\ 8)$$

Where R_1, R_2 are the chosen routes from S_1 and S_2 that produce the following routes of the offsprings O_1 and O_2 after swapping the middle sequence.

$$R_1\ (1\ 2 \mid 3\ 1\ 4 \mid 6\ 7)$$
$$R_2\ (5\ 2 \mid 3\ 4\ 5 \mid 9\ 6\ 7\ 8)$$

The new solutions need to be repaired in a way that no double or missing customers exist. The final routes of the offsprings can then look like this:

$$R_1\ (5\ 2 \mid 3\ 1\ 4 \mid 6\ 7)$$
$$R_2\ (2\ 1 \mid 3\ 4\ 5 \mid 9\ 6\ 7\ 8)$$

The best of the two offsprings is then used to update the population.

Population Management

After the generation of the initial population, the algorithm starts with the selection of individual solutions for recombination. After the recombination operator has generated an offspring, *pop* is updated in a steady state fashion (Whitley, 1987), which means that new solutions are allowed to enter *pop* if they are fitter than the worst solution in *pop*. The population is implemented as an array of chromosomes sorted by their fitness values. To save computational time, fitness values for whole solutions as well as for individual routes are stored in the chromosomes, and need only to be re-evaluated when a change in the chromosome occurs. The detection of clones is based on this stored fitness value, where identical values are handled as identical solutions. Additionally the best feasible solution is saved. In the case that no feasible solution exists in the population at the end of the calculation, this solution then represents the final solution.

Mutation

A Stochastic Local Search procedure based on Variable Neighbourhood Search (VNS) (Hansen & Mladenović, 1999) is applied to modify existing solutions as well as newly generated ones. The goal of this procedure is to better explore the search space as well as to overcome local optima. It follows the ideas of the work done by Polacek et al. (2004) and uses CROSS neighbourhoods (Taillard et al., 1997) in the shaking phase. CROSS swaps two sequences of customers

belonging to different routes. This leads to the possibility of reaching more distant neighbourhoods. The maximum allowed sequence length is fixed as well as the number of depots involved in a move. The 12 different neighbourhoods used in our VNS frame ($\kappa = 1, \ldots, 12$) are shown in Table 1 where C_k denotes the number of customers assigned to route k. After the swapping of the sequences, a 3-opt operator, that is restricted to sequence length sl, is used to bring the newly generated routes into local optimum.

The Stochastic Local Search Procedure is applied to each newly generated offspring with probability p_1 and to each solution in *pop* with probability p_2. The VNS stopping criterion is set to a small amount of iterations it_{vns}, to be relatively inexpensive with regard to computational time.

POPMUSIC algorithm

POPMUSIC general description

The POPMUSIC framework was proposed by Taillard & Voss (2001) for dealing with large problem size. The idea is to decompose a given solution S into p parts $s_1, ..., s_p$. Once these parts are identified, some of them are aggregated to build a sub problem. The latter is tentatively improved with an optimiser. If parts and sub problem are well defined, to every improvement of a sub problem corresponds to an improvement of the whole solution S. The process is then repeated until all the sub problems that can be built with parts have been optimised. The basic POPMUSIC framework is described in Figure 2.

Obtaining an initial solution by clustering

For creating the initial solution, the customers are first clustered. Each cluster of customers builds a smaller MDVRPTW. The partition of customers is obtained by solving a relaxation of a capacitated p-Median problem (see Hakimi, 1965, Taillard, 2003, Waelti et al., 2002). To build clusters of customers for which the overall demand is balanced, the distances between customers are modified with Lagrangian multipliers, one for each cluster. Initially, all multipliers are set to 0 and a standard p-Median problem is solved. Then, for each cluster c the overall demand Q_c is computed and is compared to a global capacity V_c that is allowed for this cluster. The quantity V_c corresponds to the capacity of the vehicles that are allotted to cluster c. If $V_c < Q_c$, it is not possible to deliver all customers allotted to cluster c. In this case, the Lagrangian multiplier λ_c associated to

cluster c is increased by a quantity that depends on Q_c/V_c. The true distance d_{ij} between customers i and j is modified, resulting in a new distance measure Π_{ij} between customers:

$$\Pi_{ij} = d_{ij} + \lambda_c \cdot d_i$$

Since the new distance mixes length and demand units, the Lagrangian coefficients λ_c must be multiplied by a factor that balances the influence of both units. Once new distances are computed, the problem is decomposed again with the p-Median solver and the process repeats until a feasible decomposition is found or an iteration limit is reached.

At this stage of the process, it is not very important to get a decomposition that strictly respects the capacity constraints, since the routes have still to be built and customers can be moved from one route to another. The main advantage of this relaxation is that constraints other than capacity (pick-up, time windows) can be added while using a common p-Median solver. The main layout of the algorithm is presented in Figure 3.

Better balancing customers between clusters

In the initial phase, customers are assigned to clusters $s_1, ..., s_p$ using the p-Median decomposition procedure. A typical solution of the p-Median decomposition can be seen in Figure 4 for an example of our real world problem instances. The city of Vienna is shown expanded in Figure 5. It can be seen that most of the customers are located in a small geographic region in the centre. As a result, the solution of the p-Median decomposition procedure may consist of clusters with a high amount of customers as well as clusters with only a handful of customers. Since this feature may not be a good starting point for further evaluation, a preprocessing procedure, as described by Figure 6, tries to level out the number of customers inside clusters.

In detail, routes are build by it_{ini} iterations of the sub problem optimiser to generate the first solution for each cluster. After the optimiser has built routes for each cluster, clusters that exceed a certain amount of customers $csize$, are split in the following way. The centre of gravity (Reimann et al., 2004) is calculated for each route in the cluster to represent its aggregated customers. The Sweep algorithm (Gillet & Miller, 1974) is then applied and splits the cluster by the centres of its routes. The starting point is randomly defined and the Sweep algorithm then sequentially adds routes until $csize$ customers are reached. When this limit is reached the cluster is split and the routes are

removed from the cluster and inserted into a new cluster until no more new clusters can be generated. The remaining clusters smaller then *csize* are then checked if they could be merged to form new clusters with a size smaller than *csize*. Clusters are then merged by a greedy heuristic that uses the distance of the centres of gravity of each cluster.

Note that the number of available vehicle in the real world problem tackeled is far sufficient to perform the deliveries. Also, the time windows of customers are wide (basically: morning, afternoon or whole day). So, no unfeasible solutions (customers not delivered) have to be managed.

POPMUSIC customisation

In order to implement POPMUSIC, it is necessary to specify its principal components in relation to the problem at hand. For the MDVRPTW, a part is defined as a route. The proximity measure (*relation*) between routes is defined as the distance between the centres of gravity of the entities (routes, clusters).

As mentioned previously, for creating the initial solution, the customers are first clustered by solving a capacitated p-Median problem and are then balanced as described in in the previous section. Each cluster is considered as a small MDVRPTW. Initially, each route only visits customers that belong to the same cluster. Instead of selecting a single part as seed for initiating the creation of sub problems (as in a regular POPMUSIC framework) all the routes of a cluster are considered as a seed part. The centre of gravity of the entities composing the cluster is computed. Then, the seed part is extended by adding routes until it contains r routes. Routes are chosen by their proximity to the cluster they will be added to.

A sub problem is therefore a subset of routes that can be treated as a small, independent MDVRPTW. All resulting sub problems are then optimised by a MA that stops either if a computational time or an iteration limit is reached.

Computational Analysis

The problem considered is a large real world problem of an Austrian carrier company that owns two depots in or near Vienna. The depots serve around 4000 different customers by a total fleet of 160 vehicles that are split equally between the depots. Two weeks, consisting of 10 workdays, were chosen for evaluation with customers ranging between 743 and 1848 per day. Note that

some customers are visited on several days. To present more concise results, the problem instances are merged into three classes (S,M,L) according to their size. Table 2 shows the three classes and the assignment of the individual days to each class.

The parameters used for all calculations are shown in Table 3. Further we decided to set *csize* to 75 customers as the MA provides good solutions in a reasonable amount of time for this problem size. In order to find appropriate *csize* the MA was tested on standardised instances used in (Cordeau et al., 2001, Polacek et al., 2004). It turns out that the MA could find the best known solutions for all instances up to size 75 except one (where it only deviates by 0.08%) within reasonable computation time. Because no related work was done on the same problem and no data for comparison exists, we set up three strategies to tackle the large real world problem.

Strategy I (no decomposition) is the most basic one, which tries to solve the problem as a whole by the described MA until a certain amount of time is elapsed .

Strategy II (fixed decomposition) is based on the initial clustering by the p-Median algorithm. In this case all clusters where treated as individual problems and solved by the MA respectively. The resulting problems vary in size and therefore the time t_i allowed for each problem s_i is dependent on the square of its size C_{s_i} in relation to the total problem size C_{s_n} and the maximum time t_{max} allowed . It is calculated as follows; $t_i = t_{max} \cdot (C_{s_i})^2 / \sum_{n=0}^{p} (C_{s_n})^2$. The whole solution was then generated by simply merging the solutions of the single clusters.

Strategy III (POPMUSIC) was executed with different parameter settings. The POPMUSIC Algorithm was implemented with longer runtime in $IIIa$ and shorter runtime in $IIIb$. Furthermore the optimiser was given less time in $IIIb$ to improve the sub problems than in $IIIa$ to put some emphasis on faster descent of the solution quality. Strategies I, II and $IIIa$ were given $t_{max} = 28800$ seconds for each individual run, with 10 runs each. Furthermore eight starting clusterings by the p-Median procedure where evaluated for Strategy II and III ranging from 16 to 80 clusters. Table 6 gives an overview about the different parameters retained for each strategy.

The sum of the length of the routes is provided in Table 4 for each strategy. Table 5 provides the percentage of deviation of each strategy relative to Strategy I. All runs where executed on Pentium Dual 3.2 GHz with 1 GB Ram, but they had to share the 1 GB of Ram since both processors were used at the same time for different instances.

Solving the problem without decomposition resulted in the worst solution quality. The decomposition of the problem into intelligently chosen parts (Strategy II) is a very simple and easy to implement. This strategy is able to provide significantly better results; about -13% compared to the most basic strategy. Detailed computational results are given in the Appendix.

Looking at the average results of strategy $IIIa$ compared to strategy I, nearly 20% improvement in solution quality shows that the POPMUSIC framework can solve large scale problems efficiently. This fact is underlined when looking at the results of the short POPMUSIC results $IIIb$. Even though the algorithm has only about 6% of the time available it can beat the simple Strategy I by nearly 14% and the decomposition Strategy II by around 0.75%.

The POPMUSIC strategy that uses a MA shows that large scale problems derived from real world data can be solved more efficiently than by simple approaches that focus on the problem as a whole. The set-up of Strategy III with short runtimes and reduced number of iterations it_{ini} for the MA (Strategy $IIIb$) also shows that relatively satisfying solutions can be found in a reasonable amount of time. The average objective values as well as a 95% confidence interval for Strategies I, $IIIa$ and $IIIb$ over the runtime are shown in Figure 7 for day 6. Strategy II is not represented in these tables because clusters are solved sequentially.

Conclusion

We have shown that a decomposition strategy such as a POPMUSIC algorithm that uses a MA as optimiser can be very efficient in solving large scale MDVRPTW. We achieved an average improvement of about 20% over all considered real world instances compared to the use of the same optimiser without decomposition. The POPMUSIC framework is therefore able to automatically and efficiently reassign "borderline customers", i.e. customers that are about equally distant from both depots. Furthermore, the framework can be flexibly adapted for a faster solution finding process. The POPMUSIC framework is easy to develop for large VRP instances.

Nevertheless, further research may focus on different relatedness approaches, that are for example based on a route - route (see Semet & Rochat, 1994) or even customer - customer (see Shaw, 1998) basis instead of a route - cluster basis. More emphasis has also to be put on developing intelligent ways to maintain or create new populations which can be used for further optimisation. The algorithm presented also provides a good starting point to extend the whole

problem to a Location Routing Problem, where the number and locations of the depots are not fixed but may be chosen independently.

Acknowledgment

Support from the Austrian Science Fund (FWF) by grant #L362-N15 (Translational Research) is gratefully acknowledged. Support form the Strategic Research Funds of the University of Applied Sciences of Western Switzerland (DeProLo project, grant 12804) is also gratefully acknowledged.

References

Berger, J. & Barkaoui, M. (2004). A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, *31*, 2037–2053.

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

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

Cordeau, J.-F., Laporte, G., & Mercier, A. (2001). A unified tabu search heuristic for the vehicle routing problems with time windows. *Journal of the Operation Research Society*, *52*, 928–936.

Flaberg, T., Hasle, G., Kloster, O., & Riise, A. (2006). Towards solving huge-scale vehicle routing problems for household type applications. Workshop presentation in Network Optimization Workshop Saint-Remy de Provence, France August 2006.

Gillet, B. & Miller, L. (1974). A heuristic algorithm for the dispatch problem. *Operations Research*, *22*, 340–349.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Pub. Co.

Hakimi, S. L. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research*, *13*, 462–475.

Hansen, P. & Mladenović, N. (1999). An introduction to variable neighborhood search. In Voss, S., Martello, S., Osman, I. H., & Roucairol, C. (Eds.), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, (pp. 433–458). Kluwer Academic Publishers.

Homberger, J. & Gehring, H. (2005). A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research*, *162*, 220–238.

Kytöjoki, J., Nuortio, T., Bräysy, O., & Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, *34*, 2743–2757.

Mester, D. & Bräysy, O. (2005). Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & Operations Research*, *32*, 1593–1614.

Mester, D. & Bräysy, O. (2007). Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research*, *34*, 2964–2975.

Moscato, P. & Cotta, C. (2003). A gentle introduction to memetic algorithms. In Glover, F. & Kochenberger, G. (Eds.), *Handbook of Metaheuristics*, (pp. 105–144). Springer.

Polacek, M., Hartl, R. F., Doerner, K., & Reimann, M. (2004). A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of Heuristics*, *10*, 613–627.

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

Reeves, C. (2003). Genetic algorithms. In Glover, F. & Kochenberger, G. (Eds.), *Handbook of Metaheuristics*, (pp. 55–82). Springer.

Reimann, M., Doerner, K., & Hartl, R. F. (2004). D-ants: Saving based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, *31*, 563–591.

Salhi, S. & Sari, M. (1997). A multi-level composite heuristic for the multi-depot vehicle fleet mix problem. *European Journal of Operational Research*, *103*, 95–112.

Semet, F. & Rochat, Y. (1994). A tabu search approach for delivering pet food and flour in Switzerland. *Journal of the Operational Research Society*, *45*, 1233–1246.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. Technical report, ILOG SA.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, *32*(2), 254–265.

Taillard, E. D. (2003). Heuristic methods for large centroid clustering problems. *Journal of Heuristics*, *9*, 51–73.

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

Taillard, E. D. & Voss, S. (2001). Popmusic: Partial optimization metaheuristic under special intensification conditions. In Ribeiro, C. & Hansen, P. (Eds.), *Essays and surveys in metaheuristics*, (pp. 613–629). Kluwer Academic Publishers.

Waelti, P., Taillard, E. D., & Mautor, T. (2002). Cueillir du mimausa en écoutant de la popmusic. Technical report, MiS-TIC Institute, HEIG-Vd.

Whitley, D. (1987). Using reproductive evaluation to improve genetic search and heuristic discovery. In Grefenstette, J. J. (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms and their Applications*, (pp. 108–115). Lawrence Erlbaum Associates.

Appendix

Table 7 shows the results of Strategy I for each day that were found after 10 runs. According to these figures Table 8 was created, that shows the combined results for each of the three classes. Table 9 shows the results for solving each cluster individually. It can be seen that a small number of clusters (20, 16, 16) provides the best results. This is in direct contrast to the results of Strategy III shown in Table 10 and 11, where a larger starting number of (40, 80, 53) provides better results than the other options. As conclusion one can assume that more clusters result in a better distribution of customers in the clusters. Big chunks of customers like those occurring in densely populated areas are already divided. This allows POPMUSIC algorithm to reach better solutions in the same amount of time. Both tables 10 and 11 show that the maximum difference in solution quality when using different p-Median starting solutions does not exceed 1%. The best solution values found so far are provided in Table 12. All these best solutions where found by Strategy $IIIa$ with the initial clustering. It can be seen that most of the best solutions were found in the higher spectrum of the number of clusters, with some exceptions like days 3 and 6.

List of Figures

Figure 1: Basic Steps of the Memetic Algorithm

1. *Initialisation* Repeat $popsize$ times
   (a) Generate a solution with modified I1 insertion heuristic (Solomon, 1987)
   (b) Improve solution with 3-opt local search until reaching local optimum
   (c) Insert solution in *pop*
2. Repeat until Stopping Criterion is met
   (a) *Selection* Select two solutions from *pop* for recombination
   (b) *Recombination* Generate offsprings O_1, O_2 through route based two-point Crossover Procedure
   (c) *Improvement Step*
      i. With probability p_1, improve offspring O_1 and O_2 with Stochastic Local Search
      ii. With probability p_2, improve each solution of *pop* with Stochastic Local Search
   (d) *Population Management*
      i. If no other solution from *pop* has the same solution quality, insert the best offspring into *pop*
      ii. If an offspring was inserted, erase worst solution from *pop*
   (e) *Stopping Criterion* Stop algorithm when maximum allowed time or iterations is reached

Figure 2: Basic POPMUSIC framework

1. Input: Solution S composed of parts $s_1, ..., s_p$, parameter r
2. Set $A \leftarrow \emptyset$
3. While $A \neq \{s_1, ..., s_p\}$ repeat
   (a) Select seed part $s_i \notin A$
   (b) Create a sub-problem R_i composed of the r parts $s_{i_1}, ..., s_{i_r}$ most related to s_i
   (c) Optimise R_i
   (d) If R_i has been improved
      i. Update S (and corresponding parts)
      ii. Set $A = A \setminus \{s_{i_1}, ..., s_{i_r}\}$
   (e) Else
      i. Set $A \leftarrow A \cup \{s_i\}$

Figure 3: p-Median decomposition algorithm

1. Input: MDVRPTW, number of clusters p, iteration limit it_{dec}
2. Build p-Median problem according to MDVRPTW customers
3. Allocate K/p vehicles to each cluster c and compute maximum capacity V_c
4. Set $Q_c \leftarrow \infty \ \ \forall c$, $\lambda_c \leftarrow 0$, $i \leftarrow 0$
5. Repeat while $(Q_c > V_c \ \ \forall c)$ and $(i \leq it_{dec})$
   (a) Solve p-Median problem with modified distances Π_{ij} (see Taillard, 2003)
   (b) Compute overall capacity Q_c of each cluster c
   (c) Update λ_c coefficients according to capacity constraint violation
   (d) Set $i \leftarrow i + 1$

Figure 4: p-Median decomposition

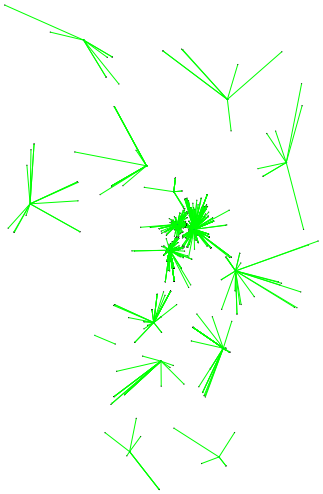


Figure 5: Zoom in on p-Median clusters

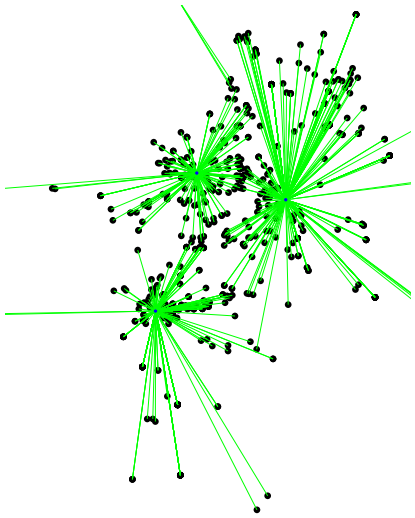
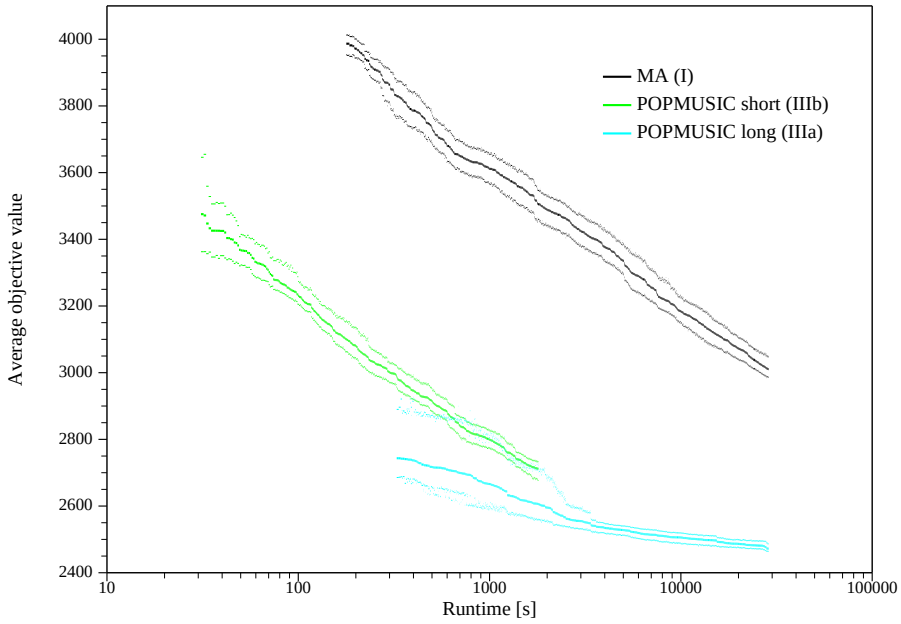


Figure 6: POPMUSIC Initialisation Phase

1. Assign customers to clusters by p-Median decomposition
2. Run it_{ini} iterations of the MA on each cluster to build initial routes
3. Split clusters that have more than $csize$ customers
   (a) Repeat until all parts $s_1, ..., s_p$ are $\leq csize$
4. Merge clusters that are smaller than $csize$ customers
   (a) Repeat until no more parts $s_1, ..., s_p$ can be merged

Figure 7: Average objective values and confidence intervals over runtime for day 6



List of Tables

Table 1: Set of neighbourhood structures

κ	Depots	Sequence length
1	1	$\min(1, C_k)$
2	1	$\min(2, C_k)$
3	1	$\min(3, C_k)$
4	1	$\min(4, C_k)$
5	1	$\min(5, C_k)$
6	1	C_k
7	2	$\min(1, C_k)$
8	2	$\min(2, C_k)$
9	2	$\min(3, C_k)$
10	2	$\min(4, C_k)$
11	2	$\min(5, C_k)$
12	2	C_k

Table 2: Problems size and class definitions

Day	1	2	3	4	5	6	7	8	9	10
Size	1201	1180	1284	1305	1175	743	889	1095	1848	1709
Class	M	M	M	M	M	S	S	S	L	L

Table 3: Global Parameters Settings

Parameter	Value
Load violation factor α	100
Duration violation factor β	100
Time window violation factor γ	100
Size of the population $popsize$	10
Mutation probability p_1	0.1
Mutation probability p_2	0.01
3-opt sequence length sl	3
Iterations for VNS it_{vns}	10
ideal cluster size $csize$	75
Number of related routes r	1

Table 4: Sum of route length for each strategy and for each problem class

Class	Strategy	Best	Mean	Worst	Stdv
S	I	10,768.38	11,157.48	11,577.30	93.82
S	II	10,476.20	10,569.07	10,679.76	33.95
S	IIIa	9,089.32	9,227.38	9,397.18	39.62
S	IIIb	9,687.67	9,966.59	10,289.52	112.26
M	I	24,608.03	25,428.30	26,260.06	243.88
M	II	21,694.29	21,923.49	22,110.66	74.94
M	IIIa	20,156.41	20,429.33	20,713.18	60.57
M	IIIb	21,450.20	21,800.15	22,237.30	76.30
L	I	14,131.32	14,532.08	14,942.90	127.30
L	II	11,970.13	12,067.94	12,162.91	37.03
L	IIIa	11,505.03	11,607.31	11,826.82	80.29
L	IIIb	12,297.36	12,473.14	12,607.14	62.68

Table 5: Comparison of the strategies

Class	RPD I/II	RPD I/IIIa	RPD I/IIIb
S–Small	-5.27%	-17.30%	-10.67%
M–Medium	-13.78%	-19.66%	-14.27%
L–Large	-16.96%	-20.13%	-14.17%
Average	-12.83%	-19.28%	-13.46%

Table 6: Parameter settings for the different strategies

	I	II	IIIa	IIIb
it_{pop}	∞	t_i	10	1
it_{ini}	-	-	200	1
t_{max}	28800	28800	28800	1800
Number of clusters	1	16	80	80

Table 7: Results of Strategy I for each day

no decomposition ($10 \cdot 10 \cdot 8h = 800h$)				
Day	Best	Mean	Worst	Stdv
1	4,560.34	4,747.76	4,938.84	106.30
2	4,829.90	4,975.89	5,138.89	102.01
3	4,860.68	5,070.26	5,315.12	125.95
4	5,348.45	5,490.42	5,628.42	97.85
5	5,008.66	5,143.97	5,238.79	73.10
6	2,872.69	3,046.42	3,253.43	119.44
7	4,030.47	4,103.11	4,159.57	38.54
8	3,865.22	4,007.95	4,164.30	99.37
9	6,861.64	7,004.53	7,216.25	106.26
10	7,269.68	7,527.55	7,726.65	145.73

Table 8: Results of Strategy I by class

no decomposition ($10 \cdot 10 \cdot 8h = 800h$)				
	Best	Mean	Worst	Stdv
Small	10,768.38	11,157.48	11,577.30	93.82
Medium	24,608.03	25,428.30	26,260.06	243.88
Large	14,131.32	14,532.08	14,942.90	127.30

Table 9: Results of Strategy II by class

fixed decomposition ($10 \cdot 10 \cdot 8 \cdot 8h = 6400h$)						
	# clusters	Best	Mean	Worst	Stdv	Rank
Small	16	10,476.20	10,569.07	10,679.76	33.95	2
	20	10,454.03	10,543.05	10,645.76	31.43	1
	22	10,826.24	10,905.97	11,068.00	29.49	3
	26	11,061.94	11,154.92	11,231.33	34.31	4
	32	11,453.13	11,546.82	11,627.81	32.36	5
	40	12,133.01	12,174.46	12,236.35	14.05	6
	53	13,253.88	13,328.14	13,377.80	27.40	7
	80	15,489.56	15,512.81	15,564.26	12.85	8
Medium	16	21,694.29	21,923.49	22,110.66	74.94	1
	20	21,759.04	21,988.95	22,305.77	73.24	2
	22	22,102.60	22,217.69	22,422.36	63.66	3
	26	22,410.12	22,546.74	22,799.75	72.60	4
	32	23,059.11	23,196.34	23,395.94	41.09	5
	40	23,900.03	24,031.28	24,160.76	36.74	6
	53	26,059.15	26,216.12	26,377.68	39.17	7
	80	30,452.64	30,547.83	30,653.02	25.66	8
Large	16	11,970.13	12,067.94	12,162.91	37.03	1
	20	12,165.35	12,238.52	12,343.51	36.75	2
	22	12,164.28	12,244.38	12,328.99	45.04	3
	26	12,324.99	12,417.89	12,507.92	34.22	4
	32	12,587.37	12,688.86	12,748.92	26.57	5
	40	12,960.88	13,012.02	13,083.13	16.11	6
	53	13,350.07	13,407.04	13,455.22	26.14	7
	80	14,402.30	14,487.32	14,594.94	23.97	8

Table 10: Results of Strategy $III a$ by class

POPMUSIC (long) ($10 \cdot 10 \cdot 8 \cdot 8h = 6400h$)						
	# clusters	Best	Mean	Worst	Stdv	Rank
Small	16	9,146.16	9,282.50	9,430.24	50.91	8
	20	9,102.46	9,249.05	9,391.41	56.45	6
	22	9,118.99	9,238.56	9,398.32	46.75	4
	26	9,127.43	9,266.33	9,408.25	49.89	7
	32	9,122.98	9,234.82	9,349.97	37.09	3
	40	9,053.41	9,218.38	9,398.19	50.84	1
	53	9,129.94	9,240.59	9,404.85	61.16	5
	80	9,089.32	9,227.38	9,397.18	39.62	2
Medium	16	20,437.59	20,686.65	20,993.94	104.61	8
	20	20,323.06	20,540.00	20,756.90	58.35	6
	22	20,268.55	20,581.12	20,848.29	85.18	7
	26	20,252.23	20,472.45	20,726.84	67.86	3
	32	20,221.06	20,483.68	20,770.48	92.92	4
	40	20,249.71	20,497.42	20,734.10	78.27	5
	53	20,269.93	20,471.12	20,810.73	102.07	2
	80	20,156.41	20,429.33	20,713.18	60.57	1
Large	16	11,532.49	11,648.60	11,759.85	50.09	7
	20	11,451.16	11,599.02	11,724.41	53.92	4
	22	11,525.01	11,662.78	11,763.10	54.16	8
	26	11,506.05	11,605.74	11,711.06	47.98	5
	32	11,395.66	11,553.61	11,688.02	47.80	3
	40	11,369.10	11,550.09	11,661.57	65.81	2
	53	11,406.17	11,545.81	11,685.24	60.97	1
	80	11,505.03	11,607.31	11,826.82	80.29	6

Table 11: Results of Strategy $IIIb$ by class

	POPMUSIC (short) ($10 \cdot 10 \cdot 8 \cdot 0.5h = 400h$)					
	# clusters	Best	Mean	Worst	Stdv	Rank
Small	16	9,966.38	10,200.80	10,454.41	84.90	8
	20	9,866.84	10,085.59	10,253.86	54.51	6
	22	9,883.93	10,140.76	10,423.39	75.67	7
	26	9,830.67	10,020.04	10,228.69	78.48	5
	32	9,793.71	9,990.40	10,218.84	76.80	4
	40	9,651.10	9,865.66	10,098.00	72.80	1
	53	9,713.36	9,932.95	10,233.72	112.97	2
	80	9,687.67	9,966.59	10,289.52	112.26	3
Medium	16	22,352.24	22,743.57	23,090.57	136.76	8
	20	22,085.14	22,541.87	23,017.33	104.78	7
	22	22,138.83	22,514.31	22,976.41	74.45	6
	26	21,888.54	22,254.29	22,662.73	110.00	5
	32	21,741.50	22,100.39	22,518.78	70.69	4
	40	21,610.46	22,013.68	22,418.18	132.56	3
	53	21,648.42	22,003.30	22,353.57	103.83	2
	80	21,450.20	21,800.15	22,237.30	76.30	1
Large	16	12,636.64	12,829.57	13,121.98	63.39	8
	20	12,575.81	12,793.70	13,035.89	97.84	7
	22	12,585.25	12,784.62	12,952.19	73.43	6
	26	12,427.22	12,625.26	12,898.93	60.77	5
	32	12,314.27	12,557.46	12,756.27	90.84	3
	40	12,332.03	12,564.40	12,766.31	57.71	4
	53	12,227.43	12,434.03	12,627.31	86.63	1
	80	12,297.36	12,473.14	12,607.14	62.68	2

Table 12: Best solution values found

Day	Best	# clusters
1	3,649.09	80
2	4,061.71	32
3	4,022.29	22
4	4,258.19	80
5	4,134.59	40
6	2,423.83	22
7	3,438.83	40
8	3,164.80	80
9	5,427.65	40
10	5,941.45	40