Node Aggregation in Vehicle Routing

Johan Oppen
Molde College, 6402 Molde, Norway
Johan.Oppen@hiMolde.no

Arne Løkketangen
Molde College, 6402 Molde, Norway
Arne.Lokketangen@hiMolde.no

Abstract

We describe a special variant of the Vehicle Routing Problem (VRP), where there are many customers per road segment. A metaheuristic tabu search algorithm for the VRP is given, and aggregation of customers is used to better the performance, with respect to both time and solution quality. The methods are tested on test instances from the literature as well as a portfolio of new test instances especially made to fit to the problem description. Test results are reported, showing that aggregation can lead to substantial improvements both in time and solution quality in this setting, especially for larger instances.

1. Introduction

In the vehicle routing literature there is a distinction between *node routing*, where the transportation tasks are focused on nodes in a road network, and *arc routing*, where the transportation tasks are defined on road segments. Planning of pickup and delivery for a transportation company represent a typical node routing problem, while gritting and snow removal are prime examples of arc routing. Vehicle routing resolution methods differ substantially between node and arc routing. For a more thorough description of arc routing problems, see Dror (2000). Chapter 1 in Toth and Vigo (2002) provides an overview of node routing problems.

We consider vehicle routing with many stops per road segment. This kind of routing problem may combine aspects of node and arc routing, and is known from i.e. waste collection and newspaper delivery. There is a desire from the industry for better solutions to this kind of problems, in terms of both the time taken to find a solution and the solution quality. In an operational setting, time is limited, and one often needs a solution to a problem in a minute or two. In practice, large instances with many customers arise, and these are often hard to solve to optimality even if a large amount of computation time is available. Aggregation of customers may then be used to reduce problem size, which allows for a larger part of the solution space to be searched.

The rest of this paper is divided as follows. The Vehicle Routing Problem is described in section 2. We describe a tabu search metaheuristic and the use of aggregation to reduce problem size in section 3, and then follow a description of test instances and test results in section 4. Finally, conclusions and future work are presented in section 5.

2. The Vehicle Routing Problem

The Vehicle Routing Problem (VRP) deals with the allocation of transportation tasks to a fleet of vehicles, and the simultaneous routing for each vehicle. The problem was first described by Dantzig and Ramser (1959). The VRP is a computationally hard optimization problem with high industrial relevance.

The classical VRP is defined on a graph $G = (N, A)$ where $N = \{0, ..., n\}$ is a vertex set and $A = \{(i, j) : i, j \in N\}$ is an arc set. Vertex 0 is the depot, the other vertices are the customers. The travel cost between customer i and j is defined by $c_{ij} > 0$ and d_i is the demand for customer i. The vehicles are usually identical, each with a capacity q. The goal is then to design a least cost set of routes, all starting and ending at the depot, where each customer is visited exactly once. The total demand of all customers on a route must be within the capacity q. This classical formulation is often referred to as the capacitated VRP or CVRP.

To make VRP models more realistic, a lot of different extensions may be added to the basic model. Examples of such extensions are distance or duration constraints for each route, time windows within which each customer must be visited, and others. For a more thorough description of the VRP and its possible extensions, see Toth and Vigo (2002).

The stringed VRP

For a given type of VRP application, there is no clear cut rule for choosing between a node routing model like the one described above, and an arc routing model. An arc routing problem may be transformed to an equivalent node routing problem and vice versa. Also, a given application may combine aspects of node and arc routing. Garbage collection as well as mail and newspaper delivery may, in applications where the number of stops per road segment is high, be borderline cases. Most industrial VRP solvers are based on node routing, and they do not address the special structure of the kind of VRP described here. Thus, they do not give satisfactory performance, with respect to both solution time and quality, for this class of problems.

This paper focuses on such borderline cases, where all demand clearly is defined on some of the nodes in a network. Traditional Euclidean CVRP instances are usually represented by a fully connected graph, where it is possible to travel between any pair of nodes by following a straight line. We consider VRP instances represented by a graph depicting a road network, where some of the nodes just represent road crossings. The road crossings are connected through arcs, but in general we cannot travel directly between any pair of road crossings. The customer nodes are situated along the roads, and some of them are close together and form “strings” or “paths” in the graph. It often seems obvious that the nodes of such a string should be visited by the same vehicle. It is easy to imagine such a string of nodes replaced by an arc between the first and last node in the string.

Figure 1 shows a simple example of such a graph. The 20 customers are shown as triangles, road crossings are shown as circles and the depot is shown as a square. Here, it seems obvious that five of the nodes between the road crossings in A and B should be visited by the same vehicle, if possible due to constraints.

In the following, VRP instances with the properties described here will be referred to as “stringed VRPs”. This is done to avoid having to give a detailed and verbose explanation every time this kind of VRP instance is referred to.

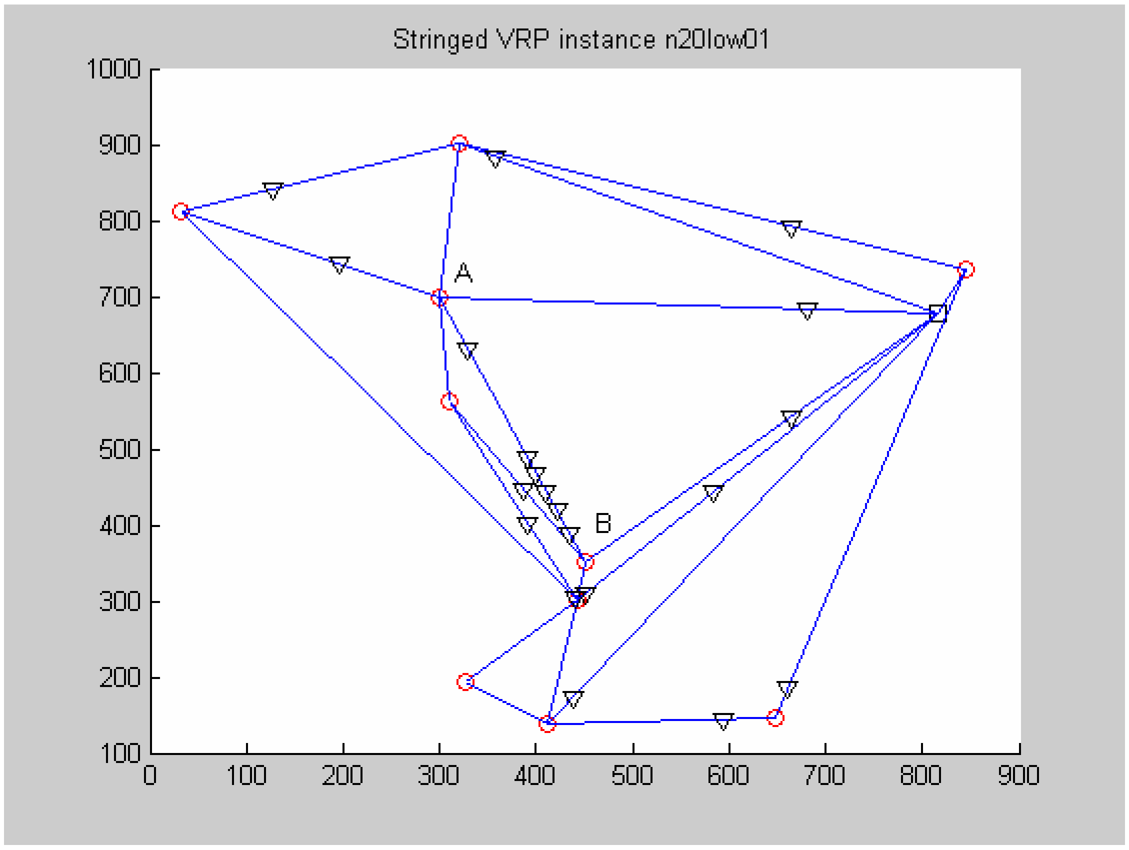

Figure 1. Stringed VRP instance n20low01.

Previous work

A variety of solution methods have been applied to the CVRP. This includes both exact and heuristic methods; see Toth and Vigo (2002) for an overview. For larger problems with more than 50 customers, exact methods in general are not able to find the optimal solution in reasonable time. Heuristic methods, on the other hand, cannot guarantee any specified solution quality, but many of them are known to give good results in short time also for large instances.

Not much seems to have been done to solve the stringed VRP, but some related kinds of VRP and Travelling Salesman Problems (TSP) have been studied. Laporte and Palekar (2002) describe applications of the clustered TSP, and Laporte, Asef-Vaziri and Sriskandarajah (1996) describe applications of the generalized TSP. There is also an article by Blais and Laporte (2003) about exact solution of the generalized routing problem through graph transformations. In all these papers, useful modeling techniques can be found that are more or less applicable to the stringed VRP. It should be mentioned, however, that there is no general conformity between a clustered problem, which is the theme in most of the papers referenced here, and our definition of a stringed problem. In the scientific literature, details like a road network are usually not included in the description of a VRP. The most usual is to use an idealized Euclidean topology, where the customers are points in the plane and the distance between them Euclidean. A clustered VRP is then a problem instance where the customers are lumped together in groups rather than being evenly or randomly spread. A stringed VRP instance is on the other hand an instance defined on a road network, with many stops per road segment, and could be clustered, random, or a combination of both.

Aggregation

Although little work seems to be done on VRPs that share the distinct properties of the VRPs studied here, much is done in the field of aggregation of optimization problems in general. Nonås (1993) describes aggregation of variables (columns) in linear problems and how this can be used to estimate both the optimal solution and the optimal objective value of the original problem. As is usual when a mathematical and theoretical approach is used to develop methods for aggregation of linear or integer problems, she does not exploit the structural information, that might or might not be present, to support the aggregation. Hjertenes (2002) uses structural information about city locations to aggregate Travelling Salesman Problems (TSP). As the VRP is a generalization of the TSP, VRP instances hold structural information, and it should be possible to exploit this information to aggregate customers also in a VRP setting.

We use aggregation of customers to find good solutions faster for stringed VRPs. Such aggregation can be viewed as an approach to arc routing, as visiting an aggregate of customers along a road segment corresponds to traversing an arc between the first and last customer in the aggregate.

By aggregating customers that are close together, problem size decreases. On the other hand, one must assume that errors are introduced as customers that do not belong to the same tour in the optimal solution, may be aggregated. The goal will then be to do the aggregation in a way that minimizes these errors, but at the same time reduces the problem size enough to allow for better solutions in the same solution time.

It should be emphasized that in many situations, the problem at hand is too large for the optimal solution to be found in reasonable time. In a real-life situation, optimality is not the goal, while getting a good solution in a short amount of time is crucial. If aggregation can help in getting such a good, but not necessarily optimal, solution quickly, it will be of interest.

3. A tabu search based metaheuristic for the VRP

As stated earlier, heuristic solution methods are needed to solve larger instances of the VRP. In many situations where we are dealing with instances that fit well into the description of the stringed VRP, there are thousands of customers. We need a reasonably good solver to find out if aggregation works for the stringed VRP, and our solver is based on tabu search. For a closer description of tabu search, see Glover and Laguna (1997).

The basic algorithm

This algorithm for the CVRP is based on ideas from Cordeau, Laporte and Mercier (2001). They describe a tabu search algorithm for the VRP with time windows, and some adjustments and changes to their approach have been done.

Moves

Let S be the set of all solutions that satisfy the following constraints: every tour starts and ends at the depot, and every customer belongs to exactly one tour. Moreover, let $A(s) = \{(i,k): \text{customer } i \text{ is visited by vehicle } k\}$ be an attribute set associated with each solution $s \in S$. The neighbourhood $N(s)$ of a solution s is defined by applying an operator that removes an attribute (i,k) from $A(s)$ and replaces it with a different attribute (i,k'), where $k \neq k'$. This gives a neighbourhood size of $|N| = n*(m-1)$, where *n* is the number of customers and *m* is the number of vehicles. The number of vehicles, *m,* is fixed. When a customer is removed from a tour, the tour is reconnected by linking the predecessor and successor of the removed customer. The insertion of a customer into a tour is done so as to minimize the increase in the length of the tour, but without changing the order of the customers already in the tour.

Move evaluation and diversification

For each solution $s \in S$, let $c(s)$ be the total length of all tours in the solution, and let $q(s)$ be the total violation of capacity constraints for the tours. Solutions are evaluated using a function $f(s) = c(s) + \alpha q(s)$, where α is a positive parameter that is dynamically adjusted during the search. To diversify the search, any solution $s' \in N(s)$ such that $f(s') > f(s)$, is given a penalty $p(s') = \lambda \sum_{(i,k) \in A(s')} \rho_{ik}$ that is added to $f(s')$. Here, ρ_{ik} is the number of times the attribute (i,k) has been part of a *good solution*, that is, a solution that is feasible and has a total length less than η times the length of the best solution found so far. The parameter λ is used to control the intensity of the diversification. These penalties are used to lead the search into less explored parts of the solution space whenever a local optimum is found. If $f(s') \leq f(s)$, the penalty term is not added to $f(s')$.

Initial solution

First, a starting solution is generated. Every customer is inserted into the first tour that has enough capacity left. The customer is inserted into the tour in the best possible way, that is, the increase in distance is minimized. If no tour with enough free capacity can be found, the customer is inserted into the last tour. This gives a starting solution where all the tours are feasible, except possibly the last tour.

Tabu search

The tabu search starts from the initial solution and moves, at each iteration, to the non-tabu neighbor that minimizes $f(s')$ for all $s' \in N(s)$. The attribute (i,k) that was removed from $A(s)$ is now declared tabu for *tabu tenure* iterations, where *tabu tenure* is the tabu length or duration. During these iterations, it is not allowed to move customer *i* back to tour *k*. By the use of a simple aspiration criterion, a tabu move can still be chosen if this leads to a solution that is the best found so far in the search. After each move, the value of the parameter α is adjusted. If the current solution is feasible (no capacity violations), the value of α is decreased to make it less costly to visit an infeasible solution. If the current solution is infeasible (one or more tours violates the capacity constraints), α is increased to lead the search back into the feasible region of the solution space.

If the current solution is feasible, has a total length less than η times the length of the best feasible solution found so far during the search, and the number of iterations performed has reached 100, the solution is considered *good*. Whenever a good solution is found, the ρ values for the attributes of the solution are incremented. In addition, a 2-opt post-optimization procedure is applied to the tours of the solution.

The search continues until a preset time limit is reached, or until a preset number of moves have been performed.

Search parameter settings

Preliminary tests have been run on a small subset of instances to find good values for the search parameters. The tabu length is, in each iteration, uniform randomly chosen from a set of integers whose limits are defined by the number of customers. The set consists of the integers from 3 to 5 if the number of customers is less than 25, from 5 to 10 for 25 to 40 customers, from 7 to 15 for 41 to 109 customers and from 10 to 20 for instances with more than 109 customers. The parameter α that gives the weight of infeasibility is initially set to 1. A feasible solution yields an increase of 0.01 to the current value, while an infeasible solution leads to a decrease of 0.5. The parameter λ is set equal to 10, η to 1.1.

All search parameters showed rather low sensitivity to minor changes, and the chosen values are those with the best overall performance. As an example, our testing showed clearly that the increase of α should be far less than the decrease, but we also found that the amounts added to and subtracted from α may be changed quite much as long as the proportion between them is kept fairly constant.

Extensions to aggregate customers

In our algorithm, aggregation is done before an initial solution is constructed and any optimization is performed. The aggregation gives a set of what we refer to as “logical customers”, which is a mix of single and aggregated customers. Construction of initial solution and optimization is then performed on this set of logical customers.

The location of an aggregate is given as an *entry point* and an *exit point*, such that the entry point is the location of the first customer in the aggregate, and the exit point is the location of the last customer. By swapping these two locations, it is possible to reverse the order in which the customers in the aggregate are visited. We also keep track of the *internal distance* in the aggregate, which is the distance from the entry point to the exit point.

Road topology

To approach a real-world situation where there are many customers per road segment, we find that traditional problem instances where customers are spread around in the Euclidean plane are not well suited. To have problems closer to the real world, we therefore construct instances where the customers are connected through a road network. Figure 1 shows an example of such an instance. In the figure, the depot is shown as a square, the road crossings are circles, and the customers are triangles. The lines are the roads, and these must be used to traverse the graph.

By using this kind of test instances it is easier to imitate the real world where distances between customers are non-Euclidean, as the vehicles must use the roads. In the real world, natural obstacles like rivers, fjords and mountains can make the real distance between two locations much longer than the straight line.

Aggregation levels

Four levels of aggregation have been defined:

- At level 0, no aggregation is done. This corresponds to the basic algorithm.
- At level 1, only pairs of customers are aggregated. This is done as follows. For each customer that is not yet part of an aggregate, the closest neighbor is found. If this customer is not yet part of an aggregate, close enough and the total demand of the two customers is small enough, the two are aggregated.
- At level 2, customers that belong to the same road segment are aggregated. This is done by first performing level 1. As soon as two customers are aggregated, one tries to extend the aggregate by including neighbors in both ends. When no more neighbors are close enough, or when the demand of the aggregate grows too big to allow for further aggregation, the next single customer is found to serve as a starting point of a new possible aggregate. No aggregation is done over road crossings.
- At level 3, aggregation is done also over road crossings. This is done the same way as level 2, but now it is also sought for neighbor customers to include in the aggregate by passing road crossings. If exactly one customer that is close enough is found, this is included, given that the customer in question is not already part of an aggregate, and that total demand allows for it. If there is more than one candidate to be aggregated, nothing is done. This is interpreted as a an indication that there may be more than one good way to proceed with the aggregation from the road crossing, and in this situation it is regarded better to be reserved.

It should be noticed that for Euclidean instances, level 2 and 3 are equal, as these instances have no road network and thus no road crossings to pass.

Aggregation parameters

Two parameters are used to guide the aggregation of customers. The parameter ω is used as a measure of how close two customers must be to allow for aggregation. If the distance between them is less than ω times the average distance between two customers, the customers can be aggregated. Another parameter τ is used to limit the total demand of the aggregated customers. If the sum of the demands of two customers is no more than τ times the vehicle capacity, the customers can be aggregated. Aggregation is done only if the customers are close enough and the total demand is small enough. Nine different combinations of ω and τ have been tested to find good values for the two parameters. These combinations are 0.1, 0.3 and 0.6 for ω and 0.2, 0.5 and 1 for τ. Different combinations gave the best results for different classes of instances and different aggregation levels. ω=0.1 and τ=1 was the best overall combination; these were also the values that performed best for the instances with most customers per road segment.

To get a view of how the aggregation works, we show a figure of the instance n100medium11 after aggregation at level 2. See section 4 for a closer description of the stringed test instances. The instance originally has 101 logical customers, as the depot is treated as a customer. Figure 3 shows the instance after aggregation. The depot is shown as a square, and the road crossings are shown as circles. Single customers are shown as triangles, while customers that are part of an aggregate are shown as stars. Customers that are start- or endpoints of an aggregate are shown as stars inside a circle. In addition, all aggregates are shown as ellipses.

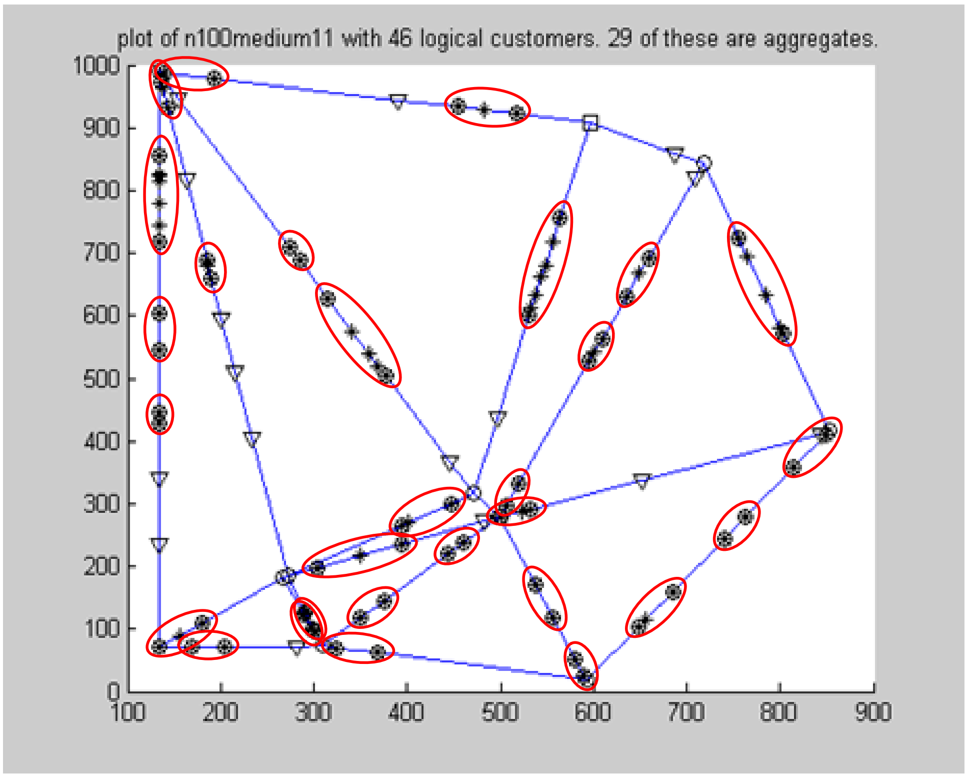

Figure 3. n100medium11 after aggregation at level 2.

4. Computational experiments

Test cases for the capacitated VRP

Test cases for the capacitated VRP from the literature have been used to test the performance of the solver. Ninety different instances have been used; these can be found at http://branchandcut.org/VRP/data/ . The optimal solution value is known for most of the instances, for the rest comparisons are done against some best-known solution value. The instances have between 16 and 262 customers.

Test cases for the stringed VRP

To find out if the methods developed for solving the stringed VRP work well, 180 test cases have been constructed for this purpose. The instances have 20, 50, 100 or 500 customers, and three different "string factors" are used for each problem size. The "string factor" indicates the number of customers per road segment, with a low string factor corresponding to a low number of customers per road segment.

The stringed test instances are named by adding a number to the name of the class, such that instance n20low17 is instance number 17 of the class n20low, which has 20 customers and low string factor. Originally, 20 instances of each class were made. 15 of these were chosen for testing, leaving out instances with road segments that are very close and almost parallel, or in other ways look particularly strange.

Results

All test instances are run with CPLEX for up to 3 hours both with and without aggregation. Aggregation is performed at the highest level (2 for the Euclidean instances, 3 for the stringed).

With the tabu search algorithm, all instances are tested in short and long runs with all levels of aggregation (3 different levels for the Euclidean instances, 4 for the stringed). The long running time is -103 + 3.69(n log n) seconds, where n is the number of customers. The short running time is 10% of the long. All runs are repeated 10 times with different random seeds.

Test results for the Euclidean CVRP

CPLEX found the optimal solution for one instance, this was the smallest one with 16 customers. In general, CPLEX was not able to find very good solutions, especially for the larger instances. CPLEX on average found better solution values after aggregation than before.

Our tabu search metaheuristic in general found the best solutions when no aggregation was done. For 32 out of 38 instances with less than 50 customers, the optimal solution was found with the long running time. On average, solution values 1.015 times the optimal or best known were found without aggregation, using long running time. For 12 of the larger instances with more than 50 customers, the use of aggregation gave better results.

Test results for the stringed VRP

The tables presented in this section give the results for the 3 classes of problems with the same number of customers. The results are given as the average proportion of the best solution value found. The column named “CPLEX” refers to the results from running CPLEX on the original problems, while the column named “CPLEX aggr” refers to the results when CPLEX was used to solve the aggregated problems. The last 8 columns are results from the tabu search algorithm. These results are split in the four aggregation levels used, and for each aggregation level the results from the short and long runs are presented. As an example, from table 1 we find that for the class n20medium, CPLEX after aggregation on average found solution values 1,049 times the best solution value found for the instances of this class, while long runs of tabu search after aggregating at level 3 on average gave solution values 1,005 times the best found.

We compare only to our own results for the stringed instances, and the purpose of our tests is to find out how aggregation works on these instances.

20 customers

CPLEX found and proved optimum for 2 of the 45 instances with 20 customers. CPLEX in general performs better on aggregated problems, even if the instances with 20 customers never are aggregated down to less than 15 customers.

The metaheuristic performs better than CPLEX for all 3 classes of problems with 20 customers. When the string factor is low, aggregation has a negative effect. We believe this is because customers that should be on different tours are aggregated together.

Class	CPLEX	CPLEX aggr	Level 0		Level 1		Level 2		Level 3	
			Short	Long	Short	Long	Short	Long	Short	Long
n20low	1,065	1,035	1,003	1,002	1,004	1,004	1,004	1,003	1,004	1,003
n20medium	1,082	1,049	1,010	1,008	1,009	1,006	1,009	1,006	1,008	1,005
n20high	1,026	1,043	1,014	1,012	1,006	1,003	1,008	1,005	1,008	1,005

Table 1. Results for stringed VRP instances, 20 customers.

When the string factor is medium or high, aggregation gives better solutions even for these small instances. We also notice that short runs with aggregation outperform long runs without aggregation in most cases when the customers are close. The different aggregation levels vary in performance, but they all do better than no aggregation for medium and high string factor.

50 customers

Class	CPLEX	CPLEX aggr	Level 0		Level 1		Level 2		Level 3	
			Short	Long	Short	Long	Short	Long	Short	Long
n50low	1,533	1,313	1,049	1,044	1,019	1,012	1,019	1,014	1,022	1,017
n50medium	1,641	1,263	1,042	1,035	1,017	1,012	1,021	1,014	1,024	1,017
n50high	1,466	1,187	1,026	1,020	1,015	1,019	1,024	1,021	1,029	1,028

Table 2. Results for stringed VRP instances, 50 customers.

CPLEX neither found the proven optimum nor the best result for any of the instances with 50 customers. We find substantial improvement when CPLEX is run after aggregation.

For all three classes, aggregation level 1 produces the best results. For low and medium string factor, level 2 and 3 are much better than no aggregation, while no aggregation is second best for the instances with high string factor. This is not quite as expected, but we notice that the difference between aggregation level 0 and 2 is small. We also believe that the 50 customer instances are too small to show distinct improvements by use of aggregation.

As for the 20 customer instances, we notice that short runs with aggregation outperform long runs without aggregation in many cases.

100 customers

Class	CPLEX	CPLEX aggr	Level 0		Level 1		Level 2		Level 3	
			Short	Long	Short	Long	Short	Long	Short	Long
n100low	1,536	1,414	1,096	1,073	1,054	1,039	1,042	1,028	1,050	1,034
n100meium	1,670	1,285	1,062	1,041	1,038	1,021	1,021	1,014	1,027	1,018
n100high	1,662	1,208	1,034	1,020	1,017	1,009	1,033	1,028	1,042	1,037

Table 3. Results for stringed VRP instances, 100 customers.

CPLEX on average found solutions with a total length of more than 150% of the best solution found. When aggregation was used, the solution quality increased to about 130% on average. It should be noted that after aggregation, the number of instances where CPLEX was unable to find an integer solution within the time limit of 3 hours, increased.

The results from running the tabu search algorithm show that aggregation always improves the solution value for the low and medium stringed instances. In both cases level 2 is the best. For high string factor, level 1 is the only aggregation level that does better than no aggregation. As for the n50 classes, we think the rather poor performance of aggregation level 2 and 3 for the instances with high string factor is due to too aggressive aggregation. The initial testing to find good values for our aggregation parameters also indicated that τ =0.2 performed better both for the n50high and the n100high instances. This corresponds to only allowing for aggregates with a total demand of 20% of the vehicle capacity. The choice of value for τ was based on the fact that τ =1.0 gave the best overall results for all problem classes.

500 customers

Class	CPLEX	CPLEX aggr	Level 0		Level 1		Level 2		Level 3	
			Short	Long	Short	Long	Short	Long	Short	Long
n500low	14,130	*1,206*	1,498	1,378	1,713	1,433	1,471	1,058	1,280	1,075
n500medium	15,328	1,071	1,278	1,270	1,395	1,239	1,097	1,056	1,072	1,070
n500high	15,925	1,026	1,146	1,142	1,219	1,141	1,021	1,021	1,025	1,025

Table 4. Results for stringed VRP instances, 500 customers.

For the 500 customer instances, CPLEX on average found solution values about 15 times the best found when no aggregation was used. After aggregation, CPLEX performs much better.

We expected the effect of aggregation to increase as the number of customers grows. The results for the 500 customer instances show that this is correct. We notice that level 1, where only pairs of customers are aggregated, hardly give any improvements at all. Except for the short runs of the instances with low string factor, aggregation level 2 and 3 give great improvements compared to what is achieved without aggregation. The aggressive aggregation policy mentioned earlier has led to a situation where the short runs for the instances with high string factor give the same results as the long runs.

5. Conclusions and future work

We use aggregation of customers to reduce problem size and thereby find better solutions to vehicle routing problems with many stops per road segment.

We have developed a tabu search based metaheuristic for VRPs that gives good results on well-known test instances for the CVRP. This algorithm is extended to aggregate customers, and a portfolio of new test instances is made. Test results both on the test instances from the literature and the new set clearly show that aggregation of customers better the performance of our solver substantially, especially for large instances with many customers per road segment. The tests also show that good results are obtained very quickly after aggregation.

We have used CPLEX for comparisons both on full-sized and aggregated problems. Aggregation also makes the results from CPLEX much better for the larger instances.

Paths for future work to find better ways to solve the stringed VRP might be:

- Try the use of arc routing methods, either alone or in combination with node routing methods.
- Investigate more sophisticated methods for aggregation; this might include mechanisms for adjusting the aggregates dynamically during the search. A hierarchical model where aggregation is done level by level might be considered.
- Improvement of the diversification mechanisms for the tabu search algorithm.
- In order to try aggregation also on real-world cases, a software vendor could add aggregation methods in an industrial VRP solver.

Acknowledgements

The work described here is done under the TOP project (http://www.sintef.no/static/am/opti/projects/top/about.html), financed by the Norwegian Research Council.

References

Blais, M; Laporte, G (2003) Exact solution of the generalized routing problem through graph transformation. *Journal of the Operational Research Society,* 54: 906-910.

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

Dantzig, G.B. and Ramser, J.H (1959) The truck dispatching problem. *Management Science,* 6:80.

Dror, Moshe, ed. (2000) *Arc Routing.* Norwell, Massachusetts: Kluwer Academic Publishers.

Glover, F; Laguna, M (1997) *Tabu Search.* Boston, Massachusetts: Kluwer Academic Publishers.

Hjertenes, Øystein M (2002) *A Multilevel Scheme for the Travelling Salesman Problem.* Department of Informatics, University of Bergen, Norway.

Laporte, Gilbert; Asef-Vaziri, Ardavan; Sriskandarajah, Chelliah (1996) Some Applications of the Generalized Traveling Salesman Problem. *Journal of the Operational Research Society,* 47: 1461-1467.

Laporte, G; Palekar, U (2002) Some applications of the clustered traveling salesman problem. *Journal of the Operational Research Society,* 53: 972-976.

Nonås, Sigrid Lise (1993) *Aggregering av lineære problemer.* Department of Informatics, University of Bergen, Norway. (in Norwegian)

Toth, Paolo; Vigo, Daniele, eds. (2002) *The Vehicle Routing Problem.* Philadelphia: Society for Industrial and Applied Mathematics.