An ant colony algorithm for the multi-compartment vehicle routing problem

Martin Reed, Aliki Yiannakou, Roxanne Evering

Department of Mathematical Sciences, University of Bath, Bath, United Kingdom

Abstract

We demonstrate the use of Ant Colony System (ACS) to solve the capacitated vehicle routing problem associated with collection of recycling waste from households, treated as nodes in a spatial network. For networks where the nodes are concentrated in separate clusters, the use of k-means clustering can greatly improve the efficiency of the solution. The ACS algorithm is extended to model the use of multi-compartment vehicles with kerbside sorting of waste into separate compartments for glass, paper, etc. The algorithm produces high-quality solutions for two-compartment test problems.

Keywords:
ant colony optimization, capacitated vehicle routing problem, clustering, multi-compartment vehicles, waste collection

1. Introduction

1.1. Vehicle routing problems

Vehicle routing problems (VRPs) are an extension of the classic Travelling Salesman Problem (TSP), in which one or more vehicles travel around a network, leaving from and returning to a depot node. Customers are located on the network and each customer must be visited by exactly one vehicle. Customers are usually located at network nodes (although in arc routing problems they are distributed along arcs of the network). The object is to find the vehicle routing(s) of minimum cost, e.g. to minimise the total route length.

An important type of VRP for practical application is the capacitated vehicle routing problem (CVRP). In this, there is a demand associated with each customer, representing an amount of goods which must be collected (or delivered). Each vehicle has a capacity which cannot be exceeded, and when the vehicle is full (or empty) it returns to the depot. There are many variants of CVRP with additional constraints, for example CVRPTW in which each customer may have a time window during which a visit must be scheduled. A survey of such variants is given in [1]. The study of CVRPs has become of increasing practical importance as distribution networks become more complex, and with the growth of online shopping, leading to a recent resurgence of research interest [2].

1.2. Waste collection

In this paper we consider a basic CVRP applied to the collection of domestic waste for recycling. Waste collection is becoming an increasingly complicated task for municipal authorities, and growing environmental concerns are gradually changing the orientation of solid waste management. In the UK, recyclable waste generated by households is organised by local authorities (LAs, for example District Councils), either with their own vehicles or contracted to private companies. Each LA decides on the nature and level of service it will provide, taking into account social and economic factors under the headings of collection, transportation and disposal [3, 4]. Reports have shown that logistics costs represent up to 95% of the total cost of recycling [5], so the importance of devising the most cost-efficient routes using the minimum number of vehicles is crucial. A new factor motivating research in this area is the imposition of large fines for missing government-set recycling targets.

1.3. Multi-compartment vehicles

Once a waste collection vehicle is full to capacity, it must move to a waste disposal site (commonly referred to as a tip) to unload. Frequently the tip is at the same location as the vehicle depot, but it may be located at a different node in the network. At the tip, the waste is sorted into its constituent types (paper, glass, etc) which are disposed of or recycled in different processes.

A recent development is the introduction of multi-compartment stillage trucks for domestic waste collection. These vehicles typically have four separate compartments, so that glass can be stored separately from paper, for example. The vehicle crew must perform kerbside sorting of the waste in customers' recycling boxes, and this slows up the collection process. This disadvantage is hopefully outweighed by the improvement in quantity and quality of the recyclable material produced. The pros and cons of kerbside sorting as against commingled collection in single-compartment vehicles, are discussed in [5, 6, 7].

2. Previous work

2.1. Solving the CVRP

Standard OR techniques of dynamic programming, or branch and cut, can be used to produce exact solutions for only small CVRP problems; as of 2007, the largest problem to be solved exactly had 135 nodes [8]. Starting with the algorithm of Clarke and Wright in 1964, several heuristics have been proposed, which construct tours and/or improve existing tours [9]. In particular we mention the improvement of a completed sub-tour (from the depot through some or all nodes and returning to the depot) by r-optimal methods [10]; the 2-opt algorithm tests whether a tour length can be shortened by crossing two of its non-adjacent arcs, i.e. by replacing arcs (a, b) and (c, d) by (a, c) and (b, d). Constructive heuristic methods are tailored for specific problems, which restricts their use in wider applications. This has led to the development of more versatile metaheuristics, which offer global search strategies for exploring the solution space. Metaheuristics which have been applied to the VRP include tabu search [11, 12] and simulated annealing [13]. More recently, soft computing techniques have proved successful in solving CVRP instances. Thangiah [14] used genetic algorithms in conjunction with these two metaheuristics in 1999, and since 2003 several papers [15, 16, 17, 18, 19, 20] have extended the use of genetic algorithms for the VRP, including with time windows. Khouadjia et al. [21] have used particle swarm optimization and variable neighbourhood search for a dynamic VRP. Adaptive Large Neighbourhood Search [8] uses a variety of heuristic algorithms (chosen by roulette wheel solution) to destroy and then repair solutions, and has been shown to solve a range of VRP variants, including CVRP, CVRPTW and multi-depot CVRP.

Perhaps the most successful soft computing approach for the TSP and related problems such as job shop scheduling and quadratic assignment, is ant colony optimization (ACO). The details are given in the next section, but essentially the algorithm as applied to the VRP is as follows. At the start of each iteration, ants (autonomous agents) are placed at random nodes of the network, ready to construct tours; their initial tour length is the distance from the depot to their starting node. They maintain a tabu list to avoid returning to already-visited nodes, and decide their next move stochastically, with probabilities based on the amount of pheromone present on the possible arcs. Once all ants have completed their tours they return to the depot node. They then update the pheromone levels on the arcs they visited (local updating), and the levels on the arcs in the tour with the shortest total length are further increased (global updating). Over a large number of iterations, the pheromone levels encourage ants to use high-quality paths through the network, resulting in shorter-length tours being discovered. The original algorithm is known as Ant System (AS), and a later, more successful variant is called Ant Colony System (ACS); both are described in Dorigo and Stützle's book [22].

Mazzeo and Loiseau [23] used the ACS to solve some benchmark CVRP problems, and report good performance in comparison with some of the methods described above. Karadimas et al. [24, 25] applied an ACO algorithm to the problem of urban solid waste collection, although they avoided the need to include capacity constraints in the ACO by first breaking the network into a set of sub-areas, each of which could be serviced by a single vehicle without unloading, an approach also followed in [26]. ACO was also applied to urban waste collection in [27], although they model the problem as a capacitated arc routing problem (CARP) extended to comply with traffic rules. They apply two versions of the ACO: the Ant System, and a populational AS in which only a subset of elite solutions update the pheromone trails. They highlight the benefits of integrating these methods with decision support systems to aid planners in their decisions. Rizzoli et al. [28] describe a commercial ACO package and its performance on a number of real-world freight distribution problems, including dynamic handling of orders. ACO algorithms for the VRP have been hybridized with scatter search [29], with genetic algorithms [32], and with savings algorithms and problem decomposition [30, 31].

2.2. Solving the CVRP with multiple compartments

The multi-compartment VRP (MCVRP) was identified already in 1979 as a variant of VRP which had practical significance; Christofides et al. [9] give as examples a delivery vehicle which has refrigerated and non-refrigerated compartments for foodstuffs, and a tanker which distributes different types of petroleum products. The problem has not however attracted the attention of researchers until recently. There have been heuristic approaches to the petroleum distribution problem in [33, 34], and the food distribution problem in [35], the latter using Lagrangean relaxation. In 2008 El Fallahi et al. [36] tackled a distribution problem in which a depot stocks m different products which must be delivered to customers by a fleet of identical vehicles, each with m compartments of limited capacities. Unlike in the multi-compartment waste collection problem, it is permitted for different vehicles to deliver different products to the same customer. The paper demonstrates a memetic algorithm (a genetic algorithm hybridised with a local search procedure) and a tabu search procedure for the MCVRP. Most recently, in 2010 Muyldermans and Pang [38] addressed the MCVRP applied to distribution. Their heuristic started by constructing the Clarke and Wright solution, and used 2-opt for improvement, coupled with guided local search. They used this algorithm to compare the costs of MC-collection against commingled collection.

3. The ACS algorithm

The two main phases of the ACO algorithm are the ants' route construction and the pheromone update. In the tour construction phase, M ants concurrently build tours beginning from starting nodes randomly chosen in the network of N customer nodes (plus the depot node). At each construction step, ant k currently at node i applies a probabilistic random proportional rule to decide which node to go to next. It selects the move to expand its tour by taking into account the following two values:

- The heuristic function η_{ij} which represents the attractiveness of the move, usually calculated as the inverse of the distance/cost on the arc from node i to node j.
- The level of pheromone on the arc (i, j), denoted τ_{ij}, which indicates how useful it has been in the past to traverse this particular arc.

Given these parameters, the probability with which the ant chooses to go to node n next is

$$p_{in}^k = \frac{(\tau_{in})^\alpha (\eta_{in})^\beta}{\sum_{l \in \mathcal{N}_i^k} (\tau_{il})^\alpha (\eta_{il})^\beta} \tag{1}$$

if node $n \in \mathcal{N}_i^k$, and $p_{in}^k = 0$ otherwise. Here, $\mathcal{N}_i^k$ is the feasible neighbourhood (i.e. the nodes which are directly accessible from node i and not previously visited), and α and β are heuristic parameters. Each ant maintains a memory (a tabu list) of the nodes already visited. Once all ants have completed a tour, the pheromone trails are updated. This is done by first lowering the pheromone levels on all arcs (to represent evaporation and in order to progressively forget bad solutions and encourage exploration of new arcs) and then adding pheromone to the arcs that have been traversed.

The Ant Colony System improves on the ACO in the following main aspects:

- **Route Construction:** During tour construction, ant k, located at node i, moves to node n chosen according to the following pseudorandom proportional rule. A random variable q uniformly distributed over $[0, 1]$ is evaluated, and if $q > q_0$ the node n is chosen according to the standard ACO rule (1), using $\alpha = 1$. Otherwise, choose n by

$$n = \arg\max_{j \in \mathcal{N}_i^k} \{(\tau_{ij})(\eta_{ij})^\beta\}. \tag{2}$$

  So, with probability q_0, the ant makes the best move described by the pheromone trails and heuristic information (exploiting the learned knowledge), while with probability $(1 - q_0)$ it performs a biased exploration of the arcs.

- **Pheromone Update:** The ACS method uses two types of pheromone updates: global and local. The local update is performed every time an ant traverses an arc (i, j) and the pheromone is modified as follows:

$$\tau_{ij} \leftarrow (1 - \xi)\tau_{ij} + \xi\tau_0 \tag{3}$$

  where $0 < \xi < 1$ and τ_0 is the initial pheromone value defined as $\tau_0 = (NL^{nn})^{-1}$ where L^{nn} is the length of the nearest neighbour tour (a tour in which each move is to the nearest unvisited node; this is used as a baseline tour length). The global update, however, is only carried out by the ant that produced the best tour so far and is implemented by the following equation for each arc of the tour:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \quad \forall(i, j) \in T^{bs} \tag{4}$$

  where $\Delta\tau_{ij}^{bs} = (L^{bs})^{-1}$, ρ is a parameter governing decay and T^{bs} is the best found tour so far with L^{bs} its length. This enables the algorithm to converge faster by directly concentrating the search around the best tour.

4. An ACS Algorithm incorporating Unloading Trips

We have extended the ACS algorithm by allowing each ant to keep a record of the amount of waste collected during its tour, and forcing a return to the depot for unloading whenever it is unable to proceed to a new node without exceeding its vehicle capacity V. This idea is relatively unexplored; most reports in the literature tackle the CVRP by implementing an heuristic savings algorithm, which combines customers into tours following a greedy strategy [30, 37]. An ACO algorithm with a path scanning heuristic involving vehicle capacity was described in [27], solving a CARP for waste collection in an urban environment.

A matrix *tour* records the nodes visited by each ant, including the depot node for unloading. Our program uses an additional input vector $v(N)$ giving the amount of waste or 'demand' v_i to be collected at node i, and so within the ants' memory structure, we include a matrix *load* recording the vehicle load volume at each step of each ant's tour.

4.1. Programming

Let the graph involve N nodes where waste is to be collected, together with the depot as node 0. The depot node has zero waste demand, and is not included in the path node selection algorithm in Section 3.1. It is directly accessible from all nodes. While it is common to use 10 ants, we have used N ants, placed randomly at each node initially, because of the increased complexity of the route to be constructed. The tour matrix is expanded to allow space in the tour for all the collection nodes plus the required number of unloading trips *mtip* to the depot node. The latter is estimated by dividing the total demand by the vehicle capacity and rounding up to the next largest integer. This is valid provided each pair of nodes is connected by an arc of finite length. We have worked with volume demand and capacity, although weight collected could easily be included as an alternative or additional calculation. The first column of the matrix *load* storing the waste volume collected by each ant, is initialised as the demand at the ant's starting node.

As an ant moves from node to node, the volume of waste collected by it is updated in successive columns of *load*. The ant selects a new node to move to using (1) or (2), from among the feasible nodes (unvisited and directly accessible) for which collection from that node will not exceed the vehicle capacity. If no collection nodes can be chosen without exceeding capacity, the ant moves instead to the depot node to unload. In this case the ant's waste volume in *load* is set to zero.

To resume its tour from the depot node, the ant selects at random an unvisited node to move to; this was found to be better than selecting the nearest unvisited node, or the furthest. When all nodes have been visited the ant makes a final move to the depot node to unload. Each ant's tour is then improved by applying the 2-opt algorithm separately to each sub-tour (i.e. each loop from the depot and back to unload), and the resulting total tour length calculated. Note that a tour consisting of K sub-tours can equivalently be considered as a solution with K vehicles each making a simple tour loop from the depot and back. Pheromone updating is not performed on arcs to or from the depot node.

The algorithm pseudocode is summarised as follows:

```
INPUT $N$ (no. of nodes), $M$ (no. of ants), $miter$ (no. of iterations)
INPUT parameters $\alpha, \beta, q_0, \rho, \xi$
INPUT coordinates of nodes 1 to $N$ and of depot node 0
Form matrix of arc lengths $dist(N,N)$ including node 0, and of heuristic
function $\eta(N,N)$
INPUT vector of nodal demands $v(N)$ and vehicle capacity $V$
$mtip$ = ceiling($\Sigma v_i/V$) {no. of unloading trips required}
$mstep = N + mtip$ {tour length (without depot at start and end)}
Calculate $\tau_0$ as in (3) and initialise pheromone matrix $\tau(N,N)$
Store values of $\tau_{ij}\eta_{ij}^{\beta}$ in separate matrix $choice(N,N)$ {to avoid repeated
recalculation in (2)}
Zero $tour(M, mstep)$ {each ant's tour of nodes}
Zero $load(M, mstep)$ {vehicle load at each step of each ant's tour}
Allocate ants randomly to starting nodes: ant $k$ starts at node $start(k)$
**for** $iter = 1$ to $miter$ **do** {iteration loop}
  Initialise tabu list {nodes visited so far by ant $k$}
  **for** $k = 1$ to $M$ **do** {loop over ants}
    $i = start(k)$ {ant starts its tour by moving from depot to node $i$}
    Add node $i$ to tabu list for ant $k$
    $tour(k,1) = i$
    $load(k,1) = v(i)$
  **end for**
  **for** $istep = 1$ to $mstep$ **do** {loop over each step of tour}
```

```
    if $istep < mstep$ then {tour not yet completed}
      for $k = 1$ to $M$ do {loop over each ant}
        $i = tour(k, istep)$ {ant $k$ currently at node $i$}
        Select next node $n$ in tour as follows:
        if $i = 0$ then {ant is at depot node}
          Select $n$ at random from feasible nodes
        else
          Choose random variable $q \in [0, 1]$
          Compile list of feasible nodes which can be visited without
          exceeding vehicle capacity, i.e. nodes $j$ for which
          $load(k, istep) + v(j) < V$
          if $\mathcal{N}_i^k$ is empty then
            $n = 0$ {return to depot to unload}
          else if $q > q_0$ then {use equation (1)}
            Calculate probabilities $p_{ij}$ for these nodes
            Choose $n$ by roulette wheel selection
          else {use equation (2)}
            Choose node $n$ which has the maximum value of $choice(i, j)$
          end if
        end if
        $tour(k, istep + 1) = n$
        if $n = 0$ then
          $load(k, istep + 1) = 0$ {vehicle has unloaded}
        else
          $load(k, istep + 1) = load(k, istep) + v(n)$
          Add node $n$ to tabu list for ant $k$
        end if
        Local pheromone updating using (3)
      end for
    end if
  end for
  Add depot node to start and (if necessary) end of each ant's tour
  Apply 2-opt improvement to each sub-tour (from depot and back to
  depot) of each tour
  Find length of each tour using $tour$ and $dist$
  Record shortest tour and apply global pheromone updating (4)
  Update best tour found so far
end for
```

```
PRINT best tour
RETURN
```

4.2. Clustering

For networks in which the nodes naturally form clusters (corresponding to centres of population), performance can be improved by first clustering the nodes, and then applying the ACS to each cluster in turn. We first determine the number of clusters by dividing the total demand by vehicle capacity and rounding up, as in Section 4.1. When there are additional constraints, a more sophisticated calculation uses an Integer Linear Programming (ILP) formulation. This model assumes a fixed number K of vehicles, and so the algorithm is run repeatedly with increasing values of K until a feasible solution is found. For the cost function we use the total time spent by vehicles on unloading trips to the depot. This model is based on the Linear Programming (LP) models of [39]; further details including the constraints used are in [40]. In a solution the program will output the number of unloading trips to be made, and the number of households to be serviced, by each vehicle.

For the clustering we employed the in-built k-means clustering algorithm within MATLAB. This uses the geographical coordinates of the nodes, iteratively allocating nodes to the cluster whose centroid is closest to that node, but does not take into account the nodal demands. We therefore need to adjust the clusters in order to balance the total load of each cluster, so that each cluster can be serviced by a single vehicle, making unloading trips when necessary. The load-balancing algorithm first sums the total demand in each cluster, calculates the average demand per cluster, and identifies the clusters with excess demand. It then tries to move an outlying node of the cluster with greatest excess demand to a neighbouring cluster, so that the total excess demand of the network is reduced. Once a node has been moved, the algorithm recalculates the centroids and cluster demands, and iterates as in the k-means algorithm. If load balancing cannot be achieved such that the total demand of each cluster does not exceed the vehicle capacity, the whole algorithm is restarted with the number of clusters increased by 1.

The final clustering information was fed into the ACS algorithm, which was run separately on each cluster. As will be seen in the next Section, the solution of several smaller ACS problems rather than a single large problem results in a much reduced processing time.

4.3. Routing vehicles with multiple compartments

We now seek to use ant colony optimisation to route a multi-compartment vehicle as described in Section 1. It is straightforward to further extend the ACS algorithm described at the start of this section, by replacing the matrix *load* recording the waste collected by each ant, with a 3D array recording the waste volume collected in each compartment for each ant at each step of its tour. The capacities of each compartment are specified, and the demands at each node are split by waste category, e.g. glass, paper, plastic and metal. The programming principles are as described above, with each compartment load being updated as the ant moves from node to node. When it is unable to make a further move without one or more compartments becoming overfull, it returns to the depot where all compartments are unloaded, i.e. reset to zero. The required number of unloading trips is predicted by calculating *mtip* as in Section 4.1 for each compartment separately, and the maximum of these is used in allocating array space. However this may now be an underestimate, if the unloading trips are driven by one type of waste in one part of the network, and by another type elsewhere. Allowance should thus be made for longer tours being necessary (This is also the case for single-compartment problems where travel is not permitted on some arcs).

The algorithm has the following modifications to Algorithm 4.1:

- Line 1: Also input L (no. of waste compartments on vehicle)
- Line 5: INPUT matrix of nodal demands $v(N, L)$ and compartment capacities $V(L)$
- Line 6: *mtip* is maximum over l of $ceiling(\Sigma v_{il})/V_l$
- Line 10: Zero $load(M, mstep, L)$
- Line 18: $load(k, 1, l) = v(i, l)$ for $l = 1$ to L
- Line 29: Nodes must satisfy $load(k, istep, l) + v(j, l) < V(l)$ for $l = 1$ to L
- Line 41: $load(k, istep + 1, l) = 0$ for $l = 1$ to L
- Line 43: $load(k, istep + 1, l) = load(k, istep, l) + v(n, l)$ for $l = 1$ to L

While this extension to multiple compartments is straightforward, it is an open question whether the ACS algorithm will perform acceptably, or indeed converge at all. The experiments in the next section seek to answer this.

Table 1: Results on benchmark problems

Problem	Published Best Solutions	ACS with 2-opt		ACS with 2-opt and clustering	
	585	536.24	5	614.66	6
C1	556	524.61		608.10	
	524.61	546.36	138.63	624.05	17.69
	900	907.46	10	1005.79	11
C2	876	877.75		957.55	
	835.26	933.15	327.07	1041.93	24.18
	887	946.29	8	1038.83	8
C3	863	919.67		978.37	
	826.14	970.45	710.57	1089.68	38.86
		1418.03	8	1193.07	8
C11		1372.95		1136.69	
	1042.11	1448.79	1676.62	1300.49	39.55
		1240.49	10	1013.10	10
C12		1147.20		862.03	
	819.56	1294.59	1077.38	1153.39	37.10

5. Computational results

The ACS and clustering algorithms described in the previous section were programmed in MATLAB as part of successive MSc projects [40, 41]. The basic ACS was first tested on a selection of the freely available TSP examples at TSPLIB
(`http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95`)
and in all cases produced the optimal solution, although in the larger examples (of over 500 nodes) required large numbers of iterations to do so.

The VRP is an NP-hard problem. As an illustration of this, Reimann et al report [31] that while their ants algorithm could find the best known solution (BKS) of a 50-node VRP problem in under 3 seconds, for a problem with 199 nodes it took up to 40 minutes to obtain a solution within 2% of the BKS. Given this situation, and due to the time and resource limitations of the projects, the experimentation performed has the limited aim of demonstrating proof of concept: that ACS with 2-opt improvement and incorporating unloading trips can produce good solutions of small and medium-size single- and multi-compartment CVRP problems in a short time. We also want to examine the potential of clustering to improve solutions. We note that in a real-world application the priority is to produce a good-quality — even if sub-optimal — solution quickly, and that any computer-generated solution will anyway be reviewed and adjusted manually by schedulers [39].

5.1. Experimental setup

There is a set of 14 CVRP problems, collected by Christofides [9] which are widely used in the literature as benchmarks. Data files are available online at:
`www.rhsmith.umd.edu/faculty/bgolden/Christofides_benchmarks.zip`.

The data set for each problem consists of:

- nodal coordinates $\{(x_i, y_i), i = 1, N\}$
- depot coordinates (x_0, y_0)
- nodal demands $\{v_i, i = 1, N\}$
- vehicle capacity V.

We have selected five of these problems to test our ACS algorithm (Section 5.2) and the effect of clustering (Section 5.3), and have extended the first of these to generate test problems for the multi-compartment ACS (Section 5.4).

Description of problems: Problems C1, C2, C3 contain $50, 75, 100$ nodes respectively, which are located randomly within the square $0 < x, y < 100$. Each pair of nodes is connected by an arc of length given by the Euclidean distance between them. Nodal demands are distributed randomly in the range $0 < v_i < 50$. For C1/C2/C3 the total demands are 777/1364/1458, and the vehicle capacities 160/140/200, meaning that the solutions should involve 5/10/8 unloading trips respectively. In the last two problems C11/C12 the dimension is 120/100, with the nodes generally located in well-defined clusters. Further description is given in Section 5.3.

Published solutions: Problems C1–C3 were solved in the 1960's by two methods — the first using the Clarke and Wright savings algorithm and the other using 3-opt improvement. The best solutions (shortest total tour length) are reported in [9] and are given here as the upper and middle values in the cells of column 2 of Table 1. In 1993 improved solutions for the Christofides benchmarks were found by Taillard [11] using a parallel Tabu Search method, and these are still the best known solutions. These are the lower values in column 2 (also for C11 and C12).

Parameters used: In all runs the ACS heuristic parameters were set as: $\alpha = 1, \beta = 2$, $q_0 = 0.9$, $\xi = \rho = 0.1$, as recommended in [22, 42].

We used $M = N$ ants, i.e. one ant is placed at each node of the network. We set a maximum of 2000 iterations because of the project limitations; the consequences of this choice are discussed below. For each problem, ten runs of the program were performed, and we summarise the results in columns 3 and 4 of Table 1. The best, average and worst tour lengths obtained are in the top left, centre and bottom left of each cell. The CPU time taken (elapsed seconds, using the MATLAB function) is in the bottom right, and in the top right of the cell is the number of sub-tours in the best solution.

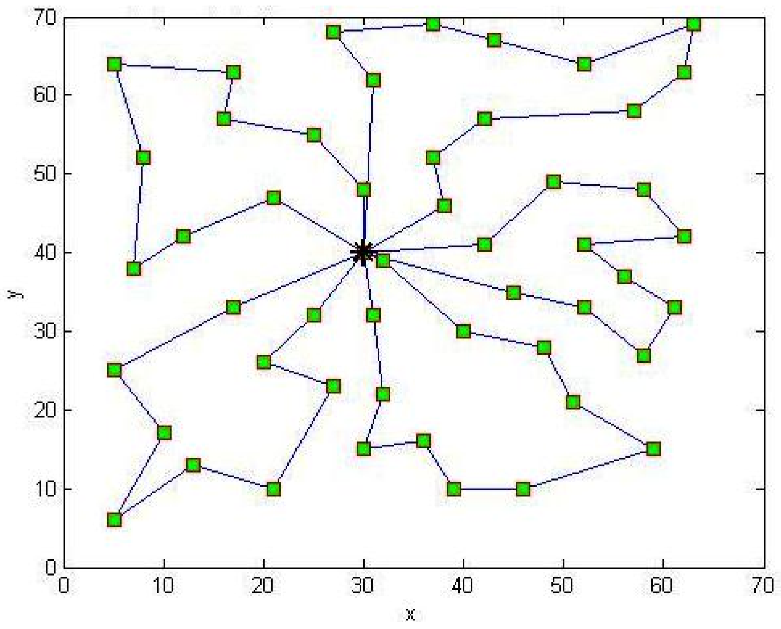

Fig. 1: Best solution for problem C1

5.2. Unstructured networks (Problems C1–C3)

The 50-node problem C1 was solved extremely well by the ACS algorithm. As can be seen in Table 1, one run produced the best known solution (BKS)

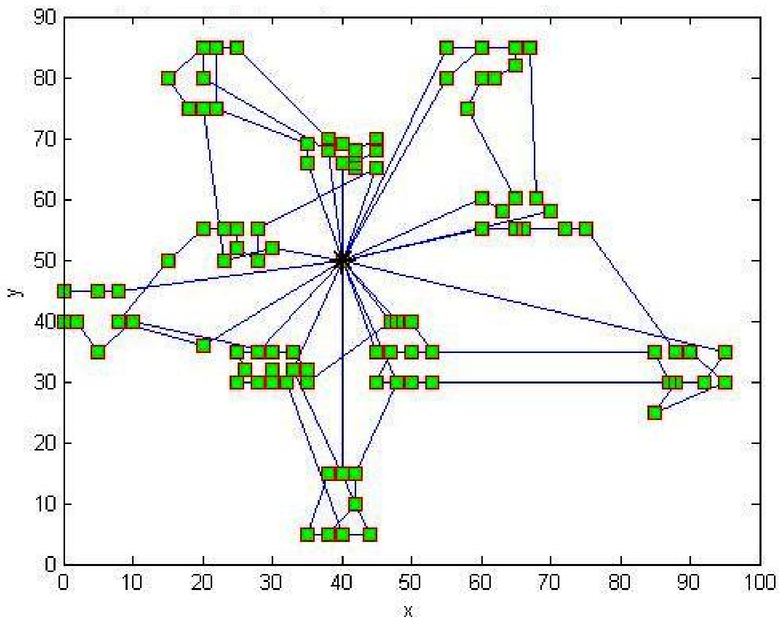

Fig. 2: Best solution for problem C12 without clustering

with optimal tour length 524.61 (Figure 1), and did so within 700 iterations. Even the worst solution found after 2000 iterations was within 4.1% of the BKS length.

When the algorithm was run without 2-opt improvement, the resulting solution lengths for problem C1 were in the range 573.21–620.06 (10 runs, 2000 iterations), but even then, in the best solution found, only one node was allocated to the wrong sub-tour, and all but one sub-tour was correctly constructed (graphs in [41] Appendix G).

On the larger problems, the algorithm performance worsened: for C2 (75 nodes) the best solution (877.75) was 5.1% away from the BKS, although comparable to the best reported using a 3-opt algorithm. For C3 (100 nodes) the best solution (919.67) was 11.3% from the BKS after 2000 iterations. Given the excellent performance on C1 and the scale-up experience of Reimann quoted above, we conclude that this degradation with problem size is due to the NP-hard nature of the CVRP problem, rather than any fault with the algorithm itself.

Not unsurprisingly, use of the clustering algorithm on these unstructured networks resulted in much poorer results (column 4 of Table 1). On problem

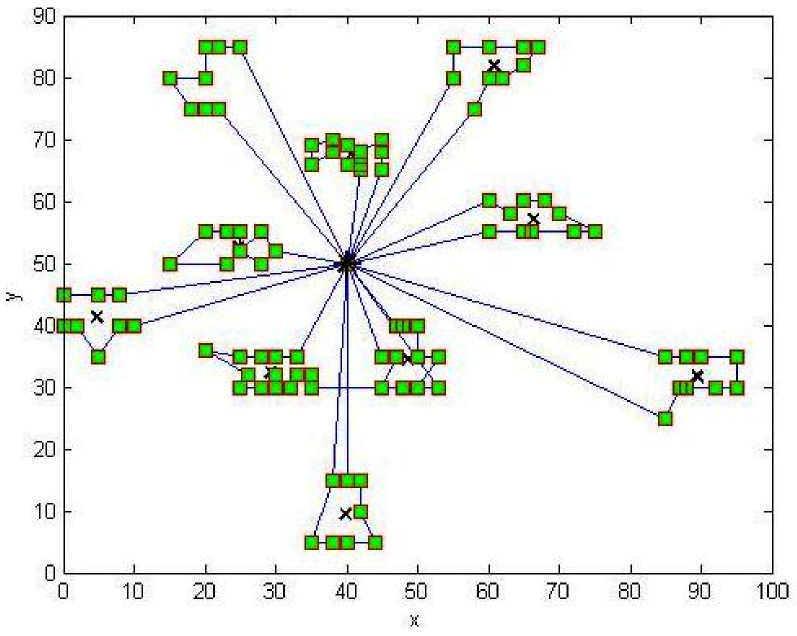

Fig. 3: Best solution for problem C12 with clustering

C1 the clustering algorithm did identify three sub-tours reasonably correctly. However on this and C2 the load-balancing algorithm forced the creation of an extra cluster. However, the fact that several small single-subtour problems were being solved by ACS separately, did result in much shorter run times (18–39 seconds, compared to 139–711 seconds without clustering) for the same number of iterations.

5.3. Structured networks (problems C11, C12)

Problem C12 has 100 nodes grouped in ten well-defined clusters, well-separated from the centrally-placed depot node (Figure 2). Each cluster is serviceable in a single sub-tour. The standard ACS algorithm is much slower in converging to a solution as compared to the same-size unstructured network C3: best solution (1147.20) is 40% away from BKS compared to 11.3% for C3 after 2000 iterations. However, when the k-means clustering algorithm is used (last cell of column 4 of Table 1) the performance is much better than for C3 (862.03 is 5.2% from BKS). In the graph of this solution (Figure 3) only one inter-cluster arc is included, and this arises because the cluster “at 5 o’clock” has a total demand exactly equal to the vehicle capacity, so one or more nodes have been moved by the load-balancing algorithm. The solution was found about 30 times faster than in the algorithm without clustering.

Problem C11 is larger (120 nodes) and the clustering is less clear-cut. There are five well-defined outlying clusters, with the remaining nodes closely surrounding the depot node located at one edge of the region (graph is in [41] Appendix G). This means that the clustering algorithm needs to add some of the surrounding nodes to each outlying cluster, and then form the remaining clusters from an unstructured data-set. Despite this, performance using clustering is still good (1136.69 is 9.1% from BKS). A more sophisticated decomposition algorithm, as in [31], would be needed in this situation.

5.4. Multi-compartment problems

To test the ACS algorithm for multi-compartment vehicles, we have constructed a pair of two-compartment problems from problem C1. The vehicle capacity of $V = 160$ units is split into a larger compartment of $V^1 = 120$ units and a smaller one of $V^2 = 40$ units; let us say that these are collecting glass and paper respectively. For simplicity we assume both compartments are unloaded at the depot. The demands v_i at each node have also been split into glass v_i^1 and paper v_i^2. Clearly, if these nodal demands were all in the ratio 3:1 the compartments would fill up at the same rate and the algorithm should perform as in the single-compartment case. In Problem C1A we have used this demand ratio at all nodes except for those in the lower left corner of the region (nodes with both coordinates in Figure 1 in the range $0 < x, y < 35$). In this region the demands are split in the ratio 2:1. As there is greater demand for paper collection in this corner region, meaning that this vehicle compartment will fill up faster, we hope to see shorter sub-tours appearing there, while the sub-tours in the rest of the graph should resemble those for the optimal single compartment case (Figure 1).

In Problem C1B we have again used the ratio 2:1 for nodes in the lower left corner, but a ratio 4:1 for the split at the remaining nodes. Now we hope to see the paper compartment filling first and triggering an unloading trip for sub-tours in the corner, while the glass compartment will fill first on the other sub-tours.

The ACS program was run ten times on each problem, again allowing 2000 iterations, and the best results found for problems C1A and C1B are shown in Figures 4 and 5 respectively. The total tour lengths were 560.74 and 564.04, which are within 7.5% of the optimal 524.61 for the single compartment problem, despite the increased constraints which have led to an additional sub-tour being required to service all nodes. In both cases the extra final sub-tour involves just two nodes, but these are nodes very close to the depot, so that the extra routing is minimised. Comparing Figure 4 with Figure 1, we do indeed see that the sub-tours away from the lower left corner in problem C1A closely resemble the optimal single compartment solution. Table 2 shows the progress of the filling of the two compartments in problem C1B for each sub-tour (starting empty from, and ending with unloading at, the depot at node 51). In the second and fifth sub-tours the smaller compartment has filled first, while in the first, third and fourth sub-tours it is the larger compartment which triggers the unloading. In all cases except for the final sub-tour, the critical compartment is almost completely full before returning to unload. This demonstrates the efficiency of the solution.

In terms of convergence, the algorithm performs just as efficiently as for the single compartment problem, with the final tour length being about 70% of the initial length, and graphs of tour length against iteration number (not shown here) looking very similar, with the best solution reached after 500–800 of the permitted 2,000 iterations. More detailed results are given in [41].

6. Conclusions

While the ACS algorithms' performance for the larger problems would have been improved by allowing a larger number of iterations, and by tuning the ACS parameters, the results from this simple feasibility study are sufficient to demonstrate the effectiveness of the method. The Ant Colony System incorporating unloading trips is competitive with other metaheuristics for the Capacitated Vehicle Routing Problem, and more significantly, an extension is able to solve multi-compartment problems with equal efficiency. The MCVRP solutions produced match the sub-tours of the best known solution in regions where the compartmentalisation has less effect, and produce efficient sub-tours (i.e. only unloading when the critical compartment is almost full) in all regions. Morevoer, the rate of convergence is the same as with the single-compartment data. Further improvement could be made by including other constructive heuristics such as the Clarke and Wright algorithm.

As local authorities and private recycling companies increasingly purchase fleets of multi-compartment vehicles for kerbside sorting, the multi-

Table 2: The cumulative vehicle load collected in each sub-tour, for problem C1B

Nodes	48	23	24	43	7	26	8	31	28	22	1	46	51
Glass	5.6	12	23.2	32	47.2	52.8	71.2	84.8	97.6	106.4	114.4	118.4	0
Paper	1.4	3	5.8	8	11.8	13.2	17.8	21.2	24.4	26.6	28.6	29.6	0
Nodes	4	41	40	19	42	44	45	33	15	37	17	51	
Glass	6	14.67	25.33	33.33	51.73	59.733	65.733	67.733	73.733	91.733	96.4	0	
Paper	3	7.33	12.67	14.67	19.27	21.27	24.27	25.27	28.27	37.27	39.6	0	
Nodes	32	2	3	36	35	20	29	16	11	51			
Glass	9.6	24.8	36.8	41.6	64	77.6	82.4	95.2	119.2	0			
Paper	2.4	6.2	9.2	10.4	16	19.4	20.6	23.8	29.8	0			
Nodes	5	49	10	39	30	34	21	50	9	38	51		
Glass	8.8	20.8	37.6	52	56	67.2	82.4	103.2	109.6	117.6	0		
Paper	2.2	5.2	9.4	13	14	16.8	20.6	25.8	27.4	29.4	0		
Nodes	27	6	14	25	13	18	51						
Glass	12	24	40.8	63.2	78.53	105.87	0						
Paper	3	6	10.2	15.8	23.47	37.13	0						
Nodes	47	12	51										
Glass	16.67	36	0										
Paper	8.33	18	0										

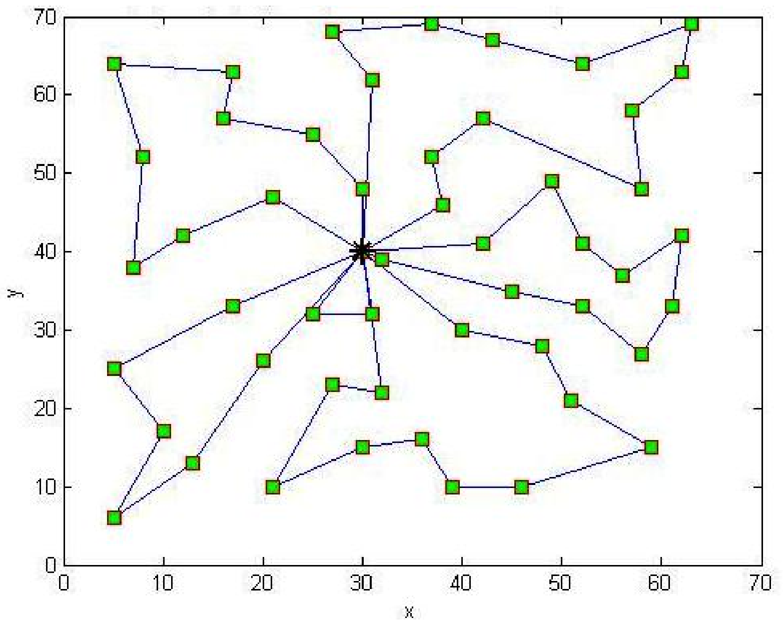

Fig. 4: Best solution for problem C1A

compartment CVRP will grow in importance. There is an immediate need for further studies of this problem, including the development of benchmark problems and the use of alternative metaheuristics.

While the clustering algorithm has improved solutions and run times for structured problems, a more sophisticated algorithm needs to be developed, which integrates the nodal demands within the clustering process, and which is itself hybridised with the ACO solution process.

Practical extensions of the simple two-compartment model considered here include the use of more compartments, and the location of the depot site separately from the vehicle depot (and possibly separate disposal sites for the different waste categories). Sensitivity analyses are also important, as the amounts of waste in the 'green boxes' set out by households cannot be accurately predicted, and will vary from one week to the next.

7. Acknowledgements

We are grateful to Maxine von Eye and her colleagues at Eunomia Research and Consulting for their involvement, including provision of data and technical guidance on aspects of recycling collection.

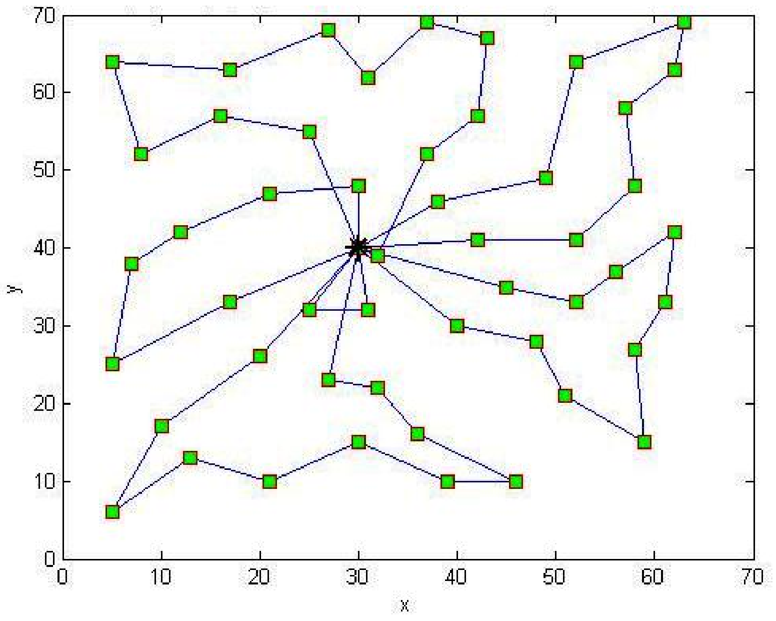

Fig. 5: Best solution for problem C1B

References

[1] L. Bodin, B. Golden, A. Assad, M. Ball, Routing and scheduling of vehicles and crews: The state of the art, Computers & Operational Research 10 (1983) 63–211.

[2] G. Laporte, P. Toth, D. Vigo, Vehicle routing: historical perspective and recent contributions, EURO Journal Transport Logistics 2 (2013) 1–4.

[3] X. Zhang, G.H. Huang, X. Nie, Y. Chen, Q. Lin, Planning of municipal waste management under dual uncertainties, Waste Management and Research 28 (2010) 673–684.

[4] E. Angelelli, M.G. Speranza, The application of a vehicle routing model to a waste collection problem: two case studies, Journal of the Operational Research Society 53 (2002) 944–952.

[5] M. Jahre, Logistics systems for recycling efficient collection of household waste. Ph.D. thesis, Chalmers University of Technology, Sweden 1995.

[6] S. Apotheker, Kerbside collection: Complete separation versus commingled collection, Resource Recycling (1990) October

[7] B. Platt, J. Zachary, *Co-collection of recyclables and mixed waste: Problems and opportunities*, Institute for Local Self-Reliance, ILSR Washington, DC, 1992.

[8] D. Pisinger, S. Ropke, A general heuristic for vehicle routing problems, Computers & Operational Research 34 (2007) 2403–2435.

[9] N. Christofides, A. Mingozzi, P. Toth, The vehicle routing problem, in: N. Christofides (Ed.), *Summer School in Combinatorial Optimization*, Wiley, 1979, pp. 315–338.

[10] N. Christofides, S. Eilon, An algorithm for the Vehicle-dispatching problem, Journal of the Operational Research Society 20 (1969) 309–318.

[11] E. Taillard, Parallel iterative search methods for vehicle routing problems, Networks 23 (1993) 661–673.

[12] M. Genreau, A. Hertz, L. Laporte, A tabu search heuristic for the vehicle routing problem, Management Science 40 (1994) 1276–1290.

[13] I.H. Osman, Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem, Annals of Operational research 41 (1993) 421–451.

[14] S.R. Thangiah, A hybrid genetic algorithms, simulated annealing and tabu search heuristic for vehicle routing problems with time windows, in: L. Chambers (Ed.), *Practical Handbook of Genetic Algorithms, vol. III: Complex Structures*, CRC Press, 1999, pp. 347–381.

[15] B.M. Baker, M.A. Ayechew, A genetic algorithm for the vehicle routing problem, Computers & Operational Research 30 (2003) 787–800.

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

[17] K.C. Tan, Y.H. Chew, L.H. Lee, A Hybrid multi-objective evolutionary algorithm for solving vehicle routing problem with time windows, Computational Optimization and Applications 34 (2005) 115–151.

[18] B. Ombuki, B.J. Ross, F. Hanshar, Multi-objective genetic algorithms for vehicle routing problem with time windows, Applied Intelligence 24 (2006) 17–30.

[19] K. Ghoseiri, S.F. Ghannadpour, Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm, Applied Soft Computing 10 (2010) 1096–1107.

[20] Z. Ursani, D. Essam, D. Cornforth, R. Stocker, Localised genetic algorithm for vehicle routing problem with time windows, Applied Soft Computing 11 (2011) 5375–5390.

[21] M.R. Khouadjia, B. Sarasola, E. Alba, L. Jourdan, E.-G. Talbi, A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests, Applied Soft Computing 12 (2012) 1426–1439.

[22] M. Dorigo, T. Stützle, *Ant Colony Optimization*, MIT Press, USA, 2004.

[23] S. Mazzeo, I. Loiseau, An ant colony algorithm for the capacitated vehicle routing problem, Electronic Notes in Discrete Mathematics 18 (2004) 181–186.

[24] N.V. Karadimas, K. Papatzelou, V.G. Loumos, Optimal solid waste collection routes identified by the ant colony system algorithm, Waste Management and Research 25 (2007) 139–147.

[25] N.V. Karadimas, G. Kouzas, I. Anagnostopoulos, V. Loumos, Urban solid waste collection and routing: the ant colony strategic approach, International Journal of Simulation: Systems, Science and Technology 6 (2007) 45–53.

[26] J.E. Bell, P.R. McMullen, Ant colony optimization techniques for the vehicle routing problem, Advanced Engineering Informatics 18 (2004) 41–48.

[27] J. Bautista, J. Pereira, Ant algorithms for urban waste collection routing, Lecture Notes in Computer Science 3172 (2004) 386–403.

[28] A.E. Rizzoli, R. Montemanni, E. Lucibello, L.M. Gambardella, Ant colony optimization for real-world vehicle routing problems, Swarm Intelligence 1 (2007) 135–151.

[29] X. Zhang, L. Tang, A new hybrid ant colony optimization algorithm for the vehicle routing problem, Pattern Recognition Letters 30 (2009) 848–855

[30] M. Reimann, M. Stummer, K. Doerner, A savings based ant system for the vehicle routing problem, in: *Proceedings of the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann Publishers, USA, 2002, pp. 1317–1326.

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

[32] M. Reimann, S. Shtovba, E. Nepomuceno, A hybrid ant colony optimization and genetic algorithm approach for vehicle routing problems solving, *Student Papers of Complex Systems Summer School-2001*, Budapest (2001) 134–141.

[33] L. Van der Brugen, R. Gruson, M. Salomon, Reconsidering the distribution of gasoline products for a large oil company, European Journal of Operation Research, 81 (1995) 460–473.

[34] P. Avella, M. Boccia, A. Sforza, Solving a fuel delivery problem by heuristic and exact approaches, European Journal of Operational Research, 152 (2004) 170–179.

[35] E.D. Chajakis, M. Guignard, Scheduling deliveries in vehicles with multiple compartments, Journal of Global Optimisation, 26 (2003) 43–78.

[36] A. El Fallahi, C. Prins, R. Wolfler Calvo, A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem, Computers & Operations Research, 35 (2008) 1725–1741.

[37] B. Bullnheimer, R.F. Hartl, C. Strauss, An improved ant system algorithm for the vehicle routing problem, Annals of Operations Research, 89 (1999) 319–328.

[38] M. Muyldermans, G. Pang, On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm, European Journal of Operational Research 206 (2010) 93–103.

[39] S. Sahoo, S. Kim, B.I. Kim, B. Kraas, A. Popov Jr., Routing optimization for waste management, Interfaces 35 (2005) 24–36.

[40] R. Evering, *Optimising a Waste Collection Service*, MSc Project Report, Dept. of Mathematical Sciences, University of Bath, UK, 2011.

[41] A. Yiannakou, *An extended model of the recycling waste collection process*, MSc Project Report, Dept. of Mathematical Sciences, University of Bath, UK, 2012.

[42] E. Bonabeau, M. Dorigo, G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. Oxford University Press, USA 1999.