Computers & Operations Research 37 (2010) 2270–2280

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities

A.M. Benjamin, J.E. Beasley *

CARISMA, Mathematical Sciences, Brunel University, Uxbridge UB8 3PH, UK

ARTICLE INFO

Available online 1 April 2010

Keywords:
Waste collection
Vehicle routing
Tabu search
Variable neighbourhood search
Variable neighbourhood tabu search

ABSTRACT

In this problem there is a set of waste disposal facilities, a set of customers at which waste is collected and an unlimited number of homogeneous vehicles based at a single depot. Empty vehicles leave the depot and collect waste from customers, emptying themselves at the waste disposal facilities as and when necessary. Vehicles return to the depot empty. We take into consideration time windows associated with customers, disposal facilities and the depot. We also have a driver rest period. The problem is solved heuristically. A neighbour set is defined for each customer as the set of customers that are close, but with compatible time windows. A procedure that attempts to fully utilise a vehicle is used to obtain an initial solution, with this initial solution being improved using an interchange procedure. We present two metaheuristic algorithms using tabu search and variable neighbourhood search that are based around the neighbour sets. We also present a metaheuristic based on variable neighbourhood tabu search, where the variable neighbourhood is searched via tabu search. Computational results are presented for publicly available waste collection problems involving up to 2092 customers and 19 waste disposal facilities, which indicate that our algorithms produce better quality solutions than previous work presented in the literature.

1. Introduction

In this paper we consider a vehicle routing problem that arises when a set of customers have waste that must be collected by vehicles. In such situations it is common for the amount of waste to be such that vehicles become full during their working day and have time to visit a waste disposal facility to empty themselves before going on to visit more customers and collect more waste. As such multiple visits to waste disposal facilities may be made during the working day. This problem is a (single period) node routing problem and is often encountered in terms of waste collection from commercial (e.g. retail) customers. The collection of waste from households, because of the large number of households that typically have to be visited, is often dealt with as an arc routing problem (see [1–4] for more as to arc routing problems).

In the problem considered in this paper we have an unlimited number of identical (homogeneous) vehicles based at a single depot. Vehicles start/end their routes at the depot empty. We have multiple disposal facilities, so that decisions must be made not only as to when a vehicle should empty itself at a disposal facility, but also which disposal facility it should use. We also have time windows, one associated with each customer that governs when waste can be collected from that customer; another associated with each disposal facility that governs when that facility is open; another associated with the depot that governs when it is open to dispatch/receive vehicles. Each vehicle has a driver rest period (associated with a lunch break during the working day), and a maximum amount of work it can do during the day (both in terms of the total amount of waste collected and the total number of customers dealt with). Our problem is a single period problem, so we are not considering a periodic routing problem where we have to design routes over multiple periods.

This paper is organised as follows: in Section 2 we review the relevant literature on the problem. In Section 3 we present our notation for the problem and define the neighbour sets for each customer. The neighbour set for a customer is those customers that are closest to it, but with compatible time windows. In this section we also present our procedures for: obtaining an initial solution; route evaluation; route improvement; and vehicle reduction. In Section 4 we present our metaheuristic algorithms using tabu search and variable neighbourhood search as well as our metaheuristic based on variable neighbourhood search, but where the neighbourhood is searched via tabu search. In Section 5 we present computational results for publicly available waste collection problems and in Section 6 we present some conclusions.

* Corresponding author. Tel.: +44 1895 266219; fax: +44 1895 269732.
E-mail addresses: aida.benjamin@brunel.ac.uk (A.M. Benjamin), john.beasley@brunel.ac.uk (J.E. Beasley).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.03.019

2. Literature survey

In this section we survey the literature as to relevant previous work dealing with the problem of routing vehicles so as to collect waste from customers. Note that we only consider node routing papers, excluding for reasons of space arc routing papers from our review.

A number of papers in the literature deal with the collection of containers/skips such as are commonly used for construction site waste. Here the vehicle is involved in collection of full skips (that have to be taken to the waste disposal facilities) from customers, and delivery of empty skips to customers. The distinguishing feature of problems of this type is that the vehicle can typically only carry one or two skips at a time, hence the number of customers that can be visited before the vehicle has to go to a disposal facility is similarly limited. This container/skip collection problem is also known as a rollon–rolloff problem (echoing the movement of skips on/off the vehicle). In our survey below we distinguish papers dealing with problems of this type from papers dealing with other problems where the vehicle can visit "many" customers before a visit to a waste disposal facility has to occur. Note here that Sbihi and Eglese [5] have discussed the importance attached to waste management and collection in terms of the "green logistics" agenda.

2.1. Container/skip problems

De Meulemeester et al. [6] dealt with the problem of delivering empty skips and collecting full skips from customers. Vehicles can carry only one skip at a time, but skips can be of different types. They stated that the problem was first considered by Cristallo [7]. Their solution approach is based on two simple heuristics and an enumerative approach. They reported computational experience with randomly generated problems involving up to 160 customers and a real-world problem involving 30 customers.

Bodin et al. [8] considered a sanitation routing problem they called the rollon–rolloff vehicle routing problem. In this problem trailers, in which waste is collected, are positioned at customers. A tractor (vehicle) can move only a single trailer at a time. Tractor trips involve, for example, moving an empty trailer from the disposal facility to a customer and collecting the full trailer from the customer. A key aspect of their work is that they assume that the set of trips to be operated is known in advance (so the problem reduces to deciding for these trips how they will be serviced by the tractors). They presented four heuristic algorithms and gave computational results for problems involving up to 199 trips and a single disposal facility.

Archetti and Speranza [9] developed a heuristic algorithm called SMART-COLL for a problem motivated by waste collection in Brescia, Italy. In their problem skips are collected from customers and the vehicle can carry only one skip at a time. They call the problem the 1-skip collection problem. They considered skips of different types and time windows are imposed on both the customers and the disposal facilities. Computational experience was reported for real-world data involving 51 customers and 13 disposal facilities.

Baldacci et al. [10] dealt with an extension of the problem considered by Bodin et al. [8]. They considered multiple disposal facilities as well as inventory facilities at which empty trailers are available. They presented an approach based on regarding the problem as a time constrained vehicle routing problem on a directed multigraph. Computational results for problems involving up to 75 customers and two disposal facilities were presented.

Le Blanc et al. [11] presented a paper dealing with the collection of containers from end-of-life vehicle dismantlers in the Netherlands. In the problem they considered vehicle can carry two containers at a time. Their heuristic is a two-steps procedure, first generating candidate routes, then selecting from these routes using a set partitioning approach. They reported potential cost savings of over 18% compared with the current system.

2.2. Other problems

In terms of other (non-skip) problems the majority of papers in the literature are case study papers, focusing on results obtained when algorithms are applied to real-world data. Only a few of these papers report computational experience with publicly available waste collection test instances.

Tung and Pinnoi [12] proposed a heuristic procedure to solve a waste collection problem in Hanoi, Vietnam. In their problem there are time windows associated with collection from customers and their heuristic first constructs routes based on an approach due to Solomon [13] and then improves them. They reported computational experience indicating that they can achieve an operating cost saving of 4.6% when compared with the current situation.

Angelelli and Speranza [14] presented an algorithm based on tabu search for the periodic version of the problem where routes must be designed over a planning horizon of more than one time period so as to meet customer service requirements. Their approach is based on the tabu search algorithm for vehicle routing of Cordeau et al. [15]. Computational results were presented for problems involving between two and six days in the planning horizon.

Angelelli and Speranza [16] proposed a model that fits three different waste collection systems to estimate operational costs. Their solution procedure is based on Angelelli and Speranza [14] and results were presented relating to two case studies: Val Trompia, Italy and Antwerp, Belgium.

Sahoo et al. [17] reported how they developed a system called WasteRoute to reduce operating costs for a large company involved in waste collection. They gave one example of an area that went from ten routes to nine, improving route productivity (as measured by amount collected per hour) by some 11%.

The heuristic used for the WasteRoute system of Sahoo et al. [17] is fully described in Kim et al. [18]. Customers have time windows for collection, and there are multiple disposal facilities, as well as a driver rest period. They extended Solomon's [13] insertion heuristic to cope with both multiple disposal facility visits and the driver rest period and used it to construct routes, which are improved using simulated annealing and a local search exchange procedure called CROSS [19]. As their work is motivated by the practical context reported in [17] they discussed a number of issues with solutions produced by this heuristic: route compactness, workload balancing and computation time. In order to deal with these issues they also presented a heuristic based on capacitated clustering that generates clusters based on the estimated number of vehicles required, and then routes customers within each cluster. Computational results were presented for ten problem instances, derived from real-world data, involving up to 2100 customers that the authors make publicly available.

Nuortio et al. [20] considered a problem based on waste collection in two regions of Eastern Finland. Their problem includes time windows and they solved the problem using Guided Variable Neighbourhood Thresholding [21].

Ombuki-Berman et al. [22] presented a multi-objective genetic algorithm that uses a crossover procedure (Best Cost Route Crossover) from Ombuki et al. [23]. They reported results from

their approach using the test problems of Kim et al. [18], but no computation times were given.

Alagöz and Kocasoy [24] considered health waste collection in Istanbul. They used a commercial vehicle routing package to consider a number of scenarios relating to the type of facility used for waste disposal. McLeod and Cherrett [25] considered a problem relating to waste collection in the UK. They used a commercial vehicle routing package and reported that vehicle mileage could be reduced by up to 14%.

Hemmelmayr et al. [26] presented a paper motivated by a real-world waste collection problem. They consider a periodic problem, where routes must be designed over a multi-day planning horizon so as to meet customer service requirements. They consider a number of constraints motivated by their underlying application and in particular in their application the vehicle need not return to the depot empty. They use dynamic programming to sequence disposal facility visits within a variable neighbourhood search approach. Computational results are presented for instances, involving up to 288 customers, derived from vehicle routing problems given in the literature.

Repoussis et al. [27] considered waste oil collection and recycling in Greece. In their problem vehicles are compartmentalised and they use a list based threshold accepting metaheuristic [28] to design vehicle routes. They reported reductions of up to 30% in the cost per unit of waste collected.

2.3. Discussion

In this paper we consider exactly the same waste collection problem as in Kim et al. [18] involving multiple disposal facilities, driver rest period and customer/depot/disposal facility time windows. Because Kim et al. [18] have made their test problems publicly available we can make a direct computational comparison with their work.

Although our paper focuses directly on waste collection from customers we would briefly mention here that in the context of deliveries to customers an analogous problem is the vehicle routing problem with intermediate replenishment facilities. In problems of this type there are intermediate facilities at which vehicles can replenish/restock with the goods that they need to satisfy demand at customers they have yet to visit before they finally return to the depot at the end of the working day.

An important difference between the collection problem with disposal facilities and the delivery problem with replenishment facilities is that in the collection problem a vehicle visits a disposal facility to empty itself immediately prior to returning to the depot. In the delivery problem there is typically no visit to a replenishment facility for restocking immediately prior to returning to the depot (or conversely immediately after leaving the depot). More as to the delivery problem with intermediate replenishment facilities can be found in Kek et al. [29], Tarantilis et al. [30], Crevier et al. [31] and Angelelli and Speranza [14].

3. Initial solution and its improvement

In this section we first present our notation and define neighbour sets. We then present our procedure to find an initial solution and our procedure to evaluate a given route, which involves inserting into the route (if necessary) disposal facility visits. Finally we present procedures to improve a solution, both in terms of the distance travelled and in terms of the number of vehicles used.

Note here that in our work in evaluating the quality of any given set of feasible vehicle routes we evaluate them using total distance travelled. We do not constrain the distance travelled by vehicles, rather we constrain vehicle operating times via the depot time window. Amending the algorithms presented below to deal with vehicle distance constraints is however a simple task.

3.1. Notation

Let C be the set of customers and D be the set of disposal facilities (recall we have just a single depot). To represent the depot, disposal facilities and customers we index them such that 0 is the depot, 1, 2, …, $|D|$ are the disposal facilities and $|D|+1$, $|D|+2$, …, $|D|+|C|$ are the customers. The travel time between i and j is denoted by t_{ij} and the distance between them by d_{ij} (where i and j may be the depot, disposal facilities or customers). Our notation may be conveniently structured as that relating to the vehicles or customers/disposal facilities/depot. For the vehicles let

- Q be the vehicle capacity, so a vehicle filled to this capacity has to be emptied at a disposal facility before any other customer can be visited;
- Q^* be the maximum amount a vehicle can deal with per day (over all customers);
- S^* be the maximum number of customers the vehicle can deal with per day and
- $[R_1, R_2]$ be the single lunch (rest) time window so that the lunch/rest period must start at some time within this period, the rest duration (how long the vehicle/driver is idle) being R_3.

For the customers/disposal facilities/depot let

- q_i be the quantity to be collected at customer $i \in C$;
- V_i be the service time for $i \in C \cup D$ such that the visit to i (for collection/disposal) takes this (fixed) time and
- $[E_i, L_i]$ be the time window for $i \in C \cup D \cup \{0\}$ such that the visit to $i \in C \cup D$ (for collection/disposal) must start within this time period and the vehicles must leave/return to the depot within $[E_0, L_0]$.

In terms of our heuristic we need to identify the nearest (in terms of travel time) open disposal facility for customer i at time T. We denote this by $n(i, T)$ and it is defined by

$$n(i,T) = \arg\min[t_{ij} | j \in D, T + t_{ij} \in [E_j, L_j]]$$

so the disposal facility associated with customer i at time T is the nearest facility that is open should the vehicle go directly from customer i to the facility. Note here that computationally we do not calculate $n(i, T)$ for all values of i and T, rather we calculate $n(i, T)$ as and when needed in the heuristic below.

3.2. Neighbour sets

A neighbour set of cardinality K for customer i, denoted by $N(i, K)$, is composed of the K customers that are closest to customer i, but with compatible time windows. A customer j is defined in our work to have a compatible time window with customer i (and hence potentially belongs to $N(i, K)$) if it is possible to visit i at some time in its time window, service i and then go directly onto j to service j without waiting for its time window to open.

As the time window for i is $[E_i, L_i]$ the time window for arrival at j after servicing i is $[E_i+V_i+t_{ij}, L_i+V_i+t_{ij}]$. Customer j potentially belongs to the neighbour set for i if this time window overlaps with its time window $[E_j, L_j]$. Two time windows overlap if and only if an end point of one time window falls within the other

time window. We can therefore define the neighbour sets using

$\forall i \in C$: set $N(i,K) = \varnothing$
  $\forall i \in C, \forall j \in C (j \neq i)$:
    if $E_j \leq E_i + V_i + t_{ij} \leq L_j$ or $E_j \leq L_i + V_i + t_{ij} \leq L_j$ or
    $E_i + V_i + t_{ij} \leq E_j \leq L_i + V_i + t_{ij}$ or $E_i + V_i + t_{ij} \leq L_j \leq L_i$
  $+ V_i + t_{ij}$ set $N(i,K) = N(i,K) \cup j$

To ensure that $N(i, K)$ has appropriate cardinality then if after the above calculation we have $|N(i, K)| > K$ we alter $N(i, K)$ to contain only the K nearest customers to i, i.e. sort the customers in $N(i, K)$ in increasing order of their travel time (t_{ij}) from i and set $N(i, K)$ to contain just the first K customers from this ordered list. Note here that we may have $|N(i, K)| < K$ if there are fewer than K customers that have compatible time windows with i.

Neighbour sets are a key element in our work. This arises for two reasons

- the nature of our metaheuristics, as will become apparent below, is that we use neighbour sets in seeking to improve a route. As such the larger the value of K the larger the neighbourhood we search; and
- by varying K we have a variable neighbourhood. This leads in a natural fashion to applying variable neighbourhood search to the problem.

Note here that $N(i, K)$ can be computed before we embark on route construction.

3.3. Initial solution

We construct an initial solution by attempting to fully utilise a vehicle over the day (thereby aiming to minimise the total number of vehicles used). Once a vehicle cannot be used any more then we start a new vehicle route with a new vehicle. To deal with the vehicle/driver rest period we attempt to schedule it as early as possible consistent with its time window. Our procedure involves a number of steps, as below.

Initialise
Set $B = C$ (B is a working set of customers that have still to be routed)
Step 1
if $|B| \neq 0$ so there are still customers to be dealt with **then**:
  start a new vehicle route at time E_0, when the depot opens

$T = E_0$	T is the current time
$s_{total} = 0$	s_{total} is the total number of customers the vehicle has visited
$Q_{total} = 0$	Q_{total} is the total load the vehicle has dealt with
$Q_{current} = 0$	$Q_{current}$ is the current load on the vehicle
$r = 0$	r is the customer at the end of the current emerging vehicle route
rest = 0	rest is one if the vehicle has had its rest period, else zero

else
  all customers have been dealt with so **stop**
end if
Step 2
Check for the rest period—here we start the rest period as soon as practicable
If the vehicle has not had a rest period (rest = 0) and $T \in [R_1, R_2]$ **then**:
  the vehicle now has its rest period

rest = 1	update rest
$T = T + R_3$	update the current time

end if
Step 3
The next customer to be visited on the current emerging route is that customer $i \in B$ such that
$i = \arg\min[t_{rj} | j \in B, T + t_{rj} \in [E_j, L_j], Q_{current} + q_j \leq Q, Q_{total} + q_j \leq Q^*, s_{\text{total}} + 1 \leq S^*, \theta + t_{j,n(j,\theta)} + V_{n(j,\theta)} + t_{n(j,\theta),0} \leq L_0, \theta \leq R_2$ if rest = 0, where $\theta = T + t_{rj} + V_j$]

This customer i is that customer that has the shortest travel time from the customer r at the current end of the route, provided i satisfies the conditions seen above. This expression is relatively complex. Here we consider only those customers j such that when the vehicle arrives at j (at time $T + t_{rj}$) it will be possible to service the customer as the visit will fall in its time window $[E_j, L_j]$ and the load to be collected at j will fit on the vehicle (in terms of the current route, the entire days work and the total number of customers visited). Also j has to be a customer such that if j is visited there is time for the vehicle to visit the nearest (open) disposal facility to j and then return to the depot before the end of the working day. In the above expression θ is the time at which the vehicle finishes servicing j, then the vehicle travels to the nearest open disposal facility $n(j, \theta)$, taking time $t_{j,n(j,\theta)}$, the disposal facility visit takes time $V_{n(j,\theta)}$, and then the vehicle travels to the depot taking time $t_{n(j,\theta),0}$, arriving before the end of the working day (at time L_0). Furthermore j has to be such that if the vehicle has not yet had its rest period (rest = 0) there is still time after servicing j for the rest period to be started ($\theta \leq R_2$).
For this step preliminary computational experience indicated that one issue which arises is that we want to avoid excess travel simply because the vehicle has some limited spare capacity. In order to gauge this we compare the travel time from the end of the current route to i (i.e. t_{ri}) to the travel time to the nearest disposal facility $n(r, T)$, i.e. $t_{r,n(r,T)}$. If $t_{ri} > t_{r,n(r,T)}$ and the vehicle is near to capacity (in our work a vehicle is defined to be near to capacity if $\max[Q_{current}/Q, Q_{total}/Q^*, s_{total}/S^*] > 0.8$) then we disregard i (i.e. we treat this situation as if we had found no customer satisfying the above expression).
If there is a customer i satisfying the above expression that we can add to the end of the emerging route **then:**
  the vehicle travels to i and services the customer

$T = T + t_{ri} + V_i$	update the current time
$r = i$	update the current customer at the end of the route
$Q_{current} = Q_{current} + q_i$	update the current vehicle load
$Q_{total} = Q_{total} + q_i$	update the daily vehicle load
$s_{total} = s_{total} + 1$	update the total number of customers visited
$B = B - \{i\}$	update the set of customers B by removing i from it

  go to step 2
end if
Step 4
We reach this step when we have not found a customer to add to the end of the emerging route.
If vehicle is not empty ($Q_{current} > 0$) **then:**
  the vehicle travels to its nearest disposal facility $n(r, T)$ to be emptied

$T = T + t_{r,n(r,T)} + V_{n(r,T)}$	update the current time
$r = n(r,T)$	update the end of the route
$Q_{current} = 0$	update the current vehicle load

  go to step 2
end if

Step 5
We reach this step when we have not found a customer to add to the end of the emerging route and the vehicle is empty ($Q_{current}=0$). In this case no more work can be done with this vehicle at the current time. It may be possible that the vehicle can do some more collections if it is idle until a time arrives such that it is possible to collect from some customer.
To deal with this situation we look for the customer whose time window will "open" as soon as possible:

$$i = \arg\min[E_j | j \in B, T + t_{rj} < E_j, Q_{current} + q_j \le Q, Q_{total} + q_j \le Q^*, s_{total} + 1 \le S^*, \theta + t_{j,n(j,\theta)} + V_{n(j,\theta)} + t_{n(j,\theta),0} \le L_0, \theta \le R_2 \text{ if rest} = 0, \text{where } \theta = E_j + V_j]$$

this expression is as above except that here the vehicle is travelling to j, arriving at $T+t_{rj}$ and waiting until time E_j to start the collection

If there is a customer i satisfying the above expression **then:**
the vehicle travels to i, waits until E_i, and then services the customer

$T=E_i+V_i$	
$r=i$	update the current time
$Q_{current}=$ $Q_{current}+q_i$	update the current customer at the end of the route
$Q_{total}=Q_{total}+q_i$	update the current vehicle load
$s_{total}=s_{total}+1$	update the daily vehicle load
$B=B-\{i\}$	update the total number of customers visited
go to step 2	update the set of customers B by removing i from it

else
the vehicle travels back to the depot (as it is empty it does not need to visit a disposal facility first) and a new vehicle must now be used
go to step 1
end if

In step 1 above we start a new vehicle route and initialise the various counters we need to keep track of the use made of the vehicle. In step 2 we attempt to schedule the rest period. In step 3 we add a customer to the end of the emerging route such that it is feasible to add the customer both in terms of the vehicle load and in terms of "look-ahead" for the vehicle to return to the depot empty. In step 4 the vehicle is emptied whilst in step 5 the vehicle waits for a customer time window to open. The above procedure terminates once all of the customers have been dealt with.

Here we have set out our initial solution procedure in detail in order that the reader can clearly see the steps involved and the counters that are updated as a route is constructed. For the remainder of this paper we, for brevity, use higher level pseudocode in terms of presenting our algorithms.

3.4. Route evaluation

In this section we indicate how we evaluate a given route. One complication here is that in the local search procedure we present later below we move customers between routes. If we move a customer onto a route then it is possible that the route after addition of the customer will be infeasible when we evaluate it (e.g. because the vehicle exceeds its collection capacity). However if we were to schedule into the route an extra disposal facility visit the route may become feasible. Preliminary computational experience indicated that incorporating extra disposal facility visits was of benefit, and so in evaluating a given route we allow such extra visits to be incorporated.

In a similar fashion there may be benefit in allowing the time at which the rest period occurs to vary from the time that was initially scheduled as we evaluate a route and so we also allow this to change (although the rest period must still occur within its time window).

In evaluating a given route we regard it as comprising a fixed sequence of places-starting at the depot, then a mix of customers and disposal facilities, finally a disposal facility (to empty the vehicle), followed by the depot. In pseudocode our procedure for route evaluation is

Start the route at time E_0 (set $T=E_0$)

Repeat until all places on the route have been dealt with:

- Perform Step 2 above to schedule the rest period if possible
- If travelling to the next place in the fixed sequence would exceed the vehicle capacity then schedule in an extra visit to the nearest disposal facility; formally suppose the current place at the end of the route is customer r and the current time is T, then insert a visit to disposal facility $n(r, T)$ at this point in the sequence
- Travel to the next place (customer/disposal facility/depot) in the fixed sequence
- If the vehicle arrives before the time window for the place opens then wait until the time window opens (we wait as we are trying to operate the sequence)
- Deal with this place (collection or disposal or arrival back at the depot).

As we run through the fixed sequence we update the current time T, also keeping track of the loads—where the procedure returns INFEASIBLE if at any point we violate the constraints of the problem (e.g. vehicle load exceeded or the vehicle arrives after the time window for a place has closed), otherwise the procedure returns FEASIBLE (also returning the sequence used since we may have added extra disposal facility visits, and the total distance associated with the route).

3.5. Route improvement—local search

In order to improve routes we adopt a local search procedure. We have two different phases associated with this procedure:

- moving customers/disposal facilities elsewhere on the same route; also changing disposal facilities on the same route; and
- interchanging the positions of two customers in the routes.

We deal with each of these in turn.

3.5.1. Phase 1

In this phase we evaluate repositioning customers/disposal facilities. In pseudocode we

For all customers $i \in C$:
For all customers $j \in N(i, K)$ such that j is on the same route as i:
- Move j immediately before/after i (here we check positioning j before/after i)
- Evaluate this new route with j moved to this new position and if it is better than the original route (FEASIBLE and of lower total distance) then keep it, else do not

end for
end for

For all routes:
For all disposal facilities $i \in D$ on the route:
- Remove i from the route
- Add i to every possible position on the route in turn
- Evaluate this new route with i added to this new position

and if it is better than the original route (FEASIBLE and of lower total distance) then keep it, else do not
end for
end for

The logic here is that for computational reasons we restrict attention in terms of moving customers to positioning them before/after those customers on the same route that are in $N(i, K)$, i.e. near to i and with compatible time windows.

In the above we simply reorder the sequence of places on a route. Preliminary computational experience indicated that we could improve routes by changing disposal facilities. This can happen (for example) if we have a disposal facility visited as the last place on the vehicle route before travel back to the depot, but in fact there is a better disposal facility to use in terms of travel back to the depot. In addition it is worthwhile to check for whether disposal facilities on the route can be removed (since in the route evaluation procedure we only ever add disposal facilities, never remove them). Therefore as part of this phase we also do:

For all routes:
For all disposal facilities $i \in D$ on the route:
- Remove disposal facility i from the route
- Evaluate this new route with the disposal facility removed and if it is better than the original route (FEASIBLE and of lower total distance) then keep it, else do not

end for
end for
For all routes:
For all disposal facilities $i \in D$ on the route:
For all disposal facilities $j \in D$, $j \neq i$:
- Replace disposal facility i on the route by disposal facility j
- Evaluate this new route with the disposal facilities changed and if it is better than the original route (FEASIBLE and of lower total distance) then keep it, else do not

end for
end for
end for

3.5.2. Phase 2

In this phase we interchange customers between vehicle routes. Here we use the neighbour set for a customer to prevent the number of customer interchanges we have to examine being excessive. In pseudocode we

For all customers $i \in C$:
For all customers $j \in N(i, K)$:
If i and j are on different routes (serviced by a different vehicle) **then:**
- we interchange customers i and j, i.e. customer i moves to the position that customer j occupied on its route and customer j moves to the position that customer i occupied on its route
- Evaluate the two routes that are involved in this interchange. If both are FEASIBLE and their total distance is $<$ the total distance for the two routes before the interchange then keep the interchange, else do not

end if
end for
end for

Computationally we repeat phases 1 and 2 in turn until no further improvement can be achieved. We will then have a locally optimal solution.

3.6. Vehicle reduction procedure

Our solution procedure has no direct control over the number of vehicles used, although as discussed above it attempts to minimise the number of vehicles used by utilising a vehicle as much as possible. Examination of preliminary computational results indicated that, for some problems, the number of customers serviced on the last vehicle route constructed was so small that (given judicious rearrangement of customers on earlier routes) it might well be possible to reduce the number of vehicles used.

In order to try and reduce the number of vehicles used we can therefore adopt a procedure whereby we attempt to move customers from the last vehicle route constructed (since that effectively corresponds to the least utilised vehicle) to earlier routes (provided that this is feasible, and irrespective of the effect on distance travelled). In pseudocode we

Repeat until no more customers can be moved from the last route:
For all customers $i \in C$ that are on the last route:
- Add i to every possible position on every other route in turn
- Evaluate this new route with i added to this new position and if it is FEASIBLE then keep it, else do not

end for
Perform phases 1 and 2 above, but excluding from consideration in those phases the last route (since we are seeking to eliminate all customers from that route)
end repeat

If all customers have been moved from the last route then keep the routes else do not

The logic here is that we, provided it is feasible, move customers off the last route to earlier routes, where we use phases 1 and 2 to reorder customers on these earlier routes (thereby potentially enabling further customers to be moved off the last route). Note here that if we are already using a minimal number of vehicles, as is the case if $\max[|C|/S^*, \sum_{i \in C} q_i/Q^*]$ (when rounded up to the nearest integer) is equal to the number of vehicles used, there is no point in applying this procedure.

3.7. Summary

In this section we have outlined our initial solution procedure and how we can improve the initial solution using the two phases discussed above. We also discussed how we can reduce the number of vehicles used. In the next section we discuss our metaheuristic algorithms for the problem.

4. Metaheuristics

In this section we discuss our metaheuristic algorithms for the problem using tabu search (TS), variable neighbourhood search (VNS) and a combined algorithm (VNTS) based on variable neighbourhood search, but where the neighbourhood is searched via tabu search. We shall assume here for reasons of space that the reader is familiar with both TS and VNS. Further information with regard to TS can be found in Glover and Laguna [32,33] and Gendreau [34]. For further information relating to VNS see Mladenović and Hansen [35], Hansen and Mladenović [36].

4.1. Tabu search (TS)

In our TS heuristic the move that we consider is an interchange of two customers, who may or may not be on the same route. This move differs from that in phase 2 above since in that phase we only considered customers that were on different routes. Here we again use the neighbour set for a customer to prevent the number of customer interchanges we have to examine being excessive. We do these interchanges however in a tabu search framework (so we allow interchanges that worsen the solution).

In our approach we apply tabu status to customers. So if a customer is tabu it cannot be considered for any possible move. Note that we do not consider disposal facilities with regard to using tabu. This is because of the fact that a disposal facility may appear on more than one route, and so considering disposal facilities for tabu would entail keeping track of which route they are on (since we may wish to move a disposal facility on one route but leave it in its current position, i.e. tabu, on another route). Customers, by contrast, can only be on one route. For our TS heuristic let

- Δ be the tabu tenure, how long a customer stays tabu for
- M be an iteration counter
- $\delta(i)$ be the last iteration at which customer i was moved, we use $\delta(i)$ to judge whether moving i is tabu or not
- $Z_{current}$ be the current solution value (this being the solution from which we are examining potential moves)
- Z_{move} be the solution value associated with the move we are currently examining
- Z_{best} be the value of the best solution we have encountered during our algorithm (before we start TS both $Z_{current}$ and Z_{best} will be the value of the locally optimal solution as derived in the previous section above)
- m be a counter of the number of times we examine all pairs of customers without improving Z_{best}
- Z_{non} be the value of the solution associated with the "best" non-improving move, and α and β be the customers associated with this best non-improving move
- ϕ be a diversification factor such that any non-improving solution from $Z_{current}$ that we consider has to have value $\geq Z_{current}+\phi$.

The role of ϕ is to diversify the solution by forcing the new solution after a non-improving move to be "far" from the current solution.

Limited computational experience indicated that appropriate values for Δ and ϕ were 7 and $Z_{best}/(20|C|)$ respectively. In pseudocode our TS heuristic is

Initialise
Set $M=m=0$ (counters set to zero); $\delta(i)=-(\Delta+1)\ \forall i\in C$ (this ensures that all customers are not tabu)
Step 1
Set $Z_{non}=\infty$; set $m=m+1$; set flag=0, this is a flag to signify whether we have changed Z_{best} or $Z_{current}$ during this step
For all customers $i\in C$:
For all customers $j\in N(i, K)$:
- Interchange customers i and j, i.e. customer i moves to the position that customer j occupied on its route and customer j moves to the position that customer i occupied on its route (these customers may be on the same route, may be on two different routes) and evaluate the routes that result
- **If** the route(s) involved in this interchange are not FEASIBLE **then** disregard the interchange and go to consider a new pair, i.e. go to DONEPAIR

Here the route(s) are FEASIBLE, check for:
- improving Z_{best}, irrespective of tabu status (so aspiration)
- improving $Z_{current}$ (if not tabu)
- a better non-improving move Z_{non} (if not tabu)

Improving the best solution (aspiration)
If the total distance, Z_{move}, associated with the entire (feasible) solution after interchange (so considering not just any routes involved in the interchange but also any routes not involved) is strictly less than Z_{best} (i.e. $Z_{move}<Z_{best}$) **then**:
we keep the interchange (i.e. keep the moved customers where they are); update Z_{best} to the value of this new improved solution, so $Z_{best}=Z_{move}$; update the current solution, so $Z_{current}=Z_{best}$; set the tabu status for customers i and j using $\delta(i)=\delta(j)=M$; set $M=M+1$; reset the value of the solution for the best non-improving move $Z_{non}=\infty$; set $m=0$ as Z_{best} has been improved; set flag=1 to indicate that the solution has changed; and go to DONEPAIR to consider a new pair.
end if
Tabu status (check for whether the customers are tabu)
If $|M-\delta(i)|\leq\Delta$ or $|M-\delta(j)|\leq\Delta$ (so the move is tabu) then go to DONEPAIR.
Improving the current solution
If $Z_{move}<Z_{current}$ (so the move improves on the current solution $Z_{current}$) **then**:
update the current solution, so $Z_{current}=Z_{move}$; set the tabu status for customers i and j using $\delta(i)=\delta(j)=M$; set $M=M+1$; reset the value of the solution for the best non-improving move $Z_{non}=\infty$; set flag=1 to indicate that the solution has changed; and go to DONEPAIR to consider a new pair
end if
Better non-improving move
If $Z_{move}<Z_{non}$ (so the move improves on the current non-improving move Z_{non}) and $Z_{move}\geq Z_{current}+\phi$ (so the move is sufficiently far from $Z_{current}$) **then**:
set the best non-improving move solution $Z_{non}=Z_{move}$ and record the customers associated with this best non-improving move using $\alpha=i$, $\beta=j$
end if
DONEPAIR:
end for
end for
Step 2
Terminate if sufficient iterations have been performed without improving the best solution Z_{best}. In our work we stop if $m=5$.
If flag=1 **then**:
we have made a change (either to Z_{best} or to $Z_{current}$), so go to Step 1
else
we have not made a change so we make the best non-improving move. In other words we interchange the customers α and β associated with the best non-improving move; update the current solution $Z_{current}$ to the solution after the move; set the tabu status for customers α and β using $\delta(\alpha)=\delta(\beta)=M$; set $M=M+1$; and go to Step 1.
end if

4.2. Variable neighbourhood search (VNS)

In our VNS heuristic we consider the same move as in our TS heuristic above. However whilst that heuristic operates with a fixed value of K, the number of neighbours a customer has, in our VNS heuristic we vary K. In our VNS heuristic define $K^*=\{$set of values of K we will consider$\}$. As for TS we start from the locally

optimal solution as derived in the previous section above. In terms of neighbourhood search (for a specified value of K) we use the same neighbourhood as in our TS heuristic. However, unlike our TS heuristic we only accept moves that improve the best solution Z_{best}. Our VNS heuristic is:

Set $\Gamma = K^*$ (initialise the set of K values we will consider)
while $|\Gamma| \neq 0$ so there are still values of K to consider:
Set $K = \min[k \,|k \in \Gamma]$ to choose the smallest value from Γ and set $\Gamma = \Gamma - \{K\}$
For all customers $i \in C$:
For all customers $j \in N(i, K)$:
- Interchange customers i and j and evaluate the routes that result
- **If** the solution after interchange is FEASIBLE and better than Z_{best}
  then: accept the move, update Z_{best}; set $\Gamma = K^*$ (as we have improved the solution we are willing to reconsider all possible values of K) **else** disregard the interchange

end for
end for
end while

Our VNS heuristic terminates when we have a solution that cannot be improved by any move associated with any of the K values in K^*.

4.3. Variable neighbourhood tabu search (VNTS)

In our VNTS heuristic we adopt the same variable neighbourhood as in our VNS heuristic above. However, whilst our VNS heuristic searches each neighbourhood for improved solutions, in VNTS we allow non-improving moves, i.e. we search each neighbourhood in a TS fashion. Our VNTS heuristic is:

Set $\Gamma = K^*$ (initialise the set of K values we will consider)
while $|\Gamma| \neq 0$ so there are still values of K to consider:
Set $K = \min[k|k \in \Gamma]$ to choose the smallest value from Γ and set $\Gamma = \Gamma - \{K\}$
Apply our TS heuristic with this value of K starting from Z_{best}
If the best solution after applying TS has improved **then** set $\Gamma = K^*$
end while

Our VNTS heuristic terminates when we have a solution that cannot be improved by TS associated with any of the K values in K^*.

5. Computational results

The metaheuristics presented in this paper were coded in C++ and run on a 3.16 GHz pc (Intel Core2 Duo) with 3.23 Gb memory. We solved the same test problems as solved in Kim et al. [18], involving up to 2092 customers and 19 waste disposal facilities, as publicly available at: http://www.postech.ac.kr/lab/ie/logistics/WCVRPTW_Problem/benchmark.html. These problems involve multiple disposal facilities, driver rest period and customer/depot/disposal facility time windows. A number of these test problems contain customers for which the amount to be collected is zero, but in our results we explicitly visit these customers (based on Kim [37]) to be comparable with the results of Kim et al. [18]. Note too here that for some of these test problems the daily vehicle capacity is such that the vehicle finishes its work and returns to the depot before the driver rest period (associated with a lunch break). For these problems we regard the driver rest period as being taken at the depot.

The results are shown in Table 1. In that table we show for each problem the number of customers and disposal facilities (further details as to these test problems can be found in [18]). Table 1 shows the results from Kim et al. [18] for their clustering heuristic (using simulated annealing), in terms of the number of vehicles used, total distance travelled and computation time. Note here that results are presented in Kim et al. [18] for an insertion heuristic (also using simulated annealing). However some of the insertion heuristic results reported are incorrect (involving for example fewer vehicles than can possibly be used) and based on Kim [37] we disregard these results. In any event for seven of the ten test problems the results for the clustering heuristic are (in terms of distance travelled) better than the results for the insertion heuristic.

Table 1 also shows the results for the metaheuristics presented in this paper, specifically the solution obtained by our initial solution procedure (denoted in Table 1 by IS), the solution after phases 1 and 2 (ISP1P2), TS, VNS and VNTS (when $K = 50$ and $K^* = \{5,10,25,50\}$). The last column in Table 1 gives the percentage improvement in distance when compared to the result of Kim et al., namely 100(Kim et al. solution distance − our solution distance)/(Kim et al. solution distance). Recall here that TS, VNS and VNTS all start from the solution given after phases 1 and 2 and the computation times given in Table 1 for each of these metaheuristics includes the time taken to generate this solution.

Examining Table 1 it is clear that our metaheuristic solutions (TS, VNS, VNTS) use less distance than those of Kim et al, on average approximately 5.6% less, with our solutions involving less distance for all but one of the ten test problems. With respect to the number of vehicles used our solutions involve (in total) 100 vehicles, those of Kim et al. 99 vehicles, so slightly worse.

It is clear from Table 1 that our three metaheuristics (TS, VNS and VNTS) produce routes of similar quality. On this basis we would be justified in choosing the metaheuristic involving the lowest computation time. From the averages presented at the foot of Table 1 it is clear that VNS is to be preferred, having a lower average time than either TS or VNTS.

Computation times for Kim et al. [18] in Table 1 are taken from their paper, and relate to a different computer than we used. Utilising Dongarra [38] it is possible to make an *approximate* estimate of the relative speed of the hardware involved. On this basis we estimate that the Kim et al. heuristic would (on average) require 31 s on our 3.16 GHz pc. So our heuristics take longer than the Kim et al. heuristic, but produce solutions involving significantly less distance. In any event the largest time seen in Table 1 for our chosen metaheuristic (VNS, 285 s for the 1932 problem with 1927 customers and 4 disposal facilities) is not especially large, approximately 5 min (equating to 0.15 s per customer), and this indicates that, in a practical setting, we would be well able to quickly produce routes on a daily basis if so required.

One issue that arises with respect to Table 1 is the added value provided by adopting our preferred metaheuristic, VNS, over our initial solution procedure in conjunction with phases 1 and 2 (ISP1P2). Above we presented the relatively complex procedures involved in ISP1P2, which (in our view) contribute to the success that ISP1P2 achieves when compared to Kim et al. One consequence of this success though is that there is then limited scope for further improvement when a metaheuristic such as VNS is applied (given that it starts from the ISP1P2 solution). From the averages at the foot of Table 1 VNS provides a reduction in distance compared to Kim et al. of some $100(5.64 - 5.43)/5.43 = 3.9\%$ over and above the reduction provided by ISP1P2. Given that the extra computation time needed by VNS is not excessive then we would argue that using VNS to achieve this extra distance reduction is worthwhile.

Table 1
Computational results.

Problem	Number of customers	Number of disposal facilities	Algorithm	Number of vehicles used	Total distance	Total computation time (s)	% Improvement in distance over Kim et al.
102	99	2	Kim et al.	3	205.1	3	
			IS	3	206.8	1	−0.83
			ISP1P2	3	183.5	2	10.53
			TS	3	183.5	4	10.53
			VNS	3	183.5	3	10.53
			VNTS	3	183.5	3	10.53
277	275	1	Kim et al.	3	527.3	10	
			IS	3	473.8	1	10.15
			ISP1P2	3	466.1	5	11.61
			TS	3	464.5	13	11.91
			VNS	3	464.5	8	11.91
			VNTS	3	464.5	8	11.91
335	330	4	Kim et al.	6	205.0	11	
			IS	6	213.3	2	−4.05
			ISP1P2	6	205.7	6	−0.34
			TS	6	204.6	16	0.20
			VNS	6	204.5	10	0.24
			VNTS	6	204.5	11	0.24
444	442	1	Kim et al.	11	87.0	16	
			IS	11	92.9	3	−6.78
			ISP1P2	11	89.2	13	−2.53
			TS	11	89.1	28	−2.41
			VNS	11	89.1	19	−2.41
			VNTS	11	89.1	18	−2.41
804	784	19	Kim et al.	5	769.5	92	
			IS	6	863.3	8	−12.19
			ISP1P2	6	757.5	39	1.56
			TS	6	756.3	73	1.72
			VNS	6	756.3	62	1.72
			VNTS	6	755.8	92	1.78
1051	1048	2	Kim et al.	18	2370.4	329	
			IS	17	2645.1	13	−11.59
			ISP1P2	17	2266.0	58	4.40
			TS	17	2250.5	116	5.06
			VNS	17	2251.6	124	5.01
			VNTS	17	2250.6	194	5.05
1351	1347	3	Kim et al.	7	1039.7	95	
			IS	8	984.3	20	5.33
			ISP1P2	8	915.4	68	11.96
			TS	8	915.1	162	11.98
			VNS	8	915.1	119	11.98
			VNTS	8	915.1	105	11.98
1599	1596	2	Kim et al.	13	1459.2	212	
			IS	14	1578.1	28	−8.15
			ISP1P2	14	1412.0	106	3.23
			TS	14	1410.4	223	3.34
			VNS	14	1410.4	172	3.34
			VNTS	14	1410.4	231	3.34
1932	1927	4	Kim et al.	17	1395.3	424	
			IS	16	1346.1	41	3.53
			ISP1P2	16	1264.4	187	9.38
			TS	16	1262.8	346	9.50
			VNS	16	1262.8	285	9.50
			VNTS	16	1262.8	335	9.50
2100	2092	7	Kim et al.	16	1833.8	408	
			IS	16	1823.6	49	0.56
			ISP1P2	16	1751.6	140	4.48
			TS	16	1749.0	332	4.62
			VNS	16	1749.0	266	4.62

Table 1 (*continued*)

Problem	Number of customers	Number of disposal facilities	Algorithm	Number of vehicles used	Total distance	Total computation time (s)	% Improvement in distance over Kim et al.
			VNTS	16	1749.0	356	4.62
Average			Kim et al.			160	
			IS			17	−2.40
			ISP1P2			62	5.43
			TS			131	5.65
			VNS			107	5.64
			VNTS			135	5.65

Table 2
Computational results, vehicle reduction procedure.

Problem	Number of customers	Number of disposal facilities	Algorithm	Number of vehicles used	Total distance	Total computation time (s)	% Improvement in distance over Kim et al.
804	784	19	Kim et al.	5	769.5	92	
			IS	6	863.3	8	−12.19
			ISP1P2	5	726.8	48	5.55
			TS	5	725.6	98	5.71
			VNS	5	725.6	72	5.71
			VNTS	5	725.6	90	5.71
1351	1347	3	Kim et al.	7	1039.7	95	
			IS	8	984.3	20	5.33
			ISP1P2	7	1012.2	161	2.64
			TS	7	1011.9	267	2.67
			VNS	7	1011.9	193	2.67
			VNTS	7	1011.9	199	2.67
1599	1596	2	Kim et al.	13	1459.2	212	
			IS	14	1578.1	28	−8.15
			ISP1P2	13	1366.3	186	6.37
			TS	13	1364.7	308	6.48
			VNS	13	1364.7	252	6.48
			VNTS	13	1364.8	321	6.47
Average			Kim et al.			160	
			IS			17	−2.40
			ISP1P2			81	5.21
			TS			153	5.43
			VNS			123	5.43
			VNTS			154	5.43

Results in Table 1 were produced without using our vehicle reduction procedure. To illustrate the effect of our vehicle reduction procedure we show in Table 2 the results obtained when it is applied (for reasons of space we only show in Table 2 those problems where a reduction in the number of vehicles was achieved). Note here that for two of the problems shown in Table 1 (problems 102 and 335) our solutions already use the minimal number of vehicles (as can be deduced from consideration of total customer demand and vehicle capacity).

Table 2 has the same format as Table 1, except that now we apply our vehicle reduction procedure to the routes that result from ISP1P2. For ease of comparison the averages shown at the foot of Table 2 are the averages over all ten problems, computed by combining the results for the three problems explicitly shown in Table 2 with the results shown in Table 1 for the other seven problems. Note here that the average time given at the foot of Table 2 includes the time for applying our vehicle reduction procedure to all problems (whether successful or not).

Considering Tables 1 and 2 then with respect to the number of vehicles used our solutions now involve (in total) 97 vehicles, those of Kim et al. 99 vehicles, so slightly better. As before it is clear that our metaheuristic solutions (TS, VNS, VNTS) use less distance than those of Kim et al., on average over these ten problems approximately 5.4% less. From the averages presented at the foot of Table 2 it is clear that VNS is still our preferred metaheuristic.

6. Conclusions

In this paper we considered a vehicle routing problem that arises when a set of customers have waste that must be collected by vehicles. Empty vehicles leave the depot and collect waste from customers, emptying themselves at waste disposal facilities as and when necessary (so typically a vehicle route would involve multiple disposal facility visits). In the problem we considered there were a significant set of constraints relating to real-world considerations. Specifically we took into consideration time windows associated with customers, disposal facilities and the depot. We also took into consideration a driver rest period.

We presented a number of metaheuristic approaches for the problem. Computational results were presented for publicly available waste collection problems involving up to 2092 customers and 19 waste disposal facilities which indicated that variable neighbourhood search was the most effective of these metaheuristics. With respect to solution quality our approaches provided better quality solutions than previous work presented in the literature.

References

[1] Golden BL, Wong RT. Capacitated arc routing-problems. Networks 1981;11: 305–15.
[2] Dror M, editor. Arc routing: theory, solutions and applications. Dordrecht, The Netherlands: Kluwer Academic Publishers; 2000.
[3] Eiselt HA, Gendreau M, Laporte G. Arc routing problems, Part I: the Chinese postman problem. Operations Research 1995;43:231–42.
[4] Eiselt HA, Gendreau M, Laporte G. Arc routing problems, Part II: the rural postman problem. Operations Research 1995;43:399–414.
[5] Sbihi A, Eglese RW. Combinatorial optimization and green logistics. 4OR—A Quarterly Journal of Operations Research 2007;5:99–116.
[6] De Meulemeester L, Laporte G, Louveaux FV, Semet F. Optimal sequencing of skip collections and deliveries. Journal of the Operational Research Society 1997;48:57–64.
[7] Cristallo G. Optimisation de Tournees de Vehicules de Transport Container. Memoire de licence en Sciences Economiques et Sociales, Facultes Universitaires Notre-Dame de la Paix, Namur, Belgium, 1994.
[8] Bodin L, Mingozzi A, Baldacci R, Ball M. The rollon–rolloff vehicle routing problem. Transportation Science 2000;34:271–88.
[9] Archetti C, Speranza MG. Vehicle routing in the 1-skip collection problem. Journal of the Operational Research Society 2004;55:717–27.
[10] Baldacci R, Bodin L, Mingozzi A. The multiple disposal facilities and multiple inventory locations rollon–rolloff vehicle routing problem. Computers & Operations Research 2006;33:2667–702.
[11] le Blanc I, van Krieken M, Krikke H, Fleuren H. Vehicle routing concepts in the closed-loop container network of ARN—a case study. OR Spectrum 2006;28: 53–71.
[12] Tung DV, Pinnoi A. Vehicle routing-scheduling for waste collection in Hanoi. European Journal of Operational Research 2000;125:449–68.
[13] Solomon MM. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 1987;35: 254–65.
[14] Angelelli E, Speranza MG. The periodic vehicle routing problem with intermediate facilities. European Journal of Operational Research 2002;137: 233–47.
[15] Cordeau JF, Gendreau M, Laporte G. A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 1997;30:105–19.
[16] Angelelli E, Speranza MG. The application of a vehicle routing model to a waste-collection problem: two case studies. Journal of the Operational Research Society 2002;53:944–52.
[17] Sahoo S, Kim S, Kim BI, Kraas B, Popov A. Routing optimization for waste management. Interfaces 2005;35:24–36.
[18] Kim BI, Kim S, Sahoo S. Waste collection vehicle routing problem with time windows. Computers & Operations Research 2006;33:3624–42.
[19] Taillard ED, Badeau P, Gendreau M, Guertin F, Potvin JY. A tabu search heuristic for the vehicle routing problem with soft time windows. Transportation Science 1997;31:170–86.
[20] Nuortio T, Kytöjoki J, Niska H, Bräysy O. Improved route planning and scheduling of waste collection and transport. Expert Systems with Applications 2006;30:223–32.
[21] Kytöjoki J, Nuortio T, Bräysy O, Gendreau M. An efficient variable neighbourhood search heuristic for very large scale vehicle routing problems. Computers & Operations Research 2007;34:2743–57.
[22] Ombuki-Berman BM, Runka A, Hanshar FT. Waste collection vehicle routing problem with time windows using multi-objective genetic algorithms. Presented at Computational Intelligence, Banff, July 2007. In: Andonie R, editor. Proceedings of Computational Intelligence (CI 2007). Calgary, Canada: Acta Press; 2007. p. 91–7.
[23] Ombuki B, Ross BJ, Hanshar F. Multi-objective genetic algorithms for vehicle routing problem with time windows. Applied Intelligence 2006;24:17–30.
[24] Alagöz AZ, Kocasoy G. Improvement and modification of the routing system for the health-care waste collection and transportation in Istanbul. Waste Management 2008;28:1461–71.
[25] McLeod F, Cherrett T. Quantifying the transport impacts of domestic waste collection strategies. Waste Management 2008;28:2271–8.
[26] Hemmelmayr V, Doerner KF, Hartl RF, Rath S. Metaheuristics for a real world solid waste collection problem. Working paper (2009) available from the third author at the Department of Business Administration, University of Vienna, Bruenner Strasse 72, 1210 Vienna, Austria, 2009.
[27] Repoussis PP, Paraskevopoulos DC, Zobolas G, Tarantilis CD, Ioannou G. A web-based decision support system for waste lube oils collection and recycling. European Journal of Operational Research 2009;195:676–700.
[28] Tarantilis CD, Kiranoudis CT, Vassiliadis VS. A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. European Journal of Operational Research 2004;152:148–58.
[29] Kek AGH, Cheu RL, Meng Q. Distance-constrained capacitated vehicle routing problems with flexible assignment of start and end depots. Mathematical and Computer Modelling 2008;47:140–52.
[30] Tarantilis CD, Zachariadis EE, Kiranoudis CT. A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. INFORMS Journal on Computing 2008;20:154–68.
[31] Crevier B, Cordeau JF, Laporte G. The multi-depot vehicle routing problem with inter-depot routes. European Journal of Operational Research 2007;176: 756–73.
[32] Glover F, Laguna M. Tabu search. In: Reeves CR, editor. Modern heuristic techniques for combinatorial problems. Oxford, UK: Blackwell Scientific Publications; 1993. p. 70–150.
[33] Glover F, Laguna M. Tabu search. Dordrecht, The Netherlands: Kluwer Academic Publishers; 1997.
[34] Gendreau M. An introduction to tabu search. In: Glover F, Kochenberger GA, editors. Handbook of metaheuristics. Dordrecht, The Netherlands: Kluwer Academic Publishers; 2003. p. 37–54.
[35] Mladenović N, Hansen P. Variable neighborhood search. Computers & Operations Research 1997;24:1097–100.
[36] Hansen P, Mladenović N. Variable neighborhood search: principles and applications. European Journal of Operational Research 2001;130:449–67.
[37] Kim BI. Private communication, 2009.
[38] Dongarra JJ. Performance of various computers using standard linear equations software. Report CS-89-85, University of Tennessee, USA, September 2009. Available from 〈http://www.netlib.org/benchmark/performance.ps〉 last accessed October 1, 2009.