Computers & Operations Research 40 (2013) 526–535

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

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

The time-dependent prize-collecting arc routing problem

Dan Black[a], Richard Eglese[b,*], Sanne Wøhlk[c]

[a] University of Edinburgh Business School, 29 Buccleuch Place, Edinburgh EH8 9JS, UK
[b] Department of Management Science, Lancaster University Management School, Lancaster LA1 4YX, UK
[c] CORAL—Cluster for OR And Logistics, Department of Economics and Business, Aarhus University, Fuglesangs allé 4, DK-8210 Aarhus V, Denmark

ARTICLE INFO

Available online 10 August 2012

Keywords:
Prize-collecting
Arc routing
Time-dependent
Full truck load

ABSTRACT

A new problem is introduced named the Time-Dependent Prize-Collecting Arc Routing Problem (TD-PARP). It is particularly relevant to situations where a transport manager has to choose between a number of full truck load pick-ups and deliveries on a road network where travel times change with the time of day. Two metaheuristic algorithms, one based on Variable Neighborhood Search and one based on Tabu Search, are proposed and tested for a set of benchmark problems, generated from real road networks and travel time information. Both algorithms are capable of finding good solutions, though the VNS approach generally shows better performance.

1. Introduction

The paper introduces a type of vehicle routing and scheduling problem that is relevant to a freight transport company with potential orders for full truck loads to be carried between pairs of pick up points and destinations. Fulfilling an order implies sending a suitable vehicle from the pick up point to the destination and might represent, for example, moving a full container or a tractor picking up a loaded trailer. In this static version of the problem, it is assumed that all potential orders are known before the vehicle starts from its depot and that all orders may be accepted or rejected. Fulfilment of an order results in a benefit or prize to the company which is also known. The objective of the problem is to decide which orders should be accepted and how the vehicle's route and schedule should be set to maximize the sum of the prizes from the accepted orders minus a travel cost proportional to the time taken to complete the accepted orders and return to the depot.

The approach taken is to construct a complete directed graph where one node corresponds to the depot and the other nodes represent pick up or delivery points. There is at least one directed arc between each pair of nodes and if there is more than one order between the same pair of pick up and delivery points, then this is represented by the corresponding number of parallel prize arcs between the same pair of nodes.

Before stating the problem to be studied in this paper, we first introduce the Prize-Collecting Arc Routing Problem which can be defined as follows.

Problem 1 (*PARP*). Given a directed graph $G(V,A)$ with a particular node designated as the depot node and with a cost, $c_{ij} > 0$ associated with each arc, let $p_{ij} \geq 0$ be a prize associated with the arc (i,j). This prize is collected the first time the arc is traversed. The goal is to construct a tour for one vehicle starting and ending at the depot node, which maximizes the sum of prizes collected minus total travel cost.

However it is common for the time required to travel between specified nodes not to be constant but to change according to the time of day and day of week. This information is increasingly collected by companies and local authorities using fixed traffic sensors on roads and/or floating vehicle data where the information is collected by tracking vehicles that have been fitted with equipment to report their location, direction and speed at regular intervals. This traffic information can be used with the underlying road network to provide a closer estimate of the time that it will take to travel between pairs of nodes. In some cases the route taken on the road network will change during the day as traffic conditions change.

This information can be used to provide more accurate and reliable plans. It is assumed that information is available to define the underlying road network and that the traffic information has been analyzed to provide the time to travel between any pair of pick up and destination points, or between any of those points and the depot. The travel times are not constant but vary according to the time of day as traffic congestion limits the speeds on different roads at different times.

This leads to the time-dependent prize-collecting arc routing problem which is defined as follows.

Problem 2 (*TD-PARP*). Given a directed multigraph $G(V,A)$ with a particular node designated as the depot node, let $\mathcal{P} \subseteq A$ be a set of

* Corresponding author. Tel.: +44 1524 593869; fax: +44 1524 592060.
E-mail addresses: Dan.Black@ed.ac.uk (D. Black), r.eglese@lancaster.ac.uk (R. Eglese), sanw@asb.dk (S. Wøhlk).

0305-0548/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2012.08.001

n profitable arcs and let $p_r \geq 0$ be a prize associated with arc $r \in \mathcal{P}$. This prize is collected the first time the arc is traversed. Furthermore, let $f_t(i,j)$ be the time it takes to travel from node i to node j starting at time t and let $f_t(r)$ be the time it takes to traverse the rth profitable arc starting at time t. The goal is to construct a tour for one vehicle starting at the depot at time $T_{\min}$ and ending at the depot node by time $T_{\max}$, which maximizes the sum of prizes collected minus total travel cost. Waiting is not permitted initially at the depot node nor at any nodes on the route.

The paper is organised as follows. In the next section, a brief review of previous work on closely related vehicle routing and scheduling problems is presented. In the following section, a precise mathematical model is formulated. This is followed by a description of two heuristic solution methods developed for the problem. The data sets used to test the solution methods are then described followed by a presentation and discussion of the results obtained. The final section contains some conclusions and proposals for further research.

2. Related literature

2.1. Prize-collecting arc routing

Unlike the Vehicle Routing Problem (VRP) or the Capacitated Arc Routing Problem (CARP), the Prize-Collecting Arc Routing Problem (PARP) has been defined in almost as many versions as papers have been published on the topic. Table 1 offers an overview of the existing literature. Here, the problems considered in the literature are classified with respect to the following parameters: (1) single or multiple vehicles. (2) Single or multiple collections of a prize from an arc. Note that if multiple collections are allowed, the number of collections may or may not be limited. (3) Additional constraint on the total route duration (being either total time or total cost). (4) Additional constraint on the capacity of the vehicle(s). (5) Additional clustering constraint. (6) Directed or undirected graph.

Two papers study the prize-collecting arc routing problem as it is defined in Problem 1, except that the directed arcs are replaced by undirected edges. In [4], Aráoz et al. study the problem using the name *Privatized Rural Postman Problem*. In that paper, the authors study the polyhedral structure of this problem as well as of a number of special cases. In [5], Aráoz et al. present a two-phase algorithm for solving the problem. The algorithm is based on the results obtained in [4] and iteratively adds violated inequalities to the initial relaxed formulation. Simultaneously, in each iteration, a feasible solution is obtained by transforming the problem into a rural postman problem, which is solved heuristically. In the second phase of the algorithm, integrality constraints are added and the algorithm continues in a branch-and-cut fashion.

A number of papers study directed variations of the problem. Malandraki and Daskin are, as far as we know, the first to consider a version of the prize-collecting arc routing problem, in their paper on the *maximum benefit Chinese postman problem* [1]. They consider a directed version of the problem where a prize can be collected from each arc multiple times, but where the size of the prize decreases with the number of collections from that particular arc. The authors present a branch-and-bound based algorithm for the problem, which repeatedly solves a minimum cost flow problem. Furthermore, the authors conclude that adding a limit on the total route duration would destroy the network structure of the problem that they exploit in their solution method and suggest that such problems be solved as multi-objective problems with minimizing the route length as one of the objectives in order to retain the network structure. Feillet et al. [3] consider a similar version where prizes can be collected from a given arc multiple times. They assume that the prize to be collected from a given arc is constant over time, but impose a limit on the number of collections from each arc. Furthermore, the authors allow for an unlimited number of vehicles, not tied to any depot but each with a limit on their tour length. The problem is solved using a branch-and-price algorithm. Euchi et al. [7] consider the same problem as Feillet et al. [3] except that they assume the cost and the length of an arc to be independent from each other. The authors solve the problem using two adaptive memory algorithms in which they embed a tabu search and a variable neighborhood search, respectively.

Archetti et al. [8] consider a new version of the problem with multiple vehicles. In addition to bounding the travel time for each vehicle, they associate a demand with each profitable edge, which is to be serviced when the prize is collected and a capacity constraint is imposed on the vehicles. They only allow for the prize of each edge to be collected once. Furthermore, in this paper, the goal is to maximize total prizes, not prizes minus cost, which is the usual goal in these problems. For this problem, the authors present a branch-and-bound algorithm for exact solution as well as both a variable neighborhood search and two tabu search algorithms. Zachariadis and Kiranoudis [9] consider the same problem, though with a modified objective function. They set a primary goal of maximizing profit and a secondary goal of minimizing total travel time. The authors solve this problem heuristically by local search using a move promise algorithm.

Pearn and Wang [2] consider the same version of the problem as Malandraki and Daskin [1] except that they work on an undirected graph. The authors present a heuristic solution based on expanding the network, finding a minimum spanning tree followed by the construction of a minimum cost matching.

Aráoz et al. [6] study the problem as it is defined in Problem 1 with the addition of a clustering requirement and using undirected edges instead of directed arcs. Clusters are generated by

Table 1
Overview of problems considered in the literature.

Paper	Year	# Vehicles	# Collections per arc	Route duration constraint	Capacity constraint	Clustered	Directed
Malandraki and Daskin [1]	1993	1	Many	No	No	No	Yes
Pearn and Wang [2]	2003	1	Many	No	No	No	No
Feillet et al. [3]	2005	Many	Many	Yes	No	No	Yes
Aráoz et al. [4]	2006	1	1	No	No	No	No
Aráoz et al. [5]	2009	1	1	No	No	No	No
Aráoz et al. [6]	2009	1	1	No	No	Yes	No
Archetti et al. [8]	2010	Many	1	Yes	Yes	No	No
Euchi et al. [7]	2011	Many	Many	Yes	No	No	Yes
Zachariadis and Kiranoudis [9]	2011	Many	1	Yes	Yes	No	No
Corberán et al. [10]	2011	1	1	No	No	Yes	No
This paper		1	1	Yes	No	No	Yes

considering the components of profitable edges and it is required that the prize is collected from either all or none of the edges in a cluster. In the paper, the authors derive new inequalities based on the clustering requirement and present a branch-and-cut algorithm using these cuts as well as those presented in [4]. Corberán et al. [10] consider the same problem as Aráoz et al. [6], though in a windy setup where the costs depend on the direction of travel. They study the polyhedral structure of the problem and present a cut-and-branch algorithm for its solution.

2.2. Time-dependent cost and time

In recent years, the availability of travel time information at different times of day and day of week has led to researchers modifying various vehicle routing and scheduling models to take advantage of this additional information, either for planning ahead or in a dynamic on-line environment.

Ichoua et al. [11] consider a vehicle routing and scheduling problem where the travel times are derived from different speeds for the traffic in different intervals of time. They demonstrate how their approach maintains the First In, First Out (FIFO) property. This means that if a vehicle starts travelling along an arc from a starting node, it will always reach the node at the end of the arc before any vehicle leaving the starting node at a later time.

Fleischmann et al. [12] consider a problem close to the one considered in this paper. They calculate time varying travel times through real time traffic information. This is used as input for a multivehicle full truckload pick-up and delivery problem with time windows. The focus in their paper is on the update mechanisms of the real time information.

Eglese et al. [13] describe how a road timetable can be constructed and used for solving a capacitated vehicle routing problem when the travel times are varying. Their approach also ensures that the FIFO property is maintained. In Maden et al. [14] a case study is described where this approach is used using real traffic data and order information from a company distributing goods in the south west of England. The model is used to estimate the potential savings in CO_2 emissions from taking the time-dependent travel time information into account compared with an approach where this information is not available.

Harwood et al. [15] investigate the trade-off between accuracy and run time in estimating the change of costs resulting from neighbourhood moves in a vehicle routing problem for a single vehicle in a time-dependent travel time environment.

Albiach et al. [16] consider a similar problem as Maden et al. [14] with the addition of time window constraints to the nodes and under certain integrality assumptions of the data. They present a way of transforming the problem into an asymmetric capacitated vehicle routing problem, which can subsequently be solved to optimality for small instances.

Gendreau et al. [17,18] consider a directed arc routing problem where the cost of servicing an arc depends on the time at which the service is initiated. They impose a restriction on the total time usage for each vehicle and separate between time for service and time for traversal. Both of these are independent of the time at which the activity is initiated.

2.3. Full truckload transportation

The final of the three problems related to Problem 2 is transportation of full truckloads, which is defined in the simplest form in Problem 3.

Problem 3 (*Vehicle routing with full loads*). Given a directed graph $G(V,A)$ with a set of depot nodes and with a cost, $c_{ij} > 0$, and travel time, t_{ij}, associated with each arc, let δ_{ij} be the number of full truck loads to be transported from i to j. Given a fleet of vehicles, initially located at the depot nodes, find a set of vehicle tours to move all loads from their origin to their destination such that the duration of each route does not exceed a given time constraint and the total cost is minimized.

Many researchers have considered this problem or close variants of it. As expressed, the demands for transportation are fixed and are not optional as in a prize-collecting problem; the objective is generally to minimize costs subject to fulfilling all the demands.

Desrosiers et al. [19] show how Problem 3 can be solved optimally through transformation to a constrained traveling salesman problem. Arunapuram et al. [20] present a branch-and-bound algorithm for solving the problem with the inclusion of time window constraints and waiting cost.

Doerner et al. [21] consider a variation of the problem where transportation is only performed among a number of distribution centers. They impose a time limit on the vehicle routes, but consider a longer planning horizon such that each vehicle can perform more than one route. They consider two objectives: firstly to minimize the number of vehicles, secondly to minimize routing cost. Many others consider variations of Problem 3 from a heuristic point of view, for instance Hirsch and Gronalt [22] and Gronalt et al. [23]. Several papers consider variations of Problem 3 in a real-time setup. Of these, we mention Jaillet et al. [24], where jobs are released online and must either be accepted or rejected, with a resulting influence on the objective function.

3. Mathematical model

We define the problem based on the directed multigraph $G(V,A)$. For any pair of nodes, i and j, and any time t, we define the function $f_t(i,j)$ to be the time it takes to travel from node i to node j when starting at node i at time t and traveling along a shortest time route. In practice we will use a road timetable to estimate this value [13]. We consider a single vehicle, which starts at the depot node, d, at time $T_{\min}$ and must return to the depot by $T_{\max}$.

We consider a set of profitable arcs, $\mathcal{P}$ indexed by r. If there is potential to collect a prize from one of the original arcs multiple times, then the original arc is replaced by a set of similar arcs between the same pair of nodes, where the number of arcs between the pair of nodes corresponds to the maximum number of times the prize can be collected. For each such profitable arc $r \in \mathcal{P}$, we let $\alpha_r \in V$ be the source node, $\omega_r \in V$ the target node, and p_r the prize. We denote by $f_t(r)$ the time it takes to traverse $r \in \mathcal{P}$ starting at time t.

For every node $j \in V$, we define $R_j^+ = \{r \in \mathcal{P} | \alpha_r = j\}$ to be the set of outgoing profitable arcs and $R_j^- = \{r \in \mathcal{P} | \omega_r = j\}$ to be the set of entering profitable arcs.

In order to make a mathematical model, we pose the problem in terms of a node duplicated network $\tilde{G}(\tilde{V},\tilde{A})$. For each node $j \in V$, we add $|R_j^+| + |R_j^-|$ nodes to $\tilde{V}$. We refer to these nodes as copies of j and refer to j as the origin $h(k)$ of a copy k of j.

We let $\tilde{G}$ be a complete graph.

For each profitable arc $r \in \mathcal{P}$, we select an arc (k,l) in $\tilde{A}$ where k is a copy of α_r and l is a copy of ω_r to be the associated profitable arc in $\tilde{G}$. We select these arcs in $\tilde{A}$ in such a way that every node in $\tilde{V}$ is adjacent to exactly one profitable arc. We use $\tilde{\mathcal{P}}$ to denote the set of these profitable arcs in $\tilde{G}$. We use $\tilde{\alpha}_{\tilde{r}}, \tilde{\omega}_{\tilde{r}}$, and $\tilde{p}_{\tilde{r}}$ to refer to the source node, target node, and profit of $\tilde{r} \in \tilde{\mathcal{P}}$.

For each profitable arc $\tilde{r} \in \tilde{\mathcal{P}}$ originating from $r \in \mathcal{P}$, we set the travel time function to $\tilde{f}_t(\tilde{\alpha}_{\tilde{r}}, \tilde{\omega}_{\tilde{r}}) = f_t(r)$. For all other arcs $(k,l) \in \tilde{A}$ we set the travel time function to $\tilde{f}_t(k,l) = f_t(h(k),h(l))$. It follows directly that $\tilde{f}_t(k,l) = 0$ if k and l belong to the same family, i.e. if $h(k) = h(l)$.

Finally, we add two nodes, denoted by d_s and d_e to $\tilde{V}$. These nodes are copies of the depot and can be thought of as a super-source and a super-sink node. For every node $k \in \tilde{V} \backslash \{d_s, d_e\}$ we add an arc (d_s,k) with travel time function $\tilde{f}_t(d_s,k) = f_t(d,h(k))$ and an arc (k,d_e) with travel time function $\tilde{f}_t(k,d_e) = f_t(h(k),d)$. Note that these travel time functions are zero if k is a copy of the depot. Finally, we add an arc (d_s,d_e) with travel time function $\tilde{f}_t(d_s,d_e) = 0$.

In Fig. 1, we give an example of an instance of the problem (left) and the resulting node duplicated network (right). Note that only profitable arcs are shown.

For the construction of a mathematical model, we use two types of decision variables as follows: for any arc (i,j) in $\tilde{G}$, let x_{ij} be a binary variable taking the value of 1 if (i,j) is traversed and zero otherwise. Let t_{ij} be the time at which the traversal of the arc (i,j) starts. t_{ij} is defined for all arcs, but for arcs where x_{ij} is zero, the value of t_{ij} is without interpretation.

$$\max \sum_{r \in \tilde{\mathcal{P}}} \tilde{p}_r x_{\tilde{\alpha}_r \tilde{\omega}_r} - \sum_{(i,j) \in \tilde{A}} \tilde{f}_{t_{ij}}(i,j) x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{j \in \tilde{V} \backslash \{d_s\}} x_{d_s j} = 1 \tag{2}$$

$$\sum_{j \in \tilde{V} \backslash \{d_e\}} x_{j d_e} = 1 \tag{3}$$

$$\sum_{j \in \tilde{V}} x_{ji} - \sum_{j \in \tilde{V}} x_{ij} = 0 \quad \forall i \in \tilde{V} \backslash \{d_s, d_e\} \tag{4}$$

$$\sum_{(i,d_e) \in \tilde{A}} (t_{id_e} + \tilde{f}_{t_{id_e}}(i,d_e) x_{id_e}) \le T_{\max} \tag{5}$$

$$t_{ki} + \tilde{f}_{t_{ki}}(k,i) x_{ki} \le t_{ij} + (2 - x_{ki} - x_{ij}) M \ \ \forall (k,i),(i,j) \in \tilde{A} \tag{6}$$

$$t_{ki} + \tilde{f}_{t_{ki}}(k,i) x_{ki} \ge t_{ij} - (2 - x_{ki} - x_{ij}) M \ \ \forall (k,i),(i,j) \in \tilde{A} \tag{7}$$

$$x_{ij} \in \{0,1\} \ \ \forall (i,j) \in \tilde{A} \tag{8}$$

$$t_{ij} \ge T_{\min} \ \ \forall (i,j) \in \tilde{A} \tag{9}$$

$$t_{d_s j} \le T_{\min} \ \ \forall (d_s,j) \in \tilde{A} \tag{10}$$

Here, (1) is the objective function maximizing profit as the prizes minus travel time. Constraints (2) and (3) ensure that the vehicle leaves the super-source depot and enters the super-sink depot using exactly one arc, respectively. (4) is the flow conservation constraint and (5) ensures that we arrive at the super-sink depot on time. Constraint (6) ensures that we do not start traversing an arc before we have reached the source of the arc. M represents a sufficiently large number. This constraint is non-binding for any arc (i,j) that is not being traversed in the solution because the corresponding t_{ij} can take the value of zero. Note here that for arc (k,i), we use the start of traversal, t_{ki}, as input time for the function $f_t(k,i)$. Constraint (7) ensures that the model does not allow for waiting in the nodes. Finally, (8) ensures binarity while (9) and (10) ensure that all start times are legal.

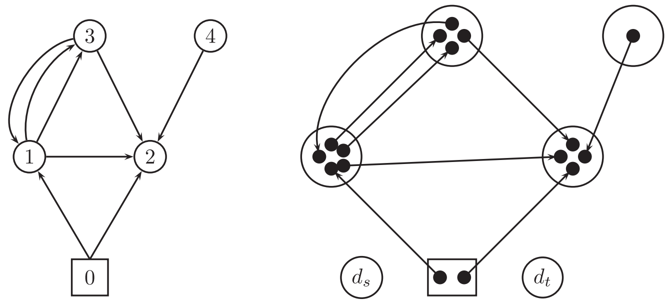

Fig. 1. Example of instance and the corresponding node duplicated network.

We stress the fact that the motivation for studying this problem is the fact that the time to travel from i to j is not constant, but rather it changes according to the time, t, the traversal takes place. This is modeled via the function $f_t(i,j)$ which changes with the parameter t. $f_t(i,j)$ directly influences the function $\tilde{f}_t(i,j)$, which appears both in the objective function and in constraints (5) through (7), resulting in a model which is not easily linearized.

4. Solution procedure

Although the solution of the mathematical model presented in Section 3 could be approached by an exact method, experience of similar routing problems suggests that for large-scale problems heuristic methods are needed to provide good solutions in an acceptable computing time.

Two solution methods have been designed. The first is a method designed specifically for the TD-PARP, based on a variable neighborhood search method and it is described in Section 4.1. The second is a method where an algorithm originally designed to solve vehicle routing problems in a time-dependent environment has been modified to provide a solution to the TD-PARP. The name for the second method is LANTIME and it is described in Section 4.2.

4.1. Variable neighborhood search

Variable Neighborhood Search (VNS) is a metaheuristic proposed by Mladenović and Hansen [25]. It has been used for a wide variety of optimization problems. For example, this style of metaheuristic is used in Carrabs et al. [26] for a pickup and delivery problem with a Last In, First Out (LIFO) constraint. Forms of VNS are also used in Euchi et al. [7], Archetti et al. [8] and Gendreau et al. [18].

We solve the problem using VNS which uses a Variable Neighborhood Descent (VND) algorithm as subroutine. We first explain our notation and some general issues, next we describe the VND and the neighborhood structures used therein, and finally, we give the details of our VNS algorithm.

We store all n profitable arcs in an ordered list L. For any $i,j \le n, i < j$, we use $L[i:j]$ as short notation for the partial list $L[i], L[i+1], \ldots, L[j]$.

The current solution of the problem is determined from L by defining a tour where the arcs $L[1:s]$ are serviced in the order dictated by the list and where the necessary traversal of other arcs are performed in between. s is called the *split-point* and is determined such that $L[1:s]$ is the solution with the highest objective value, among all feasible solutions of the type $L[1:1], L[1:2], \ldots$. That way, $T_{\max}$ imposes an upper bound on s. Note that profitable arcs in $L[s+1:n]$ are not serviced in the solution. In the description of the algorithms, we denote by x, the best feasible solution corresponding to L.

The idea in our implementation of the algorithm is that in the VNS part we select the arcs to be in the solution by moving arcs to $L[1:s]$, while arcs not to be in the solution are moved beyond the split-point. In the VND part of the algorithm, the arcs in $L[1:s]$ are reorganized. We introduce a value s', which we refer to as the *moved split-point*. In our implementation, we set $s' = \min(s+\varepsilon, n)$, where after experimentation $\varepsilon = 10$ was found to be a suitable value. When reorganizing in the VND part of the algorithm, we often use s' in place of s, and hence reorganize $L[1:s']$. Using s'

results in a more powerful VND, allowing more potential solutions to be considered.

Algorithm 1. Variable neighborhood descent.

```
1:  function VND($x'$,$l_{\max}$)
2:    $l \leftarrow 1$
3:    while $l < l_{\max}$ do
4:      $x'' \leftarrow$ best solution in $N_l(x')$
5:      if OBJ($x''$) > OBJ($x'$) then      ▷ Better solution obtained
6:        $x' \leftarrow x''$        ▷ Store as current solution
7:        $l \leftarrow 1$        ▷ Back to first neighborhood
8:      else
9:        $l \leftarrow l+1$        ▷ Next neighborhood
10:     end if
11:   end while
12:   return $x''$
13: end function
```

Algorithm 1 gives an outline of the variable neighborhood descent algorithm. We use 10 neighborhoods in this algorithm, i.e. $l_{\max} = 10$, but some of these are similar, though not identical. In the following we explain the different neighborhood structures in the order in which they are used in the algorithm. Fig. 2 offers graphical illustrations of the moves to support the verbal description.

1. Drop bad: We consider the arcs in $L[1:s']$ and drop the one that results in the highest objective value. The dropped arc is appended to the end of L.
2. Add best best: For each arc $L[j] \in L[s:n]$, we temporarily remove $L[j]$ from L and try to insert $L[j]$ at every location in $L[1:s']$. We choose the arc $L[j]$ and the insert location that results in the highest objective value.
3. Best move: For each arc $L[j] \in L[1:s']$, we temporarily move $L[j]$ to every other location in $L[1:s']$. We select the move that results in the highest objective value.

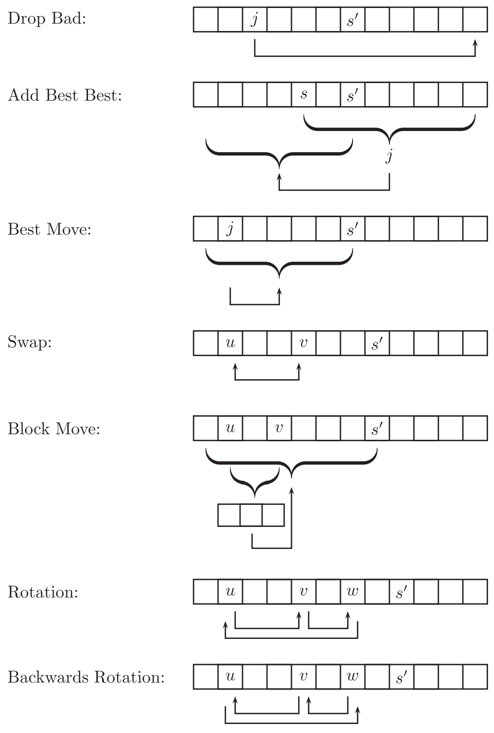

Fig. 2. Neighborhood moves used in the variable neighborhood descent.

4. Swap: We consider every pair of arcs $L[u], L[v] \in L[1:s']$. We swap the pair that results in the highest objective value.

5–8. Block move: We consider each block $L[u:v] \subseteq L[1:s']$. We temporarily remove $L[u:v]$ from L and try to reinsert the block at every location in $L[1:u-1, v+1:s']$. We choose the block and the insert location which results in the highest objective value. We use four neighborhoods with block moves, using blocks of size 2, 3, 4, and 5, respectively.

9. Rotation: We consider every triple of arcs $L[u], L[v], L[w] \in L[1:s']$ and rotate these such that $L[u]$ is inserted in $L[v]$'s location, and so forth. We choose the rotation resulting in the highest objective value.
10. Backwards rotation: We consider every triple of arcs $L[u]$, $L[v]$, $L[w] \in L[1:s']$ and rotate these backwards such that $L[u]$ is inserted in $L[w]$'s location, and so forth. We choose the rotation resulting in the highest objective value.

Algorithm 2 gives an outline of the variable neighborhood search which we used in our implementation. In the first major iteration, only strictly better solutions are accepted. In the remaining iterations, we accept solutions slightly worse than the best if they are significantly different from the current solution.

We use β to indicate the willingness to accept worse solutions and initialize $\beta = 0$. In each major iteration, we increase β by $OBJ(x_B)/800$. Given the current solution x and a new solution x'', we define the distance $\rho(x,x'')$ between them as the number of profitable arcs that are before s in either x or x'', but not in both. Now, $\beta\rho(x,x'')$ is the threshold difference between the new and best solutions for accepting solutions with lower objective value. We refer to this as accepting a skewed solution.

We use 21 neighborhoods in the VNS algorithm, i.e. $k_{\max} = 21$, but neighborhoods 9 through 21 are similar. In the following, we explain the neighborhood structures in the order in which they are considered by the algorithm.

Algorithm 2. VNS.

```
1:  function VNS ($x$,$l_{\max}$,$k_{\max}$,$t_{\max}$)
2:    $\beta = 0$
3:    $x_B \leftarrow x$
4:    while $t < t_{\max}$ do
5:      $k \leftarrow 1$
6:      while $k \le k_{\max}$ do
7:        $x' \leftarrow$ solution in $N_k(x)$        ▷ Shake
8:        $x'' \leftarrow$ VND($x'$,$l_{\max}$)
9:        if OBJ($x''$) > OBJ($x$) or
10:         OBJ($x''$) + $\beta\rho(x,x'')$ > OBJ($x_B$) then
11:         $x \leftarrow x''$        ▷ Store
12:         if OBJ($x''$) > OBJ($x_B$) then
13:           update $x_B$
14:         end if
15:         $k \leftarrow 1$        ▷ First neighborhood
16:       else
17:         $k \leftarrow k+1$        ▷ Next neighborhood
18:       end if
19:     end while
20:     $\beta \leftarrow \beta + 5$
21:     $t \leftarrow$ system time
22:   end while
23:   return $x_B$
24: end function
```

For the use in these neighborhood structures, we define γ_i for each profitable arc $L[i]$ as $\gamma_i = p_i / \min_t \{f_t(\alpha_i, \omega_i)\}$. γ_i represents an estimate of the profit obtained by accepting the order represented by arc $L[i]$ relative to the shortest time required to travel along the arc.

1. Gamma add randomly: For each arc $L[i] \in L[s+1:n]$, assign the probability $\gamma_i / \sum_{j=s+1}^{n} \gamma_j$. Randomly select an arc $L[i] \in L[s+1:n]$ using these probabilities. Insert $L[i]$ in $L[1:s]$ choosing each insert location with equal probability.
2. Gamma add best: For each arc $L[i] \in L[s+1:n]$, assign the probability $\gamma_i / \sum_{j=s+1}^{n} \gamma_j$. Randomly select an arc $L[i] \in L[s+1:n]$ using these probabilities. Insert $L[i]$ in the location of $L[1:s]$ which results in the highest objective value.
3. Gamma drop: For each arc $L[i] \in L[1:s]$, assign the probability $(1/\gamma_i)/(\sum_{j=1}^{s} 1/\gamma_j)$. Randomly remove an arc $L[i] \in L[1:s]$ using these probabilities, and append the arc to the end of L.
4. New start: Select randomly with equal probability an arc $L[i] \in L[s+1:n]$. Move $L[i]$ to the front of L.
5. Add random randomly: Choose randomly with equal probability an arc $L[i] \in L[s+1:n]$. Select randomly an insert location in $L[1:s]$ and insert $L[i]$ in that location.
6. Add random best: Choose randomly with equal probability an arc $L[i] \in L[s+1:n]$. Insert $L[i]$ in the location of $L[1:s]$ which results in the highest objective value.
7. Drop random: Select randomly with equal probability an arc $L[i] \in L[1:s]$. Move $L[i]$ to the end of L.
8. Drop best: Consider the arcs $L[i] \in L[1:s]$. Move to the end of L the arc whose move results in the highest objective value.

9–21. *k*-Block add 2–14: Choose randomly with equal probability an arc $L[i] \in L[s+1:n]$. Remove the block $L[i:\min\{i+k,n\}]$ from L. The block is inserted in the location of $L[1:s]$ which results in the highest objective value. We use 13 neighborhoods with *k*-block add moves, using blocks of size $k=2,3,\ldots,14$, respectively.

4.2. LANTIME for the TD-PARP

LANTIME was originally designed for time-dependent vehicle routing problems where the objective is to minimize the total time required to deliver goods to a set of customers, taking account of constraints on vehicle capacity and time windows for the customers. It is a metaheuristic algorithm based on tabu search and is described in Maden et al. [14].

Modifications were required to LANTIME in order to use it for the TD-PARP. Firstly the objective was changed to calculate the profit from the orders that were serviced. Secondly, the delivery to a single customer was replaced by accepting an order involving a pick-up from one customer location and delivery at another. Thirdly, as all the problems in the benchmark set are single vehicle problems, the code was modified to allow two vehicle routes: the first is the route used by the vehicle and the second is a dummy route to which orders that are not accepted are assigned. Some experimentation was carried out to determine the best parameter values to use and which neighborhoods were appropriate. This version of LANTIME uses four possible neighborhood operations: cross exchange, insertion/removal, one exchange and swap, which are based on the moves described in Maden et al. [14].

5. Computational results

5.1. Data generation and benchmark instances

A set of 41 problems has been created to use as benchmark instances to test the solution methods. They are based on two separate road networks with time-dependent travel time information that indicates how the travel time for any arc in the original road network depends on the time of day. The first set comes from a road network in the north-west of England (NW) and the second from a road network in the south-east of England in and around London (London).

For each road network, a depot node has been identified and a set of instances has been generated using different nodes to represent the source or destination of a delivery that would result in a benefit to the transport company and so correspond to a prize arc in the corresponding arc routing model. The number of prize arcs, the start time and the maximum duration of each vehicle route were set to provide a range of different problems. The sizes of the prizes for each prize arc were generated randomly.

Road timetables were created appropriate to each instance. The approach described in Eglese et al. [13] was used to produce arrays giving the shortest time to travel from one of the selected nodes to another selected node starting at different times of the day in 15-min increments over 24 h. Once a road timetable had been created for a set of selected nodes, it could be used for any instance involving a subset of those nodes as sources or destinations.

The instances based on the north-west data were divided into six clusters labelled A, B, C, D, E and F and are described below.

For the A and B instances, the source node and destination node corresponding to each prize arc were selected at random from 25 possible locations. The values of the prizes were generated at random, but for the B instances the values were smaller than that for the A instances. The five instances in each cluster varied according to the number of prizes, start time and maximum duration permitted.

The C and D instances are similar to the A instances, except that for C the prize arcs are relatively short and for D the prize arcs are relatively long.

The E instances are similar to the A instances, except that the majority of the prize arcs are inbound to one of five particular locations.

The F instances are based on prize arcs selected at random from 100 possible locations and a larger number of possible prizes and maximum time duration is allowed.

The London instances can be divided into two clusters. In the basic set (London B), the source node and destination node were selected at random from 50 possible locations. The values of the prizes were generated at random, though related to the time to travel from source to destination. In the large London set (London L), the source node and destination node were again selected at random from the 50 possible locations, but more prize arcs were generated. There are some cases where more than one prize arc is generated between the same pair of nodes.

Summary details of the problem instances used can be found in Table 2, where the fifth column gives the number of prize arcs collected in the best solution found for the computational results reported here (i.e. not necessarily the best known solution). To give an indication of the problem complexity in terms of the algorithms used, the last three columns report for each instance the average number of VNS iterations, calls to the VND, and tabu search iterations performed per minute.

When developing the solution methods, a set of eight small problems was generated based on the north-west data. The details of these instances are given in Table 3, where the second column states the number of prize arcs in the instance and the remaining columns give information regarding the optimal solution: objective value, number of prizes collected, and an ordered list of the collected prizes. These instances could be solved to optimality by complete enumeration. Both VNS and LANTIME were able to find the optimal solutions to these small problems in a short computing time, but the benchmark data sets introduced in the previous section were too large for complete enumeration to be possible.

Table 2
Problem instance details.

Instance	No. nodes	No. arcs	No. prize arcs	Best solution collected prize arcs	VNS	VND	TS
NW25-A1	25	625	50	7	745	30 735	13 267
NW25-A2	25	625	50	13	399	10 945	7503
NW25-A3	25	625	100	14	278	6509	1586
NW25-A4	25	625	100	17	162	3659	1442
NW25-A5	25	625	150	19	109	2403	489
NW25-B1	25	625	50	7	1175	37 202	9486
NW25-B2	25	625	50	13	347	11 352	5662
NW25-B3	25	625	100	13	293	6831	1335
NW25-B4	25	625	100	17	175	3842	1214
NW25-B5	25	625	150	20	88	1980	410
NW25-C1	25	625	50	11	506	15 573	7591
NW25-C2	25	625	50	16	238	5008	5794
NW25-C3	25	625	100	20	164	3507	918
NW25-C4	25	625	100	26	72	1505	759
NW25-C5	25	625	150	32	27	595	247
NW25-D1	25	625	50	4	3084	68 490	20 732
NW25-D2	25	625	50	5	1171	37 722	14 611
NW25-D3	25	625	100	6	852	24 356	3616
NW25-E1	25	625	50	6	976	35 284	13 669
NW25-E2	25	625	50	9	534	16 270	9240
NW25-E3	25	625	100	13	275	7295	1742
NW25-F1	25	625	300	22	56	1228	115
NW25-F2	25	625	400	28	45	1067	96
NW25-F3	25	625	500	29	32	779	66
NW25-F4	25	625	500	30	33	746	65
NW25-F5	25	625	600	28	29	664	71
London-B1	50	2500	75	32	43	983	2510
London-B2	50	2500	75	28	47	1044	2459
London-B3	50	2500	75	29	41	909	2146
London-B4	50	2500	75	24	77	1770	2498
London-B5	50	2500	75	22	128	2700	2720
London-B6	50	2500	75	34	35	694	2376
London-B7	50	2500	75	32	69	1517	2356
London-B8	50	2500	75	34	32	706	2308
London-B9	50	2500	75	33	39	835	2374
London-B10	50	2500	75	30	67	1567	2420
London-L1	50	2500	350	51	15	351	132
London-L2	50	2500	350	52	63	1332	143
London-L3	50	2500	350	51	32	738	180
London-L4	50	2500	350	44	41	896	128
London-L5	50	2500	350	50	20	458	170

Table 3
Specification of small instances.

Instance	No. prizes	Optimal solution		
		Value	Number	Prizes
NW25-T1	27	2154.04	9	9,15,17,19,20,24,22,11,10
NW25-T2	20	1855.04	9	2,1,7,11,13,14,15,18,16
NW25-T3	20	1085.31	6	17,15,18,19,14,8
NW25-T4	15	714.21	6	3,14,11,15,4,8
NW25-T5	15	507.01	4	11,15,4,8
NW25-T6	10	462.71	3	8,6,9
NW25-T7	10	705.16	6	5,1,4,8,6,9
NW25-T8	15	705.18	5	7,14,8,5,9

Details of the benchmark data sets and the test instances are available at http://www.optimization.dk/TD-PARP along with the actual data.

5.2. Results

The VNS algorithm was implemented on a PC with an Intel Core 2 Duo CPU, running at 2.40 GHz and with 2.97 GB of RAM and the LANTIME algorithm was also implemented on a PC with an Intel Core 2 Duo CPU, running at 2.2 GHz with 2 GB of memory. Although not identical, the two PCs have similar run time performance. The algorithms were both written in C++.

The VNS and LANTIME algorithms were originally run with a time limit of 1 min computing time. Each instance was solved 10 times using different sets of random numbers within the algorithms. The mean, maximum and minimum results for each of the benchmark instances are shown in Table 4. In the second to last column, the table shows the best result achieved with either algorithm during the 10 runs. The final column reports the best known solution in case it is better than the other results reported in the table. This result may be from one of the runs reported in subsequent tables or may be from an unreported run that was carried out when setting parameters for the algorithms.

One issue that was examined was to see whether the algorithms benefited significantly from a longer computing time. Table 5 gives the corresponding results for 10 min computing time for each instance.

The largest instances from the cluster NW100-F and London-L showed distributions of results with significant gaps to the best

Table 4
Results from 1 min computing time.

Instance	VNS				LANTIME				Best	Best known
	Mean	Median	Max	Min	Mean	Median	Max	Min		
NW25-A1	1244.34	1244.34	1244.34	1244.34	1244.34	1244.34	1244.34	1244.34	1244.34	
NW25-A2	2130.60	2138.03	2168.18	2046.87	2046.81	2023.9	2168.18	1984.96	2168.18	
NW25-A3	2311.50	2302.60	2430.92	2200.11	2180.88	2196.9	2278.84	2047.56	2430.92	2443.04
NW25-A4	2970.18	2978.54	3135.96	2785.18	2795.59	2754.15	3120.74	2509.35	3135.96	3155.24
NW25-A5	3575.84	3575.59	3799.11	3350.18	3349.91	3310.83	3682.90	3085.40	3799.11	4009.99
NW25-B1	130.34	130.34	130.34	130.34	130.34	130.34	130.34	130.34	130.34	
NW25-B2	309.23	309.23	309.23	309.23	307.95	309.23	310.62	302.64	310.62	314.18
NW25-B3	484.59	491.53	501.58	443.62	495.25	492.27	521.27	467.36	521.27	
NW25-B4	709.32	709.66	757.26	678.17	675.88	680.55	715.55	627.70	757.26	781.66
NW25-B5	802.20	796.79	904.26	748.22	692.82	706.08	790.46	606.10	904.26	915.50
NW25-C1	1830.21	1830.23	1830.23	1830.09	1821.63	1830.23	1830.23	1744.59	1830.23	
NW25-C2	2173.61	2168.70	2180.97	2168.70	2216.18	2222.6	2296.42	2149.60	2296.42	2396.68
NW25-C3	3015.77	2933.25	3357.04	2841.99	3003.26	3046.07	3129.37	2785.53	3357.04	3504.26
NW25-C4	4030.89	4155.95	4219.69	3548.58	4006.06	4020.51	4256.57	3635.96	4256.57	4440.18
NW25-C5	5593.50	5657.69	5722.76	5191.31	5151.55	5178.03	5563.63	4844.51	5722.76	5812.75
NW25-D1	762.53	755.67	772.82	755.67	772.82	772.82	772.82	772.82	772.82	
NW25-D2	1426.94	1426.94	1426.94	1426.94	1417.37	1426.94	1426.94	1370.05	1426.94	
NW25-D3	1614.08	1614.08	1614.08	1614.08	1444.01	1451.72	1554.68	1333.92	1614.08	
NW25-E1	1026.03	1026.03	1026.03	1026.03	1023.43	1026.03	1026.03	1000.07	1026.03	
NW25-E2	1953.88	1953.88	1953.88	1953.88	1865.86	1870	1904.37	1822.01	1953.88	
NW25-E3	2261.69	2264.85	2322.59	2209.69	2119.47	2117.15	2229.93	2000.51	2322.59	
NW100-F1	4667.99	4674.55	4856.56	4489.29	3852.53	3889.16	4059.51	3562.90	4856.56	5007.61
NW100-F2	4421.67	4480.04	4929.52	3873.30	3738.87	3799.15	4263.16	3106.10	4929.52	5425.43
NW100-F3	5158.37	5240.21	5587.36	4370.63	3573.76	3554.41	3891.99	3239.36	5587.36	6147.93
NW100-F4	5298.89	5312.87	5706.04	4987.70	3543.37	3528.87	4082.14	2938.08	5706.04	6253.10
NW100-F5	4544.91	4543.44	4878.57	4088.81	3561.96	3597.6	4164.24	3075.18	4878.57	5416.00
London-B1	1354.41	1358.15	1370.15	1307.22	1242.72	1245.83	1274.13	1196.64	1370.15	1373.59
London-B2	1220.64	1220.03	1294.27	1135.38	1171.49	1173.11	1194.37	1149.89	1294.27	
London-B3	1398.22	1420.32	1432.34	1294.19	1280.48	1275.24	1336.45	1236.25	1432.34	1463.77
London-B4	1041.29	1045.68	1089.37	999.18	961.84	960.73	998.42	903.17	1089.37	1096.94
London-B5	1160.45	1203.10	1272.48	990.27	1162.23	1171.4	1192.67	1108.14	1272.48	
London-B6	1357.81	1357.81	1357.81	1357.81	1292.97	1291.78	1348.46	1240.56	1357.81	1400.60
London-B7	1332.77	1392.38	1451.50	1016.11	1340.51	1336.49	1402.01	1299.14	1451.50	1491.08
London-B8	1564.51	1557.76	1632.45	1518.75	1443.10	1429.78	1546.40	1359.45	1632.45	1639.06
London-B9	1359.42	1362.29	1398.89	1275.94	1278.40	1276	1330.00	1250.00	1398.89	1400.53
London-B10	1416.11	1486.25	1556.31	1095.95	1370.85	1382.92	1435.87	1290.36	1556.31	1571.17
London-L1	2021.16	2070.52	2612.68	1178.64	1630.07	1819.4	2314.07	765.41	2612.68	2673.40
London-L2	1334.42	1069.38	2070.98	1069.38	1547.01	1634.08	1992.53	1010.09	2070.98	2533.05
London-L3	1619.26	1791.13	2076.73	1192.94	1945.01	1969.75	2158.27	1713.52	2158.27	2685.57
London-L4	1642.37	1693.46	2231.58	1011.29	1462.68	1547.75	2205.88	816.93	2231.58	2344.78
London-L5	2379.79	2347.74	2721.68	2038.22	1584.54	1325.5	2417.81	1049.46	2721.68	2806.38

ever solutions, so it was decided to allow 30 min runs for each of these instances. The results are shown in Table 6.

A summary of the results showing the mean percentage gaps from the best ever solutions for each cluster of instances is presented in Table 7.

Looking at the results for a 1 min run time, it is clear that there are differences in the results obtained for each run from the two heuristic algorithms, however the better results are generally given by the VNS method. From Table 7 it is can be seen that when the results for each cluster are amalgamated, the VNS outperforms LANTIME for each cluster except for the maximum result for the largest set (London-L). When the computing time is increased to 10 min, VNS still generally outperforms LANTIME. However the advantage is not so clear and, for example, LANTIME now does better in the amalgamated results for sets NW25-B and NW25-C.

Both methods found improvements for the NW100-F and London-L sets when given additional computing time, though one method did not consistently outperform the other.

The summary of the results also suggest that for both VNS and LANTIME, better results are obtained by taking the best (maximum) result from 10 one-minute runs, rather than a single ten-minute run, though this is not always the case.

Both methods are capable of providing good solutions to the problem. It is perhaps not surprising that VNS showed the better performance as it had been designed with the prize-collecting arc routing problem in mind, while LANTIME had been adapted from a vehicle routing algorithm.

6. Concluding remarks

The paper has introduced a new problem to the literature: the Time-Dependent Prize Collecting Arc Routing Problem (TD-PARP). Related research has been surveyed and a precise mathematical formulation has been provided.

Two heuristic methods have been proposed and tested on a set of benchmark problems: VNS and LANTIME. Both algorithms

Table 5
Results from 10 min computing time.

Instance	VNS				LANTIME				Best	Best known
	Mean	Median	Max	Min	Mean	Median	Max	Min		
NW25-A1	1244.34	1244.34	1244.34	1244.34	1244.34	1244.34	1244.34	1244.34	1244.34	
NW25-A2	2167.76	2168.18	2168.18	2163.92	2165.78	2023.9	2168.18	2162.68	2168.18	
NW25-A3	2407.12	2420.60	2443.04	2308.11	2291.39	2284.96	2397.46	2211.78	2443.04	
NW25-A4	2970.76	2969.16	3117.45	2721.16	2988.39	3000.44	3120.74	2845.90	3120.74	3155.24
NW25-A5	3669.57	3681.72	3956.74	3310.20	3522.67	3512.42	3710.07	3365.67	3956.74	4009.99
NW25-B1	130.34	130.34	130.34	130.34	130.34	130.34	130.34	130.34	130.34	
NW25-B2	309.30	309.23	309.92	309.23	312.41	314.18	314.18	309.23	314.18	
NW25-B3	506.84	506.25	521.27	487.51	517.87	518.85	521.27	514.04	521.27	
NW25-B4	732.95	738.27	781.66	678.17	761.22	767.79	781.66	727.01	781.66	
NW25-B5	823.39	822.55	902.01	748.22	798.71	793.43	856.45	759.14	902.01	915.50
NW25-C1	1830.23	1830.23	1830.23	1830.23	1830.23	1830.23	1830.23	1830.23	1830.23	
NW25-C2	2178.15	2168.70	2238.63	2168.70	2333.04	2341.52	2396.68	2270.87	2396.68	
NW25-C3	2975.45	2863.38	3422.22	2777.18	3294.94	3306.11	3504.26	3142.09	3504.26	
NW25-C4	4131.05	4155.95	4413.13	3970.05	4221.06	4210.54	4440.18	4049.30	4440.18	
NW25-C5	5560.84	5654.51	5752.05	5265.49	5442.55	5445.04	5587.37	5235.25	5752.05	5812.75
NW25-D1	764.25	764.25	772.82	755.67	772.82	772.82	772.82	772.82	772.82	
NW25-D2	1426.94	1426.94	1426.94	1426.94	1426.94	1426.94	1426.94	1426.94	1426.94	
NW25-D3	1614.08	1614.08	1614.08	1614.08	1585.76	1580.81	1614.08	1554.68	1614.08	
NW25-E1	1026.03	1026.03	1026.03	1026.03	1026.03	1026.03	1026.03	1026.03	1026.03	
NW25-E2	1953.88	1953.88	1953.88	1953.88	1909.29	1903.67	1953.88	1883.87	1953.88	
NW25-E3	2298.10	2299.64	2322.59	2267.43	2210.39	2212.15	2237.96	2181.49	2322.59	
NW100-F1	4685.80	4705.35	4984.03	4342.35	3970.23	3959.14	4439.30	3678.12	4984.03	5007.61
NW100-F2	4826.09	4866.90	5312.66	4232.73	3845.65	3875.66	4263.16	3282.23	5312.66	5425.43
NW100-F3	5660.62	5642.38	6147.93	5194.85	4053.63	4169.22	4862.05	3323.48	6147.93	
NW100-F4	5427.56	5503.19	6130.10	4471.59	4096.83	4106.57	4834.15	3736.11	6130.10	6253.10
NW100-F5	4993.64	5045.53	5309.17	4536.36	3777.62	3846.05	4404.67	3192.91	5309.17	5416.00
London-B1	1299.82	1287.19	1372.03	1253.20	1292.09	1288.18	1317.15	1275.39	1372.03	1373.59
London-B2	1219.52	1226.01	1274.38	1140.03	1205.22	1195.85	1251.79	1175.46	1274.38	1294.27
London-B3	1405.29	1429.90	1463.77	1294.19	1316.39	1312.9	1359.07	1286.31	1463.77	1463.77
London-B4	1065.37	1070.31	1092.75	1014.15	996.02	992.29	1043.20	975.18	1092.75	
London-B5	1160.44	1201.84	1250.48	1013.46	1187.36	1183.41	1214.30	1177.63	1250.48	1272.48
London-B6	1357.81	1357.81	1357.81	1357.81	1336.51	1336.63	1378.35	1298.05	1378.35	1400.60
London-B7	1406.95	1418.15	1487.41	1297.40	1387.29	1386.09	1423.01	1350.64	1487.41	1491.08
London-B8	1572.89	1593.48	1629.99	1485.04	1503.55	1502.74	1546.40	1461.31	1629.99	1639.06
London-B9	1360.28	1378.79	1400.39	1288.53	1325.40	1326.5	1358.00	1306.00	1400.39	1400.53
London-B10	1379.03	1412.49	1526.76	1070.66	1423.58	1431.6	1457.88	1370.88	1526.76	1571.17
London-L1	2144.04	2359.66	2564.45	1528.81	2188.57	2222.74	2452.84	1711.06	2564.45	2673.40
London-L2	1184.78	1069.38	1573.59	1027.13	1838.56	1883.85	2011.91	1622.53	2011.91	2533.05
London-L3	2038.09	2032.63	2517.97	1608.92	2201.76	2109.87	2630.41	1881.09	2630.41	2685.57
London-L4	1690.75	1661.95	2309.65	1072.99	1847.74	1830.73	2205.88	1595.37	2309.65	2344.78
London-L5	2411.85	2452.93	2666.30	1924.04	2151.08	2119.6	2451.34	1884.03	2666.30	2806.38

Table 6
Results from 30 min computing time.

Instance	VNS				LANTIME				Best	Best known
	Mean	Median	Max	Min	Mean	Median	Max	Min		
NW100-F1	4723.56	4794.62	5007.61	4302.09	3987.21	3963.87	4439.30	3794.98	5007.61	
NW100-F2	4880.97	4902.58	5285.08	4343.03	4075.42	4107.08	4306.10	3451.66	5285.08	5425.43
NW100-F3	5396.28	5363.37	6126.85	4628.48	4509.27	4553.77	4862.05	4064.91	6126.85	6147.93
NW100-F4	5636.64	5847.85	6150.61	4417.01	4379.10	4335.27	5025.90	3884.29	6150.61	6253.10
NW100-F5	5034.77	5038.30	5416.00	4497.06	4023.48	4035.23	4404.67	3850.58	5416.00	5416.00
London-L1	1972.19	1848.83	2673.40	1209.32	2282.02	2270.96	2452.84	2140.09	2673.40	2673.40
London-L2	1857.01	1843.03	2533.05	1069.38	1963.71	1954.98	2068.95	1845.04	2533.05	2533.05
London-L3	2022.64	1992.82	2528.92	1192.94	2310.19	2297.58	2630.41	2085.24	2630.41	2685.57
London-L4	1665.78	1620.61	2229.24	1333.31	1894.12	1851.03	2205.88	1777.01	2229.24	2344.78
London-L5	2420.04	2562.17	2742.25	1527.07	2187.16	2190.99	2451.34	1904.57	2742.25	2806.38

are capable of producing good solutions on large problems. However, the algorithm specifically designed for the TD-PARP, VNS, has a generally better performance than the algorithm adapted from another more general purpose heuristic, LANTIME.

Future research will consider extensions of the TD-PARP to cover situations where more than one vehicle is available to deal with orders. This may require more significant changes to the VNS approach than LANTIME and may allow LANTIME to become more competitive.

Table 7
Summary of results showing the average percentage gaps from the best known solutions.

Time	Instance	VNS			LANTIME		
		Mean	Max	Min	Mean	Max	Min
1	A instances	4.76	1.27	8.74	8.84	3.19	13.63
	B instances	6.05	1.94	9.60	8.97	4.65	13.50
	C instances	7.25	3.94	11.84	8.69	4.66	14.05
	D instances	0.44	0.00	0.74	3.74	1.23	7.11
	E instances	0.87	0.00	1.62	4.50	2.17	7.72
	F instances	14.54	7.99	22.52	34.46	26.68	42.80
	London-B	5.65	1.03	14.12	10.37	6.79	14.01
	London-L	31.32	10.21	50.70	35.83	12.87	58.14
10	A instances	3.17	0.51	7.39	4.75	2.09	7.12
	B instances	4.13	0.57	7.92	3.32	1.29	5.41
	C instances	7.10	2.12	10.05	3.99	0.78	6.86
	D instances	0.37	0.00	0.74	0.59	0.00	1.23
	E instances	0.35	0.00	0.79	2.37	1.21	3.22
	F instances	9.28	1.30	19.10	29.46	18.70	38.42
	London-B	5.47	1.05	12.55	7.33	4.61	9.41
	London-L	27.82	10.94	45.61	19.80	7.89	31.76
30	F instances	8.97	0.91	21.02	25.24	17.93	31.98
	London-L	24.07	2.61	51.37	16.53	7.39	23.34

References

[1] Malandraki C, Daskin MS. The maximum benefit Chinese postman problem and the maximum benefit traveling salesman problem. European Journal of Operational Research 1993;65:218–34.

[2] Pearn W, Wang K. On the maximum benefit Chinese postman problem. Omega 2003;31:269–73.

[3] Feillet D, Dejax P, Gendreau M. The profitable arc tour problem: solution with a branch-and-price algorithm. Transportation Science 2005;39(4):539–52.

[4] Aráoz J, Fernández E, Zoltan C. Privatized rural postman problems. Computers and Operations Research 2006;33:3432–49.

[5] Aráoz J, Fernández E, Meza O. Solving the prize-collecting rural postman problem. European Journal of Operational Research 2009;196:886–96.

[6] Aráoz J, Fernández E, Franquesa C. The clustered prize-collecting arc routing problem. Transportation Science 2009;43(3):287–300.

[7] Euchi J, Chabchoub H, Yassine A. Metaheuristics optimization via memory to solve the profitable arc tour problem. In: 8th International Conference of Modeling and Simulation; 2010.

[8] Archetti C, Feillet D, Hertz A, Speranza MG. The undirected capacitated arc routing problem with profits. Computers and Operations Research 2010;37:1860–9.

[9] Zachariadis E, Kiranoudis C. Local search for the undirected capacitated arc routing problem with profits. European Journal of Operational Research 2010;210:358–67.

[10] Corberán Ángel, Fernández E, Franquesa C, Sanchis JM. The windy clustered prize-collecting arc routing problem. Transportation Science 2011;45(3):317–34.

[11] Ichoua S, Gendreau M, Potvin J-Y. Vehicle dispatching with time-dependent travel times. European Journal of Operational Research 2003;144:379–96.

[12] Fleischmann B, Gnutzmann S, Sandvoß E. Dynamic vehicle routing based on online trafic information. Transportation Science 2004;38(4):420–33.

[13] Eglese R, Maden W, Slater A. A road timetable^{TM} to aid vehicle routing and scheduling. Computers and Operations Research 2006;33:3508–19.

[14] Maden W, Eglese R, Black D. Vehicle routing and scheduling with time-varying data: a case study. Journal of the Operational Research Society 2010;61:515–22.

[15] Harwood KG, Mumford CL, Eglese R. Single vehicle routing problems with time varying traversal costs. Working Paper.

[16] Soler D, Albiach J, Martínez E. A way to optimally solve a time-dependent vehicle routing problem with time windows. Operations Research Letters 2009;37:37–42.

[17] Tagmouti M, Gendreau M, Potvin J-Y. Arc routing problems with time-dependent service costs. European Journal of Operational Research 2007;181(1):30–9.

[18] Tagmouti M, Gendreau M, Potvin J-Y. A variable neighborhood descent heuristic for arc routing problems with time-dependent service costs. Computers and Industrial Engineering 2010;59:954–63.

[19] Desrosiers J, Laporte G, Sauve M, Soumis F, Taillefer S. Vehicle routing with full loads. Computers and Operations Research 1988;15(3):219–26.

[20] Arunapuram S, Mathur K, Solow D. Vehicle routing and scheduling with full truckloads. Transportation Science 2003;37(2):170–82.

[21] Doerner K, Hartl RF, Reimann M. A hybrid ACO algorithm for the full truckload transportation problem. Working Paper. Vienna University of Economics and Business.

[22] Hirsch P, Gronalt M. Tabu search based solution methods for scheduling log-trucks. In: TRISTAN VI—the sixth triennial symposium on transportation analysis; 2007.

[23] Gronalt M, Hartl RF, Reimann M. New savings based algorithm for time constrained pickup and delivery of full truckloads. European Journal of Operational Research 2003;151:520–35.

[24] Yang J, Jaillet P, Mahmassani H. Real-time multivehicle truckload pickup and delivery problems. Transportation Science 2004;38(2):135–48.

[25] Mladenović N, Hansen P. Variable neighborhood search. Computers and Operations Research 1997;24:1097–100.

[26] Carrabs F, Cordeau JF, Laporte G. Variable neighborhood search for the pickup and delivery traveling salesman problem with LIFO loading. INFORMS Journal on Computing 2007;19:618–32.