Matheuristics for a Multi-attribute Profit Collecting Vehicle Routing Problem

Güneş Erdoğan [*] Fraser McLeod [†] Tom Cherrett [†]
Tolga Bektaş [*]

August 29, 2013

Abstract

This paper studies a multi-attribute profit collecting vehicle routing problem, which arises in the collection operations of a charity organisation in the UK. The problem involves a heterogeneous fleet consisting of vehicles of different capacities, mandatory visits to a subset of vertices, time windows, rest requirements associated with maximum driving and working times, and partial collection. A mixed integer programming formulation of the problem is provided, along with three matheuristics based on Tabu Search and Large Neighbourhood Search. Computational results on instances derived from a case study are presented, as well as the results of the real-world implementation.

1 Introduction

In this paper, we focus on a variant of the *Vehicle Routing Problem* (VRP) with profit collection that arises daily in a charity organisation in the UK. We are given a heterogeneous fleet operating out of a single depot and the quantity of donations at the *banks* and *shops* owned by the charity organisation. Mostly staffed by volunteers, each shop has an associated time window, the start and end times of which vary based on the parking restrictions. Each type of vehicle has an associated travel cost per mile and load capacity, all vehicles are subject to limits on the driving time and the working time, and the drivers are required to rest if they exceed a given driving time limit or a given working time limit. The problem is to determine routes for the vehicles that visit each location no more than once, with the objective of maximising the net profit, computed as the difference of the value of collected donations and

[*]School of Management, University of Southampton, SO17 1BJ, UK {G.Erdogan,T.Bektas}@soton.ac.uk

[†]Transportation Research Group, University of Southampton, SO17 1BJ, UK {F.N.Mcleod,T.J.Cherrett}@soton.ac.uk

the travel cost incurred. We refer to this problem as the Multi-attribute Profit Collecting Vehicle Routing Problem (MPCVRP).

The closest problem to MPCVRP is a variant of the *Vehicle Routing Problem with Private Fleet and Common Carrier* (VRPPC) studied by Ceschia et al. (2011). This problem differs from MPCVRP in terms of the objective function, the existence of a multi-day planning horizon, and soft time windows. The authors have presented a Tabu Search (TS) algorithm for the problem, and have provided results for 18 instances with 56 customers and a CPU time limit of 500 seconds. Baldacci et al. (2010) have presented a unified exact method for solving many variants of the VRP, which can handle most of the attributes of MPCVRP, including time windows, heterogeneous fleet, and vehicle capacity. However, their approach does not cover profit collection and rests. To the best of our knowledge, MPCVRP has not been studied before.

Regarding other related studies, there is a large body of literature on *Traveling Salesman Problems with Profits* (TSPP), for which we refer the reader to the excellent survey by Feillet et al. (2005). There exist three main categories of TSPP: 1) The *Profitable Tour Problem*, in which the objective is to find a tour that minimises the travel cost minus the profits collected. 2) The *Orienteering Problem* (OP), in which the travel cost (or distance, or time) is constrained by an upper bound and the objective is to maximise the profits collected. 3) The *Prize-Collecting Traveling Salesman Problem*, in which the profit collected is constrained by a lower bound and the objective is to minimise the travel cost. Due to the type of objective function and the time window of the depot enforcing a maximum working time, we categorise MPCVRP to be in the intersection of categories 1 and 2. The most popular category is observed to be the OP, for which we refer the interested reader to the recent survey by Vansteenwegen et al. (2011). The generalization of OP to multiple vehicles is named as the *Team Orienteering Problem* (TOP), and has received ample attention from the research community. To the best of our knowledge, the most successful heuristics for the TOP are due to Ke et al. (2008) and Souffriau et al. (2010), whereas the most successful exact algorithm is that of Archetti et al. (2009).

Matheuristics are combinations of metaheuristics and exact optimization methods, which combine the diversification ability of the former and the intensification ability of the latter. Recently, Subramanian et al. (2013) have applied a combination of Iterated Local Search, Random Variable Neighbourhood Descent, and a Set Partitioning Problem formulation to seven variants of the VRP, and reported a number of new best solutions. Similarly, Villegas et al. (2013) have applied a combination of Greedy Randomised Adaptive Search, Iterated Local Search, and a Set Partitioning Problem formulation to the *Truck and Trailer Routing Problem*, and reported results that outperform state-of-the-art results. Matheuristics have also been successfully applied to production-distribution planning, inventory routing, and healthcare (Raa et al. 2013, Coelho et al. 2012, Allaoua et al. 2013).

In this paper, we describe the MPCVRP, propose a mathematical programming formulation and describe three matheuristics that are based on the the TS and Large Neighbourhood Search (LNS) algorithms. We present our computational results

based on instances derived by imitating real data. The rest of this paper is organised as follows. In Section 2, we provide the details of the problem and provide a mixed integer programming formulation. In Section 3, we present our metaheuristic and matheuristic algorithms. In Section 4, we give the results of our algorithms. Finally, we provide our conclusions in Section 5.

2 Problem Definition

The current collection process is based on a fixed schedule of stops at the shops and selective stops at the banks for every day of the week. Albeit simple, this strategy has a number of disadvantages. Firstly, it brings the possibility of inefficient use of vehicle capacity, due to set visits to shops that may not contain enough donations. Secondly, the banks that are already full but cannot be visited run the risk of overfilling and losing donations. Finally, the donations have a significant monetary value and theft of uncollected donations occurs frequently. Hence, a better strategy is to utilise dynamic fill information and to determine the vehicle routes on a daily basis. We now describe the components of the daily collection problem in detail and define the associated parameters.

2.1 Parameters

As mentioned in the introduction, the collection points are composed of banks and shops. Less than half of the collection locations consist of banks: sites containing one or more large bins fitted with infrared remote monitoring sensors that transmit their fill levels twice a day. The remaining collection locations are the shops, which have a larger storage capacity and can relay the fill level as frequently as required. The banks do not need staff to operate and do not normally have associated time windows of work. On the contrary, the shops may have preferences not only about the time of day but also the day of the week of the visit. Shops dictate the day of the visit, which introduces *mandatory* visits into the routes. It is allowed for the vehicles to arrive at shops early and wait until the beginning of the time window. A visit to a location is expected to collect all the donations that have accumulated, with the exception of the last stop before returning to the depot, in which a partial collection due to the capacity constraint is allowed.

Let us denote the set of locations as $V = \{0, 1, ..., n\}$, where 0 corresponds to the depot. For the sake of brevity, we refer to the collection locations as $V_C = V \setminus \{0\}$. We also denote the set of mandatory locations as $T \subseteq V$, with $\{0\} \subseteq T$. We define $[a_i, b_i]$ to be the time window of location $i \in V$, and l_i to be the loading time of the vehicle at the location. The donation profile of a location depends on the demographics of the surrounding district and the profit and quantity collected vary accordingly, which we denote by p_i and q_i, respectively.

The vehicles use the road network connecting the locations, and it is tacitly assumed that they use the shortest path between any two points, which may not be

the case in the existence of heavy traffic or an accident. Let us define the set of arcs as $A = \{(i, j) : i, j \in v, i \neq j\}$. The charity organisation currently owns two types of vehicles, vans and trucks, where the former type has a smaller capacity and can be driven by people with a regular driving license. We denote the set of vehicles as K and weight capacity of vehicle $k \in K$ as Q_k. Due to the difference in size and weight the vehicles have different travel costs and times, which we denote by c_{ijk} and t_{ijk} for arc $(i, j) \in A$ and vehicle $k \in K$, respectively.

EU regulations require the drivers to rest for 45 minutes for every 4.5 hours of accumulated driving. Similarly, the policy of the charity organisation is such that the drivers should rest for 45 minutes for every 6 hours of continuous work. We parametrise these values as T^v_{max} for the accumulated driving time limit, T^w_{max} for the accumulated working time limit, and r for the length of the rest period. Although it is possible for for the drivers to rest at suitable locations on the road, we assume that the rests only occur at collection locations.

2.2 Mixed integer programming formulation

We now present our mixed integer programming formulation for MPCVRP using the following decision variables. Let x_{ijk} be equal to 1 if vehicle k travels from location i to j, and 0 otherwise. Let y_{ik} be equal to 1 if vehicle k visits location $i \in V_C$, and 0 otherwise. Let $\hat{y}_{ik}$ be equal to 1 if the driver of vehicle k rests at location $i \in V_C$, and 0 otherwise. Let u_{ik} be the arrival time of vehicle k at location i. Let v_{ik} be the driving time accumulated by the driver of vehicle k upon arrival at location i. Let w_{ik} be the working time accumulated by the driver of vehicle k upon arrival at location i. Finally, let z_{ik} be the amount picked up by vehicle k at location $i \in V_C$. The corresponding formulation is given below.

(F1)

$$\text{maximise} \quad \sum_{i \in V} \sum_{k \in K} p_i z_{ik} - \sum_{(i,j) \in A} \sum_{k \in K} c_{ijk} x_{ijk} \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in V} x_{ijk} = y_{ik} \quad (i \in V_C, k \in K) \tag{2}$$

$$\sum_{j \in V} x_{jik} = y_{ik} \quad (i \in V_C, k \in K) \tag{3}$$

$$\sum_{j \in V_C} x_{0jk} \leq 1 \quad (k \in K) \tag{4}$$

$$\sum_{j \in V_C} x_{0jk} = \sum_{j \in V_C} x_{j0k} \quad (k \in K) \tag{5}$$

$$\sum_{k \in K} y_{ik} = 1 \quad (i \in T) \tag{6}$$

$$\sum_{k \in K} y_{ik} \leq 1 \quad (i \in V_C \setminus T) \tag{7}$$

$$\sum_{i \in S, j \in V \setminus S} x_{ijk} \geq y_{tk}$$
$$(S \subset V : 2 \leq |S| \leq |V| - 2, T \setminus S \neq \emptyset, t \in S; k \in K) \quad (8)$$
$$\hat{y}_{ik} \leq y_{ik} \quad (i \in V_C, k \in K) \quad (9)$$
$$\sum_{k \in K} z_{ik} \leq Q_k \quad (k \in K) \quad (10)$$
$$z_{ik} \leq q_i y_{ik} \quad (i \in V_C, k \in K) \quad (11)$$
$$z_{ik} \geq q_i \sum_{j \in V_C} x_{ijk} \quad (i \in V_C, k \in K) \quad (12)$$
$$u_{jk} \geq u_{ik} + (t_{ijk} + l_i)x_{ijk} + r\,\hat{y}_{ik} - b_0(1 - x_{ijk})$$
$$((i,j) \in A : j \in V_C, k \in K) \quad (13)$$
$$u_{jk} \leq b_0 - (t_{j0k} + l_j)x_{j0k} - r\,\hat{y}_{jk} \quad (j \in V_C, k \in K) \quad (14)$$
$$v_{jk} \geq v_{ik} + t_{ijk}x_{ijk} - T^v_{max}(1 - x_{ijk} + \hat{y}_{ik})$$
$$((i,j) \in A : j \in V_C, k \in K) \quad (15)$$
$$v_{jk} \leq T^v_{max} - t_{j0k}(x_{j0k} - \hat{y}_{jk}) \quad (j \in V_C, k \in K) \quad (16)$$
$$w_{jk} \geq w_{ik} + (t_{ijk} + l_i)x_{ijk} - T^w_{max}(1 - x_{ijk} + \hat{y}_{ik})$$
$$((i,j) \in A : j \in V_C, k \in K) \quad (17)$$
$$w_{jk} \leq T^w_{max} - (t_{j0k} + l_j)(x_{j0k} - \hat{y}_{jk}) \quad (j \in V_C, k \in K) \quad (18)$$
$$0 \leq u_{ik} \leq b_i - l_i \quad (i \in V, k \in K) \quad (19)$$
$$0 \leq v_{ik} \leq T^v_{max} \quad (i \in V, k \in K) \quad (20)$$
$$0 \leq w_{ik} \leq T^w_{max} \quad (i \in V, k \in K) \quad (21)$$
$$x_{ijk} = 0 \text{ or } 1 \quad ((i,j) \in A, k \in K) \quad (22)$$
$$y_{ik} = 0 \text{ or } 1 \quad (i \in V_C, k \in K) \quad (23)$$
$$\hat{y}_{ik} = 0 \text{ or } 1 \quad (i \in V_C, k \in K) \quad (24)$$
$$z_{ik} \geq 0 \quad (i \in V_C, k \in K). \quad (25)$$

The objective function (1) maximises the profit collected minus the travel cost. Constraints (2) and (3) state that a vehicle must enter and exit a location it is visiting, respectively. Constraints (4) enforce a maximum of one route per vehicle. Constraints (5) are the flow conservation constraints at the depot for the vehicles. Constraints (6) and (7) state that a mandatory location must be visited by a vehicle and other locations may not be visited by more than one vehicle, respectively. Constraints (8) ensure that there is a path from a visited location to a mandatory location. Constraints (9) ensure that a driver can only take a rest at a visited location. Constraints (10) set the upper limit of collection by a vehicle to the capacity of the vehicle. Constraints (11) require a visit at a location for a vehicle to collect at that location. Constraints (12) enforce the collection of all the demand, except when the vehicle is immediately going back to the depot, allowing partial collection only for the last visited location. Constraints (13) and (14) set the upper and lower bounds for the total time accumulated by a vehicle, respectively. Constraints (15) and (16)

set the upper and lower bounds for the driving time accumulated by the driver of a vehicle, respectively. Constraints (17) and (18) set the upper and lower bounds for the working time accumulated by the driver of a vehicle, respectively. Constraints (19) state the time windows for each location. Constraints (20) and (21) state the limits on driving time and working time, respectively. Finally, constraints (22), (23), (24) are binary constraints and constraints (25) are nonnegativity constraints.

As a closing remark for this section, we would like to state that MPCVRP contains many well-known routing problems as special cases, e.g. the TOP is a special case of the MPCVRP with a fleet of a single vehicle type, time windows for the customers that are set larger than that of the depot, and the time window of the depot representing the travel budget of the vehicles. This relationship also proves that MPCVRP is NP-Hard.

3 Matheuristic Algorithms

Formulation F1 presented in the previous section, albeit capturing all the characteristics of the problem, is unable to provide a solution within a reasonable time due to the high number of variables and constraints. We now describe three matheuristics that are capable of finding high quality results in a short time. The matheuristics we describe here have two main components: (i) *intensification* through mathematical programming, and (ii) *diversification* using metaheuristics. We now present the two components in greater detail.

3.1 Mathematical programming component

To present the mathematical programming component of the matheuristics, we need to define a number of parameters. Let us define the set of *types* of vehicles as $\bar{K}$, and denote the number of available vehicles of type $\bar{k} \in \bar{K}$ as $m_{\bar{k}}$. Let us define R to be a set of vehicle routes the members of which are feasible for at least one type of vehicle, with route $i \in R$ having a net profit of $\bar{p}_i$. Let $R_i \subseteq R$ to be the subset of routes that visit vertex $i \in V_C$, with $\cup_{i \in V_C} R_i = R$. Furthermore, let $R^{\bar{k}} \subseteq R$ be the set of routes that are feasible for vehicle type $\bar{k} \in \bar{K}$, with $\cup_{\bar{k} \in \bar{K}} R^{\bar{k}} = R$.

We now provide a formulation that selects routes $i \in R$ such that they cover all mandatory locations and do not use more than the available number of vehicles, so as to maximise the net profit. Let $\bar{x}_i$ be equal to 1 if route i is selected to be a part of the solution, and 0 otherwise. The resulting formulation is given below.

(F2)

$$\text{maximise} \quad \sum_{i \in R} \bar{p}_i \bar{x}_i \tag{26}$$

$$\text{s.t.} \quad \sum_{i \in R_j} \bar{x}_i = 1 \quad (j \in V_C \cap T) \tag{27}$$

$$\sum_{i \in R_j} \bar{x}_i \leq 1 \quad (j \in V_C \setminus T) \tag{28}$$

$$\sum_{i \in R^{\bar{k}}} \bar{x}_i \leq m_{\bar{k}} \quad (\bar{k} \in \bar{K}) \tag{29}$$

$$\bar{x}_i = 0 \text{ or } 1 \quad (i \in R) \tag{30}$$

The objective function (26) maximises the net profit. Constraints (27) ensure that one and only one vehicle visits a mandatory collection location. Constraints (28) state that at most one vehicle can visit a collection location that is not mandatory. Constraints (29) state that the number of routes of a given vehicle type is limited by the number of vehicles available of the type. Finally, constraints (30) are integrality constraints.

Note that formulation F2 can find the optimal solution for the MPCVRP if R contains all feasible routes, which requires either explicit enumeration of the routes or an exact column generation technique, the former being infeasible due to the exponential size of the set and the latter due to a strict CPU time constraint. In order to keep F2 to a manageable size, we set the limit of the cardinality of the route set R to be R_{max}.

3.2 Metaheuristic component

In the remainder of this section, we present the three mathheuristic algorithms we have developed, based on TS and LNS. We refer to the matheuristic algorithms by appending a * sign next to the name of the metaheuristic is it based on. Within the algorithms, we refer to well known route operators of *vertex addition*, *vertex removal*, *vertex relocation*, and *vertex swap*. Vertex addition returns a vertex, the addition of which will result in a maximal profit increase with respect to the current solution. The other three operators behave similarly, with respect to the operations they refer to.

3.2.1 Tabu Search

TS has been successfully applied, both in academia and practice, in many diverse fields such as Group Theory (Gallego et al. 2013), automated code generation (Hyde

et al. 2013), and rostering (Edleston and Bartlett 2012) as well as routing (Moccia et al. 2012). Diversification is achieved through the use of a *tabu list*, which temporarily prohibits the reversal of the moves within the local search neighbourhood. We provide below the TS based matheuristic we have implemented for MPCVRP.

(TS*)

Step 1 (Initialization): Initialize the incumbent solution, the best known solution, tabu list, $R = \emptyset$, and the iteration counter $k = 1$.

Step 2 (Stopping condition): If the time limit is exceeded, stop and report the best known solution.

Step 3 (Local search): Select and apply the best among the operators of vertex addition, vertex removal, and vertex swap, honoring the tabu list.

Step 4 (Route set update): If the incumbent solution is feasible, add the routes in the solution to R.

Step 5 (Intensification): If $|R| > R_{max}$, solve F2. Replace the incumbent solution with the solution of F2. Delete all routes in R and add the routes in the solution of F2 to R.

Step 6 (Best solution update): If the incumbent solution is feasible and its objective value is higher than the best known solution, update the best known solution.

Step 7 (Tabu list update): Add the vertex (or vertices) in the selected operator to the tabu list and increase their tenure by 1. Remove vertices with a tabu tenure that is greater than a pre-specified tenure limit from the tabu list. Increment k and go to Step 2.

3.2.2 Large Neighbourhood Search

LNS has been presented by Shaw (1998) and successfully used by Pisinger and Ropke (2007) for solving five variants of the VRP. LNS achieves diversification by removing a part of a given solution (parametrised as $\alpha\%$) and repairing the solution by a constructive heuristic. The LNS based matheuristic we have implemented for MPCVRP is given below.

(LNS*)

Step 1 (Initialization): Initialize the incumbent solution, the best known solution, $R = \emptyset$, and the iteration counter $k = 1$.

Step 2 (Stopping condition): If the time limit is exceeded, stop and report the best known solution.

Step 3 (Break): Randomly select and remove $\alpha\%$ of the vertices from the incumbent solution.

Step 4 (Repair): Choose the vertex with maximum incremental profit to be added to the solution, until no more vertices can be added.

Step 5 (Polishing): Select and apply the best among the operators of vertex relocation and vertex swap , until no further improvement is possible.

Step 6 (Route set update): If the incumbent solution is feasible, add the routes in the solution to R.

Step 7 (Intensification): If $|R| > R_{max}$, solve F2. Replace the incumbent solution with the solution of F2. Delete all routes in R and add the routes in the solution of F2 to R.

Step 8 (Best solution update): If the incumbent solution is feasible and better than the best known solution, update the best known solution. Increment k and go to Step 2.

3.2.3 Large Neighbourhood Search - Tabu Search

Our third matheuristic is a hybrid of TS and LNS that aims to combine the diversification features of both algorithms. To the best of our knowledge, this is the first attempt to combine the two algorithms. Simply put, we replace the Repair step of LNS with the TS algorithm. We define k_{max} to be the maximum number of TS iterations before termination. We now present the LNS-TS based matheuristic we have implemented for MPCVRP.

(LNS-TS*)

Step 1 (Initialization): Initialize the incumbent solution, the best known solution, $R = \emptyset$, and the tabu list.

Step 2 (Stopping condition LNS): If the time limit is exceeded, stop and report the best known solution.

Step 3 (Break): Randomly select and remove $\alpha\%$ of the vertices from the incumbent solution.

Step 4 (Repair - TS): Initialize the TS iteration counter $k = 1$.

Step 4.1 (Local search): Select and apply the best among the operators of vertex addition, vertex removal, and vertex swap, honoring the tabu list.

Step 4.2 (Best solution update): If the incumbent solution is feasible and its objective value is higher than the best known solution, update the best known solution.

Step 4.3 (Tabu list update): Add the vertex (vertices) in the selected operator to the tabu list and increase their tenure by 1. Remove vertices with a tabu tenure that is greater than a pre-specified tenure limit from the tabu list.

Step 4.4 (Stopping condition TS): Increment k. If $k > k_{max}$, go to Step 5. Else, go to Step 4.1.

Step 5 (Polishing): Select and apply the best among the operators of vertex relocation and vertex swap , until no further improvement is possible.

Step 6 (Route set update): If the incumbent solution is feasible, add the routes in the solution to R.

Step 7 (Intensification): If $|R| > R_{max}$, solve F2. Replace the incumbent solution with the solution of F2. Delete all routes in R and add the routes in the solution of F2 to R.

Step 8 (Best solution update): If the incumbent solution is feasible and better than the best known solution, update the best known solution, and go to Step 2.

4 Computational Experiments

In this section, we provide the details of data generation, the parameter settings, and the results of our computational experiments.

4.1 Data generation and parameter settings

The locations of the depot and the collection locations were provided by the charity organisation, which have been in turn used for obtaining the driving distances and times from a commercial software package. A map showing the locations of the depot and the collection locations consisting of 58 banks and 75 shops are depicted in Figure 1. A bank consists of up to four bins, each of which has a capacity of 270 kg, hence the capacity of a bank is contained in the set $\{270, 540, 810, 1080\}$. The capacities of the shops, the days of the mandatory visits, and the time windows were provided by the charity organisation. The dwell time at a location was determined as the average dwell time for the previous visits. The profit per kilogram of donation was estimated to be £0.80 for the banks and £0.50 for the shops, reflecting the fact that unsold goods tend to have a lower value than goods donated at banks. The fleet of vehicles consisted of a single van and five trucks at the time of writing, the capacities of which were known through their technical specifications. The parameters for the rests were derived from the EU regulations and the company policy, as mentioned in Section 2. The data mentioned so far formed the static setting, and the computational experiments were conducted through varying the daily donations. We have modeled the daily amount of donations at a collection location as a Gaussian random variable. Samples consisting of negative values and values that exceeded the capacity of the collection have been truncated to 0 and the capacity, respectively.

We have generated a total of 50 instances using the static and dynamic data described above. A set of preliminary experiments were conducted to determine the algorithmic parameters. Based on these experiments, the best performance of the TS* algorithm was observed to be for a tabu tenure of 20 and the best results for the LNS* algorithm were observed for $\alpha = 10$. For LNS-TS*, we have used the same values, and have set $k_{max} = 75$. For all three algorithms, we have used $R_{max} = 1500$,

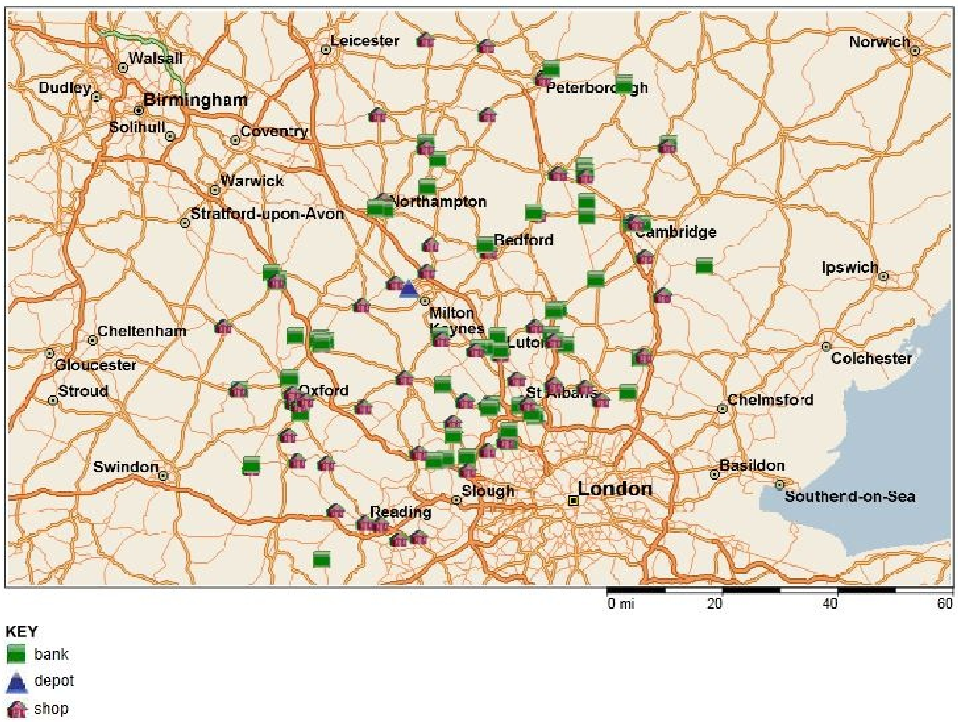

Figure 1: The depot and the collection locations

to ensure that R contained a reasonable number of routes to ensure diversity and yet was of manageable size for F2 to be solved quickly.

4.2 Computational results

All three algorithms were coded using C++ and CPLEX 12.5. The executables were run on the IRIDIS 4 cluster, consisting of 2.6 GHz cores and 4 GB of memory per core. A CPU time limit of 10 minutes was imposed on all three algorithms, as required by the planning process of the charity organisation. We have run the algorithms with and without the intensification step, to be able to compare the performance enhancement provided by the mathematical programming component. A comparison of the deviations of all six heuristics from the best known solutions are given in Table 1. The detailed results are presented in the Appendix, in Table 3 for the metaheuristics and Table 4 for the matheuristics, where the best known solutions are indicated in boldface.

Table 1: Comparison of all six algorithms

	TS	**LNS**	**LNS-TS**	**TS***	**LNS***	**LNS-TS***
Minimum	−19.20%	−3.11%	−7.39%	−19.20%	−1.73%	−7.65%
Average	−12.11%	−0.33%	−4.29%	−10.92%	−0.22%	−4.30%
Maximum	−5.40%	0.00%	−1.47%	−3.92%	0.00%	−1.84%

LNS and LNS* clearly outperform the other algorithms, the former finding the best known solution in 21 instances out of 50, and the latter in 35 instances. The intensification step improves the performance of TS by 1.19% and that of LNS by 0.11% , but has an insignificant effect on LNS-TS. The performances of TS and TS* are remarkably worse than those of the other algorithms. Our intuition is that the algorithmic structure of the TS, in particular the tabu list, is incompatible with mandatory vertices. A mandatory vertex being removed from the solution cannot return to the solution until it reaches the tabu tenure, causing the incumbent solutions for the iterations in-between to be infeasible.

Table 2: Comparison of all six algorithms for the case with no mandatory collection visits

	TS	**LNS**	**LNS-TS**	**TS***	**LNS***	**LNS-TS***
Minimum	−14.96%	−13.56%	−2.32%	−13.96%	−12.80%	−2.14%
Average	−6.91%	−2.73%	−0.50%	−6.30%	−2.76%	−0.27%
Maximum	−1.64%	0.00%	0.00%	−1.41%	0.00%	0.00%

In order to test our conjecture about the effect of mandatory vertices, we have also run the algorithms on the same set of instances with the mandatory collection visits removed, i.e. $T = \{0\}$. Similar to the analysis above, the comparison of the deviations of all six heuristics from the best known solutions are given in Table 2. The detailed results the of the metaheuristics and the matheuristics are provided in the Appendix, in Tables 5 and 6, respectively. In this case, the performances of TS and LNS are quite similar, with LNS still outperforming TS. Remarkably, the distances traveled are significantly shorter for LNS, making it a more desirable algorithm for the practitioners. The performances of LNS-TS and LNS-TS* are significantly better the other algorithms, since both successfully exploit the diversification ability of TS and LNS algorithms. The effect of the intensification step on the average is less pronounced in this case, yet significant improvements for the worst performances have been observed. LNS-TS* successfully finds the best known solutions for 28 instances out of 50, where LNS-TS finds 13, and LNS and LNS* find 5 each.

4.3 Implementation results

The LNS algorithm was used in practice by the charity organisation over a period of 36 working days between 9 May and 19 July 2013 with routes being computed one day in advance of being implemented. The choice of the algorithm was due to its robust performance with the mandatory shop visits. Historical data were used to estimate daily fill levels at the other banks or where the remote monitoring sensors were not functioning satisfactorily. The use of these data and the algorithm resulted in an overall 28% reduction in the number of bank visits, from 953 to 685 visits over the period

The routes that were proposed by the LNS algorithm were examined by the charity organisation's transport manager, with some manual alterations being made

to fit in with the various day-to-day operating conditions that applied, some of which were not considered in the model. The main reasons for changes made were:

- Access: Routes were often reordered manually (sometimes being undertaken in reverse) to avoid the risk of arriving at a shop too early or too late due to changes in the prespecified time window or heavy traffic.
- Balancing workloads: The metaheuristics and matheuristics were not designed to balance workloads between vehicles and crew, and sometimes produced one or more suggested routes that had a relatively heavy or light workload.
- Clustering: This refers to the human preference of assigning clusters of collections to a single route. In several cases the vehicle routing algorithm would assign different collections in the same city to two vehicles, which was altered by the transport manager.
- Inclusion of urgent requests: On a few occasions the transport manager had to respond to additional service requests from shop and bank managers. The extra tasks had to be included manually and sometimes required restructuring of the routes.
- Forced or delayed collections: In a number of cases the algorithm omitted a bank collection, yet the transport manager preferred to force a visit as delaying it might have caused a problem due to the bank overfilling, as it would not have been convenient to visit the bank later in the week. Conversely, sometimes a collection scheduled by the algorithm was preferred to be delayed until later in the week.
- Unavailability of a vehicle or staff: Unforeseeable events such as a last minute breakdown of a vehicle, or a member of staff calling in sick, required the routes to be modified.

As it was not possible to measure the exact monetary value of the collection, the performance of the LNS algorithm was evaluated by comparing times and distances between the routes undertaken during the trial and those that would normally have been undertaken (i.e. the fixed routes). Distance savings, day by day, are shown in Figure 2. The total estimated distance saved over all 36 days was 1159 km, equating to a 3.2% reduction and an average savings of 32 km per day across the vehicle fleet. On most days, total distance was reduced due to a reduction in the numbers of bank visits made; the greatest savings on one day was 191 km on Tuesday 4 June, where the number of bank visits was reduced from 21 to 12. Distance increases of up to 110 km were observed on some days mainly due to the inclusion of one or more bank visits that did not fit onto any of the vehicle routes well. The total estimated time saved was 19.2 hours, equating to a 2.8% reduction and an average of 32 minutes per day across the vehicle fleet. The transport manager appreciated this additional flexibility in the round, 'this gave me the opportunity to phone some shops (e.g. Oxford area) to offer additional collections which the shop managers liked.' A total

CO2 saving of 464 kg was estimated, based on an assumed average emissions rate of 400 g/km and the predominant use of trucks with a carrying capacity of 6 tonnes (den Boer et al. 2011).

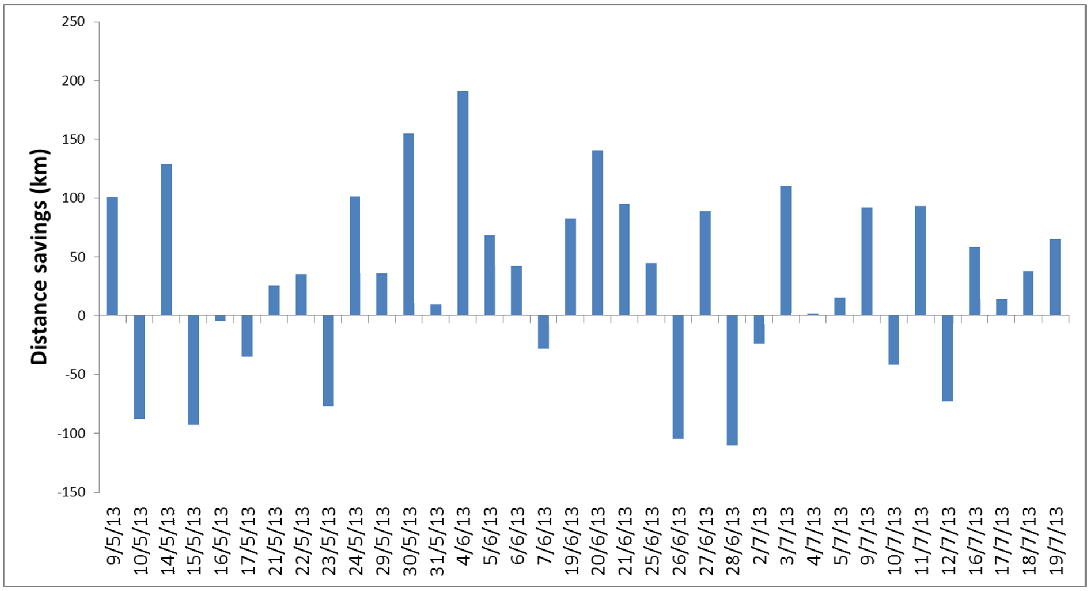

Figure 2: Estimated distance savings by day

5 Conclusions

In this paper, we have studied a multi-attribute profit collecting vehicle routing problem, which arises in a charity organisation in the UK. We have described the problem in detail and provided a mixed integer programming formulation. Constrained by a strict time limit, we have resorted to matheuristics that combine the power of metaheuristics and mathematical programming. We have provided three mathheuristics as well as their metaheuristic counterparts. The computational experiments show that in general the matheuristics outperform their metaheuristic counterparts. For the case with mandatory vertices, LNS and LNS* outperform the others, contrary to the case without the mandatory vertices for which LNS-TS and LNS-TS* perform the best. The LNS algorithm was implemented in practice, and resulted in average round distance and time savings of 3.2% and 2.8% respectively per day.

Acknowledgements: This study has been gratefully supported by the Sixth Sense Transport project (`www.sixthsensetransport.com`), the STRAIGHTSOL project (`www.straightsol.eu`) and the Centre for Operational Research, Management Science and Information Systems (CORMSIS) based within the University of Southampton.

References

H. Allaoua, S. Borne, L. Létocart, and R. Wolfler Calvo. A matheuristic approach for solving a home health care problem. *Electronic Notes in Discrete Mathematics*, 41 (0):471 – 478, 2013.

C. Archetti, D. Feillet, A. Hertz, and M. G. Speranza. The capacitated team orienteering and profitable tour problems. *Journal of the Operational Research Society*, 60:831–842, 2009.

R. Baldacci, E. Bartolini, A. Mingozzi, and R. Roberti. An exact solution framework for a broad class of vehicle routing problems. *Computational Management Science*, 7: 229–268, 2010.

S. Ceschia, L. Gaspero, and A. Schaerf. Tabu search techniques for the heterogeneous vehicle routing problem with time windows and carrier-dependent costs. *Journal of Scheduling*, 14(6):601–615, 2011.

L.C. Coelho, J.-F. Cordeau, and G. Laporte. Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, 24:270 – 287, 2012.

E. den Boer, M. Otten, and H. van Essen. Comparison of various transport modes on a EU scale with the STREAM database. Technical Report 11.4377.53, CE Delft, Delft, Netherlands, 2011.

O.S.S.T. Edleston and L.M. Bartlett. A tabu search algorithm applied to the staffing roster problem of Leicestershire police force. *Journal of the Operational Research Society*, 63:489–496, 2012.

D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39:188–205, 2005.

M. Gallego, M. Laguna, R. Marti, and A. Duarte. Tabu search with strategic oscillation for the maximally diverse grouping problem. *Journal of the Operational Research Society*, 64:724–734, 2013.

M.R. Hyde, E.K. Burke, and G. Kendall. Automated code generation by local search. *Journal of the Operational Research Society*, 2013. Forthcoming.

L. Ke, C. Archetti, and Z. Feng. Ants can solve the team orienteering problem. *Computers & Industrial Engineering*, 54:648–665, 2008.

L. Moccia, J.-F. Cordeau, and G. Laporte. An incremental tabu search heuristic for the generalized vehicle routing problem with time windows. *Journal of the Operational Research Society*, 63:232–244, 2012.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, 2007.

B. Raa, W. Dullaert, and E.-H. Aghezzaf. A matheuristic for aggregate production-distribution planning with mould sharing. *International Journal of Production Economics*, 2013. Forthcoming.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*, pages 417–431, Springer, New York, 1998.

W. Souffriau, P. Vansteenwegen, G. Vanden Berghe, and D. Van Oudheusden. A path relinking approach for the team orienteering problem. *Computers & Operations Research*, 37:1853–1859, 2010.

A. Subramanian, E. Uchoa, and L.S. Ochi. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10):2519 – 2531, 2013.

P. Vansteenwegen, W. Souffriau, and D.V. Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1 – 10, 2011.

J.G. Villegas, C. Prins, C. Prodhon, A.L. Medaglia, and N. Velasco. A matheuristic for the truck and trailer routing problem. *European Journal of Operational Research*, 230(2):231 – 244, 2013.

Appendices

Table 3: The results of the metaheuristics

	TS			LNS			LNS-TS		
Instance	**Collection (kg)**	**Distance (km)**	**Profit (£)**	**Collection (kg)**	**Distance (km)**	**Profit (£)**	**Collection (kg)**	**Distance (km)**	**Profit (£)**
1	6216.0	1061.6	2275.0	6089.0	729.9	**2727.0**	6110.0	840.0	2562.8
2	5536.0	842.9	2088.8	5869.0	749.9	**2528.4**	5866.0	821.3	2396.1
3	5618.0	1054.7	2061.0	5548.0	801.1	**2454.1**	5565.0	849.6	2365.5
4	6370.0	808.0	2728.3	6701.0	784.0	**3044.6**	6726.0	869.5	2893.9
5	7473.0	964.9	2982.1	7489.0	820.1	3239.6	7490.0	883.2	3130.4
6	6164.0	1054.9	2317.6	6089.0	779.3	2727.3	6139.0	913.0	2540.7
7	5717.0	1031.4	1942.0	5628.0	742.4	**2369.2**	5642.0	819.0	2246.1
8	5353.0	964.5	1894.6	5418.0	787.6	**2262.5**	5459.0	872.2	2126.1
9	7536.0	992.5	3347.4	7520.0	800.5	3687.7	7523.0	871.4	3547.2
10	6919.0	1037.8	2600.5	6849.0	840.6	2879.3	6913.0	911.0	2807.9
11	5954.0	973.2	2194.3	5815.0	773.0	2436.2	5895.0	874.3	2323.0
12	6060.0	934.5	2426.9	6028.0	734.7	**2725.2**	6059.0	850.0	2571.2
13	6202.0	1082.6	2382.4	6096.0	806.4	2771.0	6228.0	916.3	2694.5
14	6170.0	1056.1	2262.6	6057.0	828.1	**2583.2**	6169.0	943.9	2466.5
15	7501.0	1030.5	2999.2	7344.0	780.8	3319.3	7503.0	891.3	3241.4
16	5893.0	1089.1	2035.7	5868.0	819.4	2489.3	5893.0	899.7	2359.5
17	6342.0	867.2	2645.7	6482.0	740.1	2975.5	6585.0	876.5	2824.1
18	5858.0	1034.2	2108.3	5923.0	893.8	2428.7	5923.0	895.0	2406.2
19	8159.0	979.4	3685.0	8192.0	798.6	4036.7	8164.0	840.5	3935.2
20	8283.0	1006.3	3569.9	8153.0	780.3	3814.4	8268.0	915.3	3708.7
21	5975.0	1080.1	2061.3	6014.0	820.9	**2551.1**	6077.0	905.3	2464.3
22	7020.0	958.0	3029.4	6905.0	741.3	**3326.4**	7108.0	906.2	3191.6
23	6155.0	996.0	2540.9	6038.0	798.2	2792.5	6204.0	907.9	2732.4
24	6666.0	982.7	2559.6	6567.0	747.4	**2893.1**	6663.0	867.9	2759.3
25	7905.0	955.1	3387.0	7669.0	748.9	3568.9	7913.0	899.5	3507.7
26	6205.0	1070.2	2242.0	6129.0	819.0	2613.3	6205.0	909.1	2514.4
27	5582.0	939.2	2042.5	5483.0	700.6	2371.2	5576.0	822.1	2230.7
28	5747.0	1035.7	2158.8	5589.0	778.5	2484.9	5744.0	883.5	2418.8
29	6859.0	1030.9	2710.2	6727.0	796.3	**3010.7**	6859.0	924.9	2896.9
30	8075.0	903.3	3600.9	8036.0	779.5	3792.3	8179.0	899.7	3703.0
31	6575.0	1044.1	2611.3	6410.0	808.1	2887.1	6580.0	920.6	2826.0
32	7094.0	1111.3	2696.9	6986.0	765.0	**3208.7**	7094.0	898.0	3063.1
33	5260.0	1056.0	1878.2	5147.0	809.2	2168.7	5253.0	925.8	2072.5
34	7417.0	1011.3	3135.0	7395.0	808.8	**3473.3**	7416.0	846.8	3403.7
35	7811.0	1025.8	3088.4	7628.0	765.3	3387.7	7857.0	927.5	3308.2
36	6016.0	1022.7	2281.2	5907.0	824.0	**2550.7**	5999.0	909.9	2455.6
37	5680.0	831.8	2290.7	5916.0	755.0	2615.3	5957.0	884.7	2422.2
38	5564.0	1018.4	2066.1	5385.0	762.8	2383.7	5558.0	886.4	2278.0
39	6303.0	762.8	2645.0	6769.0	815.0	2892.5	6714.0	874.2	2776.0
40	7452.0	1005.4	3018.4	7419.0	811.7	3333.9	7452.0	888.6	3208.4
41	6441.0	1079.8	2408.9	6295.0	782.3	2807.3	6406.0	876.1	2731.9
42	5560.0	862.3	2089.6	5693.0	753.2	**2391.5**	5748.0	874.5	2221.3
43	5412.0	985.9	1936.7	5272.0	774.1	**2177.2**	5376.0	870.2	2102.3
44	6878.0	1000.7	2799.6	6881.0	799.8	3112.2	6877.0	870.2	3021.9
45	6629.0	937.2	2544.0	6542.0	755.5	**2803.6**	6634.0	882.6	2664.7
46	5719.0	1087.9	1985.7	5509.0	814.0	2290.8	5714.0	945.8	2218.2
47	6650.0	921.8	2762.6	6726.0	745.5	**3138.5**	6793.0	875.2	2945.8
48	5474.0	974.6	1889.8	5544.0	787.4	2262.2	5646.0	876.1	2197.4
49	7572.0	944.9	3228.5	7601.0	794.9	**3527.0**	7601.0	857.3	3407.1
50	7525.0	984.4	2929.7	7443.0	752.6	**3269.9**	7455.0	845.8	3116.4

Table 4: Results of the matheuristics

	TS*			LNS*			LNS-TS*		
Instance	**Collection (kg)**	**Distance (km)**	**Profit (£)**	**Collection (kg)**	**Distance (km)**	**Profit (£)**	**Collection (kg)**	**Distance (km)**	**Profit (£)**
1	5936.0	904.1	2326.6	6122.0	771.3	2710.2	6110.0	840.0	2562.8
2	5698.0	859.6	2190.1	5866.0	753.9	2510.9	5893.0	835.6	2391.7
3	5570.0	975.9	2161.4	5548.0	801.1	**2454.1**	5565.0	849.6	2365.5
4	6495.0	843.1	2766.8	6701.0	784.0	**3044.6**	6703.0	876.4	2885.2
5	7472.0	941.4	3016.5	7463.0	784.3	**3256.6**	7490.0	883.2	3130.4
6	6160.0	1021.0	2367.3	6089.0	778.0	**2729.6**	6139.0	913.0	2540.7
7	5498.0	876.6	2031.2	5643.0	761.1	2355.5	5745.0	806.5	2318.5
8	5425.0	923.1	2029.5	5418.0	793.2	2252.7	5459.0	872.2	2126.1
9	7645.0	1061.7	3309.6	7514.0	787.7	**3696.5**	7535.0	876.7	3549.3
10	6921.0	1002.9	2655.4	6737.0	764.2	**2925.3**	6911.0	903.8	2818.9
11	5784.0	901.4	2187.0	5816.0	773.0	**2437.0**	5895.0	874.3	2323.0
12	5891.0	947.2	2262.9	5950.0	716.6	2702.8	6007.0	775.9	2659.1
13	6146.0	1021.3	2444.8	6096.0	802.9	**2777.1**	6228.0	916.3	2694.5
14	6170.0	1042.7	2285.4	6057.0	828.1	**2583.2**	6169.0	943.9	2466.5
15	7492.0	907.3	3209.3	7483.0	828.0	**3340.2**	7491.0	894.7	3230.1
16	5894.0	1010.1	2160.6	5869.0	815.1	**2497.6**	5893.0	899.7	2359.5
17	6507.0	827.0	2849.6	6504.0	747.4	**2980.4**	6606.0	888.3	2823.0
18	5866.0	1035.0	2113.3	5923.0	889.4	**2442.2**	5923.0	900.0	2397.3
19	8211.0	1008.8	3680.1	8192.0	798.1	**4037.6**	8164.0	840.5	3935.2
20	8282.0	995.2	3579.9	8230.0	803.7	**3835.0**	8281.0	935.3	3683.8
21	5975.0	1080.1	2061.3	5898.0	772.4	2542.7	6077.0	905.3	2464.3
22	6918.0	919.1	3033.2	6881.0	729.9	3323.6	7110.0	916.8	3174.9
23	6202.0	1014.2	2567.4	6037.0	797.2	**2793.4**	6204.0	907.9	2732.4
24	6664.0	1024.4	2489.5	6517.0	725.0	2888.8	6663.0	867.9	2759.3
25	7939.0	988.7	3358.4	7669.0	747.8	**3580.3**	7912.0	899.3	3507.2
26	6205.0	1051.0	2275.4	6134.0	827.1	**2618.8**	6205.0	909.1	2514.4
27	5446.0	801.8	2174.2	5416.0	667.4	**2384.5**	5560.0	824.8	2212.6
28	5747.0	1035.7	2158.8	5589.0	776.8	**2487.8**	5745.0	886.7	2416.0
29	6862.0	981.4	2788.4	6812.0	845.1	3000.5	6859.0	924.9	2896.9
30	8142.0	1019.6	3458.6	7962.0	731.1	**3815.6**	8179.0	899.7	3703.0
31	6574.0	1036.4	2622.4	6465.0	832.1	**2900.9**	6584.0	925.1	2821.3
32	7074.0	1012.9	2846.1	6906.0	750.4	3171.0	7094.0	906.5	3055.9
33	5256.0	1023.8	1931.4	5125.0	805.8	**2197.1**	5262.0	936.5	2062.7
34	7415.0	1041.8	3078.9	7415.0	800.1	3465.9	7416.0	846.8	3403.7
35	7785.0	972.5	3147.3	7642.0	756.1	**3415.0**	7858.0	917.1	3326.0
36	6010.0	1015.9	2288.3	5907.0	824.0	**2550.7**	5999.0	909.9	2455.6
37	5680.0	831.8	2290.7	5916.0	755.0	**2615.4**	5957.0	886.1	2419.4
38	5530.0	1039.2	2003.0	5458.0	793.4	**2388.6**	5560.0	917.0	2248.3
39	6303.0	762.8	2645.0	6768.0	815.3	**2932.9**	6714.0	874.2	2776.0
40	7452.0	1001.9	3024.1	7417.0	748.1	**3440.8**	7451.0	889.2	3208.0
41	6403.0	1039.6	2450.1	6295.0	782.6	**2813.9**	6405.0	890.6	2716.1
42	5606.0	822.7	2196.0	5605.0	728.0	2362.6	5748.0	879.0	2208.6
43	5376.0	930.9	2004.2	5270.0	778.2	2176.0	5379.0	872.3	2101.6
44	6883.0	1048.1	2713.9	6836.0	808.1	**3115.6**	6877.0	870.2	3021.9
45	6631.0	958.9	2508.2	6556.0	784.4	2762.6	6634.0	882.6	2664.7
46	5719.0	1087.9	1985.7	5508.0	792.7	**2311.3**	5714.0	945.8	2218.2
47	6703.0	865.8	2904.2	6740.0	765.7	3100.6	6740.0	858.5	2937.2
48	5485.0	942.8	1951.2	5568.0	790.6	**2295.4**	5656.0	884.9	2189.9
49	7601.0	1081.1	3024.8	7601.0	794.9	**3527.0**	7601.0	868.1	3386.9
50	7314.0	801.6	3062.1	7323.0	725.5	3213.4	7456.0	837.1	3130.6

Table 5: Results of the metaheuristics for the case with no mandatory collection visits

	TS			LNS			LNS-TS		
Instance	**Collection (kg)**	**Distance (km)**	**Profit (£)**	**Collection (kg)**	**Distance (km)**	**Profit (£)**	**Collection (kg)**	**Distance (km)**	**Profit (£)**
1	9184.0	970.0	3885.8	8960.0	750.2	4088.5	9674.0	935.4	4203.8
2	8445.0	987.2	3373.4	7667.0	758.2	3340.0	8672.0	889.4	3649.5
3	9062.0	970.9	3877.1	9238.0	917.5	4064.9	9314.0	933.2	4074.2
4	8950.0	996.7	3758.5	8488.0	755.9	3906.3	9011.0	924.2	3949.5
5	9270.0	984.0	3853.1	9263.0	892.4	4004.9	9532.0	915.3	4091.8
6	9684.0	974.1	4197.5	9480.0	835.6	4299.1	10032.0	867.3	**4554.1**
7	8618.0	1014.3	3366.9	8058.0	752.4	3532.1	8784.0	899.5	**3671.6**
8	9328.0	1040.6	3737.9	9987.0	892.3	4305.2	10173.0	901.2	**4395.4**
9	10093.0	1006.7	4568.8	10102.0	898.1	4762.6	10272.0	909.5	4824.7
10	9201.0	999.9	3743.1	8508.0	769.0	3795.8	9293.0	916.4	3972.3
11	9753.0	1038.9	3907.3	10017.0	908.5	4303.0	10142.0	917.5	4340.9
12	9214.0	914.3	3984.9	8684.0	757.9	3954.4	9368.0	899.3	4096.9
13	9489.0	940.1	4153.7	9564.0	899.9	4299.2	9630.0	894.9	4346.2
14	8816.0	937.7	3725.5	8298.0	755.8	3729.0	8960.0	944.0	**3819.1**
15	9143.0	1001.4	3847.9	9229.0	875.5	4104.3	9426.0	913.8	4153.5
16	8731.0	987.4	3549.3	8104.0	797.5	3567.2	8779.0	918.3	3762.9
17	9309.0	986.3	3876.5	9596.0	899.0	4222.5	9686.0	912.4	**4276.4**
18	9553.0	952.5	4053.2	9763.0	899.2	4283.3	9789.0	934.6	4225.0
19	10457.0	1021.5	4741.7	10562.0	882.7	5057.1	10750.0	917.9	5087.8
20	10154.0	993.3	4454.9	10316.0	933.7	4682.5	10273.0	941.3	4661.5
21	9818.0	928.3	4194.4	8905.0	735.5	4007.2	9853.0	910.5	4259.9
22	9505.0	959.0	4258.0	8860.0	747.1	4271.2	9873.0	908.6	**4518.1**
23	10037.0	959.7	4471.6	10348.0	906.7	4734.1	10513.0	927.0	4787.1
24	9101.0	924.3	3837.9	9632.0	873.9	4177.3	9648.0	893.7	4152.7
25	9484.0	984.1	4098.8	9631.0	868.7	**4350.4**	9760.0	953.5	4300.6
26	9547.0	984.7	4008.1	9707.0	923.1	**4198.6**	9582.0	941.7	4120.3
27	9022.0	957.9	3694.8	8575.0	735.8	3810.8	9148.0	906.7	3890.7
28	9090.0	1004.4	3794.6	9168.0	869.1	4090.2	9435.0	928.5	4139.4
29	8801.0	1007.5	3683.3	9099.0	874.6	4023.4	9328.0	926.0	4079.6
30	10000.0	955.5	4461.0	10001.0	889.6	4578.4	10208.0	906.9	4651.1
31	9304.0	978.0	4040.9	9460.0	903.3	4303.1	9683.0	900.5	**4400.4**
32	9555.0	1016.6	4024.2	9922.0	852.0	4463.6	10181.0	917.1	**4515.3**
33	8299.0	1012.8	3421.2	7567.0	774.1	3390.8	8478.0	894.1	**3711.8**
34	9857.0	953.5	4417.3	9027.0	764.8	4311.7	9982.0	909.4	**4550.0**
35	9286.0	979.3	3850.1	9114.0	814.5	4027.0	9699.0	907.6	4196.4
36	9461.0	1011.0	3981.2	9943.0	900.3	4398.1	9949.0	899.7	4437.4
37	8690.0	973.4	3614.3	8956.0	894.6	3871.1	9079.0	902.0	**3910.8**
38	9260.0	953.3	3939.6	9500.0	858.4	**4224.6**	9559.0	906.3	4206.0
39	9518.0	993.7	3933.9	8767.0	777.1	3875.4	9632.0	887.2	**4168.3**
40	9536.0	894.5	4229.3	9557.0	882.0	4254.3	9560.0	881.1	4265.7
41	10227.0	932.7	4466.8	10189.0	869.8	4582.1	10346.0	894.2	4620.1
42	8605.0	947.4	3447.2	8981.0	899.3	**3724.8**	8949.0	890.6	3702.1
43	8442.0	1043.3	3306.1	7180.0	635.8	3285.8	8957.0	930.3	3756.2
44	9706.0	1010.9	4127.2	9624.0	867.5	4348.8	10025.0	912.4	**4495.1**
45	8856.0	1027.4	3485.6	8373.0	725.3	3726.6	9351.0	950.9	3863.9
46	9697.0	944.0	4179.0	9906.0	923.8	4293.6	9951.0	914.3	4345.9
47	9418.0	998.8	4052.5	9476.0	933.5	4166.5	9491.0	928.5	4180.3
48	9101.0	981.8	3680.6	9680.0	877.7	4121.9	9699.0	906.7	4137.5
49	10179.0	941.1	4535.3	10479.0	922.0	4699.3	10480.0	935.5	4692.1
50	9632.0	995.7	3915.7	9498.0	809.1	**4154.9**	9795.0	914.5	4140.1

Table 6: Results of the matheuristics for the case with no mandatory collection visits

	TS*			LNS*			LNS-TS*		
Instance	**Amount collected**	**Distance (km)**	**Profit (£)**	**Amount collected**	**Distance (km)**	**Profit (£)**	**Amount collected**	**Distance (km)**	**Profit (£)**
1	9537	961.4	4069.5	9059	774	4089.2	9640	905.8	**4229.7**
2	8403	977.5	3379.3	7850	**748.3**	3427.6	**8682**	904	**3651.2**
3	8807	983.8	3670.5	9153	**870.9**	4072	**9355**	952.9	**4091.2**
4	8982	994.5	3789.4	8425	**736.6**	3910.1	**9049**	909.8	**3986**
5	9427	979.5	3943.3	8786	**749.3**	3942.1	9524	914.1	**4108.6**
6	9697	970.2	4209.8	9248	**735.3**	4338.8	**10032**	867.8	4551.8
7	8530	996.9	3357.3	8124	760.8	3543.9	8734	898.1	3650.4
8	9624	1018.4	3920.3	10122	913.9	4339.9	10164	898	4394.9
9	10081	1020.4	4555.6	10140	**863.1**	4816	**10346**	927.4	**4839.8**
10	9147	1009.9	3702.9	9298	886	**3999.1**	**9326**	913.9	3993.8
11	9930	1002.1	4059.5	9919	**860.2**	4306	**10188**	920.1	**4364.9**
12	9149	915.2	3960.1	8689	**746.4**	3981.7	**9467**	897.5	**4133.6**
13	9333	919.1	4135.6	8686	**793.3**	4012.7	**9641**	913.3	**4364.9**
14	8787	945.9	3704.1	8249	**750.3**	3711.4	8897	932	3784.6
15	9088	996	3839	8535	**724.7**	4008	**9454**	918.7	**4160.8**
16	8166	961.6	3350.7	7113	**609.4**	3321.4	**8909**	931.2	**3809.1**
17	9284	991.7	3849.5	8937	**780.9**	4056.2	**9695**	922.2	4240.4
18	9652	962.7	4084.6	9830	**896.6**	**4325.5**	**9849**	943.6	4240.2
19	10474	1010.6	4758.3	10649	898.1	5055.3	**10808**	916.9	**5114.7**
20	10078	925.4	4530.7	10304	926.1	4665.6	**10393**	950.6	**4696.8**
21	9602	913.1	4125.5	8928	742.5	4025	**9877**	920.4	**4264.2**
22	9518	956.6	4270.8	9046	767.5	4320.9	**9966**	934.4	4516.9
23	10377	958.2	4635.8	9567	**770.5**	4534.5	10492	904.4	**4837.5**
24	9239	905.3	3924	8907	**760.6**	3999.8	**9663**	882.9	**4179.3**
25	9580	995	4153.8	9511	**847.3**	4321.4	**9788**	939.7	4305.2
26	9508	965	4024.9	8870	**740.4**	4065.1	9683	949.8	4146.4
27	9015	945.5	3706.5	8577	741.3	3810.3	**9172**	911.3	**3899.3**
28	8935	1021.7	3698.9	9186	869.7	4091.5	9371	886.4	**4146**
29	9172	972.7	3906.3	8463	**756.3**	3851.9	**9359**	929.8	**4087.1**
30	10031	961	4464.4	10166	908.8	4642	**10274**	915.1	**4671.8**
31	9505	997.8	4125.3	9581	910.2	4349.5	9624	**887.6**	4389.7
32	10111	938.9	4451.5	9936	**851.5**	4482.4	10133	906.7	4514.6
33	8267	976.9	3458.1	7740	**737.4**	3569.5	8455	895.3	3698.2
34	9849	952.7	4413.3	9072	**759.3**	4338.2	9961	912.9	4544.8
35	9453	947.6	3992.8	9280	865.2	4038	**9746**	913.8	**4206.7**
36	9482	1008.1	3989.3	10040	912	4452.7	**10101**	914.2	**4484.2**
37	8608	974.3	3562.3	8981	895.1	3876.1	8938	928.7	3827
38	9257	959	3933.7	9511	867	4219.2	9541	903.2	4208.4
39	9341	917.4	3985	9554	868.2	**4168.3**	9486	865	4144.7
40	9566	915.5	4211.7	9623	892.3	4290.9	**9627**	**875.4**	**4309.2**
41	10139	951.4	4404.9	9407	**719.2**	4391.9	**10401**	906.4	**4651.6**
42	8654	957.7	3462.1	8785	**868.4**	3679.5	8974	893.8	3717.4
43	8624	1028.5	3419.5	8116	747.3	3592.3	**8997**	917.3	**3801.2**
44	9739	1009.5	4146.4	9247	**771.5**	4273	9811	882.2	4441.1
45	8934	961.7	3648.9	8607	743.2	3821.4	9308	923.2	**3895.1**
46	9863	946	4239.7	9903	925.5	4298.8	**10003**	**910.3**	**4366.1**
47	9472	980.1	4096.3	9481	**886.7**	**4239.4**	**9564**	937.9	4210.3
48	8956	1016.6	3570.1	9699	878.7	**4149.5**	**9765**	924.3	4136.6
49	10041	976.4	4402.4	10255	**876.6**	4681.9	**10507**	926.1	**4714.4**
50	9716	1055.4	3855.1	9296	**744**	4138	9788	915.7	4131.2