A Bi-Level Approach to Vehicle Fleet Reduction: Successful Case Study in Community Healthcare

Alexander E.I. Brownlee
alexander.brownlee@stir.ac.uk
University of Stirling
Computing Science and Mathematics
Stirling, UK

Sarah L. Thomson
s.thomson4@napier.ac.uk
Edinburgh Napier University
Computing, Engineering & The Built Environment
Edinburgh, UK

Rachael Oladapo
ooladapo@mail.com
NHS Scotland
Manchester, UK

ABSTRACT

We report on a case study application of metaheuristics with Argyll and Bute Health and Social Care Partnership in the West of Scotland. The Partnership maintains a fleet of pool vehicles that are available to service visits of staff to locations across a largely rural area. Maintaining such a fleet is important but costly: we show how the allocation of fleet vehicles can be formulated as a bilevel optimisation problem. At the upper level, vehicles are allocated to 'base' locations such as hospitals. At the lower level, vehicles are allocated to specific jobs. We explore local-search approaches to solving this problem. We show that some blurring of the distinction between upper and lower levels can be helpful for this problem. We also demonstrate, for our case study, a 7.1% reduction in the vehicle fleet while still being able to meet all demand.

CCS CONCEPTS

• **Mathematics of computing → Graph algorithms**; *Combinatorial algorithms*; • **Theory of computation → Evolutionary algorithms**.

KEYWORDS

Optimal job scheduling, evolutionary computation, bilevel optimization

ACM Reference Format:
Alexander E.I. Brownlee, Sarah L. Thomson, and Rachael Oladapo. 2024. A Bi-Level Approach to Vehicle Fleet Reduction: Successful Case Study in Community Healthcare. In *Genetic and Evolutionary Computation Conference (GECCO '24 Companion), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3638530.3664137

1 INTRODUCTION

As the UK seeks to switch to zero emission electric vehicles [12] to benefit from reduced emissions [13], public sector bodies are increasingly prioritising such a switch for their own fleet transportation. One large scale user of fleet transportation is the National Health Service (NHS), which maintains a large fleet of leased vehicles to facilitate home visits and connectivity between many small health facilities. The fleet is particularly important in rural locations, and reducing the leasing costs of the fleet is of high importance, as money spent on vehicles is not spent directly on care. This will also ease the switch to electric: although the reduced fuel costs represent long-term savings, initial leasing costs for electric vehicles are higher than for petrol and diesel. Therefore, it is required to reduce the scale of the fleet by removing any excess capacity, while still being able to complete all required operations. Yet there are many complex requirements for the fleet that make optimisation of vehicle numbers and types challenging.

We focus on a case study with Argyll and Bute NHS in the West of Scotland. Argyll and Bute is a largely rural area with a small number of moderately sized towns, with fleet vehicles required to cover a very large area. Statistical analysis of the fleet's operations revealed that vehicles were in use (driving on the road) around 25% of the time. While much of this 'idle' time is necessary (e.g., while a vehicle, having transported staff to a remote location, waits for the return journey), this data suggested that there was potential scope for reducing the fleet size by careful placement of the vehicles.

In this paper, we formulate the reduction of a vehicle fleet as a bilevel optimisation problem [6, 10]. The upper level attempts to optimise the overall goal, with the lower level acting as a constraint on the upper level solutions.

- The *upper level* allocates vehicles to base locations and is similar to a facility location problem with discrete locations. This is an optimisation problem with the objective of reducing the number of vehicles.
- The *lower level* determines whether the given allocation of vehicles can serve the job requirements of that location. It is the decision version of the optimal job scheduling problem, and is solely concerned with determining whether a feasible solution exists. That is, one in which all jobs are successfully completed.

We develop a stochastic local search for the upper level and a constructive heuristic search for the lower level problem, and demonstrate their combined performance on our case study when compared to random search baselines. We show that some blurring of the distinction between levels is useful, whereby the lower level algorithm is able to refine the allocations of vehicles while allocating them to jobs. For the real-world application, we are able to determine that a potential reduction to the fleet of 7.1% is possible.

2 RELATED WORK

Many problems can be formulated as bilevel and many successful approaches have been developed in recent years [10]. Metaheuristics have proven to be a popular solution approach for these typically very difficult problems [6]. Research in this area has also extended

GECCO '24 Companion, July 14–18, 2024, Melbourne, VIC, Australia
2024. ACM ISBN 979-8-4007-0495-6/24/07.
https://doi.org/10.1145/3638530.3664137

to other matters such as handling uncertainty [3] and multiple objectives [15].

Our upper level is similar to a Facility Location Problem, which has frequently been formulated as a bilevel problem [1, 4, 7, 9], including [11] which includes freight transport routing. Typically the levels are determining which facilities to open, and which customers to assign to each facility. Our upper level is similar, in that vehicles (facilities) are allocated to one of a fixed number of locations. A key difference with our target problem is that the low level is a time-series problem of allocating the vehicles to jobs, rather more like optimal job scheduling than allocating customers to a particular location. We also allow for vehicles to change bases (i.e. facilities change location) from time to time.

Related to optimal job scheduling, flow shop scheduling has itself been formulated as a bilevel problem, whereby the upper level is the allocation of jobs to machines and the lower level is scheduling on each machine [2, 14]. In our case, the problem at each base can still be solved in a single algorithm run, so this further division of the problem is unnecessary.

3 PROBLEM DEFINITION

The overall problem is to determine the allocation of vehicles to bases that minimises the total fleet size while still being able to complete the same set of jobs. The overall problem is treated as a bilevel optimisation problem:

- The upper level (outer) problem is an allocation of vehicles to each base. (number of vehicles of each of the three types; these can change over time). Objective: minimise the total number of vehicles.
- The lower level (inner) problem determines whether a specific allocation of vehicles made at the upper level will be able to service all the jobs (i.e., is the allocation feasible). Job start and end times are fixed. The search operators are designed to keep changes from the existing allocation of vehicles to a minimum in order to minimise disruption as far as possible.

We now define the notation used in the rest of the paper, the assumptions underpinning the problem, and set out the lower and upper level problems.

3.1 Definitions

We begin with some definitions:

- Base: a location at which vehicles are parked. These include hospitals and local health centres, generally near to a cluster of staff using the vehicles.
- Job: a journey that starts and ends at a base, such as a series of home visits performed by a single member of staff, or the transport of equipment.
- Vehicle: a car or van represented by a registration number. Vehicles are grouped into three types: smallcar, estate, van, each being best suited to particular types of jobs. Each vehicle is assigned to a specific base for a period of time.
- Swap: reallocating a vehicle from one base to another.

Table 1 lists the major elements of notation related to these definitions used later in the paper.

Symbol	Description
$B = \{b_1, b_2, \ldots, b_m\}$	all bases
$V = \{v_1, v_2, \ldots, v_n\}$	all vehicles
$J = \{j_1, j_2, \ldots, j_p\}$	all jobs
$V^b = \{v_1^b, v_2^b, \ldots\}$	vehicles allocated to base b
$J^b = \{j_1^b, j_2^b, \ldots\}$	jobs at base b

Table 1: Notation

3.2 Assumptions

Building on the definitions above, there are also some rules / assumptions that constrain the problem:

- Vehicles are only allowed to swap between certain bases. Swaps between bases on different islands or the mainland are too costly and impractical to be implemented routinely.
- It is preferred for a vehicle of the same type to do a job, but occasionally a different type of vehicle can be used. Jobs performed by a smallcar can be performed by any other vehicle; estates can be replaced by vans and vice-versa.
- A new swap cannot be introduced if it involves a ferry journey; however existing swaps that involve a ferry can be retained.

3.3 Low level: allocating vehicles to jobs

Despite the use of vehicles and jobs, the lower level problem is not quite an instance of Vehicle Allocation Problem [8], which involves allocating a fleet of vehicles to attend to the expected demand for freight transportation between terminals along a finite multiperiod planning horizon. Rather, it is closer to an instance of optimal job or machine scheduling [5, 16]. Here, we have a list of jobs (trips to visit a list of patients) that need allocated to vehicles (machines). The goal is to successfully allocate all jobs to a machine. There are several constraints:

(1) All jobs have a specific category (the vehicle type), and machines can service specific categories (van and estate can service any job, smallcar can only service smallcar jobs).
(2) Start and end times are fixed per job.
(3) Machines are independent of each other and can only service one job at a time.

Let $J^b = \{j_1^b, j_2^b, \ldots, j_p^b\}$ be the set of jobs for a given base b and $V^b = \{v_1^b, v_2^b, \ldots, v_n^b\}$ be the set of vehicles allocated to b (the vehicles can be viewed as the 'machines'). Each vehicle can only service one job at a time. Each job has fixed start and end times. Furthermore, we introduce the possibility of adding vehicles to V^b by taking one not in use from the vehicles of another base V^{other}. The lower-level problem's goal is to determine whether there exists an assignment x of all jobs in J^b to a member of V^b without any of the J^b that share a common V^b also overlapping in time.

3.4 Upper level: allocating vehicles to bases

The upper level problem is the high level allocation of vehicles to each base. More formally, let $B = \{b_1, b_2, \ldots, b_m\}$ be the set of bases and $V = \{v_1, v_2, \ldots, v_n\}$ be the set of vehicles in the fleet. Our goals are to minimise the number of vehicles in the fleet n while

allocating a subset of V to each V^b such that the low level problems can all be satisfied. The original allocation is actually treated as a starting point, because the low level algorithm is then also able to swap vehicles between bases at later points in time.

4 SOLUTION METHODOLOGY

A telematics system was installed in all vehicles of the NHS fleet targeted by our optimisation approach. The telematics device tracks the activities of the vehicles and drivers' performances, by collating and collecting data such as the vehicle mileage, safety score, location, idle time, and drive time, fuel consumption and drivers' performance, etc. This provides us with a rich historical data set of vehicle movements, specifying where and when each vehicle was used over a long period of time. Further details of this data set are given in Section 5.

We begin all searches from the historical fleet and its actual schedule. Vehicles are placed at the bases where they were located at the start of our historical data set, and allocated to the jobs they originally performed. A stochastic local search algorithm was applied at the upper level, allocating vehicles to bases, with a constructive heuristic at the lower level allocating vehicles to jobs. The top level algorithm calls the low level algorithm to determine whether particular vehicle allocations are feasible. The algorithm starts with the current allocations of vehicles and then tries to improve on that allocation, to minimise the changes from the current setup.

As this is a novel applied problem, to provide a baseline for comparison at both levels we also compare with a random search. The remainder of this section details our algorithm implementations.

4.1 Lower level: constructive heuristic search

The lower-level problem is the assignment of vehicles to jobs (trips) to determine whether the upper level solution is feasible (i.e., enough vehicles have been allocated to service all jobs). This is a decision version of the optimal job scheduling problem. Before all searches, we initialise the starting solution as the historical version of the schedule: that is, we assign vehicles to the jobs that they actually were assigned to in the data. After that, we apply constructive heuristic search. These lower-level searches happen within the scope of one base (although other bases can become involved if vehicle swaps occur). This is because the upper-level search proposes to remove a single vehicle from a specified base at every iteration. The lower-level search then checks if this removal can still result in a valid assignment of vehicles to jobs for that base.

This is the process shown in Algorithm 1. A list of eligible vehicles for the base is built. Those are — in order — the vehicles already at that base, and also vehicles at other bases which have historically done jobs for that base at least once. For each job, we first consider vehicles already at the base. If the vehicle is not in use, it is assigned to the job. If it is in use, but there is time between the end time of its current job and the new proposed job, it is assigned. If there are no available vehicles at the current base, then vehicles from other bases are considered. In these cases, a **satcheck** takes place: a check for whether the other base can still handle all its jobs with the loss of the proposed vehicle. If it can, the vehicle is assigned to the job and swaps to the base. We define the process of considering any vehicle for assignment to a job as a **fitcheck**.

For the random search baseline variant of this algorithm, there is only one difference: there is a random shuffling of the order in which vehicles are considered for assignment. We also considered a variant where swaps between bases were disallowed at the lower level.

4.2 Upper level: stochastic local search

The upper search is a stochastic local search, where the operator is removing vehicles from the fleet. We begin from the actual historical fleet and try to reduce from there. Removals are only allowed when there is a feasible job allocation on the lower level possible with the new reduced fleet (Algorithm 1). Only if that constraint is satisfied is the new fleet accepted as the incumbent solution on the upper level; the neighbourhood is explored looking for an improvement. The process for our upper search is captured in Algorithm 2. The operation which removes vehicles from the fleet is a semi-guided mutation: it is subject to a parameter m. This is used in the following way: a group of vehicles of type t at a given base are only considered for removal if the group has a size of at least m. Our experiments considered two values for m: 1 (i.e., vehicles may be removed from any base) and, based on our observations of typical vehicle counts at each base, 4.

We also implement a random search baseline variant of this level: the process is similar to Algorithm 2, except at each iteration a random number r between 1 and 10 is generated. After that, r vehicles are chosen uniformly at random and removed from the fleet; thereafter, checks are carried out to see if this is still a valid fleet (that is, for each individual vehicle removal, we check with the lower level search in Algorithm 1 whether a valid job assignment is still possible for the affected base). After all iterations are complete, the fleet with the largest reduction which is also **valid** is the output of the search.

5 CASE STUDY DETAILS

Our experiments focus on optimisation of the vehicle fleet operated by NHS Argyll and Bute, which covers an area of approximately 70 000km^2 around the West coast of Scotland and serves around 85 000 people. Within the region are 18 bases from which the fleet vehicles operate. Figure 1 shows the area, with each base highlighted by a marker.

Our case study focuses on the 84 active vehicles with telematics fitted. This excludes several fleet vehicles that were undergoing scheduled maintenance or could certainly not be removed for operational reasons. The telematics devices are able to track the activities of the vehicles, allowing us to model the current usage patterns of the fleet and identify any opportunities for efficiencies. We were provided with data spanning a period of approximately 4 months, from 14/07/2022 00:58:33 to 16/11/2022 21:46:59. Some basic data cleaning was required to remove errors (e.g., zero-length trips due to an engine being turned on then off again, or where the telematics device failed to log vehicle coordinates correctly). The data actually contains point-to-point trips (engine-on to engine-off). This cleaning reduced the number of point-to-point trips from 57 850 to 52 999. These trips were then grouped. 43 659 trips were grouped into 13 208 jobs (i.e., journeys starting and ending at a known base),

and 9340 trips were grouped into 3797 swaps (i.e. journeys starting and ending at different bases). Each job also has an associated vehicle type (the vehicle which originally performed the job).

Thus our case study problem has 84 vehicles of 3 types, 18 bases, 13 208 jobs, and 3797 swaps between bases. The existing swaps in the historical data were retained in all solutions to ensure that equipment and staff movements were accounted for. As the jobs were constructed from known movements to always start and end at the same base (i.e., being round trips), we ensured that idle time caused by waiting for staff to complete work at a remote location was never eliminated. Repairs to vehicles also took place at remote locations and were accounted for in the same way (a repair counting as any other job). Thus, any slack removed by our approach was genuinely idle time that could be removed.

6 EXPERIMENTAL SETUP

We combined the different configurations described in Section 4 to form 9 variants. We have adopted a coding system for these in the results, detailed in Table 2. Following this pattern, e.g., u1l2m4 means an algorithm having random search at the upper level, heuristic search at the lower level, and removals limited to cases where there are four or more vehicles of the same type at a base. We use a * to denote all variants matching a configuration at particular levels. u1l2* means all variants with random search at the upper level and heuristic search at the lower level.

Algorithm 1 Lower level constructive heuristic search for one vehicle base

Output: x ▷ assignment of vehicles to jobs for b
Initialisation:
$J^b = \{J_1^b, J_2^b, \ldots, J_p^b\}$ ▷ jobs for this base
$V^{cb} = V_1^{cb}, V_2^{cb}, \ldots$ ▷ vehicles at current base
$V^{ob} = V_1^{ob}, V_2^{ob}, \ldots$ ▷ vehicles at other bases eligible for swap
$V^a = V^{cb} \cup V^{ob}$ ▷ all eligible vehicles in V; sorted with those at current base first
▷ Note: mX variant has $V^a = V^{cb}$, excluding vehicles from other bases
for j_i^b in J^b **do**
  for v_k in V^a **do** ▷ **fitcheck**: checking a vehicle for assignment to a particular job
    if $v_k \in V^{cb}$ **then**
      if v_k is not in use **then**
        assign v_k to j_i in x
      else
        if v_k.currentjob.endTime < j_i.startTime **then** ▷ i.e., v_k is in use, but clear in time for current job
          assign v_k to j_i in x
        end if
      end if
    end if
    if $v_k \in V^{ob}$ **then**
      sat = `satcheck`($v.currentBase$) ▷ **satcheck**: whether the base can satisfy its jobs without v
      if sat **then**
        if v_k is not in use **then**
          assign v_k to j_i in x
          `swapvehicle(v_k, v_k.currentBase, cb)` ▷ swap the vehicle to the present base
        end if
      end if
    end if
  end for
end for

Algorithm 2 Upper level stochastic local search

Input: CF ▷ historical fleet encoded as counts for vehicle types at each base
Output: CF ▷ fleet, potentially reduced in size
for i in 1..$iterations$ **do**
  MF = `removevehicle(CF, m)` ▷ remove vehicle subject to m; MF is mutated fleet
  sat = `lowersearch(MF, re)` ▷ check whether the base can satisfy its jobs without the vehicle proposed for removal with Alg. 1
  if $sat = true$ **then** ▷ this solution is an improvement: fewer vehicles while completing all jobs
    $CF \leftarrow MF$
  end if
end for

Figure 1: The geographic region covered by the case study. Red markers show the locations of bases, with labels giving base name and lat/lon. Approximate width of this figure is 200km. Maps data © Google, Data SIO, NOAA, U.S. Navy, NGA, GEBCO. Imagery: Landsat / Copernicus.

We conduct 100 independent runs of each algorithm variant and report the aggregate results of these in the following section.

7 RESULTS

The current best reduction we have found is as follows: All 13208 jobs have vehicles allocated to them. The fleet of 84 was reduced by 6 vehicles, all smallcars, representing a reduction of 7.1% and an annual saving of £19 662, assuming a typical annual leasing charge for an electric vehicle of £3277.

7.1 Reductions to vehicle count

Table 3 reports the headline results for each algorithm in terms of the number of vehicles removed and the increase in distance travelled due to vehicles swapping bases. 'Median reduction' is the median number of vehicles removed over all 100 repeat runs of the algorithm. The only algorithms where this was non-zero (meaning that at least half the runs removed at least one vehicle) were those without random search at either level; algorithm IDs u2l2*.

'% removing at least' are the percentages of the repeat runs where at least 1...6 vehicles where removed. Only the u2l2* variants were able to find solutions removing 4 or more vehicles, and of those, u2l4m4 most frequently found solutions removing each number of vehicles. u2l4m4 was the only approach able to find any solutions removing 6 vehicles.

'Median swap distance' is the median extra distance travelled due to new swaps being added, over the runs removing 1...6 vehicles. u2l2m4 was able to find greater reductions in vehicles by being more free to make swaps between bases; this then comes at the cost of some additional distance travelled. However, the total distance covered by the jobs to be completed and the existing swaps in the historical data is 93 723 miles, so this additional distance covered is, in practice, negligible.

Strictly speaking, allowing swaps between bases in the low level problem is a blurring of the distinction between upper and lower levels. The upper level algorithm is primarily responsible for allocating vehicles to bases. However, for the application at hand, allowing the lower algorithm some capacity to move vehicles between bases (i.e., lower level problems) when it becomes apparent that this is necessary means that it is able to find better solutions overall. Thus u2l2m4 is able to find better solutions than u2l2mX.

Table 2: Algorithm variant codes

Code	Description
u1	Upper level random search
u2	Upper level stochastic local search
l1	Lower level random search
l2	Lower level constructive heuristic search
m1	Upper level can remove any available vehicle
m4	Upper level can only remove vehicles where at least 4 of same type are present
mX	Upper level as for m4, lower level no swaps allowed

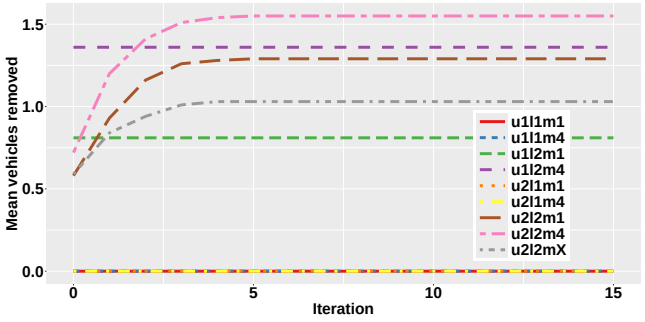

Figure 2: Vehicle reductions over outer loop iterations. The lineas for u1l1m1, u1l1m4, u2l1m1, and u2l1m4 are overlaid on each other at zero.

7.2 Upper search convergence

The upper level search was limited to 15 iterations per run. The reason for this is that, if a reduction was going to be made, it would happen quickly, and more iterations did not lead to further fleet reductions. Figure 2 reports the mean reduction at each iteration over all repeat runs of each algorithm configuration. In all cases, the upper level search had converged by 6 iterations.

7.3 Computational overhead

The previous section notes that the upper level algorithms were limited to 15 iterations. The lower level algorithms account for most of the computational overhead of our approach. Figure 3 shows the distribution of run times on our experimental platform,

Table 3: Main results for vehicle reductions, and extra distance travelled by vehicles swapping bases, for each algorithm. 'Median reduction' is the median number of vehicles removed over all 100 repeat runs of the algorithm. '% removing at least' are the percentages of the repeat runs where at least 1...6 vehicles where removed (so, 41% of u2l2m4 runs removed one or more vehicles). 'Median swap distance' is the median extra distance travelled due to swaps, over the runs removing 1...6 vehicles.

Algorithm	Median	% removing at least:						Median swap distance (miles)					
	Reduction	1	2	3	4	5	6	1	2	3	4	5	6
u1l1m1	0	0	0	0	0	0	0	0	0	0	0	0	0
u1l1m4	0	0	0	0	0	0	0	0	0	0	0	0	0
u1l2m1	0	13	5	3	0	0	0	0	0	0	0	0	0
u1l2m4	0	18	12	4	0	0	0	0	0	0	0	0	0
u2l1m1	0	0	0	0	0	0	0	0	0	0	0	0	0
u2l1m4	0	0	0	0	0	0	0	0	0	0	0	0	0
u2l2m1	1	54	29	18	9	2	0	0	0	0	0	22.5	0
u2l2m4	1	61	41	25	13	5	3	0	0	45	45	45	45
u2l2mX	1	59	25	11	6	2	0	0	0	0	0	0	0

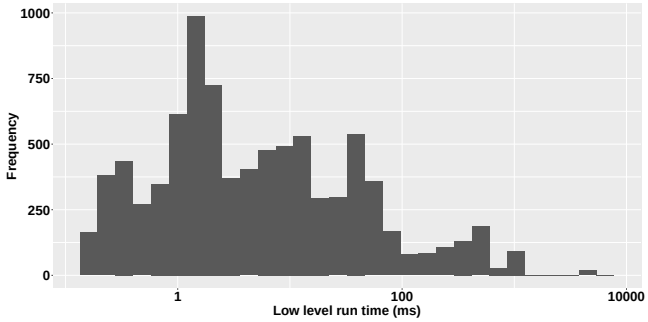

Figure 3: Distribution of run times over all repeat runs of all algorithms. The times are the wall-clock time, in milliseconds, to complete one run of the lower level algorithm. Note that the times are presented on a log scale due to the long tail.

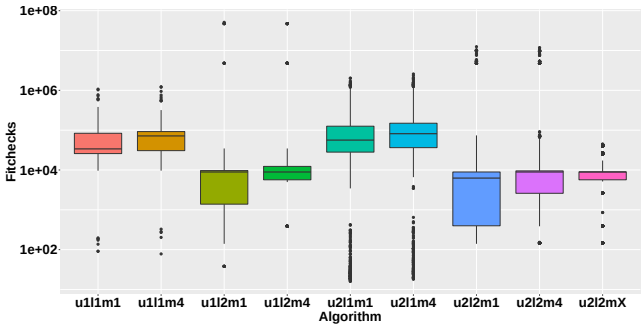

Figure 4: Number of calls to fitcheck by each algorithm across all repeat runs.

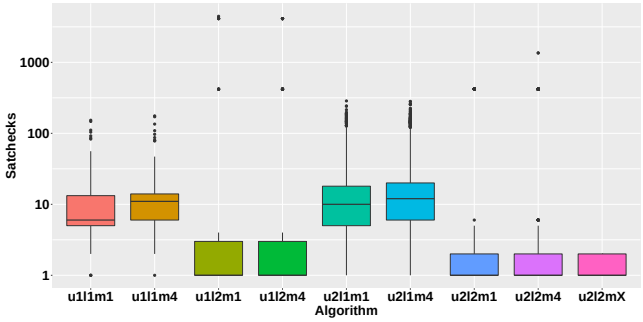

Figure 5: Number of calls to satcheck by each algorithm across all repeat runs.

a machine with 20x 12th Gen Intel i9-12900HK cores and 32GB of memory, running Ubuntu 22.04. As is often the case with run time measurements, there is a long tail. Most of the run time is due to the checks to determine whether a vehicle can be swapped with another (fitcheck) and whether a base can lose a proposed vehicle (satcheck), which are called many times. Table 4 and Figures 4 and 5 compare the approaches in terms of the number of calls to these functions, broken down by algorithm. In general, the approaches using random search at the low level (*l1*) are longer running that the guided approaches (*l2*), with l2 approaches using fewer calls to both fitcheck and satcheck. u2l2fX uses fewer calls than any other approach because it disallows swaps between bases. However, this comes at the cost of being unable to reduce the vehicle count by as much as u2l2f4.

8 CONCLUSIONS

In this paper we have formulated a bilevel optimisation problem for reducing the vehicle fleet of a rural health and social care service. In our practical case study, we have shown that it is possible to reduce an existing fleet of 84 by 6 cars (7.1%), potentially saving £19 662 annually. In achieving this, negligible extra distance travelled was added in the form of additional swaps of vehicles between bases.

We demonstrated that, for this problem, some blurring of the distinction between upper and lower levels is helpful. The upper level algorithm is primarily responsible for allocating vehicles to bases, but our results (Section 7.1) show that allowing the lower level algorithm to occasionally request a vehicle from another base

Table 4: Number of calls to fitcheck and sat check by each iteration of each algorithm. Figures are the minimum, 1st quartile, median, 3rd quartile, and maximum, across all iterations of all repeat runs.

	satcheck					fitcheck				
	min	q1	med	q3	max	min	q1	med	q3	max
u1l1m1	1	5	6	13.25	153	91	25 817	33 858	83 513	1 072 472
u1l1m4	1	6	11	14	176	78	30 649	71 669	92 124	1 231 012
u1l2m1	1	1	1	3	4457	38	2620	8904	9659	51 076 752
u1l2m4	1	1	1	3	4136	386	5670	8904	12 307	47 397 191
u2l1m1	1	5	10	18	286	16	28 096	56 281	125 777	2 030 720
u2l1m4	1	6	12	20	282	18	36 362	81 498	149 324	2 576 057
u2l2m1	1	1	1	2	419	140	397	6261	8904	12 480 305
u2l2m4	1	1	1	2	1356	145	2599	8904	9496	11 747 683
u2l2mX	1	1	1	2	2	145	5669	8904	8967	44 817

(i.e. the difference between algorithms u2l2l4 and u2l2lX) offers an overall improvement in performance for the full bilevel problem.

There is far more to be done with this problem. The modelling makes some assumptions: in particular, we have assumed all jobs to be fixed, whereas each could be treated as a routing problem (similar to travelling thief considering distance, timing and emissions), adding a third level to the overall problem. We also ignored the practicality of swaps in terms of personnel being available to move the vehicles (beyond checking that there is enough time for each swap to happen). In the long term the goal over moving towards an electric vehicle fleet will also require consideration of times and locations to recharge, adding further complexity but potentially greater gains.

REFERENCES

[1] Karen Aardal, Martine Labbé, Janny Leung, and Maurice Queyranne. 1996. On the two-level uncapacitated facility location problem. *INFORMS Journal on Computing* 8, 3 (1996), 289–301.
[2] Samir A Abass. 2005. Bilevel programming approach applied to the flow shop scheduling problem under fuzziness. *Computational Management Science* 2 (2005), 279–293.
[3] Yasmine Beck, Ivana Ljubić, and Martin Schmidt. 2023. A survey on bilevel optimization under uncertainty. *European Journal of Operational Research* (2023).
[4] Vladimir Beresnev and Andrey Melnikov. 2019. Approximation of the competitive facility location problem with MIPs. *Computers & Operations Research* 104 (2019), 139–148.
[5] Jacek Blazewicz, Klaus Ecker, Erwin Pesch, Günter Schmidt, and J Weglarz. 2019. *Handbook on scheduling*. Springer.
[6] José-Fernando Camacho-Vallejo, Carlos Corpus, and Juan G Villegas. 2023. Metaheuristics for bilevel optimization: A comprehensive review. *Computers & Operations Research* (2023), 106410.
[7] Martha-Selene Casas-Ramírez and José-Fernando Camacho-Vallejo. 2017. Solving the p-median bilevel problem with order through a hybrid heuristic. *Applied Soft Computing* 60 (2017), 73–86.
[8] Cesar Alvarez Cruz, Pedro Munari, and Reinaldo Morabito. 2020. A branch-and-price method for the vehicle allocation problem. *Computers & Industrial Engineering* 149 (2020), 106745. https://doi.org/10.1016/j.cie.2020.106745
[9] Ivan Davydov, Yury Kochetov, and Stephan Dempe. 2018. Local search approach for the competitive facility location problem in mobile networks. *International Journal of Artificial Intelligence* 16, 1 (2018), 130–132.
[10] Stephan Dempe. 2020. Bilevel optimization: theory, algorithms, applications and a bibliography. *Bilevel Optimization: Advances and Next Challenges* (2020), 581–672.
[11] Leila Hajibabai, Yun Bai, and Yanfeng Ouyang. 2014. Joint optimization of freight facility location and pavement infrastructure rehabilitation under network traffic equilibrium. *Transportation Research Part B: Methodological* 63 (2014), 38–52.
[12] HM Government. 2021. Transitioning to zero emission cars and vans: 2035 delivery plan. https://www.gov.uk/government/publications/transitioning-to-zero-emission-cars-and-vans-2035-delivery-plan Retrieved 24/1/2023.
[13] Sinan Küfeoğlu and Dennis Khah Kok Hong. 2020. Emissions performance of electric vehicles: A case study from the United Kingdom. *Applied Energy* 260 (2020), 114241.
[14] Zrinka Lukač, Kristina Šorić, and Višnja Vojvodić Rosenzweig. 2008. Production planning problem with sequence dependent setups as a bilevel programming problem. *European Journal of Operational Research* 187, 3 (2008), 1504–1512.
[15] Jesús-Adolfo Mejía-de Dios, Alejandro Rodríguez-Molina, and Efrén Mezura-Montes. 2023. Multiobjective Bilevel Optimization: A Survey of the State-of-the-Art. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2023).
[16] Michael L Pinedo. 2012. *Scheduling*. Springer.