Applying Route Optimisation to Rota Generation For Home-Help Services

Author – Naomi Davidson

Supervisor – Richard Booth

Moderator – Natasha Edwards

CM3202 One Semester Individual Project – 40 credits

School of Computer Science and Informatics, Cardiff University

ABSTRACT

Rota creation for home-help services is often a complicated, manual task, with very few solutions available that can factor in the routes that carers will travel to complete their daily work. What's more, carers on zero-hours contracts are often left dealing with unpaid travel time and job assignments which appear unfair.

This project investigates this problem from a combinatorial optimisation perspective, studying the social and mathematical background, and formulating a solution based on a vehicle routing problem. A multi-objective integer linear program is implemented in Python to optimise travel time, fairness, and optional similarity to existing rotas, and results are visualised and evaluated for effectiveness.

ACKNOWLEDGEMENTS

I am grateful to my supervisor, Dr Richard Booth for his guidance and support throughout this project, and for his engaging teaching in the Introduction to the Theory of Computation and Combinatorial Optimisation modules which inspired my choice of project.

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

1.1 THE PROBLEM

Carers who provide home-help to elderly clients perform a wide range of duties, including housework, washing and physical assistance, cooking meals, and running errands. Whatever the needs of the client, the travel required to get to their home is an essential part of every job.

Companies offering this service create rotas for their staff, with some giving more consideration than others to how far each carer will be expected to travel between jobs. Rota creation is often time-consuming, and it can be near impossible to manually produce rotas which avoid excess travel. Additionally, a good rota maintains client satisfaction with carer-client familiarity, by consistently sending clients the same carers. These repeated pairings also contribute to carer job satisfaction, as does fair and transparent distribution of work between carers.

We define our problem as follows: given a set of locations, let k carers be located individually at their k home locations. Let the remaining n locations be the home locations of n clients. Our problem is to find routes for the k carers such that each of the n clients is visited at home exactly once by exactly one carer, with each carer starting and ending their route at their own home. The total travel time by all carers must be minimised, subject to the routes being fair for all carers, and optionally subject to similarity of the routes to an existing set of routes from a previous rota.

1.2 COMBINATORIAL OPTIMISATION

Combinatorial (or discrete) optimisation is one of the most active fields in the interface of operations research, computer science, and applied mathematics (Du and Pardalos 1998). Its applications include corporate planning, database query design, computational biology, and, most interestingly to us, both scheduling and vehicle routing.

In a combinatorial optimisation problem, there exists a discrete (but normally large) set of feasible solutions, and an objective function which can be applied to any given solution, the minimising (or maximising) of which will provide an optimal solution.

1.3 INTEGER LINEAR PROGRAMMING

Integer linear programming is commonly used as an approach to solving combinatorial optimisation problems. In an integer linear program (ILP) being used to solve a combinatorial optimisation problem, integer values will be assigned in various combinations to a set of variables, with each combination able to represent a feasible solution. In the case of a routing problem, this might mean a variable x_{ij} for each directed edge of a graph where i and j are the locations joined by that edge. Each variable will be assigned a value of 1 if the edge is travelled and 0 if it is not. To fully represent a combinatorial optimisation problem, the assignment of value to ILP variables is subject to a set of relevant constraints, and an objective function provides a way to measure the value of each solution.

1.4 THE PROJECT

The aim of this project is to create and implement an algorithm which can take as an input the addresses of carers and clients, along with other relevant details. The implementation of this algorithm should produce an ordered list of job assignments for each carer which is optimised for overall travel time while still being fair, and optionally consistent with an existing rota, all with the aim of enhancing the ability of home-help services to create efficient rotas. Treating this project as a combinatorial optimisation problem, rather than using other artificial intelligence approaches such as machine learning, provides us with an appropriate level of control over how we represent needs and preferences in the algorithm, and will allow us to evaluate highly explainable results. We will be able to encode the sorts of logic used by humans in manual rota creation, but with the added ability of a computer to quickly explore thousands of routing options.

The well-established use of ILPs to solve route optimisation problems makes them an ideal candidate for tackling this project. In the Background section of this report, we explore the context of our problem, as well as existing literature on the use of ILPs to solve related routing problems and some definitions of fairness. In the Basic Solution section, we use what we have discovered to formulate an ILP which is specific to solving the problem of creating home-help rotas, following which we create a dataset and a Python implementation. An Advanced Solution section sees the ILP expanded to solve the problem with an improved definition of fairness and the addition of a client-carer familiarity feature in order to increase the usefulness of the solution in a business setting. We test and evaluate our ILPs in the Results and Evaluation section for route optimisation, fairness, and consistency with existing rotas. A Future Work section suggests ways that this limited proof of concept could be expanded to create a more comprehensive solution.

2 BACKGROUND

2.1 CONTEXT

In the UK, employers are required to pay home-help care workers (also known as domiciliary care workers) at least the National Minimum Wage (NMW) for time spent working, which includes travel time between appointments. This is in addition to any reimbursement for travel costs. Employers will either pay for travel time explicitly, or instead pay just for time spent caring with wages that are high enough to be the equivalent of

NMW once travel time is accounted for (Minimum wage for different types of work 2022). Despite this, Rubery et al. (2015) finds that,

> An index of fragmented time practices among 52 independent-sector domiciliary care providers reveals widespread tendencies to use zero-hours contracts and limit paid hours to face-to-face contact time, leaving travel time and other work-related activities unpaid.

Indeed, all domiciliary care workers interviewed in another study (Hebson et al. 2015) reported high job satisfaction overall, but several expressed that terms and conditions of employment were a source of dissatisfaction, specifically: pay, working time and staff shortages. The study explored the rewarding nature of care work and how a resulting willingness to tolerate low pay is taken for granted by employers.

After outlining the difficulties of packed rotas and unforeseen client needs in home-help care, Wibberley (2013) finds that,

> These time pressures are exacerbated by the lack of travel time *allocated on rotas, particularly in the private sector…* It is very difficult to reduce the amount of time travelling and, therefore, domiciliaries have to save time elsewhere.

In addition to the lack of time and pay allocated for travel between appointments, issues over the distribution of the jobs themselves are common. Ravalier et al. (2018) studied the stress of working as a domiciliary care worker on a zero-hours contract, as approximately 60% of domiciliary care workers are employed in this way. The study reports,

> Respondents felt that there was often a lack of fairness in the way in which hours were offered across those with zero-hours contracts, with a lack of clarity in the way that hours were offered adding to this feeling of uncertainty and unfairness.

There is sufficient motivation here to design a program that could reduce the incentive for employers to shift a burden of unpaid labour onto carers, and to include provision for a fair distribution of the work. An ideal solution would minimise carers' overall travel time through the optimal assignment of jobs, with sufficient provision in the rota for the amount of time required for travel. This increased efficiency would make it possible for carers to spend a higher percentage of a working day carrying out home-help care, increasing income for the business, without obliging carers to travel backwards and forwards to their own homes between appointments at their own expense.

2.2 COMMON APPROACHES

While home-help rotas have traditionally been created without the help of computerised systems, a national review in Wales reported that the use of scheduling software is becoming increasingly commonplace (Care and Social

Services Inspectorate Wales 2016). As well as creating simple schedules, some more sophisticated programs are able to:

- Factor in travel times between calls.
- Link to satellite navigation systems to give travel directions to care workers and provide the agency with real time location of where their care workers are. This helps providers estimate call times, contact service users and bring in extra care workers if needed.
- Calculate wages and allow for holidays.
- Identify replacement care workers who the person receiving care already knows.
- Predict and calculate care worker continuity.

Current popular examples of these programs include offerings from Access (PeoplePlanner 2022) and Care Planner Ltd. (CarePlanner 2022). They provide solutions to the problem of a lack of travel time allocated on rotas, but are not able to optimise routes to minimise this time.

The same national review detailed a common approach for companies who attempt to minimise travel time: the region is split up into smaller areas and each carer's workload is clustered in just one area. This is certainly better than no optimisation as carers are not having to drive long distances between jobs, but can still give rise to the complaint that they are having to travel backwards and forwards across an area in the course of a day.

2.3 Algorithmic Problems

There are many well-known routing problems for which solutions have been formulated as ILPs. Here we explore a series of the problems which most closely resemble our own, with increasing generalisation until we reach a formulation which we can adapt to suit our unique requirements.

2.3.1 Travelling Salesperson Problem

Almost synonymous with route optimisation is the Travelling Salesperson Problem (TSP). Although mention of this problem can be found as far back as 1832, it was formulated mathematically by both Kirkman and Hamilton in 1856 (Schrijver 2005). Generalising the problem of finding a Hamiltonian cycle in a graph, the TSP is: given n cities and the distances between each pair of cities, find a shortest route traversing each city exactly once and returning to the city of origin (see Figure 1).

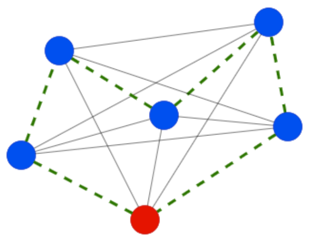


Figure 1: Example feasible solution for a TSP

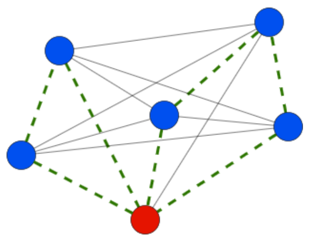


Figure 2: Route with subtours - these are not permitted

The TSP is one of the most well studied optimisation problems, with a variety of exact algorithms and heuristics recorded. It can also be formulated as an ILP, done notably by Dantzig, Fulkerson and Johnson (1954) utilising n^2 binary variables x_{ij} as recorded by Roberti and Toth (2012):

$$Minimise \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \qquad (1)$$

s.t.

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1,\dots,n \qquad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1,\dots,n \qquad (3)$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \le |S| - 1, \qquad S \subset V : S \neq \emptyset \qquad (4)$$

$$x_{ij} \in \{0,1\}, \qquad i,j = 1,\dots,n \qquad (5)$$

Where V={1,...,n} is the vertex set, x_{ij} (5) is equal to 1 if and only if arc (i, j) (i=1,...,n; j=1,...,n) is in the optimal tour. The objective (1) is to minimise the tour length which is the total cost c_{ij} of every arc x_{ij} traversed. Constraints (2) and (3) impose that the in-degree and out-degree of each vertex, respectively, is equal to one, while constraints (4) are Subtour Elimination Constraints (SECs) and impose that no partial circuit exists (see Figure 2).

The most basic case of our problem can be said to be a TSP as it involves finding the shortest route for one carer to visit all client locations, starting at and returning to the carer's own home.

2.3.2 Multiple Travelling Salesperson Problem

The TSP can be generalised to a Multiple Travelling Salesperson Problem (MTSP). The most basic version of this problem is as the TSP, but with m salespeople located at the depot city 1. As in Figure 3, each other city must appear exactly once in exactly one tour, and the overall distance cost to visit all cities must be minimised.

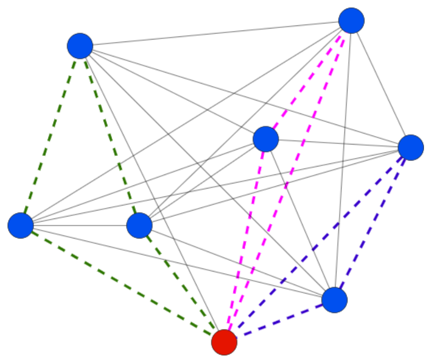


Figure 3: Example feasible solution for an MTSP with three salespeople

Kara and Bektas (2006) adjust the problem to include upper and lower bounds on the number of cities a salesperson must visit. Their ILP formulation for an MTSP with a single depot is as follows:

$$Minimise \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{6}$$

s.t.

$$\sum_{j=2}^{n} x_{1j} = m \tag{7}$$

$$\sum_{j=2}^{n} x_{j1} = m \tag{8}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 2, \ldots, n \tag{9}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 2, \ldots, n \tag{10}$$

$$u_i + (A-2)x_{1i} - x_{i1} \leq A - 1, \qquad i = 2, \ldots, n \tag{11}$$

$$u_i + x_{1i} + (2-B)x_{i1} \geq 2, \qquad i = 2, \ldots, n \tag{12}$$

$$x_{1i} + x_{i1} \leq 1, \qquad i = 2, \ldots, n \tag{13}$$

$$u_i - u_j + A x_{ij} + (A-2)x_{ji} \leq A - 1, \qquad 2 \leq i \neq j \leq n \tag{14}$$

$$u_i \in \{1, \ldots, A\} \qquad i = 1, \ldots, n \tag{15}$$

$$x_{ij} \in \{0,1\}, \qquad i, j = 1, \ldots, n \tag{16}$$

We define x_{ij} as a binary variable equal to 1 if arc (i,j) is in the optimal solution and 0 otherwise (16). For any salesperson, u_i (15) is the number of nodes visited on that salesperson's path from the origin up to node i (i.e., the visit number of the ith node). A is the maximum number of nodes a salesperson may visit; thus, $1 \leq u_i \leq A$ for all $i \geq 2$. In addition, let B be the minimum number of nodes a salesperson must visit, i.e., if $x_{i1} = 1$, then $B \leq u_i \leq A$ must be satisfied.

Once again, the objective (6) is to minimise the tour length which is the total cost c_{ij} of every arc x_{ij} traversed. In this formulation, constraints (7) and (8) ensure that exactly m salespeople leave from and return to the depot. Constraints (9) and (10) are the degree constraints, ensuring that each non-depot node is visited exactly once. The inequalities given in (11) and (12) serve as upper and lower bound constraints on the number of nodes visited by each salesperson, and initialize the value of u_i to 1 if and only if i is the first node on the tour for any salesperson. Constraints (11) and (12) we call bounding constraints, and were completely new introductions by Kara and Bektas for the MTSP. Inequality (13) forbids a vehicle from visiting only a single node. The inequalities given in (14) ensure that $u_j = u_i + 1$ if and only if $x_{ij} = 1$. Thus, they prohibit the formation of any subtour between non-depot nodes, so they are the SECs of the formulation.

We will need to generalise this MTSP further to reach a formulation relevant to our problem, but can take inspiration from Kara and Bektas' novel introduction of bounding constraints to pursue our aim of a fair distribution of jobs.

2.3.3 Multi-Depot Multiple Travelling Salesperson Problem

Multi-Depot Multiple Travelling Salesperson Problems (MDMTSPs), generalise MTSPs with the introduction of more than one depot and a number of salespeople initially located at each depot. A common variation is the nonfixed destination MDMTSP in which salespeople do not have to return to their original depots but the number of salespeople at each depot should remain the same at the end as it was in the beginning (see Figure 4).

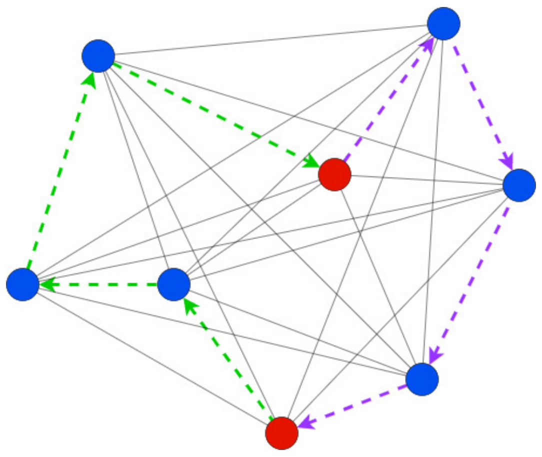


Figure 4: Example feasible solution for a nonfixed destination MDMTSP with two depots and two salespeople

Also due to Kara and Bektas (2006) is this ILP formulation of an MDMTSP:

$$Minimise \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{17}$$

s.t.

$$\sum_{j \in V'} x_{ij} = m_i, \qquad i \in D \tag{18}$$

$$\sum_{i \in V'} x_{ij} = m_j, \qquad j \in D \tag{19}$$

$$\sum_{i \in V} x_{ij} = 1, \qquad j \in V' \tag{20}$$

$$\sum_{j \in V} x_{ij} = 1, \qquad i \in V' \tag{21}$$

$$u_i + (A-2)\sum_{k \in D} x_{ki} - \sum_{k \in D} x_{ik} \leq A - 1, \qquad i \in V' \tag{22}$$

$$u_i + \sum_{k \in D} x_{ki} + (2-B)\sum_{k \in D} x_{ik} \geq 2, \qquad i \in V' \tag{23}$$

$$x_{ki} + x_{ik} \leq 1, \qquad k \in D, \qquad i \in V' \tag{24}$$

$$u_i - u_j + Ax_{ij} + (A-2)x_{ji} \leq A - 1, \qquad i \neq j; \;\; i,j \in V' \tag{25}$$

$$u_i \in \{1, \dots, A\} \qquad i = 1, \dots, n \tag{26}$$

$$x_{ij} \in \{0,1\}, \qquad i,j = 1, \dots, n \tag{27}$$

We partition the node set such that V = D ∪ V', where the first d nodes of V are depot set D, there are m_i salespeople located at depot i initially and the total number of salespeople is m. Also, let V' = {d + 1, d + 2, ..., n} be the set of customer nodes. x_{ij} variables, A, B and u_i variables are defined as before.

In this formulation, for each i ∈ D, m_i outward and m_i inward arcs are guaranteed by (18) and (19). Equations (20) and (21) are the degree constraints for the customer nodes. Constraints (22) and (23) impose bounds on the number of nodes a salesperson visits together with initializing the value of the u_i variables as 1 if i is the first node visited on the tour. Constraints (24) prohibit a salesperson from serving only a single customer. Finally, constraints (25) are SECs in that they break all subtours between customer nodes. Observe that there are O(n^2) binary variables and O(n^2) constraints in this formulation. It is easily seen that the MDMTSP model reduces to that of the MTSP when D contains only one depot node.

While we can begin to see parallels between this MDMTSP problem and our own, we will need to generalise again to find a problem studied in the literature on which to model a new ILP, which takes into account carers leaving from and returning to their own homes, along with timings for client visits and other related features.

2.3.4 Vehicle Routing Problem

While the formulations we have seen so far all have applications in logistics, Vehicle Routing Problems (VRPs) take this further still and are another group of widely studied problems. VRPs tend to be generalisations of MTSPs or MDMSTPs, with added constraints placed on each vehicle such as capacity or maximum distance. For this project, we are specifically interested in fixed destination VRPs, where each vehicle must end their route at the same depot at which they started (see Figure 5).

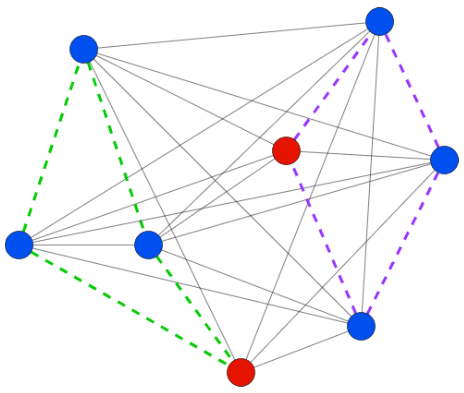


Figure 5: Example feasible solution for a fixed destination VRP with two depots and two vehicles

Ramos et al. (2019) reported the following VRP, with binary variables x_{ijk} that indicate whether vehicle k travels directly from node i to node j:

Indices

i,j	node index
k	vehicle index

Sets

$\bar{V}$	the set of nodes $\bar{V} = \{1, \dots, n + w\}$; $\bar{V} = V_c \cup V_d$
V_c	the subset of customer nodes $V_c = \{1, \dots, n\}$
V_d	the subset of depots nodes $V_d = \{n + 1, \dots, n + w\}$
K	the set of vehicles $K = \{1, \dots, r\}$; $K = K_1 \cup \dots \cup K_i$
K_i	the subset of vehicles belonging to depot i

Parameters

d_{ij}	distance between nodes i and j
r_{ij}	travelling time from node i to node j
Q_k	capacity of vehicle k
p_i	customer i demand
t_i	service duration at customer i
T	maximum time allowed for a route

$$Minimise \quad \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} x_{ijk} d_{ij} \tag{28}$$

s.t.

$$\sum_{i \in V} \sum_{k \in K} x_{ijk} = 1, \qquad \forall j \in V_c \tag{29}$$

$$\sum_{j\in V}\sum_{k\in K} x_{ijk} = 1, \qquad \forall i \in V_c \tag{30}$$

$$\sum_{i\in V} x_{ihk} - \sum_{j\in V} x_{hjk} = 0, \qquad \forall k \in K, \quad \forall h \in V \tag{31}$$

$$\sum_{i\in V_c}\sum_{j\in V} p_i x_{ijk} \leq Q_k, \qquad \forall k \in K \tag{32}$$

$$\sum_{i\in V}\sum_{j\in V} t_i x_{ijk} + \sum_{i\in V}\sum_{j\in V} r_{ij} x_{ijk} \leq T, \qquad \forall k \in K \tag{33}$$

$$\sum_{j\in V_c} x_{ijk} \leq 1, \qquad \forall k \in K_i, \quad \forall i \in V_d \tag{34}$$

$$\sum_{i\in V_c} x_{ijk} \leq 1, \qquad \forall k \in K_j, \quad \forall j \in V_d \tag{35}$$

$$\sum_{i\in V_c} x_{ijk} = 0, \qquad \forall j \in V_d, \quad \forall k \notin K_j \tag{36}$$

$$\sum_{j\in V_c} x_{ijk} = 0, \qquad \forall i \in V_d, \quad \forall k \notin K_i \tag{37}$$

$$u_i - u_j + n \times x_{ijk} \leq n - 1, \qquad 1 \leq i \neq j \leq n, \quad \forall k \in K \tag{38}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i \in V, \quad \forall j \in V, \quad \forall k \in K \tag{39}$$

$$u_i \in \{1, \dots, n\} \qquad \forall i \in V_c \tag{40}$$

The objective function (28) states that total distance travelled is to be minimised. Constraints (29) and (30) ensure that each customer is visited exactly once by a single vehicle. Route continuity is guaranteed by constraint (31), i.e., if a vehicle enters a site, it must exit that site. Constraint (32) ensures that the vehicle capacity is not exceeded. Similarly, constraint (33) guarantees that route duration (including service time duration and travelling time between nodes) does not exceed the maximum time allowed. Constraints (34) and (35) ensure that each vehicle will leave and return to its home depot at most once. Constraint (36) and (37) jointly ensure that a vehicle cannot leave and return to a depot other than its home depot. Constraint (38) is a redefined version of the Miller–Tucker–Zemlin SEC (Miller et al. 1960). Finally, constraints (39) and (40) set the variable domains.

While adaptations will be required to achieve aims such as fairness, this is the most appropriate problem we have seen from which to build a solution to the problem of applying route optimisation to rota generation for home-help services.

2.4 NP-Hardness

As mentioned previously, the most basic case of our problem can be said to be a TSP. The TSP is known to be NP-hard, so we can also say that all generalisations of the TSP are NP-hard, including MTSPs and VRPs. NP-hard problems are not solvable in polynomial time, unless P=NP[1], so it is likely that any algorithm to find the optimal solution to our problem will have a high time cost for large companies. It is not, however, a given that this cost will be prohibitively high, as there appear to be considerable differences in complexity within the class of NP-hard problems (Lenstra and Kan 1981).

2.5 Fairness

As we match carers to clients, one of our main aims is a fair assignment of jobs. In the fields of mathematics, economics, and computer science there are several established fairness measures. Hoang et al. (2016) summarise some of the most commonly used definitions of fairness to have emerged in the allocation problem literature in the last six decades or so, using the example of a cake cutting problem (usually a cake with different toppings, so a heterogeneous resource) with a finite set of players N and a cake CAKE. A division of the cake is a vector $x = (x_1,..., x_n)$ where $x_i \subset \text{CAKE}$ is the share of player i and $\bigcup_{i \in N} x_i = \text{CAKE}$. Each player i has a utility function u_i that associates a real number to any x_i. Player i prefers share x_i to x'_i, if and only if, $u_i(x_i) > u_i(x'_i)$. The utility function is additive; that is, for any disjoint subsets x_k and x_l, we have:

$$u_i(x_k \cup x_l) = u_i(x_k) + u_i(x_l).$$

In particular, this implies that the utility of an empty allocation is equal to zero, i.e., $u_i(\emptyset) = 0$, and by the normalisation of the whole-cake value, we have $u_i(\text{CAKE}) = 1, \forall i \in N$.

In our problem, the 'cake' to be divided is the set of all available caring jobs. The utility function for each carer will vary based on whether or not the employer pays for travel explicitly, and could relate to amount of paid work, amount of paid work proportional to unpaid work, or some combination of these.

Using our cake cutting problem notation, we have the following four definitions:

Exact fairness

A division is exact if all players' allocations are identical, i.e., exchanging shares will not affect any player's outcome. So, for any player $i \in N$, $u_i(x_j) = 1/n, \forall j \in N$. We know intuitively that this will not be possible to achieve in our problem given any utility measure involving travel, as is it extremely unlikely that a carer would have exactly the same travel time when swapping for otherwise identical jobs in alternative locations.

[1] See https://www.britannica.com/science/P-versus-NP-problem for more, accessed May 2022

Proportionality
A division is proportionally fair if every player prefers its allocation to an allocation from an exact division. If we suppose that the cake is fully allocated among n agents, then proportionality can be interpreted as an allocation where each agent prefers its share to the average of what they would get if allocations were given away uniformly randomly. So, a division is proportionally fair if any player gets at least 1/n, i.e., $u_i(x_i) \geq 1/n, \forall i \in N$. Given the difficulties we encounter with exact fairness and its relationship to proportionality, we do not pursue this fairness measure in the project. We do, however, discuss its use in the Future Work section.

Envy-freeness
A division is envy-free if every player prefers its allocation to any other player's allocation. So, for any player i, $u_i(x_i) \geq u_i(x_j), \forall j \in N$. In our problem, as in many others, this is not a realistic measure. We cover this in more detail later in this report.

Equitable fairness
A division is equitable if all players have the same utility for their respective shares, i.e., $u_i(x_i) = u_j(x_j), \forall i,j \in N$. Related to our issues with exact fairness, given any utility measure involving travel, it is extremely unlikely that a carer will have exactly the same utility as another carer for their respective jobs.

Although none of these definitions can be straightforwardly applied to our problem, we will be able to use relaxations of both exact fairness and equitable fairness to inspire the ILPs we formulate for route-optimised rota generation.

With an inability to guarantee a division of jobs which is exact or equitable, we might look to envy-freeness as a better approach, especially as route optimisation involves choosing for each party the route that is most appropriate for them. But Bouveret and Lang (2008) note that,

> A key concept in the literature on fair division is envy-freeness: an allocation is envy-free if and only if each agent likes her share at least as much as the share of any other agent. Ensuring envy-freeness is considered a desirable property; however, envy-freeness alone does not suffice as a criterion for finding satisfactory allocation

Indeed, for many problems there can be no envy-free allocation, and this is further complicated in our case by the high likelihood of our aims of travel minimisation being at odds with a pursuit of envy-freeness, as we will see later. This being said, we saw from Ravalier et al. (2018) in section 2.1 that,

> Respondents felt that there was often a lack of fairness in the way *in which hours were offered across those with zero-hours*

> contracts, with a lack of clarity in the way that hours were offered adding to this feeling of uncertainty and unfairness.

Envy-free allocations may not exist for all/any of our rota-creation problems, but the spirit of envy-freeness is relevant: clarity and impartiality in the system can reduce envy, and we can borrow intuitively from envy-freeness as we evaluated the strength of our proposed solutions, and test whether we are able to create rotas for which there would be little incentive for carers to swap jobs with one another in the pursuit of fairness. We will need to test this empirically, and the availability of visualisations will play a role here.

2.6 Existing Work

Here we identify three articles which present important problems and solutions related to this project.

An et al. (2012) present a two-phase heuristic algorithm which makes use of a mixed-integer program (MIP) to optimise the ordering of a nurse's pre-assigned jobs with the objective of minimising total travel time.

An algorithm based on particle swarm optimisation for home care worker scheduling in the UK due to Akjiratikarl et al. (2007) has very similar aims to this project but with some notable differences. They base their algorithm on a VRP with time windows, which is beyond the scope of this project, but which will be mentioned in the Future Work section.

And finally, Luo et al. (2021) formulate a route and speed optimization problem in home health care as an MIP and then propose an alternative ant colony optimisation based heuristic approach that can better handle large-scale instances. This problem deals with instances where clients must be visited by multiple carers simultaneously, and in this problem all carers must start the day at a central office and end the day at a medical laboratory.

In contrast to these three approaches, our project will add to route optimisation a focus on carer and client satisfaction, with attention given to fairness and carer-client familiarity. In Akjiratikarl et al.'s study a straight-line distance is assumed for travel between clients, whereas we will include real travel times.

3 Basic Solution

3.1 Approach

As we are primarily interested in the role of route optimisation in home-help rota generation, we first formulate an ILP which can model the assigning of carers to ordered lists of clients while minimising the total distance travelled by all carers. Starting by adapting the VRP encountered in section 2.3.4 allows for a gradual introduction of our unique features of fairness and similarity. We choose this VRP for its fixed-destination problem and its handling of travel time and visit duration, which will allow us to expand into an advanced solution once we have visualised its behaviour.

We use a Python program to solve the ILP for sets of addresses chosen at random from a pool of real UK addresses across adjacent towns. These addresses have themselves been collected randomly and are not provided with this report as they constitute data relating to real individuals[2]. Visualisations of the optimal solutions generated by the program serve to demonstrate the effectiveness of the ILP and support the later stages of the project (Figure 6).

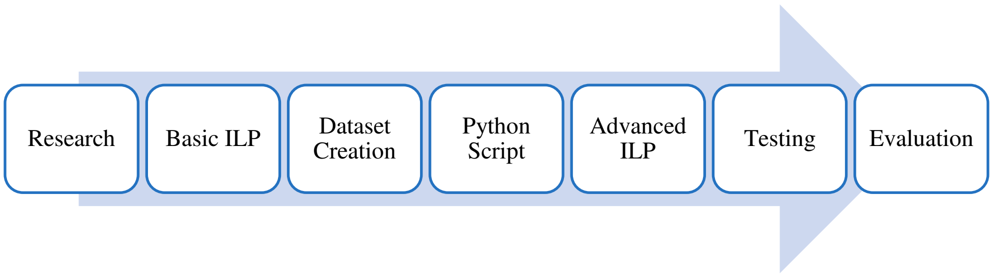


Figure 6: Project milestones

We assume that our program would be used to divide a chosen number of jobs between a chosen number of carers rather than, for example, to select the most appropriate carer from a pool of many to carry out a job. We can therefore assume that the number of clients will always be greater than or equal to the number of carers.

3.1.1 Fairness Definition

We assume that it is desirable for carers to be allocated an equal share of the available work. For this basic solution, we use the concept we saw in our background research of exact fairness, and divide the number of available jobs to be shared between carers so that all carer's allocations are identical. We treat all jobs as having an equal utility for all carers in the set of carers N, so that, using our notation from section 2.5, for an exact division for any carer $i \in N$, $u_i(x_j) = 1/n, \forall j \in N$.

Of course, this is a naïve approach to pursuing exact fairness, both because there will be cases where the jobs do not divide exactly between the carers, and because we do not account for probable large variations in the total time individual carers will be required to spend travelling and therefore the variation in carers' utility for any given assignment of jobs. However, this approach is worth exploring as it will provide a level of fairness while allowing flexibility for the algorithm to find a solution which truly minimises overall travel time. It is also of interest as it is likely a common method for dividing work between carers in situations where route optimisation is not considered, making it an useful basic case. We assume jobs of equal length (one hour) for this basic solution.

[2] These individuals are not known to us, and the school's ethics committee has granted a favourable opinion on the use of this data for the project

To ensure fairness according to this measure, we will introduce upper and lower bounds on the number of jobs a carer must be assigned. These bounds will be based on the result of dividing the total number of jobs by the total number of carers. If the resulting number is an integer (that is, if the job number is exactly divisible by the carer number) then this will be the value of both the upper and lower bounds, guaranteeing that the jobs are assigned with exact fairness. Otherwise, the result will be rounded up to the nearest integer for the upper bound and down for the lower bound, ensuring that the difference between the number of jobs assigned to any two carers is no more than one.

3.1.2 Integer Linear Programming Formulation

We begin by adapting the ILP we saw earlier from Ramos et al. (2019):

Indices

i,j	node index
k	carer index

Sets

$\bar{V}$	the set of nodes $\bar{V} = \{1, \dots, n + w\}$; $\bar{V} = V_c \cup V_d$
	where n is the number of clients to visit
	and w is the number of carer homes (depots)
V_c	the subset of client nodes $V_c = \{1, \dots, n\}$
V_d	the subset of depots nodes $V_d = \{n + 1, \dots, n + w\}$
K	the set of carers $K = \{1, \dots, l\}$; $K = K_1 \cup \dots \cup K_i$
	where l is the number of carers
K_i	the subset of carers belonging to carer home i

Parameters

r_{ij}	travelling time from node i to node j
t_i	visit duration at customer i
T	maximum time allowed for a route
A	maximum number of clients a carer may visit
B	minimum number of clients a carer may visit

$$Minimise \quad \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} x_{ijk} r_{ij} \tag{41}$$

s.t.

$$\sum_{i \in V} \sum_{k \in K} x_{ijk} = 1, \qquad \forall j \in V_c \tag{42}$$

$$\sum_{j \in V} \sum_{k \in K} x_{ijk} = 1, \qquad \forall i \in V_c \tag{43}$$

$$\sum_{i \in V} x_{ihk} - \sum_{j \in V} x_{hjk} = 0, \qquad \forall k \in K, \ \ \forall h \in V \tag{44}$$

$$\sum_{i \in V_c} \sum_{j \in V} t_i x_{ijk} + \sum_{i \in V} \sum_{j \in V} r_{ij} x_{ijk} \leq T, \qquad \forall k \in K \tag{45}$$

$$\sum_{j \in V_c} x_{ijk} = 1, \qquad \forall k \in K_i, \ \ \forall i \in V_d \tag{46}$$

$$\sum_{i \in V_c} x_{ijk} = 1, \qquad \forall k \in K_j, \ \ \forall j \in V_d \tag{47}$$

$$\sum_{i \in V} x_{ijk} = 0, \qquad \forall j \in V_d, \ \ \forall k \notin K_j \tag{48}$$

$$\sum_{j \in V} x_{ijk} = 0, \qquad \forall i \in V_d, \ \ \forall k \notin K_i \tag{49}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i \in V, \ \ \forall j \in V, \ \ \forall k \in K \tag{50}$$

The objective function (41) has been adapted to minimise the total travel time rather than distance. Constraints (42) and (43) ensure that each client is visited exactly once by a single carer. Route continuity is guaranteed by constraint (44), i.e., if a carer arrives at a client's home, they must also depart from that client's home. Constraint (45) guarantees that route duration (including client visit duration and travelling time between homes) does not exceed the maximum time allowed for a carer's working day. Note that we now include service times for client homes only, not carer homes. Constraints (46) and (47) have been adapted to ensure that each carer will leave and return to their home exactly once. Constraint (48) and (49) have been adapted to jointly ensure that a carer cannot travel to or from any carer's home that is not their own. Constraint (50) sets the variables domain.

Additional parameter A is the upper bound on the number of clients a carer may visit, while B is the lower bound. Inspired by the MTSPs we saw from Kara and Bektas (2006), these bounds are introduced to ensure a fair division of jobs.

We therefore include the following constraints:

$$u_i - u_j + A \times x_{ijk} \leq A - 1, \qquad 1 \leq i \neq j \leq n, \ \ \forall k \in K \tag{51}$$

$$u_i \in \{1, \dots, A\} \qquad \forall i \in V_c \tag{52}$$

$$\sum_{i \in V} \sum_{j \in V} x_{ijk} \geq B + 1, \qquad i \neq j, \forall k \in K \tag{53}$$

We replace the adapted Miller–Tucker–Zemlin SEC seen in the background section (38) with an adapted SEC (51) which also introduces the upper bound A by using u_i variables which denote, for each client i, where they are placed in an ordered list of visits. The constraint enforces that no carer may have an (A+1)th client visit. Constraint (52) sets the related variables domain. We also create a new constraint (53) to impose the lower bound, where the total number of journeys made by each carer must be greater than or equal to one more than the minimum number of visits required (since, for example, visiting five clients will involve six journeys including those starting and ending at the carer's home).

3.2 IMPLEMENTATION

3.2.1 Gurobi

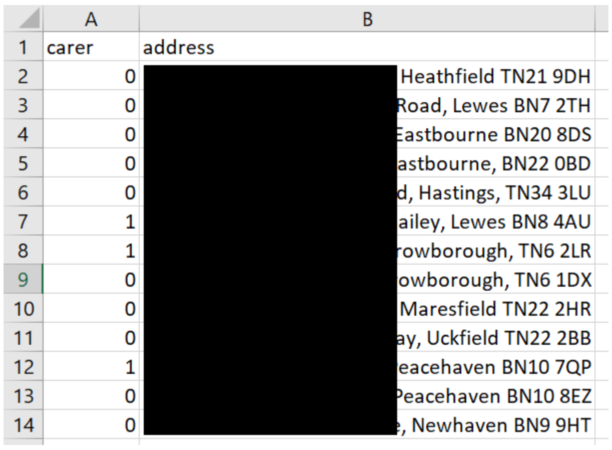
To solve our ILP we turn to Gurobi who claim to be the fastest and most powerful mathematical programming solver available for linear programs (Gurobi Optimizer. 2022). Their widely used, state-of-the-art optimiser offers interfaces for many popular programming languages, including Python, and provides free academic licenses. We make use of the gurobipy module, for which there is extensive documentation. When presented with a linear program as part of a Python script, the solver is able to select the most appropriate from a range of optimisation algorithms without input from the user, including simplex, parallel barrier, cutting planes, and branch and bound.

3.2.2 Routingpy

It is important that this project maintains a focus on developing a proof-of-concept solution which is relevant to the real world in which our problem is based. To that end, several options were explored to find an API for translating UK addresses into latitude and longitude coordinates which could be used to create a matrix of the car travel times from each address to every other address. Python 3 client routingpy enables easy and consistent access to third-party spatial webservices to request route directions, isochrones or time-distance matrices (routingpy. 2022). routingpy facilitated easy testing of services such as Here Maps, Google Maps, Graphhopper, and Mapbox Valhalla, acquiring API keys through each service's website but using nearly the same code for each test. Ultimately Mapbox Valhalla was chosen as the free version offered all the functionality required for this project. Calls to the API for address to latitude and longitude conversion, however, are limited to one per second, so when running repeated tests to evaluate our ILP we will use an adapted testing script which accepts a dataset with pre-converted addresses.

3.2.3 The Program

The command line interface takes as an argument a csv file which has two columns: carer and address. On each row, the address column contains the home address of either a client or carer, and the carer column indicates whether the address belongs to a client or carer with a 0 or a 1 respectively (see Figure 7). The program in its current form has a sample variable which can be set to False or True to indicate whether input should be treated as an example problem or as a dataset from which to randomly select a sample of addresses to represent a chosen number of clients and carers.



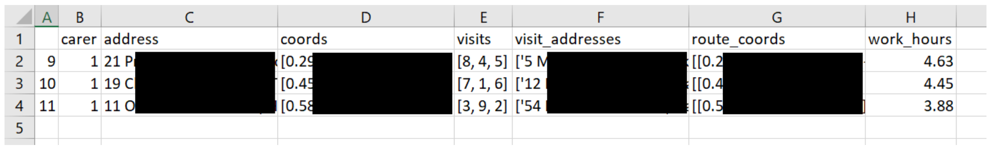
	A	B
1	carer	address
2	0	[redacted] Heathfield TN21 9DH
3	0	[redacted]Road, Lewes BN7 2TH
4	0	[redacted]Eastbourne BN20 8DS
5	0	[redacted]astbourne, BN22 0BD
6	0	[redacted]d, Hastings, TN34 3LU
7	1	[redacted]ailey, Lewes BN8 4AU
8	1	[redacted]rowborough, TN6 2LR
9	0	[redacted]owborough, TN6 1DX
10	0	[redacted] Maresfield TN22 2HR
11	0	[redacted]ay, Uckfield TN22 2BB
12	1	[redacted]eacehaven BN10 7QP
13	0	[redacted]Peacehaven BN10 8EZ
14	0	[redacted], Newhaven BN9 9HT

Figure 7: Example input csv with exact address details redacted

Functions are called to generate coordinates from the addresses and to create a travel time matrix, after which a make_model function produces the sets, parameters, objective function, and constraints described in section 3.1.2. Creating this function involved acquiring knowledge of gurobipy syntax, but functions including Gurobi's quicksum() and well written documentation made this a relatively straightforward process. More complicated was extracting the results once the model had been optimised. After the x_{ijk} variables with a value of 1 have been identified as the active arcs, for each carer k we collect into a list the arcs travelled by that carer and then a recursive get_route function is called with the carer's home location indicator as an initial argument. This function adds each visit, in order, to a route which starts as a list containing only the carer's home. The pseudocode for this function is as follows:

```
Function get_route( current_location )
    For arc ijk in arc_list
        If arc first value == current_location
            Add arc second value to route
                If arc second value == carer_home
                    End
                Else
                    get_route( arc second value )
```

A csv file (Figure 8) is produced as an output with a row for each carer, identifying them by their address and coordinates, and containing details of the ordered list of jobs they will undertake along with an expected total time for their route.



	A	B	C	D	E	F	G	H
1		carer	address	coords	visits	visit_addresses	route_coords	work_hours
2	9	1	21 P[redacted]	[0.29[redacted]	[8, 4, 5]	['5 M[redacted]	[[0.2[redacted]	4.63
3	10	1	19 C[redacted]	[0.45[redacted]	[7, 1, 6]	['12 [redacted]	[[0.4[redacted]	4.45
4	11	1	11 O[redacted]	[0.58[redacted]	[3, 9, 2]	['54 [redacted]	[[0.5[redacted]	3.88
5								

Figure 8: Example csv output with exact location details redacted

Plots using latitude and longitude coordinates and straight lines between them give a basic visualisation of the problem space and the solution given by the program, as shown in Figure 9.

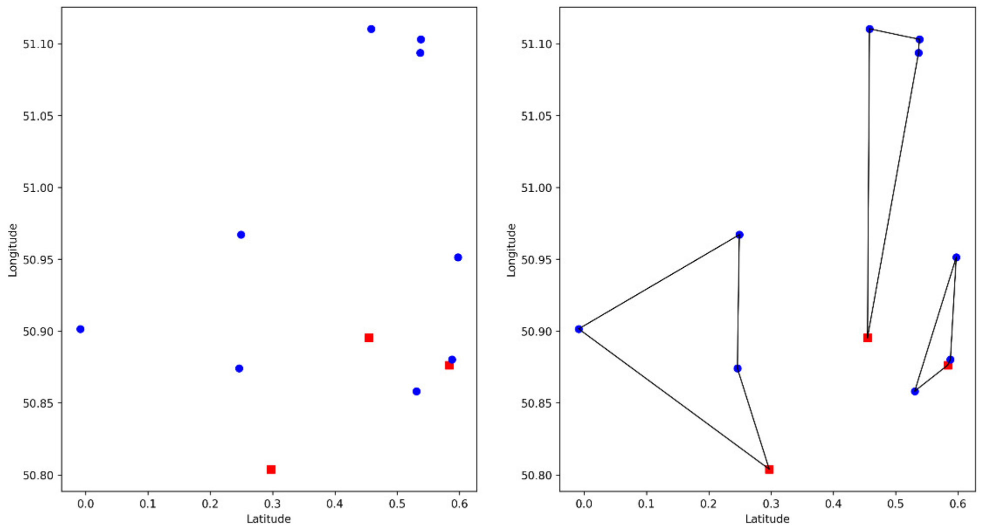


Figure 9: Plotted job assignments seen in Figure 8 of nine clients between three carers

Code adapted from a raw.githubusercontent.com notebook (Notebooks. 2022) provides a look at the exact routes proposed on a map of the area. In Figure 10 we see a map of the East Sussex area on which carer home locations are visualised as red squares while blue circles denote client locations. Exact efficient driving routes provided by Mapbox Valhalla's API are shown in black.

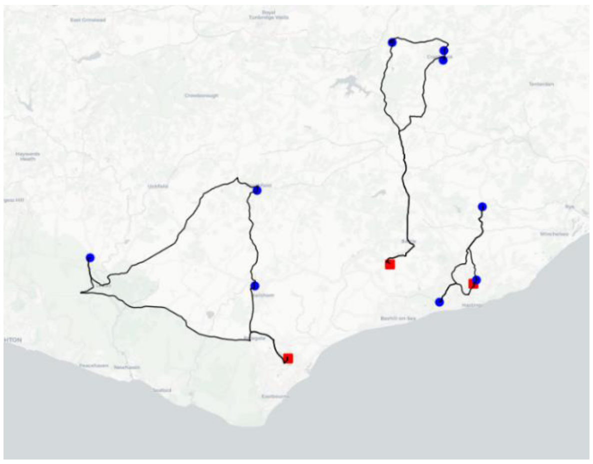


Figure 10: Carer routes for Figure 8 job assignments

4 Advanced Solution

We use insights from our basic solution and background research to introduce an improved fairness measure in section 4.1. Following this, a feature is created in section 4.2 which can optionally generate job assignments which favour similarity to a previous rota.

4.1 Improved Fairness

4.1.1 Fairness Definition

We know from exploring the context of our problem that time spent caring is only a part of the working day for home-help care workers, and that it is far from the only factor determining employee satisfaction. Our problem is motivated by a lack of attention given to the travel carried out by carers in the course of a working day, and so we must take this into account when considering fairness.

We see in Figure 11 an example of a job assignment that is allowed according to our fairness measure of a relaxed 'exact' division of jobs, but which results in a significant difference in the amount of time each carer spends working, and the distance travelled, both overall and proportional to the number of jobs assigned to them. The carer with three jobs will spend 60 minutes travelling, while one carer with four jobs will spend 75 minutes travelling and the other with four must travel for 145 minutes, nearly twice as many.

Intuitively we know that this is not a fair assignment, and that the utility each carer gains from their assigned jobs is not equal to their utility for the other carers' assignments. In cases where carers' pay is calculated based on the amount of time spent caring, there is a clear unfairness here between the two carers who will be paid the same amount when one's working day will last over an hour longer than the others. And in cases where caring and travel time are paid alike, unfairness lies in the over two-hour difference between the longest and shortest working days offered to carers.

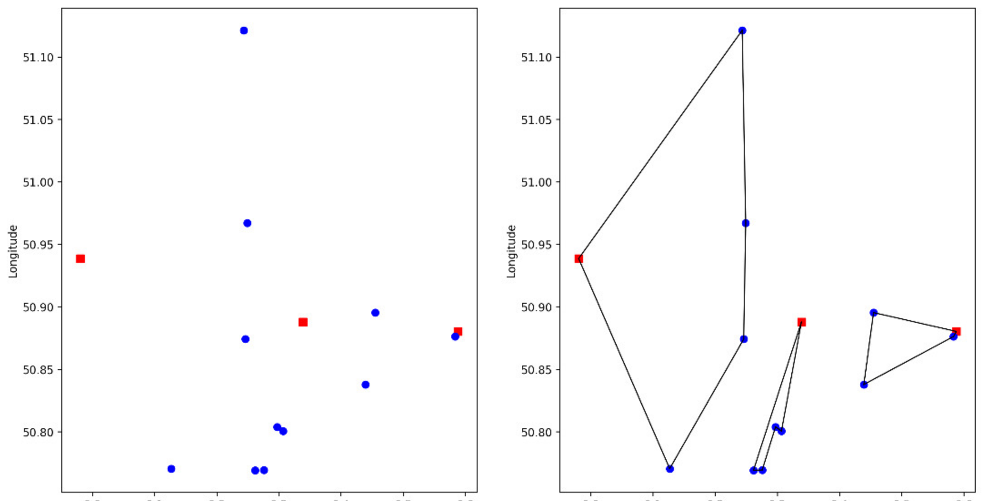


Figure 11: Assignment provided by the basic solution for 11 clients between three carers

For a more intelligent approach to ensuring fairness, we will adapt our basic solution to take into account total travel time for each carer. Whether or not carers are paid explicitly for time spent travelling greatly impacts what would constitute a fair assignment of jobs. As we are interested in a solution which will increase efficiency by lowering a company's overall travel time, we will assume that we are developing a solution for companies who will pay carers a set, time-based wage for both travel time and caring time.

The concept of equitable fairness is relevant here. We saw in section 2.5 that a division is equitable if all players have the same utility for their respective shares, i.e., $u_i(x_i) = u_j(x_j)$, $\forall i,j \in N$. In our problem, we may now define a carer's utility for an assignment as the amount of paid working time that it will take them to complete the jobs assigned to them. While requiring precisely the same utility for all carers is not sensible, requiring the same amount of work within a reasonable range, 60 minutes for example, could generally be expected, and this is what we will seek to achieve.

It is worth noting that here we consider an alternative definition of utility which relates to the working time associated with a particular allocation, independent from the amount of working time which would be associated with an allocation of the whole set of available jobs. In this definition, a carer's utility function is no longer additive. That is to say, $u_i(x_k \cup x_l) \neq u_i(x_k) + u_i(x_l)$ because, for example, adding a carer's utility (total working time) for an assignment of two jobs to their utility for an assignment of another three jobs will not be equal to that carer's utility for an assignment of all five jobs as the total amount of required travel time will be different in each case.

Although the scope of this project does not extend to implementing care appointments of varying lengths, an ILP formulation using this equitable fairness measure will allow for the introduction of varying appointment lengths for client visits, which would not have been easily integrated into our basic solution of equitably dividing the jobs between carers. No adjustment to the ILP would be required to accommodate such an input – the Python program would simply require a few extra lines of code to use the input to create a set of visit times relating to the set of clients.

4.1.2 Multi-Objective VRP

Now that our measure of fairness is no longer directly tied to the number of jobs assigned to each carer, we need a way to achieve our new objective of equitable fairness while maintaining our existing objective of route optimisation. It makes sense, therefore, to remove our existing fairness constraints (51), (52) and (53), and instead create a second objective function, with our ILP becoming a multi-objective VRP. Bowerman et al. (1995), with their paper on multi-objective optimisation for bus routing, provide inspiration for the following objective function:

$$Minimise \quad \sum_{k \in K} \left| \sum_{i \in V_c} \sum_{j \in V} t_i x_{ijk} + \sum_{i \in V} \sum_{j \in V} r_{ij} x_{ijk} - \frac{\sum_{i \in V_c} \sum_{j \in V} \sum_{k \in K} t_i x_{ijk} + \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} r_{ij} x_{ijk}}{l} \right| \quad (54)$$

This function (54) works by minimising the total of the absolute differences between each carer's total route time and the mean total route time for all carers.

Having removed previous fairness constraints, we must reintroduce an SEC (55) and related variables (56) into the ILP:

$$u_i - u_j + n \times x_{ijk} \leq n - 1, \qquad 1 \leq i \neq j \leq n, \quad \forall k \in K \tag{55}$$

$$u_i \in \{1, \dots, n\} \qquad \forall i \in V_c \tag{56}$$

4.1.3 Implementation

Gurobi supports two approaches to combining objective functions: blended and hierarchical (Working With Multiple Objectives. 2022).

The choice of method to combine our two functions is important because, with the removal of the upper and lower bounds implemented in the basic solution, the optimum result according to our fairness objective function will be vastly different to the optimum result according to our original objective function. We see this in Figure 12 and Figure 13, which show the two objective functions producing very different optimal assignments when applied independently. In this example, minimising travel time produces one route of 2.42 hours and one of 9.24 hours, while the two routes produced by minimising unfairness each take 6.69 hours to complete.

This difference is relevant when considering whether to use the blended or hierarchical approach. While the blended approach uses programmer-defined weights to consider multiple objective functions simultaneously, the hierarchical approach optimises according to each objective function in a programmer-determined order. Once optimal solutions have been found for the first objective, Gurobi's algorithm finds the solution from among these which is optimal according to the second objective. The programmer can choose an amount by which to allow the program to degrade the solution favoured by the first objective, with the program choosing from the previously explored solutions within the allowed value range to find that which optimises the second objective. However, the process of optimising the first objective function is not repeated, so only the solutions which were initially found in pursuit of an optimum are available to be considered against the second objective. The benefit of Gurobi's optimisation algorithms is that they efficiently explore the solution space without needing to enumerate every feasible solution. This is ideal for optimisation – exploring every feasible solution would result in extremely poor performance. This does mean, however, that a majority of solutions explored are close to the optimum solution, so in a case where we seek to combine two objective functions with very different optimal solutions, optimising each in order is ineffective as it is likely that solutions important for compromise will remain unexplored by the first objective and therefore unavailable to the second.

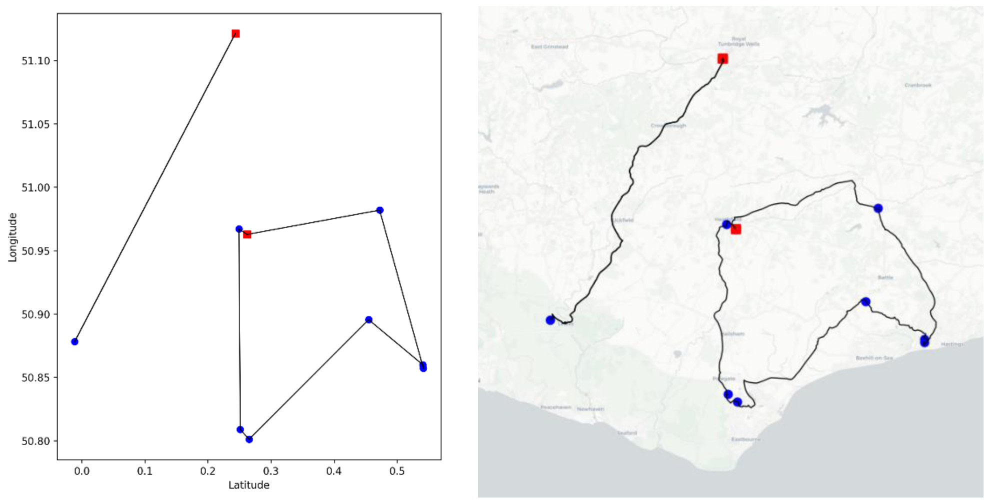


Figure 12: Assignment of eight clients to two carers with only the travel time objective

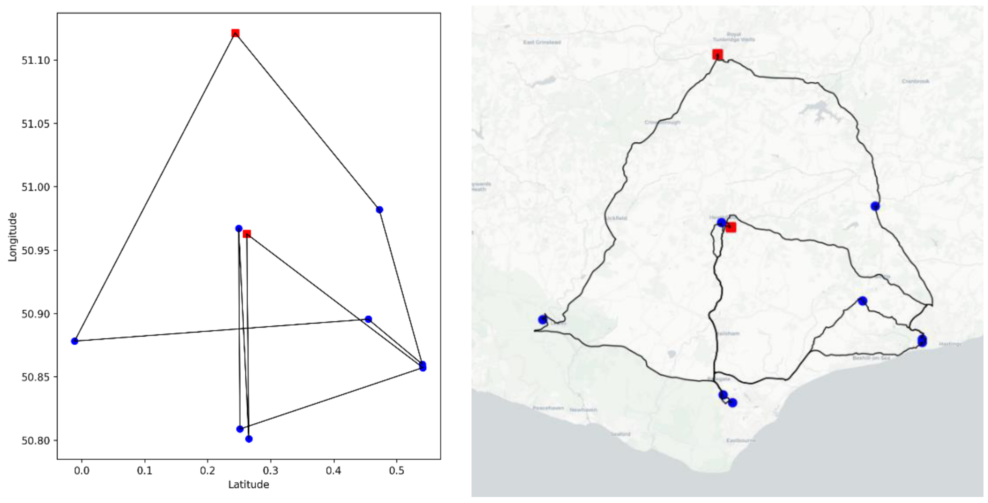


Figure 13: Assignment of eight clients to two carers with only the fairness objective

A blended approach, therefore, is the better of Gurobi's options for combining our objectives of minimising both travel time and unfairness. We use the default weight for the travel time objective and in section 5 we test three values as weights for the fairness objective, including weights less than and greater than the default.

Figure 14 shows the optimal solution given by a Python program implementing the new algorithm with a default weight of 1 for travel time and a weight of 0.8 for fairness. We see that the number of job assignments given to each carer is balanced with the time spent travelling, with the carer with three jobs rather than four travelling the furthest. The difference in total time spent working is now just 12 minutes compared to the 145 minutes given by the basic solution. In the basic solution the total working time for all carers was 940 minutes, whereas the advanced solution gives 960 minutes working time. So, for this example, the cost to the business for a dramatic increase in fairness would be an additional 20 working minutes.

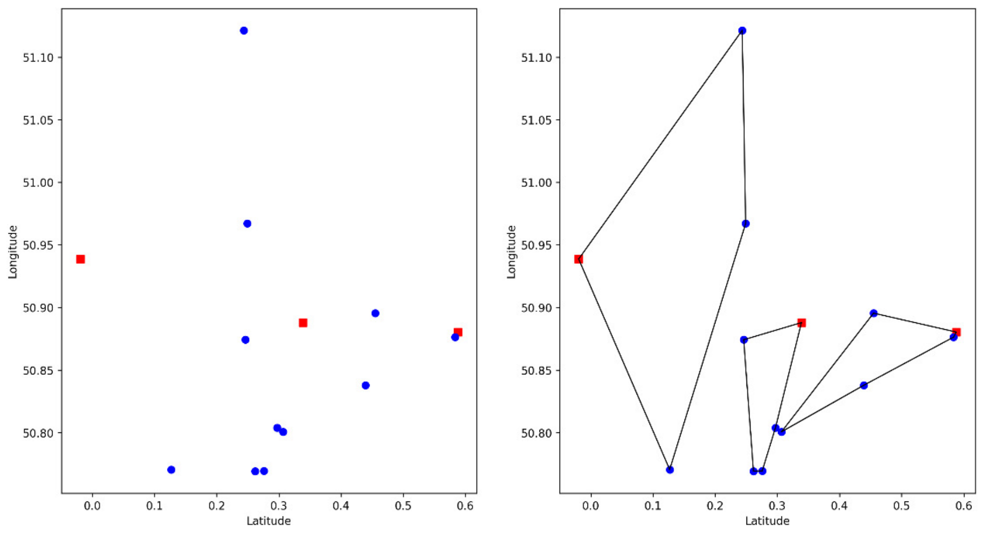


Figure 14: Problem seen in Figure 11 with assignments by new ILP

4.2 Similarity to Previous Rota

We identify two distinct areas of usefulness for the creation of a rota with similarities to a previous rota, using these to motivate a similarity feature which will address both.

4.2.1 Legacy Rota

Given a lack of route optimisation in the common approaches to rota creation seen in section 2.2, we assume that there would be companies already operating with a rota who could be interested in changing their approach to using a program implementing our solution. We call their existing rota a legacy rota, and anticipate that the ability to optimise for travel time and fairness while still maintaining some of the existing carer-client pairings present on a legacy rota, could contribute to ongoing carer and client satisfaction and therefore be a desirable feature when transitioning to using a route optimising program.

4.2.2 Live Rota

New rotas may be created on a daily or weekly basis, and may also need to be generated when unforeseen circumstances arise, such as carer absence or a carer needing to stay longer with a client to wait for an ambulance. In either case, there will be circumstances in which it is useful to be able to reoptimise job assignments based on a rota already generated by our algorithm, but now with an adjusted set of carers and/or clients.

4.2.3 Similarity Objective Function

As with fairness, we seek an approach which can balance our aims, rather than maintaining similarity at the expense of all route optimisation. We will therefore create a third objective function for similarity which can be combined with the ILP’s two existing functions.

Although not pursued here, carer-client pairings which are strictly necessary, as opposed to preferred, could be modelled as a constraint in the ILP to ensure that certain pairings are maintained regardless of effect on overall travel time.

We introduce the following new parameter and objective function:

s_{ki}	carer-client pairing from existing rota, with a value of 1 if carer k is paired with client i in existing rota, 0 otherwise

$$Minimise \quad -\sum_{i\in V_c}\sum_{j\in V}\sum_{k\in K} x_{ijk}s_{ki} \tag{57}$$

This function (57) maximises (by minimising the negative, in order to be consistent with the existing objectives) the total similarity value of a set of assignments, effectively maximising the number of instances of a client being visited by a carer who would have visited them in the existing rota.

4.2.4 Implementation

We add to the command line interface the ability to take a second csv file as an optional additional argument. This csv file must contain an address column and a visit_addresses column. On each row, the address column must contain the street address of a carer and the visit_addresses column must contain the street addresses of all clients visited by that carer in an existing rota. The program's output (Figure 8) fits this description and can be used as an input with no adjustments needed. If this second csv is not provided, the program will model the ILP without the similarity objective.

The program extracts relevant information from the previous rota, using a get_ccp_matrix function to identify only those existing carer-client pairings which relate to carers and clients who are both part of the dataset from which a new rota should be constructed. This information is used to construct a carer client paring matrix which functions as the s_{ki} parameter for the new objective function (57).

To meet our aim of rota similarity, we will select the set of assignments which contains the most carer-client pairings found in the existing rota, from the best assignment options found by our existing algorithm. To achieve this, both of Gurobi's hierarchical and blended approaches to combining objective functions are used. The priority parameter for both the travel time and fairness objective functions is set to a 2, with the similarity objective function given a lower priority value of 1. This means that Gurobi will first fully optimise using the blended higher priority functions, before turning its attention to the similarity objective function. As described in section 4.1.3, we allow Gurobi to degrade higher priority objectives by a limited amount, selecting the solution with the highest similarity value from among the best solutions explored in pursuit of the blended objective's optimum (Working With Multiple Objectives. 2022). We set the amount of permitted degradation using the higher priority functions' ObjNRelTol attribute. Where optimising produces an objective value a, further steps may degrade this value by no more than ObjNRelTol*|a|. For example, Figure 15 shows an optimal job assignment for 15 clients shared between five carers.

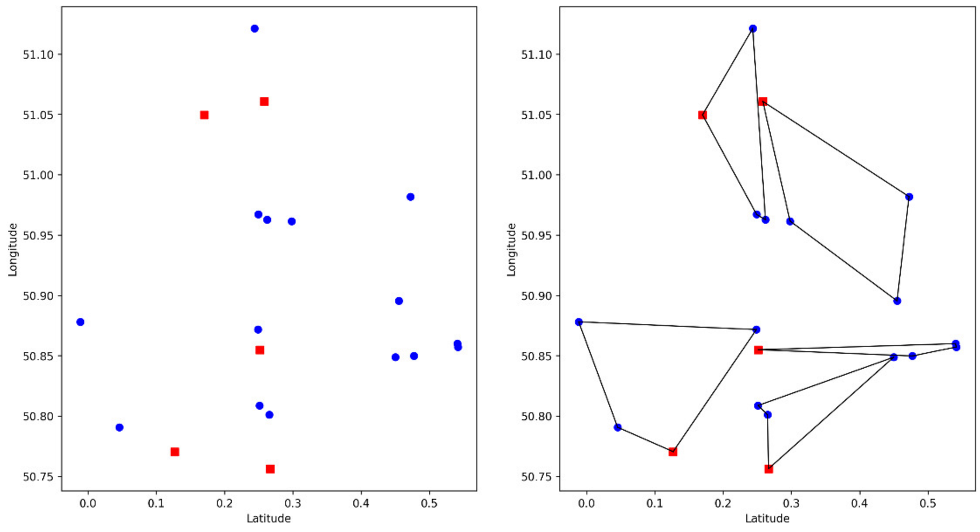


Figure 15: Plotted job assignments of 15 clients between five carers

Given the scenario of a (randomly selected) carer calling in sick to work and their jobs requiring redistribution, Figure 16 shows the set of new assignments chosen by the algorithm with no use of the similarity feature, maintaining four of the 12 original carer-client pairings held by the remaining carers. Figure 17 shows a set of assignments which has been created with the existing rota given as an input and the similarity objective function permitted to degrade travel time and fairness by up to 0.25 and 0.1 respectively, maintaining 11 of the 12 original pairings.

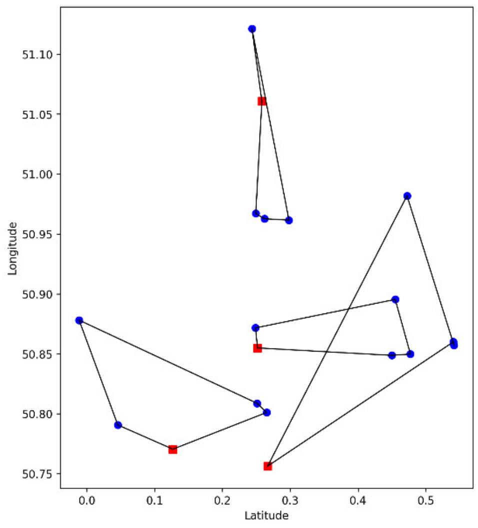


Figure 16: Plotted job assignments of 15 clients between four carers

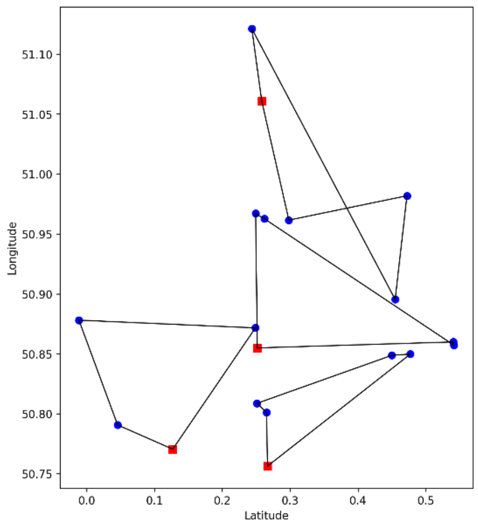


Figure 17: Plotted job assignments of 15 clients between four carers with similarity

4.3 Final Advanced Integer Linear Program

For completeness, we record the final ILP, including advanced fairness and similarity.

Indices

i,j	node index
k	carer index

Sets

$\overline{V}$	the set of nodes $\overline{V} = \{1, \dots, n + w\}$; $\overline{V} = V_c \cup V_d$
	where n is the number of clients to visit
	and w is the number of carer homes (depots)
V_c	the subset of client nodes $V_c = \{1, \dots, n\}$
V_d	the subset of depots nodes $V_d = \{n + 1, \dots, n + w\}$
K	the set of carers $K = \{1, \dots, l\}$; $K = K_1 \cup \dots \cup K_i$
	where l is the number of carers
K_i	the subset of carers belonging to carer home i

Parameters

r_{ij}	travelling time from node i to node j
s_{ki}	carer-client pairing from existing rota, with a value of 1 if carer k is paired with client i in existing rota, 0 otherwise
t_i	visit duration at customer i
T	maximum time allowed for a route

$$Minimise \quad \sum_{i\in V}\sum_{j\in V}\sum_{k\in K} x_{ijk} r_{ij} \tag{58}$$

$$Minimise \quad \sum_{k\in K}\left|\sum_{i\in V_c}\sum_{j\in V} t_i x_{ijk} + \sum_{i\in V}\sum_{j\in V} r_{ij} x_{ijk} - \frac{\sum_{i\in V_c}\sum_{j\in V}\sum_{k\in K} t_i x_{ijk} + \sum_{i\in V}\sum_{j\in V}\sum_{k\in K} r_{ij} x_{ijk}}{l}\right| \tag{59}$$

$$Minimise \quad -\sum_{i\in V_c}\sum_{j\in V}\sum_{k\in K} x_{ijk} s_{ki} \tag{60}$$

s.t.

$$\sum_{i\in V}\sum_{k\in K} x_{ijk} = 1, \qquad \forall j \in V_c \tag{61}$$

$$\sum_{j\in V}\sum_{k\in K} x_{ijk} = 1, \qquad \forall i \in V_c \tag{62}$$

$$\sum_{i\in V} x_{ihk} - \sum_{j\in V} x_{hjk} = 0, \qquad \forall k \in K, \quad \forall h \in V \tag{63}$$

$$\sum_{i \in V_c} \sum_{j \in V} t_i x_{ijk} + \sum_{i \in V} \sum_{j \in V} r_{ij} x_{ijk} \leq T, \qquad \forall k \in K \tag{64}$$

$$\sum_{j \in V_c} x_{ijk} = 1, \qquad \forall k \in K_i, \quad \forall i \in V_d \tag{65}$$

$$\sum_{i \in V_c} x_{ijk} = 1, \qquad \forall k \in K_j, \quad \forall j \in V_d \tag{66}$$

$$\sum_{i \in V} x_{ijk} = 0, \qquad \forall j \in V_d, \quad \forall k \notin K_j \tag{67}$$

$$\sum_{j \in V} x_{ijk} = 0, \qquad \forall i \in V_d, \quad \forall k \notin K_i \tag{68}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i \in V, \quad \forall j \in V, \quad \forall k \in K \tag{69}$$

$$u_i - u_j + n \times x_{ijk} \leq n - 1, \qquad 1 \leq i \neq j \leq n, \quad \forall k \in K \tag{70}$$

$$u_i \in \{1, \dots, n\} \qquad \forall i \in V_c \tag{71}$$

Objective functions (58), (59) and (60) set the objective of minimising total travel time, unfairness and dissimilarity respectively. Constraints (61) and (62) ensure that each client is visited exactly once by a single carer. Route continuity is guaranteed by constraint (63), i.e., if a carer arrives at a client's home, they must also depart from that client's home. Constraint (64) guarantees that route duration (including client visit duration and travelling time between homes) does not exceed the maximum time allowed for a carer's working day. Constraints (65) and (66) ensure that each carer will leave and return to their home exactly once. Constraints (67) and (68) jointly ensure that a carer cannot travel to or from any carer's home that is not their own. Constraint (70) is a modified Miller–Tucker–Zemlin SEC (Miller et al. 1960), and (69) and (71) set the variables domains.

5 RESULTS AND EVALUATION

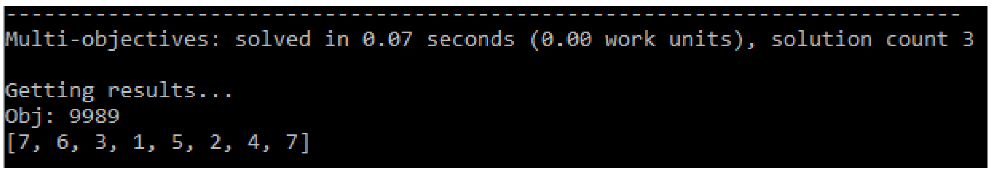
Although results have already been presented in the previous sections of this report in the form of visualisations which appear, intuitively, to show correct functioning of our algorithm, proving this correctness is challenging. It is not possible to verify our solutions are optimal by enumerating all feasible solutions and comparing them, nor it is possible to check our ILP against an existing solver in a majority of cases.

We can, however, check our ILP against an existing solver for sample sizes in which we have only one carer. As we saw in section 2.31, this is a special case of our problem, which can be said to be a TSP as it involves finding the shortest route for one carer to visit all client locations, starting at and returning to the carer's own home.

After using an online TSP solver to verify our algorithm for the simplest case of our problem, several hundred examples are tested to help us evaluate the performance of the algorithm. To evaluate the business value of our advanced fairness measure, we test it against our basic ILP.

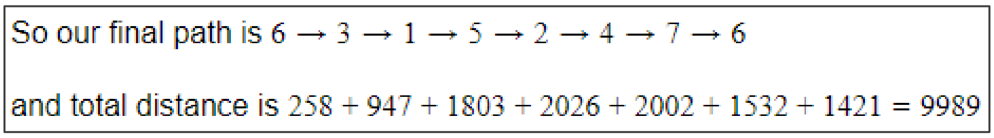
5.1 TSP Verification

AtoZmath.com's TSP solver (Shah 2022), using the branch and bound (penalty) method, was used to verify the result given by our program for one carer and six clients. Our dataset of 60 addresses was randomly sampled to produce a set of home addresses. A travel time matrix constructed by our program was given to both solvers (see Figures 25 and 26 in the Appendix section), which produced the results seen in Figure 18 and Figure 19.



```
----------------------------------------------------------------------------
Multi-objectives: solved in 0.07 seconds (0.00 work units), solution count 3

Getting results...
Obj: 9989
[7, 6, 3, 1, 5, 2, 4, 7]
```

Figure 18: Solution given by our program



So our final path is $6 \to 3 \to 1 \to 5 \to 2 \to 4 \to 7 \to 6$

and total distance is $258 + 947 + 1803 + 2026 + 2002 + 1532 + 1421 = 9989$

Figure 19: Solution given by the online TSP solver

The two solutions are the same route between the locations, just with a different starting point. They also give the same distance (in seconds) of 9989 for the route. From these results, from the knowledge that our ILP is based on existing formulations of problems which minimise distance, and on the strength of the visualisations presented in this report which appear to show short, sensible routes, we may trust that our ILP's travel time minimisation is functioning correctly.

5.2 Fairness

We have seen from Figure 12 and Figure 13 in section 4.1.3 that the objective of fairness is somewhat at odds with the objective of travel time minimisation. However, we have also seen (in section 2.1) that fairness is important for carer satisfaction, the neglect of which can lead to higher staff turnover, which can in turn negatively impact client satisfaction.

In light of the need to balance these objectives carefully, 3,193 tests were run on data randomly sampled from our dataset of 60 addresses, the results of which appear in Figure 27 in the appendix section. For each sample size, 100 random samples were each optimised by four algorithms: three versions of our advanced ILP with different weights placed on the fairness objective function, and the basic ILP to function as a baseline.

Due to limits on time and computational power, the number of clients and carers in each sample size were limited. A time limit of 3600 seconds (one hour) was set for each problem.

	mean time range (hours)	mean total time (hours)	mean runtime (sec)
0.4_fairness	0.91	11.75	18.69
0.8_fairness	0.64	11.94	33.75
1.5_fairness	0.35	12.29	82.07
basic_fairness	1.33	11.70	6.34

Figure 20: Comparison of average results

In Figure 20, we can observe that all variations of our advanced ILP have the advantage over the basic ILP of a smaller mean average time range, which is the difference between the maximum and minimum carer route times for test, with a lower number equating to a fairer division of jobs.

On the other hand, each advanced ILP performs worse than the basic ILP for total travel time, which we also seek to minimise. This is to be expected in a situation where we have somewhat opposing objectives. A worsening of the programs' runtimes also correlates with an increase in the weight of the fairness objective.

From the results seen is Figure 20 and Figure 27 results, we rule out the program with a fairness weight of 0.4 for having a mean time range which is too close to our preferred limit of one hour, especially with a standard deviation of up to 0.77 for some sample sizes. The results for the weight of 1.5, however, show too great a cost in exchange for the fairness achieved. Increasing the average overall time by 0.59 hours in what are relatively small sample sizes suggests that there are cases in which routes are made deliberately inefficient in order to balance travel time (Figure 13 would be an extreme example of this phenomenon). The program with a weight of 0.8 for fairness is able to more than halve the average time range while adding less than a quarter of an hour to the overall time.

Having identified 0.8 as an appropriate weight for our fairness objective function, we visualise the benefits of the advanced fairness objective function using the results from optimising routes for three carers and 13 clients. We choose this sample size because being one of the larger sample sizes in our test set makes it closer to an expected real world use case.

Figure 21 and Figure 22 together show the significant benefit of the ILP with advanced fairness measure which we have introduced in this project; a marginal increase to the overall travel time suggested in our basic ILP yields a significant increase in fairly distributed work for carers when our advanced ILP is used. And it is worth noting that we are comparing performance with a program with route optimisation and basic fairness, so the potential benefit given by the use of this advanced ILP when compared to a manual system which does not optimise for travel time or fairness is likely to be even greater.

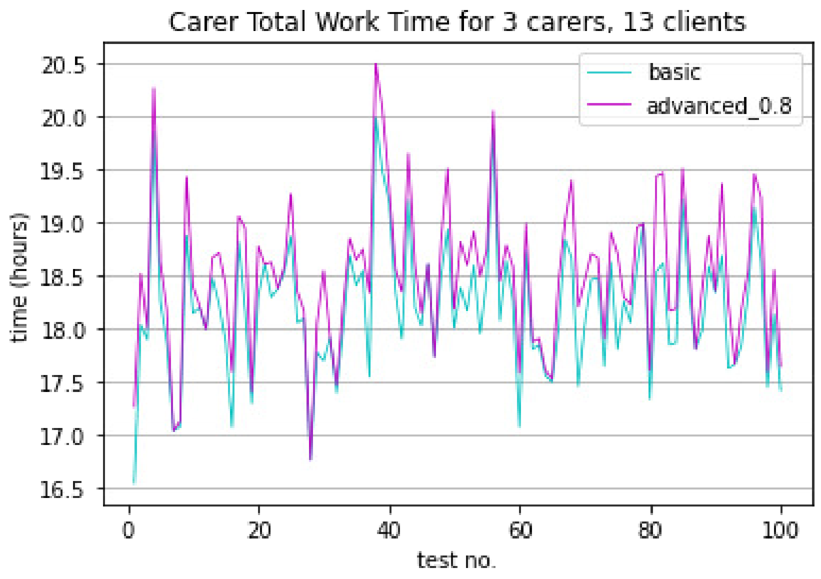


Figure 21: Graph showing difference in total time between basic and advanced solutions

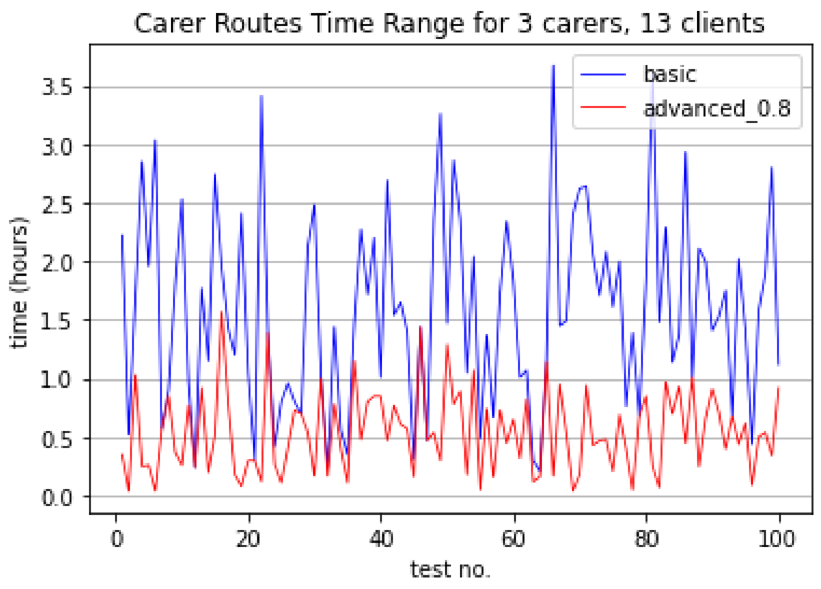


Figure 22: Graph showing difference in fairness between basic and advanced solutions

These figures help to show the value of our ILP: able to maximise carer satisfaction while also maximising the amount of time spent caring proportionately over the time spent travelling. But they also show evidence of achieving a level of relaxed equitable fairness, with all carers have the approximately (in this case within 1.5 hours) the same utility for their respective shares, i.e., $u_i(x_i) \approx u_j(x_j)$, $\forall i,j \in N$.

5.2.1 Evaluating Envy-freeness

As we have taken a combinatorial optimisation approach to solving this problem, we are interested to see what outcomes can be proved in terms of our initial aims. In section 2.5 we introduced the fairness measure of envy-freeness and judged that the spirit of envy-freeness is relevant to our problem of the fair and efficient division of home-help care jobs, even if the exact definition does not apply.

As we established, a division is envy-free if every carer prefers their allocation to any other carer's allocation. So, for any carer i, $u_i(x_i) \geq u_i(x_j)$, $\forall j \in N$. If both travel time minimisation and relaxed equitable fairness are achieved as intended, we may conjecture that envy-freeness, as defined here, will not be achieved.

> If we have that carers' home locations have been a factor in determining appropriate division of the available jobs based on minimising travel, and that they have therefore been assigned a route based on its proximity to their home,
>
> Then it follows that the sum of: the distance from the home of carer i to their first job, and the distance from their last job back to their home, and the distance from the home of carer j to their first job, and the distance from their last job back to their home, will be less than it would be if they were to travel to and from the first and last jobs of one another.
>
> Then it is the case that at least one of carer i or j would have a higher travel time, and therefore utility, given the share of the other.
>
> In order for it to be the case that neither would benefit from a swap, the existing assignment would need to involve unnecessary travel, and we have already established that for a correct assignment this will not be the case.
>
> So for our solution, and indeed for any reasonable solution which seeks to avoid unnecessary travel, envy-freeness with this definition cannot be achieved.

However, assuming that carer-client familiarity is desirable for carers as well as clients, it may be possible to model a carer's utility function for a share of the available jobs in a way that includes both the amount of paid work and the similarity of the assignment to previous assignments, which could yield more provably envy-free divisions.

The logic we have used here also applies to whether or not we can prove that a division is optimised for fairness based on whether carers would be interested in swapping individual jobs with one another to improve their utility. If the aim for carers is more hours, then of course swaps could decrease efficiency to increase time spent travelling for both carers, and in this sense likely all divisions are unstable. However, if we limit swaps (unless necessary for reasons other than fairness) to those which do not increase travel time for both carers, then there is no longer a mutual, time-based incentive to swap.

If we assume, on the other hand, that unnecessary travel is undesirable to all parties, then there will be cases where swapping jobs assigned by our algorithm could reduce the overall travel time, but these cases would also result in a time increase for one carer and a decrease for the other (or an decrease for both with a third colleague retaining a larger amount of work time), which would be unfair if our main basis for fairness is the equitable distribution of work time (but sensibly, with the desire to minimise travel).

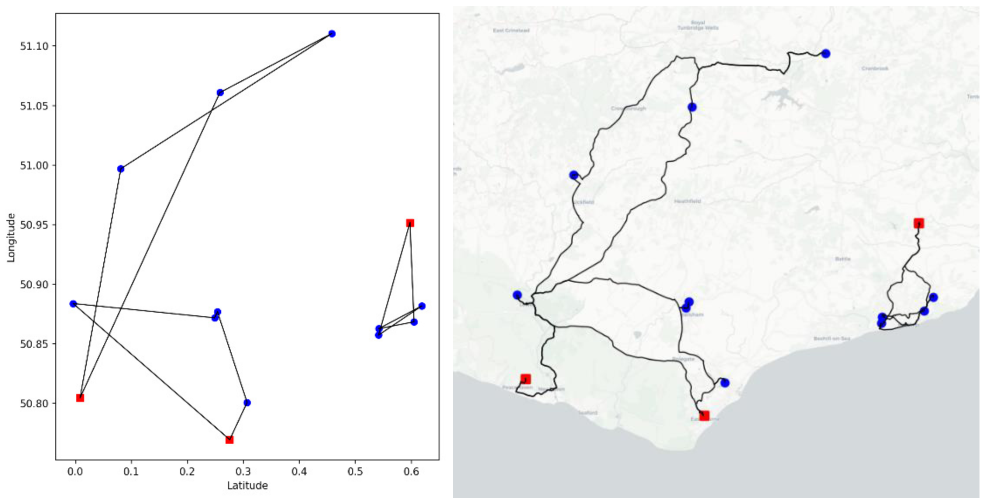


Figure 23: Assignment of 11 clients to three carers with inefficient travel used to compensate for what would otherwise be a larger difference in route durations

These cases, in which the swapping of individual jobs or entire routes would actually result in reduced travel time for both carers involved, are a casualty of our current ILP formulation. While some additional mileage may be deemed acceptable in order to maintain carer-client pairings, it is not so acceptable for the algorithm to reorder a route so that a carer's travel time increases to bring their overall work time in line with the work time given to the other carers. Finetuning the balance used by Gurobi to combine objective functions helps to prevent this phenomenon in a majority of cases, but it would still need to be addressed before commercial use of such an algorithm.

Although envy-freeness is not possible for our problem with the given definition, a program has been created which is able to assign jobs in a way that offers more fairness and clarity than other solutions, and thus would be able to reduce envy.

5.3 RUNTIME

A problem encountered in testing was that of large runtimes. Runtime averages have been seen in Figure 20 and in the appendix in Figure 27, and the chart in Figure 25 with a logarithmic scale further highlights the exponential complexity of our NP-hard problem.

Despite the otherwise promising results produced by the program, the issue of prohibitively large runtimes would need to be addressed before our ILP could see any commercial use.

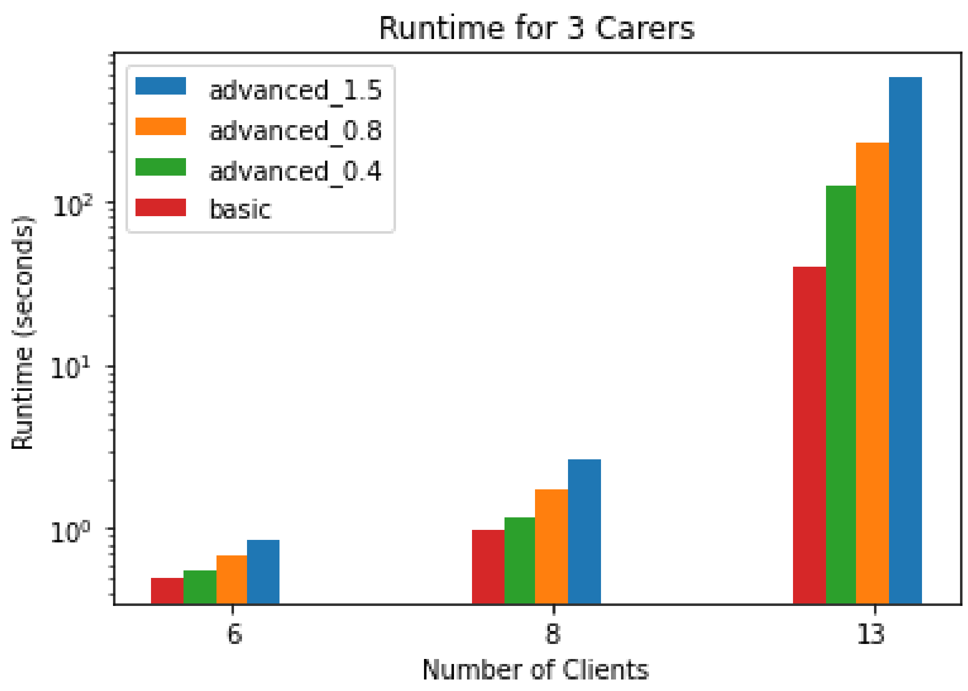


Figure 24: Chart with logarithmic scale showing average runtimes for different programs

6 Future Work

The complex social and mathematical contexts make applying route optimisation to rota generation for home-help services a very rich subject area. The results achieved in this project show potential for further development of the work, while the issues with runtime point to the exploration of alternative approaches for creating and implementing a similar algorithm.

6.1 Extending This Work

The process of researching and implementing this project led to several interesting potential avenues for expansion. Here we outline some of the most relevant.

6.1.1 Varied Appointment Lengths

As discussed in section 4.1.1, this project's final algorithm is able to carry out route, fairness, and similarity optimisation for a set of job assignments containing jobs of varying lengths. This was not included in the implementation, in order to maintain the simplicity of the route visualisations produced by the program, with appointments of the same length providing plots which allow the reader to understand the effectiveness of the route optimisation feature. However, the program would be adapted for use to take as an input an appointment length value for each client.

6.1.2 VRPTW

We saw in our Existing Work section (2.6) that work has already been done on the use of vehicle routing problems with time windows (VRPTWs) (Akjiratikarl et al. 2007). Time windows could be added to this project's ILP to include the ability to identify set periods of time during the day in which a client should receive, or not receive, a visit from a carer. In terms of approach, these time periods would be set as constraints if they are strictly necessary, or included in the objective function with associated costs if they are preferences.

The inclusion of time windows in the ILP could also be used to enhance the similarity feature, offering clients a consistent visit time as well as a consistent carer.

6.1.3 CMTSP

The coloured multiple travelling salesperson problem (CMTSP) is a problem in which some cities must be visited by a particular salesperson or subgroup of salespeople. In one example (Li et al. 2015), two types of city groups are defined, i.e., each group of exclusive cities of a single colour for a salesman to visit and a group of shared cities of multiple colours allowing all salesmen to visit.

This logic of a CMTSP could be applied to our problem in order to restrict the assigning of certain clients, based on their needs and preferences, to carers who fall into a certain gender, lifting ability, or medical skill category.

6.1.4 Traffic

Although our program goes further than any examples we have seen by using real travel times to optimise overall travel time, it does not account for augmented travel times due to poor traffic. Although some traffic problems are unpredictable, certain times on certain days of the week are known to have increased travel times. If implementing a VRPTW, the ILP could be set up to augment expected travel times if a majority of a journey is to take place within a time window known to be associated with poor traffic conditions.

6.1.5 Varied Days

Although the scope of this project has been limited to the assigning of jobs to be carried out in one working day, an ideal solution would extend this to a rota spanning several days, in which workload is balanced for fairness across the days rather than treating each day individually for fairness. This could allow for variation in the length of time worked each day.

Given the complexity of balancing workloads over just one day in our existing problem, it is to be expected that a new approach would be needed to realistically implement multi-day rota generation with fairness. Indeed, when using an ILP formulation to solve a multi-day VRP, which has in common with ours its three-index formulation and three objective functions, Ribeiro & Lourenço (2001) concluded that in order to improve running times they would need to use a heuristic, iterated local search approach in their further work.

6.1.6 Different working hours

Although we have assumed a scenario in which carers desire an equal amount of paid work, there are certainly situations in which carers working for the same home-help service would have a variety of lengths of working day. Although it would be straightforward to implement a carer-specific upper limit to the amount of time allowed for a day's route, the current fairness objective function does not account for such scenarios.

Instead of minimising the total absolute difference between each route time and the mean route time, the difference between each route time as a percentage of a carer's working hours and the mean of each route time as a percentage of respective working hours could instead be minimised. In other words, instead of seeking to give carers similar length working days, fairness could be reinterpreted as assigning working hours which are equally as close (proportionately or otherwise) to each worker's desired load.

6.1.7 Proving Proportionality

As we saw in section 2.5, a division is proportionally fair if any player gets at least 1/n, i.e., $u_i(x_i) \geq 1/n$, $\forall i \in N$. That is, if a carer's utility for their assigned jobs is at least an nth of the utility they would have if assigned all available jobs.

It was beyond the scope of this project to create an algorithm to find the exact utility function for a carer that would associate a real number to any share of any division of job assignments between N carers. Doing this in future would create the means to prove whether or not our solution creates assignments for which every carer receives a share for which their utility is at least an nth of their utility in the case where they are assigned all jobs.

6.1.8 Open Path MSTP

Given our motivation of the benefits of reduced travel for all involved, and our motivation of fairness, we have modelled our problem as a closed path, fixed destination VRP in which carers start and end the day at their own homes. But it is not actually the case that they must be paid for the whole route. Of home-help care work in the UK, Pennycook (2013) states,

> The law on travel time is relatively clear: unless genuinely self-employed, a worker travelling for the purposes of duties carried out in the course of his or her work will be required to be paid at least the minimum wage, excluding the first and last journeys of the day and travel to and from breaks.

It would therefore be legitimate to reformulate our solution as an open path MTSP, in which we find the n shortest routes between clients to be assigned to n carers, with or without consideration given to distances between each carer's home location and the start or end of their route.

This amount of time spent travelling by a carer in the unpaid first and last journeys of the day would impact on a sense of fairness, although it could be argued that this is not the responsibility of the employer as long as the travel is still within the agreed geographical area covered by the business.

6.1.9 Front-End Application

For use in a commercial setting, a user interface would need to be developed for the program, including an easy method for importing both carer-client data (including addresses, needs and preferences) and existing rotas. Alternatively, an adapted version of our algorithm could be incorporated into an existing offering, to add the novel functionalities of route optimisation, fairness and similarity to currently available software already being used to generate rotas for home-help services.

6.2 Alternative Approaches

In the literature encountered in this report there are several examples of VRPs and MTSPs which have been formulated as ILPs or MIPs before being adapted. When their MIP approach was too slow for large companies, Luo et al. (2021) formulated an ant colony optimisation (ACO) algorithm. And Akjiratikarl et al. (2007), discussing the NP-hardness of their problem, state,

> The number of possible solutions for VRPTW grows exponentially with the problem size. Using an exact algorithm will lead to an excessive computational requirement. As an alternative, heuristic techniques that are fast and yield good quality solutions should be applied. PSO [particle swarm optimisation] is such a technique that has many desirable characteristics as stated above and it performs well for scheduling when compared to various other heuristics. This indicates that the PSO-based algorithm is potentially suitable to efficiently solve the home care worker scheduling problem.

A significant issue encountered in testing our algorithm has been the exponential growth in the complexity of the problem as the sample size increases. We would therefore look to adapt our algorithm using combinatorial optimisation approaches other than ILPs, choosing from among existing meta-heuristics such as those mentioned above, which have been used to approximate optimal solutions to similar problems.

7 Conclusions

An in-depth study of the problem has been undertaken, and a novel solution produced which achieves the aims set out at the start of this project. The central goal of rota generation with route optimisation has proven effective, and we have added to our algorithm successful objective functions to make this route optimisation subject to fairness and optional carer-client familiarity.

The three objective functions in our final ILP join together to offer considerable value to home-help services. To be a viable solution, the algorithm would need to be adapted into an alternative approach which could solve runtime issues, and occasional inefficiencies arising from balancing the objectives of travel time and fairness optimisation would need to be fixed.

While real life circumstances may require refinements to the program, it has already been shown that the algorithm could improve both efficiency and fairness in the rotas of home-help services, and that it is possible to package this solution as one which can take a company's existing rota as an input, in order to ease the transition to a more efficient rota.

Although business, employee and client needs can often appear to be at odds with one another, we have found a way to satisfactorily balance the needs of all three in an algorithm which has the potential to create a positive outcome for all involved.

8 REFLECTION ON LEARNING

This project was my first introduction to more in-depth academic research and I enjoyed making extensive use of Google Scholar, knowing that my ability to learn all that I needed for this project had been built through the course of my degree.

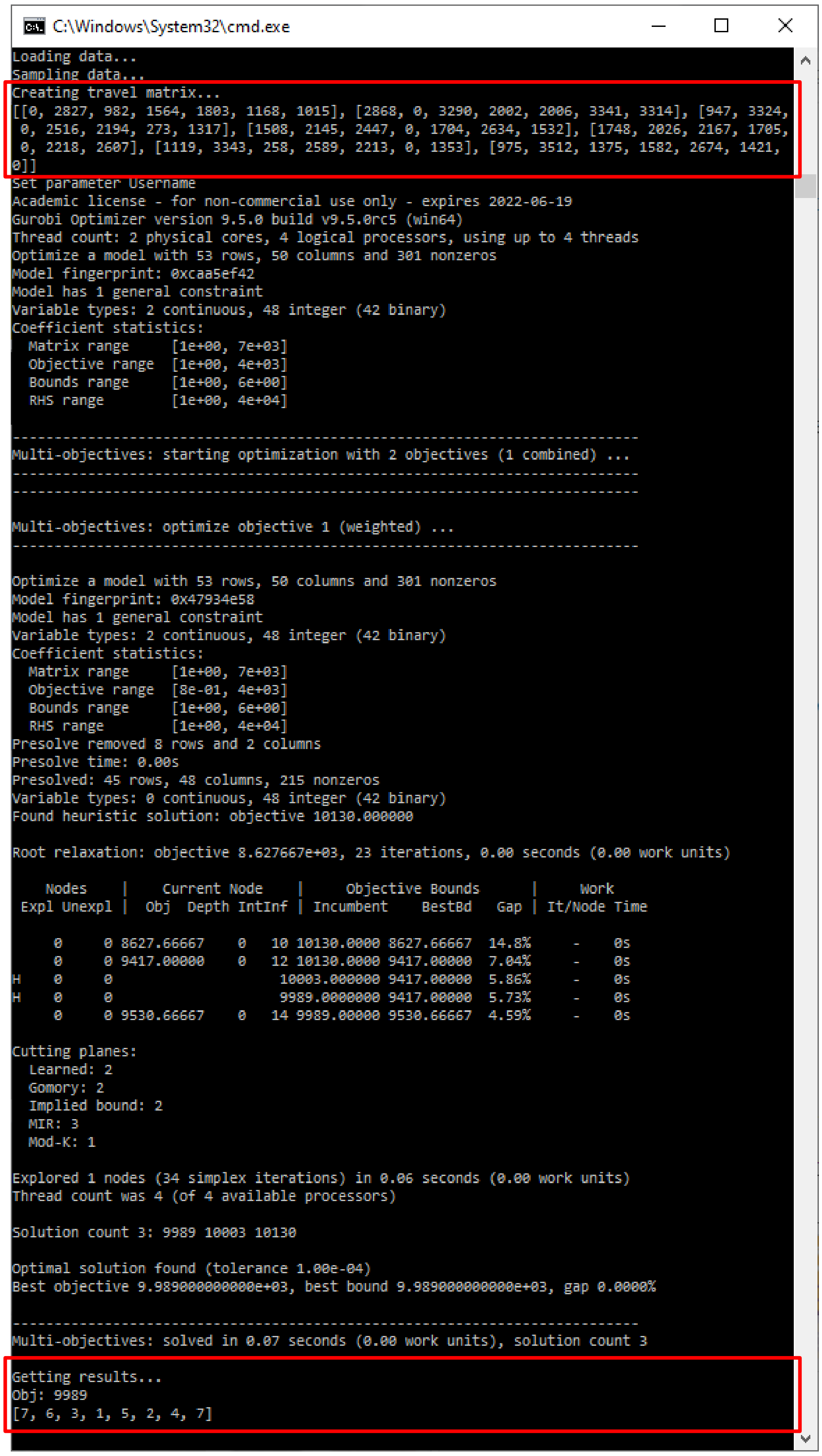
Although I made use of reference material throughout the project, going through my full background research thoroughly after having developed my solutions inspired some last-minute improvements to the project. This reminded me of the learning process that unfolds in the implementation stage, which sheds light on the literature in a way that simply reading it does not.

I found my initial plan to be an extremely useful document, and I benefitted greatly from the weekly supervision meetings in which I was able to present my work and seek guidance. Making use of both the plan and these meetings helped me to stay on track with my project and to keep my aims at the forefront of my mind. When there were unpredictable changes such as when I underestimated time for literature review and overestimated time for the initial algorithm formulation, I was especially grateful for the way that laying out the project stages helped with time management and motivation. When I lost the majority of a seven week period to illness (which also impacted time for testing and access to a more powerful computer), the progress I had already made in creating my solution thanks to good project planning meant that I was able to adapt my plan and still complete the project with the help of a two week extension. Although I had informally tested the similarity feature and knew it to be effective, I was disappointed not to be able to show off more of my work due to the limited resources I had available for testing already being used to create a full set of results to test the more nuanced fairness objective.

9 TABLE OF ABBREVIATIONS

API	Application Programming Interface
CMTSP	Coloured Multiple Travelling Salesperson Problem
ILP	Integer Linear Program
MDMTSP	Multi-Depot Multiple Travelling Salesperson Problem
MIP	Mixed-Integer Program
MTSP	Multiple Travelling Salesperson Problem
NMW	National Minimum Wage
SEC	Subtour Elimination Constraint
TSP	Travelling Salesperson Problem
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows

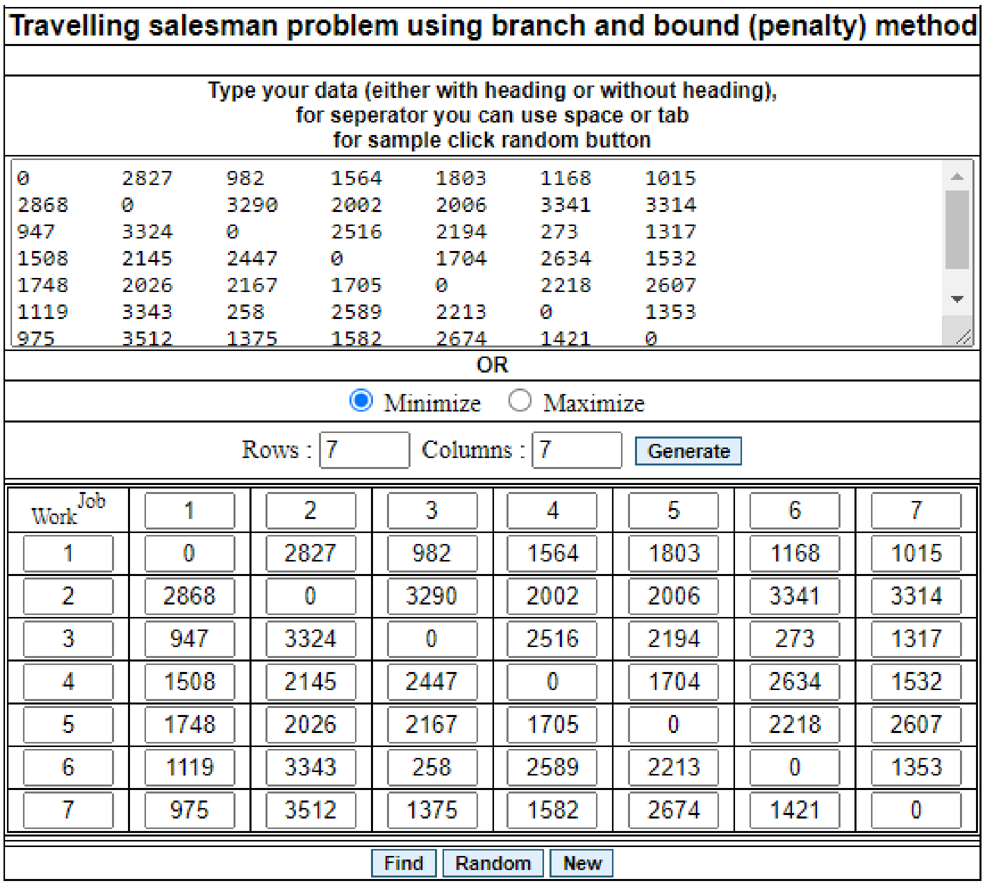
10 APPENDIX



C:\Windows\System32\cmd.exe

```
Loading data...
Sampling data...
Creating travel matrix...
[[0, 2827, 982, 1564, 1803, 1168, 1015], [2868, 0, 3290, 2002, 2006, 3341, 3314], [947, 3324,
 0, 2516, 2194, 273, 1317], [1508, 2145, 2447, 0, 1704, 2634, 1532], [1748, 2026, 2167, 1705,
 0, 2218, 2607], [1119, 3343, 258, 2589, 2213, 0, 1353], [975, 3512, 1375, 1582, 2674, 1421,
0]]
Set parameter Username
Academic license - for non-commercial use only - expires 2022-06-19
Gurobi Optimizer version 9.5.0 build v9.5.0rc5 (win64)
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads
Optimize a model with 53 rows, 50 columns and 301 nonzeros
Model fingerprint: 0xcaa5ef42
Model has 1 general constraint
Variable types: 2 continuous, 48 integer (42 binary)
Coefficient statistics:
  Matrix range     [1e+00, 7e+03]
  Objective range  [1e+00, 4e+03]
  Bounds range     [1e+00, 6e+00]
  RHS range        [1e+00, 4e+04]

---------------------------------------------------------------------------
Multi-objectives: starting optimization with 2 objectives (1 combined) ...
---------------------------------------------------------------------------
---------------------------------------------------------------------------

Multi-objectives: optimize objective 1 (weighted) ...
---------------------------------------------------------------------------

Optimize a model with 53 rows, 50 columns and 301 nonzeros
Model fingerprint: 0x47934e58
Model has 1 general constraint
Variable types: 2 continuous, 48 integer (42 binary)
Coefficient statistics:
  Matrix range     [1e+00, 7e+03]
  Objective range  [8e-01, 4e+03]
  Bounds range     [1e+00, 6e+00]
  RHS range        [1e+00, 4e+04]
Presolve removed 8 rows and 2 columns
Presolve time: 0.00s
Presolved: 45 rows, 48 columns, 215 nonzeros
Variable types: 0 continuous, 48 integer (42 binary)
Found heuristic solution: objective 10130.000000

Root relaxation: objective 8.627667e+03, 23 iterations, 0.00 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 8627.66667    0   10 10130.0000 8627.66667  14.8%     -    0s
     0     0 9417.00000    0   12 10130.0000 9417.00000  7.04%     -    0s
H    0     0                    10003.000000 9417.00000  5.86%     -    0s
H    0     0                    9989.0000000 9417.00000  5.73%     -    0s
     0     0 9530.66667    0   14 9989.00000 9530.66667  4.59%     -    0s

Cutting planes:
  Learned: 2
  Gomory: 2
  Implied bound: 2
  MIR: 3
  Mod-K: 1

Explored 1 nodes (34 simplex iterations) in 0.06 seconds (0.00 work units)
Thread count was 4 (of 4 available processors)

Solution count 3: 9989 10003 10130

Optimal solution found (tolerance 1.00e-04)
Best objective 9.989000000000e+03, best bound 9.989000000000e+03, gap 0.0000%

---------------------------------------------------------------------------
Multi-objectives: solved in 0.07 seconds (0.00 work units), solution count 3

Getting results...
Obj: 9989
[7, 6, 3, 1, 5, 2, 4, 7]
```

Figure 25: Program output for one carer visiting six clients


Travelling salesman problem using branch and bound (penalty) method

Type your data (either with heading or without heading),
for seperator you can use space or tab
for sample click random button

```
0       2827    982     1564    1803    1168    1015
2868    0       3290    2002    2006    3341    3314
947     3324    0       2516    2194    273     1317
1508    2145    2447    0       1704    2634    1532
1748    2026    2167    1705    0       2218    2607
1119    3343    258     2589    2213    0       1353
975     3512    1375    1582    2674    1421    0
```

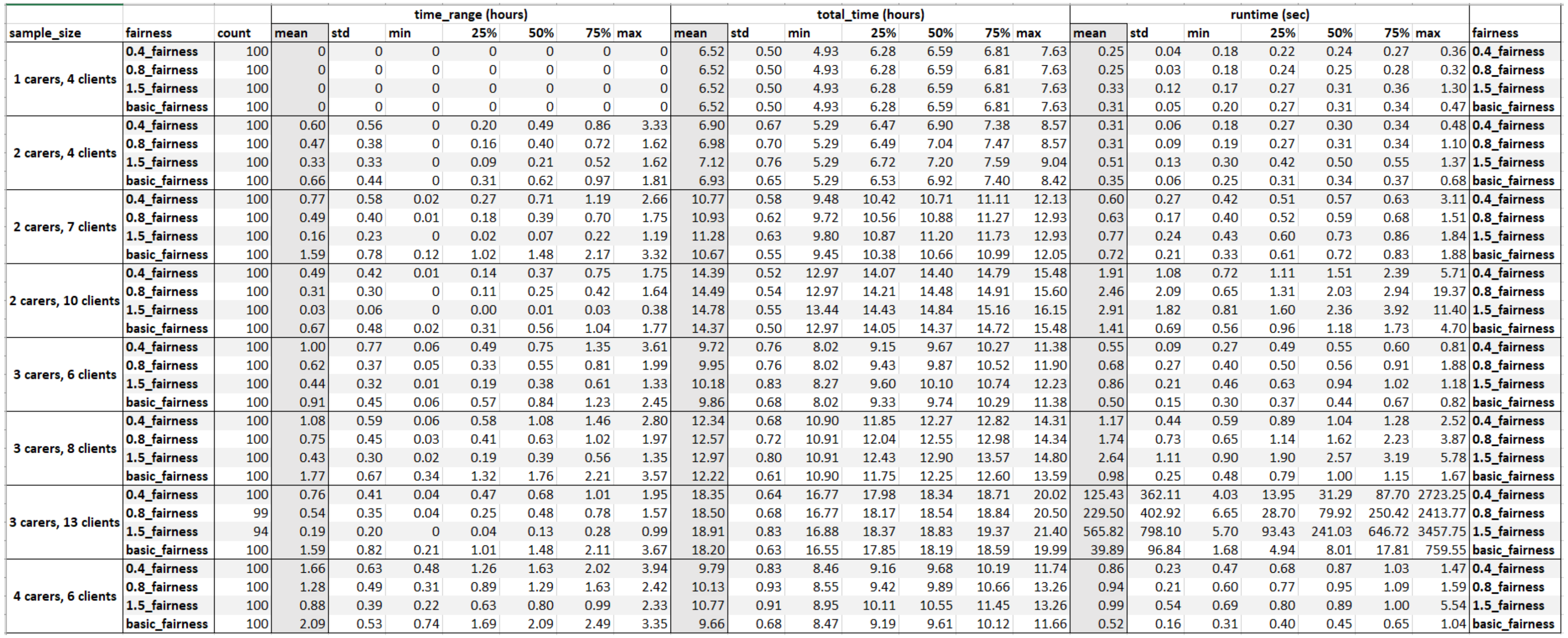
OR

[✓ Minimize] Maximize

Rows : 7 Columns : 7 Generate

Work \ Job	1	2	3	4	5	6	7
1	0	2827	982	1564	1803	1168	1015
2	2868	0	3290	2002	2006	3341	3314
3	947	3324	0	2516	2194	273	1317
4	1508	2145	2447	0	1704	2634	1532
5	1748	2026	2167	1705	0	2218	2607
6	1119	3343	258	2589	2213	0	1353
7	975	3512	1375	1582	2674	1421	0

Figure 26: One carer six client problem input for online TSP solver



			time_range (hours)							total_time (hours)							runtime (sec)							
sample_size	fairness	count	mean	std	min	25%	50%	75%	max	mean	std	min	25%	50%	75%	max	mean	std	min	25%	50%	75%	max	fairness
1 carers, 4 clients	0.4_fairness	100	0	0	0	0	0	0	0	6.52	0.50	4.93	6.28	6.59	6.81	7.63	0.25	0.04	0.18	0.22	0.24	0.27	0.36	0.4_fairness
	0.8_fairness	100	0	0	0	0	0	0	0	6.52	0.50	4.93	6.28	6.59	6.81	7.63	0.25	0.03	0.18	0.24	0.25	0.28	0.32	0.8_fairness
	1.5_fairness	100	0	0	0	0	0	0	0	6.52	0.50	4.93	6.28	6.59	6.81	7.63	0.33	0.12	0.17	0.27	0.31	0.36	1.30	1.5_fairness
	basic_fairness	100	0	0	0	0	0	0	0	6.52	0.50	4.93	6.28	6.59	6.81	7.63	0.31	0.05	0.20	0.27	0.31	0.34	0.47	basic_fairness
2 carers, 4 clients	0.4_fairness	100	0.60	0.56	0	0.20	0.49	0.86	3.33	6.90	0.67	5.29	6.47	6.90	7.38	8.57	0.31	0.06	0.18	0.27	0.30	0.34	0.48	0.4_fairness
	0.8_fairness	100	0.47	0.38	0	0.16	0.40	0.72	1.62	6.98	0.70	5.29	6.49	7.04	7.47	8.57	0.31	0.09	0.19	0.27	0.31	0.34	1.10	0.8_fairness
	1.5_fairness	100	0.33	0.33	0	0.09	0.21	0.52	1.62	7.12	0.76	5.29	6.72	7.20	7.59	9.04	0.51	0.13	0.30	0.42	0.50	0.55	1.37	1.5_fairness
	basic_fairness	100	0.66	0.44	0	0.31	0.62	0.97	1.81	6.93	0.65	5.29	6.53	6.92	7.40	8.42	0.35	0.06	0.25	0.31	0.34	0.37	0.68	basic_fairness
2 carers, 7 clients	0.4_fairness	100	0.77	0.58	0.02	0.27	0.71	1.19	2.66	10.77	0.58	9.48	10.42	10.71	11.11	12.13	0.60	0.27	0.42	0.51	0.57	0.63	3.11	0.4_fairness
	0.8_fairness	100	0.49	0.40	0.01	0.18	0.39	0.70	1.75	10.93	0.62	9.72	10.56	10.88	11.27	12.93	0.63	0.17	0.40	0.52	0.59	0.68	1.51	0.8_fairness
	1.5_fairness	100	0.16	0.23	0	0.02	0.07	0.22	1.19	11.28	0.63	9.80	10.87	11.20	11.73	12.93	0.77	0.24	0.43	0.60	0.73	0.86	1.84	1.5_fairness
	basic_fairness	100	1.59	0.78	0.12	1.02	1.48	2.17	3.32	10.67	0.55	9.45	10.38	10.66	10.99	12.05	0.72	0.21	0.33	0.61	0.72	0.83	1.88	basic_fairness
2 carers, 10 clients	0.4_fairness	100	0.49	0.42	0.01	0.14	0.37	0.75	1.75	14.39	0.52	12.97	14.07	14.40	14.79	15.48	1.91	1.08	0.72	1.11	1.51	2.39	5.71	0.4_fairness
	0.8_fairness	100	0.31	0.30	0	0.11	0.25	0.42	1.64	14.49	0.54	12.97	14.21	14.48	14.91	15.60	2.46	2.09	0.65	1.31	2.03	2.94	19.37	0.8_fairness
	1.5_fairness	100	0.03	0.06	0	0.00	0.01	0.03	0.38	14.78	0.55	13.44	14.43	14.84	15.16	16.15	2.91	1.82	0.81	1.60	2.36	3.92	11.40	1.5_fairness
	basic_fairness	100	0.67	0.48	0.02	0.31	0.56	1.04	1.77	14.37	0.50	12.97	14.05	14.37	14.72	15.48	1.41	0.69	0.56	0.96	1.18	1.73	4.70	basic_fairness
3 carers, 6 clients	0.4_fairness	100	1.00	0.77	0.06	0.49	0.75	1.35	3.61	9.72	0.76	8.02	9.15	9.67	10.27	11.38	0.55	0.09	0.27	0.49	0.55	0.60	0.81	0.4_fairness
	0.8_fairness	100	0.62	0.37	0.05	0.33	0.55	0.81	1.99	9.95	0.76	8.02	9.43	9.87	10.52	11.90	0.68	0.27	0.40	0.50	0.56	0.91	1.88	0.8_fairness
	1.5_fairness	100	0.44	0.32	0.01	0.19	0.38	0.61	1.33	10.18	0.83	8.27	9.60	10.10	10.74	12.23	0.86	0.21	0.46	0.63	0.94	1.02	1.18	1.5_fairness
	basic_fairness	100	0.91	0.45	0.06	0.57	0.84	1.23	2.45	9.86	0.68	8.02	9.33	9.74	10.29	11.38	0.50	0.15	0.30	0.37	0.44	0.67	0.82	basic_fairness
3 carers, 8 clients	0.4_fairness	100	1.08	0.59	0.06	0.58	1.08	1.46	2.80	12.34	0.68	10.90	11.85	12.27	12.82	14.31	1.17	0.44	0.59	0.89	1.04	1.28	2.52	0.4_fairness
	0.8_fairness	100	0.75	0.45	0.03	0.41	0.63	1.02	1.97	12.57	0.72	10.91	12.04	12.55	12.98	14.34	1.74	0.73	0.65	1.14	1.62	2.23	3.87	0.8_fairness
	1.5_fairness	100	0.43	0.30	0.02	0.19	0.39	0.56	1.35	12.97	0.80	10.91	12.43	12.90	13.57	14.80	2.64	1.11	0.90	1.90	2.57	3.19	5.78	1.5_fairness
	basic_fairness	100	1.77	0.67	0.34	1.32	1.76	2.21	3.57	12.22	0.61	10.90	11.75	12.25	12.60	13.59	0.98	0.25	0.48	0.79	1.00	1.15	1.67	basic_fairness
3 carers, 13 clients	0.4_fairness	100	0.76	0.41	0.04	0.47	0.68	1.01	1.95	18.35	0.64	16.77	17.98	18.34	18.71	20.02	125.43	362.11	4.03	13.95	31.29	87.70	2723.25	0.4_fairness
	0.8_fairness	99	0.54	0.35	0.04	0.25	0.48	0.78	1.57	18.50	0.68	16.77	18.17	18.54	18.84	20.50	229.50	402.92	6.65	28.70	79.92	250.42	2413.77	0.8_fairness
	1.5_fairness	94	0.19	0.20	0	0.04	0.13	0.28	0.99	18.91	0.83	16.88	18.37	18.83	19.37	21.40	565.82	798.10	5.70	93.43	241.03	646.72	3457.75	1.5_fairness
	basic_fairness	100	1.59	0.82	0.21	1.01	1.48	2.11	3.67	18.20	0.63	16.55	17.85	18.19	18.59	19.99	39.89	96.84	1.68	4.94	8.01	17.81	759.55	basic_fairness
4 carers, 6 clients	0.4_fairness	100	1.66	0.63	0.48	1.26	1.63	2.02	3.94	9.79	0.83	8.46	9.16	9.68	10.19	11.74	0.86	0.23	0.47	0.68	0.87	1.03	1.47	0.4_fairness
	0.8_fairness	100	1.28	0.49	0.31	0.89	1.29	1.63	2.42	10.13	0.93	8.55	9.42	9.89	10.66	13.26	0.94	0.21	0.60	0.77	0.95	1.09	1.59	0.8_fairness
	1.5_fairness	100	0.88	0.39	0.22	0.63	0.80	0.99	2.33	10.77	0.91	8.95	10.11	10.55	11.45	13.26	0.99	0.54	0.69	0.80	0.89	1.00	5.54	1.5_fairness
	basic_fairness	100	2.09	0.53	0.74	1.69	2.09	2.49	3.35	9.66	0.68	8.47	9.19	9.61	10.12	11.66	0.52	0.16	0.31	0.40	0.45	0.65	1.04	basic_fairness

Figure 27: Fairness measure test results

REFERENCES

Akjiratikarl, C. et al. 2007. PSO-based algorithm for home care worker scheduling in the UK. Computers &amp; Industrial Engineering 53(4), pp. 559-583. doi: 10.1016/j.cie.2007.06.002.

An, Y. et al. 2012. Scheduling healthcare services in a home healthcare system. Journal of the Operational Research Society 63(11), pp. 1589-1599. doi: 10.1057/jors.2011.153.

Bouveret, S. and Lang, J. 2008. Efficiency and Envy-freeness in Fair Division of Indivisible Goods: Logical Representation and Complexity. Journal of Artificial Intelligence Research 32, pp. 525-564. doi: 10.1613/jair.2467.

Bowerman, R. et al. 1995. A multi-objective optimization approach to urban school bus routing: Formulation and solution method. Transportation Research Part A: Policy and Practice 29(2), pp. 107-123. doi: 10.1016/0965-8564(94)e0006-u.

Care and Social Services Inspectorate Wales 2016. 'Above and beyond': national review of domiciliary care in Wales. Merthyr Tydfil: Care and Social Services Inspectorate Wales., pp. 102-108.

CarePlanner 2022. Available at: https://www.care-planner.co.uk/ [Accessed: 11 May 2022].

Cheikhrouhou, O. and Khoufi, I. 2021. A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy. Computer Science Review 40, p. 100369. doi: 10.1016/j.cosrev.2021.100369.

Dantzig, G. et al. 1954. Solution of a Large-Scale Traveling-Salesman Problem. Journal of the Operations Research Society of America 2(4), pp. 393-410. doi: 10.1287/opre.2.4.393.

Du, D. and Pardalos, P. 1998. Handbook of combinatorial optimization. Kluwer Academic Publishers.

Gurobi Optimizer. 2022. Available at: https://www.gurobi.com/products/gurobi-optimizer/ [Accessed: 20 May 2022].

Hebson, G. et al. 2015. Rethinking job satisfaction in care work: looking beyond the care debates. Work, Employment and Society 29(2), pp. 314-330. doi: 10.1177/0950017014556412.

Hoang, L. et al. 2016. Measuring unfairness feeling in allocation problems. Omega 65, pp. 138-147. doi: 10.1016/j.omega.2016.01.005.

Kara, I. and Bektas, T. 2006. Integer linear programming formulations of multiple salesman problems and its variations. European Journal of

Operational Research 174(3), pp. 1449-1458. doi: 10.1016/j.ejor.2005.03.008.

Lenstra, J. and Kan, A. 1981. Complexity of vehicle routing and scheduling problems. Networks 11(2), pp. 221-227. doi: 10.1002/net.3230110211.

Li, J. et al. 2015. Colored Traveling Salesman Problem. IEEE Transactions on Cybernetics 45(11), pp. 2390-2401. doi: 10.1109/tcyb.2014.2371918.

Luo, H. et al. 2021. An ACO-based heuristic approach for a route and speed optimization problem in home health care with synchronized visits and carbon emissions. Soft Computing 25(23), pp. 14673-14696. doi: 10.1007/s00500-021-06263-6.

Miller, C. et al. 1960. Integer Programming Formulation of Traveling Salesman Problems. Journal of the ACM 7(4), pp. 326-329. doi: 10.1145/321043.321046.

Minimum wage for different types of work 2022. Available at: https://www.gov.uk/minimum-wage-different-types-work [Accessed: 2 May 2022].

Notebooks. 2022. Available at: https://tinyurl.com/3f5us8xa [Accessed: 2 March 2022].

Pennycook, M., 2013. Does it pay to care. Under-payment of the National Minimum Wage in the social care sector, Resolution Foundation.

PeoplePlanner 2022. Available at: https://www.theaccessgroup.com/en-gb/health-social-care/care-management-software/peopleplanner/ [Accessed: 11 May 2022].

Ramos, T. et al. 2019. Multi-depot vehicle routing problem: a comparative study of alternative formulations. International Journal of Logistics Research and Applications 23(2), pp. 103-120. doi: 10.1080/13675567.2019.1630374.

Ravalier, J. et al. 2018. Zero-hour contracts and stress in UK domiciliary care workers. Health and Social Care in the Community 27(2), pp. 348-355. doi: 10.1111/hsc.12652.

Ribeiro, R. and Lourenço, H. 2001. A multi-objective model for a multi-period distribution management problem. SSRN Electronic Journal.

Roberti, R. and Toth, P. 2012. Models and algorithms for the Asymmetric Traveling Salesman Problem: an experimental comparison. EURO Journal on Transportation and Logistics 1(1-2), pp. 113-133. doi: 10.1007/s13676-012-0010-0.

routingpy. 2022. Available at: https://pypi.org/project/routingpy/ [Accessed: 20 May 2022].

Rubery, J. et al. 2015. “It's All About Time”: Time as Contested Terrain in the Management and Experience of Domiciliary Care Work in

England. Human Resource Management 54(5), pp. 753-772. doi: 10.1002/hrm.21685.

Schrijver, A. 2005. On the History of Combinatorial Optimization (Till 1960). In: Aardal, K. et al. ed. Discrete Optimization. Elsevier, pp. 1-68.

Shah, P. 2022. Travelling salesman problem using branch and bound (penalty) method calculator. Available at: https://cbom.atozmath.com/CBOM/Assignment.aspx?q=tsp [Accessed: 26 May 2022].

Wibberley, G. 2013. ‘Getting the Bodies of the Workers to the Bodies of the Clients: the Role of the Rota in Domiciliary Care’ in: Wolkowitz, C. et al. ed. Body/sex/work. Basingstoke: Palgrave Macmillan, pp. 223-238.

Working With Multiple Objectives. 2022. Available at: https://www.gurobi.com/documentation/9.5/refman/working_with_multiple_obje.html [Accessed: 15 March 2022].