Exploring fairness in food delivery routing and scheduling problems

Antonio Martínez-Sykora, Fraser McLeod, Tom Cherrett,
Adrian Friday

PII:	S0957-4174(23)02990-1
DOI:	https://doi.org/10.1016/j.eswa.2023.122488
Reference:	ESWA 122488

To appear in: *Expert Systems With Applications*

Received date :	7 June 2023
Revised date :	3 October 2023
Accepted date :	5 November 2023

Exploring fairness in food delivery routing and scheduling problems

Antonio Martínez-Sykora[a] (a.martinez-sykora@soton.ac.uk), Fraser McLeod[b] (f.n.mcleod@soton.ac.uk), Tom Cherret[b] (t.j.cherrett@soton.ac.uk), Adrian Friday[c] (a.friday@lancaster.ac.uk)

[a] Centre for Operational Research, Management Science and Information Systems (CORMSIS), University of Southampton, Southampton, United Kingdom
[b] Transportation Research Group, University of Southampton, Southampton, United Kingdom
[c] Department of Computing and Communications, Lancaster University, Lancaster, United Kingdom

Corresponding Author:
Antonio Martínez-Sykora
Centre for Operational Research, Management Science and Information Systems (CORMSIS), University of Southampton, Southampton, United Kingdom
Email: a.martinez-sykora@soton.ac.uk

Exploring fairness in food delivery routing and scheduling problems

Antonio Martínez-Sykora[a,*], Fraser McLeod[b], Tom Cherrett[b], Adrian Friday[c]

[a] *Centre for Operational Research, Management Science and Information Systems (CORMSIS), University of Southampton, Southampton, United Kingdom*
[b] *Transportation Research Group, University of Southampton, Southampton, United Kingdom*
[c] *Department of Computing and Communications, Lancaster University, Lancaster, United Kingdom*

Abstract

Demand for delivery of take-away meals to customers has been growing worldwide, with deliveries often performed by non-specialised gig economy couriers working for online platform operators such as Deliveroo or Just Eat. This has led to the introduction of the 'meal delivery problem', characterised by a series of individual pickup and delivery tasks to be assigned to available couriers. While there is a vast set of algorithms proposed in the literature that aim to minimise total workload, very little attention has been given to equitably distributing work between couriers. We propose a new multi-objective problem that is aiming at distributing orders equitably between couriers as well as minimising total workload, where all information is known upfront. We propose an integer linear programming (ILP) model with a weighted objective function that is used to derive the Pareto front in small-scale problems by exploiting the $\epsilon-$constraint approach. This formulation has been proven to solve in a reasonable time for problems with up to 60 orders, however, the optimal Pareto front can only be computed within a reasonable time for problems up to 30 orders. For problems with more orders, we propose a Variable Neighbourhood Search

*Corresponding author.
Email addresses: `a.martinez-sykora@soton.ac.uk` (Antonio Martínez-Sykora), `f.n.mcleod@soton.ac.uk` (Fraser McLeod), `t.j.cherrett@soton.ac.uk` (Tom Cherrett), `a.friday@lancaster.ac.uk` (Adrian Friday)

(VNS) algorithm, for which the fitness evaluation evolves in order to explore a wider set of the solution space. The VNS is compared against the ILP and also tested on more realistic size instances with up to 3123 orders, improving the performance over the business as usual and shows that equitable distribution of work can be achieved alongside reducing the total travelled distance.

Keywords: food delivery, vehicle routing, fairness, variable neighbourhood search, integer linear programming

1. Introduction

The demand for delivery of take-away meals to customer homes, workplaces or other locations has been increasing rapidly, worldwide, in recent years, enabled by online platform technology and growth in companies that offer customers a range of restaurant menu options, manage payments and arrange deliveries (Dablanc et al., 2017). In the United Kingdom, the meal delivery market was estimated to be worth around 8.5 billion British pounds, in 2019, with around 11 million users ordering from major operators such as Just Eat, Domino's, Deliveroo and Uber Eats (Chartered Institute of Environmental Health (2020)).

This growth has led to increasing interest from both practitioners and the OR community in how to arrange an effective delivery service, where, typically, the main objectives are fast deliveries (e.g. within one hour) at low cost. Delivery is often performed by so-called gig economy couriers: non-specialised people registered with one or more online platform companies to undertake individual delivery tasks using their own transport (e.g. bike, motorbike, van or car).

This operating model and practice has received much criticism about perceived unfair treatment of couriers relating to low pay, lack of employment rights, lack of managerial support and job insecurity (Nolan (2018), Broughton et al. (2018), Field & Forsey (2018) and Cant (2020)), in addition to courier concerns relating to personal safety, and security of their vehicles while working (Dablanc et al. (2017) and Christie & Ward (2018)). Low pay is often linked to the practice of being paid a fixed amount for a given order completed, rather than an hourly rate, which means that the courier is effectively unpaid for any excess waiting time at restaurants (when meal preparation has been delayed) or at delivery addresses (e.g. when customers cannot be found) (Cant (2020)). In addition, this operating model can encourage the platform companies to hire too many couriers for the available work at any given time, as this does not directly cost the company and is of benefit to them in terms of ensuring fast delivery. However, this has a negative impact on individual couriers as they will receive fewer orders and wait longer between them, with this further source of waiting time also being unpaid. Some couriers also complain about perceived app biases that appear to favour some couriers over others in allocating orders (Cant (2020)). The purpose of this paper is to consider food delivery routing and scheduling methods that are fairer to couriers by distributing work more equitably between them, without unduly compromising delivery cost or time taken.

Section 2 includes a review of related literature. Section 3 introduces the problem and formalises it in Section 4 by proposing an Integer Linear Programming (ILP) model. The model includes several objectives, which are embedded into an $\epsilon-$constraint approach proposed in Mavrotas & Florios (2013) to derive or approximate the Pareto front, including balancing orders, minimising travel time and minimising waiting time. To solve larger and more practical problems, we propose a Variable Neighbourhood Search (VNS), described in Section 5. The VNS starts with a solution obtained by greedy rules typically used by platforms to sequentially assign orders to couriers (Section 5.1). These solutions are aimed to capture behaviour of the business-as-usual (BAU) solutions used by platforms. Finally, in Section 6 we compare the VNS against the approximated Pareto front obtained by the ILP models for a small-scale problem and against the business-as-usual (BAU) solution for larger problems.

2. Literature review

The meal delivery problem (MDP) is relatively new within OR literature, introduced by Reyes et al. (2018), where their main goal was to maximise the number of delivered orders while also considering delivery costs, based on fixed payments per order and a guaranteed minimum wage, and customer waiting times (time between ordering and delivery). The authors built a comprehensive set of problem instances generated from real world historic data, available at `https://github.com/grubhub/mdrplib` and used in this paper.

Vidal et al. (2020) presented a comprehensive guide to emerging vehicle routing problem variants in which different variants of pickup and delivery problems were discussed. MDP are pickup and delivery vehicle routing problems (PDVRP) where goods are to be collected from specific locations (e.g. restaurants) and delivered to customers. They differ from the commonly studied capacitated vehicle routing problem (CVRP), where all goods are picked up from a depot. Another variant of PDVRP considers items that need to be collected and delivered from customers at the same time. A review paper on this variant can be found in Koç et al. (2020). In Arslan et al. (2018), the authors assessed the impact of crowdsourcing in the dynamic PDVRP for parcel delivery under various assumptions about the behaviour of the ad hoc couriers.

The main difference between the general PDVRP and MDP derives from the time window constraints. The MDP has extremely tight delivery time windows, to the extent that only orders from the same restaurant at similar times can potentially be combined with one another (Reyes et al. (2018), Yildiz & Savelsbergh (2019)) and, in UK practice, Cant (2020) reported that opportunities to combine orders were very rare. This makes a major difference since two consecutive pickups for different customers are generally not allowed in the MDP and, therefore, the flexibility on designing a routing plan is more limited. However, the assignment of orders to couriers has a direct impact on the quality of the solution and the efficient use of the couriers, which commonly are measured by the total waiting time of the couriers and the distance travelled by couriers between orders.

There are exact algorithms based on mathematical formulations for some of these problems. In Aziez et al. (2020), a rolling horizon framework was proposed to solve the dynamic PDVRP with ad hoc couriers and they developed an exact solution approach that solves the matching problem between ad hoc couriers and parcels every time new information is available. In Cosmi et al. (2019), the authors explored the single courier, single restaurant, MDP. In this case, there was no routing to be planned and the authors reduced it to a single machine scheduling problem. A dynamic programming model was then proposed, which can be solved in polynomial time when the slacks are bounded. Baldacci et al. (2011) proposed an exact algorithm based on the set partitioning integer linear programming model of the CVRP, in which the authors improved by enhancing valid inequalities and reducing the problem size by using the dual solution. In Dahle et al. (2019), the authors extended the PDVRP by considering occasional couriers, and the exact algorithm proposed was able to optimally solve the problem with up to 70 requests.

There are more recent papers in the literature targeting the efficiency from the platform's perspective. In Zheng et al. (2022) the authors explored an online food delivery problem in which the problem is decomposed into two coupled problems: an order assignment problem (assigning orders to couriers), and a stochastic vehicle routing problem. The objective function considered in Zheng et al. (2022) aims to minimise the total time taken, which might result in unbalanced solutions that can be considered unfair from a courier's perspective.

In most of the publications on the MDP the main focus is on minimising the total cost and maximising the number of orders delivered. A relatively quick delivery time is usually offered to the customer to attract their custom and using couriers efficiently is key to this goal. In common practice, couriers are paid an agreed amount per delivery order with no guarantee of earning a minimum hourly rate (Cant (2020)) whereas Reyes et al. (2018) modelled additional compensation payments to couriers where the number of orders assigned to them did not meet an agreed minimum number. Meal delivery platform companies will typically assign the next order to whichever courier is available and nearest to the pickup point with no consideration of fairly distributing work between couriers ((Cant (2020)). This can result in unbalanced earnings between couriers as some receive more orders than others either through luck or courier experience in being in the right place at the right time, or created by any inherent platform biases where some couriers or vehicle types are prioritised over others.

There has been recent interest in finding balanced solutions in several routing problems, as well as new variants and mathematical formulations to address fairness in various optimisation problems (see Bektaş & Letchford (2020)). In Bektaş et al. (2019), a branch and cut algorithm for the balanced vehicle routing problem (BVRP) is proposed, where the aim is to balance the number of nodes on each route in the final solution. This problem was first introduced in Gouveia & Salazar-González (2010). This is aligned with the aims of this work since one node could represent a specific order. However, in Bektaş et al. (2019), the balancing feature was addressed in the constraints, by assuming a lower bound and an upper bound on the number of orders in any route. In this paper we introduce this fairness component as one of the terms in the objective function.

The delivery problem addressed in this work accounts for four different objectives, and the multi-criteria aspect brings to the problem the possibility to define many different fitness functions that can be used in any heuristic algorithm to approximate the Pareto front. These fitness functions are usually derived by setting different weights to the objectives and then solving a single-objective problem. However, the exact Pareto front (or good approximations) is much more complex to compute in practice, and generally rely on $\epsilon-$constraint methods, where all the objectives but one are regarded as constraints, and then the model is solved many times by changing the constraint set in a structured way. In Mavrotas (2009) the author proposed an efficient implementation of the $\epsilon-$constraint method, named as AUGMECON (AUGMented Epsilon CONstraint), accelerating the process by avoiding the calculation of weak solutions and reducing overall iterations of the process. A more effective approach is presented in Mavrotas & Florios (2013), named as AUGMECON2, which is capable of efficiently approximating the Pareto front to produce competitive results for larger problems, compared to multi-objective meta-heuristics. In this work we have used the AUGMECON2 algorithm to derive good approximations on small-scale problems.

3. Problem description

In this paper we consider different aspects of fairness as objectives in the optimisation algorithm. We assume that the number of orders (I) and the set of couriers (J) is known and that $|J|$ is sufficiently large to meet all the orders. All couriers are assigned to a shift with a maximum working time ('shift length') of T_{max}, and they can have different starting times across the day. The aim is to assign all the orders to couriers in such a way that the following objectives, in order of importance, are optimised:

1. The range in the number of orders undertaken by couriers is minimised. (For example, if all undertake the same number of orders then the range is zero.)
2. The travel time between orders (from delivery to next pickup) is minimised. (Note: the travel time within an order, from pickup to delivery, is assumed to be fixed for any given mode of transport.)
3. The total waiting time between orders is minimised
4. The range in the proportion of waiting time (i.e. ratio of waiting time to shift length) across all the couriers is minimised.

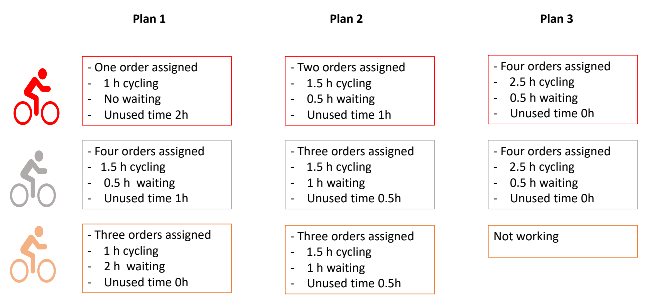
The four objectives consider both fairness and efficiency aspects. We consider balancing orders between couriers to be the primary objective as, in practice, this dictates the pay they will receive. The second and third objectives both relate to how efficiently the work is undertaken. The fourth objective also relates to fairness, aiming at balancing the proportion of waiting time each courier incurs, as this is effectively unpaid time.

Each order $i \in I$ has an associated pickup location, c_i, a delivery location, d_i and a required pickup time t_i. Immediate delivery is assumed (i.e. no waiting for another order), so delivery time is known once the mode of transport used by the courier assigned to the order is determined. We represent by t_j^s (t_j^f) the start (finish) working time of courier $j \in J$ and m_j is the mode of transport used by courier j, $m_j \in \{1, \ldots, \overline{m}\}$, where $\overline{m}$ is the number of different modes of transport available. This differentiation between transport modes is only needed when computing the travel times of the route plan for the couriers. We denote by $t_{a,b}^m$ the travel time between locations a and b by transport mode m. In our experiments we only consider bikes ($m = 1$) and motorbikes ($m = 2$) but this can be easily extended to any other combination using any type of transport. In this paper we assume that the travel times $t_{a,b}^m$ are given. In order to assess the impact of the transport modes in Section 6 we created several problem instances by sampling the travel times from real data. However, in order to be able to use the model proposed in this work some estimates on travel times will be required.

The length of each shift is limited by a specified maximum value, which, in practice, would be determined either by the courier or by the company they work for. We denote by T_{max} the maximum time allowed in a shift for any courier. Therefore, in any solution or plan it is assumed that each courier $j \in J$ is assigned to a shift with a starting time t_j^s and finish time $t_j^s + T_{max}$. If a courier arrives at a collection point before the meal is ready to collect then the courier should wait at that location, and some waiting time is incurred. The time between the last delivery t_j^f and the end of the shift, $t_j^s + T_{max}$, is regarded as *unused time*, which differs from waiting time. We assume that the courier can use that time for other purposes, e.g. going home early or working with another platform.

In Figure 1 we present an illustrative example with three cycle couriers, eight orders and three different solutions (plans). Lets assume in this example that each courier is available to work for up to 3 hours. The first plan, using all three couriers, has a total travel time (cycling) of 3.5 hours, which is the lowest of the three plans shown, but the orders are unevenly distributed between the

Figure 1: Example with 3 couriers, 8 orders, comparing three plans.



couriers, whereas plan 2 has a more even distribution of orders but the total travel time increases to 4.5 hours. Even though the number of couriers is not being minimised in this work, it is worth pointing out that plan 3 provides more work (= more pay) for the two couriers used and would be best from the couriers' perspective, so limiting the number of couriers is key to offering sustainable shifts.

4. ILP model

Let R^m be the set of all the feasible routes by transport mode $m \in \{1, \ldots, \overline{m}\}$. Each route $r \in R^m$ is defined by a subset of orders, $r = \{i_1, \ldots, i_{|r|}\}$. We can assume that the orders in r are sorted by increasing collection time (or delivery time). Then, route r stops in the following locations in the given order $(c_{i_1}, d_{i_1}, c_{i_2}, d_{i_2}, \ldots, c_{i_{|r|}}, d_{i_{|r|}})$. The travel time within the orders (from pickup to delivery) on route r can be computed as

$$D_r^w = \sum_{i \in r} t_{c_i, d_i}^m$$

The travel time between orders (from delivery to next pickup) of route r can be calculated as

$$D_r^b = \sum_{i \in \{1,\ldots,|r|-1\}} t^m_{d_i,c_i+1}$$

Let $R = \bigcup_{m \in \{1,\ldots,\overline{m}\}} R^m$. For each route $r \in R$ we define a binary variable, x_r, which takes the value one if route r is used in the solution. We denote by W_r the waiting time incurred by route r, defined, here, as all non-travel time between the first pickup and the last delivery. The proportion of waiting time to total time for any given route is denoted by ω_r, and can be calculated as $\omega_r = W_r/T_r$, where T_r is the total time taken between the first pickup and the last delivery $(t_j^f - t_j^s)$. Note that $T_r \leq T_{max}$ for any valid route r.

To avoid quadratic terms in the first objective function (in Section 3), we define two continuous variables, y_{max} and y_{min}, which represent the maximum and minimum number of orders assigned to any route used in the solution, respectively. Similarly, we define variables z_{max} and z_{min} as the maximum and minimum proportion of waiting time by any route in the solution, so the fourth objective described in Section 3 can also be linearised. In Table 1 we summarise the notation used in this paper.

To capture the four objectives (1-4) listed in Section 3, we consider the four objective functions described in, (1), (2), (3) and (4) all of them aimed at being minimised.

$$\text{Minimise} \quad y_{max} - y_{min} \tag{1}$$

$$\text{Minimise} \quad \sum_{r \in R} D_r^b x_r \tag{2}$$

$$\text{Minimise} \quad \sum_{r \in R} W_r x_r \tag{3}$$

$$\text{Minimise} \quad z_{max} - z_{min} \tag{4}$$

This objective can be combined into a single weighted objective function by

Table 1: Notation - Parameters and variables

Parameters	
I	set of orders
J	set of couriers
$m \in \{1, \ldots, \overline{m}\}$	mode of transport
R^m	Set of all feasible routes for mode m
R	Set of all feasible routes
W_r	Waiting time in route $r \in R$
T_r	Total time to perform route $r \in R$
T_{max}	Maximum time length allowed in a shift
ω_r	proportion of waiting time in shift defined by $r \in R$
D_r^w	Total travel time from collections nodes to delivery nodes
D_r^b	Total travel time from delivery to the next collection
Variables	
x_r	value 1 if route r is used in solution
y_{max}	maximum number of orders performed by any used route
y_{min}	minimum number of orders performed by any used route
z_{max}	maximum proportion of waiting time performed by any used route
z_{min}	minimum proportion of waiting time performed by any used route

adding weights w_1, w_2, w_3 and w_4, as follows.

$$\begin{aligned} F = w_1(y_{max} - y_{min}) + w_2 \sum_{r \in R} D_r^b x_r \\ + w_3 \sum_{r \in R} W_r x_r + w_4(z_{max} - z_{min}) \end{aligned} \tag{5}$$

Even though this is a widely used approach on multi-objective problems, the way the weights are set up clearly determines the quality of the Pareto front. In this work we will use the efficient implementation of the $\epsilon-$constraint method used in Mavrotas & Florios (2013). In order to compute the optimal Pareto front (or good approximations) by using the $\epsilon-$constraint method, we define the following Integer Linear Programming model (ILP), where M is a big-M constant used to deactivate constraint (13) if route r is not used in the solution.

$$
\begin{aligned}
\text{Minimise} \quad & y_{max} - y_{min} + \epsilon(S_2/r_2 + 10^{-1}S_3/r_3 + 10^{-2}S_4/r_4) && && (6)\\
s.t. \quad & \sum_{r \in R} D^b_r x_r - S_2 = e_2 && && (7)\\
& \sum_{r \in R} W_r x_r - S_3 = e_3 && && (8)\\
& z_{max} - z_{min} - S_4 = e_4 && && (9)\\
& \sum_{r \in R, i \in r} x_r = 1 && \forall i \in I && (10)\\
& \sum_{r \in R} x_r = |J| && && (11)\\
& y_{max} \geq |r| x_r && \forall r \in R && (12)\\
& y_{min} \leq |r| + M(1 - x_r) && \forall r \in R && (13)\\
& z_{max} \geq W_r x_r && \forall r \in R && (14)\\
& z_{min} \leq W_r x_r && \forall r \in R && (15)\\
& x_r \in \{0, 1\} && \forall r \in R && (16)
\end{aligned}
$$

Where constraints (7), (8) and (9) are mapped to objectives (2), (3) and (4). Parameters e_2, e_3 and e_4 are set up for each iteration used in the AUGMECON2 method described in Mavrotas & Florios (2013). It is worth highlighting that, even though the original algorithm is considering a maximisation problem, the same approach can be used for a minimisation problem. Parameters r_2, r_3 and r_4 are the ranges of the respective objective function, and S_2, S_3 and S_4 are the surplus variables of the respective constraints, which are required to formulate the main model used in the AUGMECON2 method. Finally, in this work we have considered $\epsilon = 10e - 4$, which is sufficient to guarantee that the optimal solution matches with the optimal solution of the problem where the relevance of the objectives are ranked in lexicographic order, helping to reduce the number of MILP models being optimally solved.

Constraints (10) force each order to be served, and constraint (11) forces all of the couriers to be used. Finally, constraints (12), (13), (14) and (15) are needed to properly link variables y_{min}, y_{max}, z_{min} and z_{max} with variables x_r. Constraints (16) force variables x_r to be binary.

4.1. Implementation details of AUGMECON2

As it is shown in Mavrotas & Florios (2013), objective function (6) is already performing a lexicographic optimisation for objectives 2,3 and 4. This is an improvement from previous $\epsilon-$constraint methods that did not consider the weights and had more erratic behaviour of the ultimate algorithm when there are alternative optimal solutions.

In order to compute the range for objectives 2,3 and 4, we first need to compute the payoff table, which is the table containing the results for the individual optimisation of all the objective functions separately. It is worth highlighting that the lower bound of each objective (best value) is easily obtained, however, the drawback of this method is to compute an efficient nadir value for each objective. In this work, this is approximated with the maximum value obtained when solving all the individual optimisations of the objective functions. Once the payoff table is calculated we can compute the ranges r_2, r_3 and r_4 for objectives 2,3 and 4.

Due to the vast CPU time needed when using this method, in our computational experiments we have then divided the ranges into 10 intervals for objectives 2, 3 and 4, which limit the number of ILP problems that need to be optimally proven to 1000 (accounting for all the combinations of the 10 intervals for objectives 2,3 and 4) and we identified that these parameters provide a very good approximation of the Pareto front. Furthermore, by using the AUGMECON2 method the number of ILPs solved is reduced considerably, but this reduction is not sufficient to compute the optimal Pareto front (by considering a unity step when defining the range) since the time consumed to solve a single ILP model can easily be the order of hours for small-scale problems, and the number of problems that should be considered is increasing exponentially.

5. Variable Neighbourhood Search Algorithm

Variable neighbourhood search (VNS) algorithms have been proven to be quite efficient when solving vehicle routing problems (Kytöjoki et al. (2007), Sze et al. (2017)). The problem presented in Section 3 differs substantially from the standard vehicle routing problems, where the neighbourhoods considered in the literature can be classified in two types, the inter-route neighbourhoods such as interchange or cross-exchange between routes, where two or more routes are modified simultaneously, and the intra-route neighbourhoods, like the 2-opt for the travelling salesman problem, where only one route is modified and the only changes in the solutions are related to modifying the visiting order of the nodes within the route.

In the MDP, the time windows constraint plays an important role, defining a unique sequence for each feasible route. Therefore, the routing problem is somehow defined and the main problem is to allocate the right set of orders to each route (courier). In this work we heavily exploit the efficiency of simple intra-route neighbourhoods based on the 1-insertion and the interchange (or swap), but we include different stages of the algorithm for which the fitness function keeps changing to account for the multiple objectives. One of the main differences with the VNS proposed in this work and the first VNS proposed, Hansen et al. (2008), is that in this paper we are using the neighbourhood to improve the first objectives as the kick to escape from the local optima on the other objectives, however, the other neighbourhoods are used in a simliar way than in the original VNS setting. We now give a detailed explanation of the construction heuristic, the two neighbour structures, and how they are used in a VNS framework where the fitness function is modified during the search.

5.1. Initial construction heuristic

The initial construction heuristic is an iterative process where the orders are assigned one at a time to the first available courier from the list. We first sort the orders by non-decreasing collection time and couriers are sorted at random or by some criteria, e.g. using predefined priorities or based on the courier waiting time. Algorithm 1 runs in linear time on the number of orders. The first for

Algorithm 1 Constructive algorithm

```
1: Sort $I$ by non-decreasing collection time
2: Sort $J$ by non-decreasing preference
3: for each $i \in I$ do
4:     Initialise $j = 1$
5:     while $i$ cannot be allocated to courier $j$ do
6:         $j = j + 1$
7:     end while
8:     Assign $i$ to $j$, i.e, $r_j = r_j \cup \{i\}$
9:     Update ordering in $J$ if needed
10: end for
```

loop (line 3) ensures that all orders are explored and in lines 5-7 the algorithm computes the first courier available to deliver the order. The condition in line 5 is true if courier j has been already assigned an order which clashes with order i. The order in which the list of couriers is sorted is key in this construction heuristic since it has some similarities with the first fit decreasing or the best fit decreasing in bin packing problems. In line 9 of Algorithm 1 we allow the order to be modified during the algorithm to try a best fit approach. To reflect what some meal delivery platforms do in practice, we tried the following different options as potential 'business as usual' (BAU) scenarios:

1. The list of couriers J is sorted by a given fixed priority order and the order will always be offered to an available courier with highest priority. This approach will result in higher priority couriers receiving more orders than

lower priority couriers and not all couriers will be used since the aim is to minimise the number of couriers used.

2. The list of couriers J is sorted by waiting time. The courier that has been waiting longest will receive the order.
3. The list of couriers J is sorted by time required to get to the collection point of the order. The courier that can get to the collection point soonest will be assigned to the order.

In Table 10 in the appendix we show the solutions obtained by applying these three rules. It can be observed that the first strategy leads to more unbalanced shifts and the second strategy is the least efficient in terms of total travel time, which make sense since in the first strategy the couriers are ranked by priority and in the second strategy, despite the waiting time being more balanced, the travel time is not considered. However, all of these strategies result in unbalanced shifts, where the range of orders is around 5. The algorithm proposed in this paper aims to reduce the range as well as balance the waiting time and workload across the gig workers, aiming for a range of 1 order on the same shifts.

5.2. Neighbourhood structures

In this paper we explore two well-known fast and greedy neighbourhood structures. The first neighbourhood we consider is the insertion of one order from one courier into a different courier in such a way that the total distance travelled is minimised. However, in order to explore efficient solutions in the different objectives (criteria) described in Section 3 we add two different stages on the search mapped. The first stage is targeting the fairness, corresponding to objectives 1 and 4, and the second stage is aimed at improve the traveling efficiency (the routing). It is important to highlight that these neighbourhoods do not use the weighted objective function described in (5), but only the criteria being considered.

Objectives 1 and 4 are looking at fairness across the couriers, and generally clash more often with objectives 2 and 3, which are aiming at increasing the efficiency of routing and reducing the total travel time. Therefore, when applying the first neighbourhood we will consider two phases by introducing one parameter, $h \in H = \{1, 2\}$. When $h = 1$ we are aiming to improve the fairness of the solution and only objectives 1 (range) and 4 (balancing waiting time) of those listed in Section 3 are considered as the fitness and the other objectives are ignored within this neighbourhood. We start by accepting new solutions that improve performance in objective 1, and, when there is no further improvement, we then use the same insertion strategy, where solutions that improve performance in objective 4 are accepted, maintaining the solution quality achieved in objective 1.

This neighbourhood $N1(h)$, $h \in H$ is described in Algorithm 2, which runs in linear time with respect to the number of orders. We use a first improvement acceptance strategy for $N1(h)$, $h \in H$, and in the case that $h = 2$ we sort the orders randomly at every iteration of the algorithm (see line 4). However, when $h = 1$, the algorithm is trying to balance the quality of the route among couriers (i.e. balancing the number of orders assigned to the couriers) and, therefore, the orders will always be selected from the top of the list, i.e, the courier with the most convenient route in terms of objectives 1 and 4, and then they will be inserted to the courier on the back of the list, i.e, the courier with the worst route in terms of objectives 1 and 4. For example, if we are minimising the range of the number of orders on a given stage of the algorithm we will be selecting orders from the courier with more orders assigned and these will be added to the courier with fewer orders assigned and, in this case, only removing orders from the first courier on the list or adding orders to the last courier would be explored since this is aiming to reduce the range.

The second neighbourhood ($N2$) is the swap of two orders between two couriers. This is explained in Algorithm 3. This algorithm is quadratic on the number of orders and it only aims to improve objectives 2, 3 and 4 since it is not really helpful to improve fairness. As in $N1(h)$, we perform any improvement found during the search process.

Algorithm 2 N1(h)

```
1: Let $\overline{J} \subseteq J$ be the set of couriers being used
2: Routes $r_j,\ j \in \overline{J}$ are sorted by non-increasing number of orders ($|r_j|$)
3: while There is an improvement do
4:   Sort $I$ randomly
5:   for each $i \in I$ do
6:     Find best insertion of order $i$ into a different route(courier).
7:     if Fitness $h$ improved then
8:       Improvement found
9:       Update solution
10:      Break for loop
11:    end if
12:  end for
13: end while
```

Algorithm 3 N2

```
1: Let $\overline{J} \subseteq J$ be the set of couriers being used
2: Routes $r_j,\ j \in \overline{J}$ are sorted by non-increasing number of orders ($|r_j|$)
3: while There is an improvement do
4:   Sort $I$ randomly
5:   for each $r_j, r_k,\ j, k \in \overline{J}$ do
6:     for each possible swap do
7:       Compute value of fitness
8:       if Fitness improved then
9:         Improvement found
10:        Update solution
11:        Break both for loops
12:      end if
13:    end for
14:  end for
15: end while
```

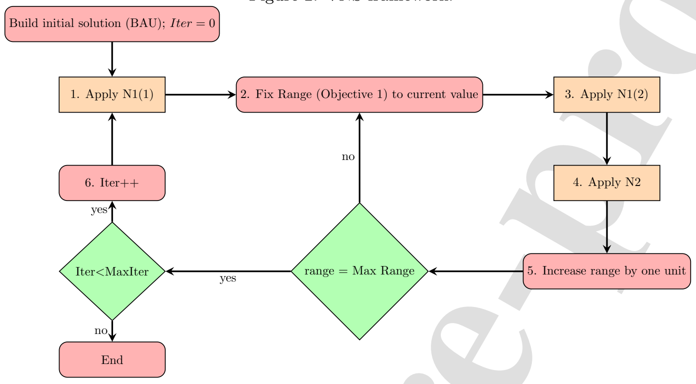
5.3. VNS framework

An overview of the VNS algorithm is shown in Figure 2. We first build an initial solution by using Algorithm 1 with the first option described in Section 5.1. This is determining the business-as-usual (BAU) solution and dictating the number of couriers needed; we assume that this number would be provided by the platform and is not allowed to change during the algorithm.

Then, in a first step, we apply the neighbourhood N1, which will find a solution aiming at reducing the first objective, the range on the number of jobs assigned to all the couriers. Then, the value of the first objective (range) is fixed in the second step, and it remains fixed when both neighbourhoods N1(2) and N2 are applied, steps 3 and 4, aiming at improving objectives 2 and 3. Every time a non-dominated solution is found after applying each neighbourhood then it is added in the Pareto set, and this set is updated by removing all the solutions that are dominated by the new solution. Once there is no further improvement in step 4 we allow the range of the first objective to increase by one unit (step 5). If the range is maximum then we complete one iteration of the VNS, and we increase the iteration number by one unit (step 6) if the maximum number of iterations of the VNS has not been reached.

It is worth highlighting that the solution that is returned from step 6 to step 1 usually performs badly in the first objective, which is again optimised in step 1, ignoring the other objectives. However, during one iteration of the algorithm we try to improve the other objectives again, and we iterative allow more flexibility by allowing to explore more unfair solutions (by worsening the first objective). The stopping criteria of the algorithm is the number of iterations, $MaxIter$. In all of our experiments we used $MaxIter = 5000$, providing a good balance between time and solution quality for all instances.

Figure 2: VNS framework.


Build initial solution (BAU); *Iter* = 0
1. Apply N1(1)
2. Fix Range (Objective 1) to current value
3. Apply N1(2)
4. Apply N2
5. Increase range by one unit
range = Max Range
no
yes
Iter<MaxIter
yes
6. Iter++
no
End

6. Computational results

In this section we present the results of the algorithm described in Section 5.3 using two different sets of instances. In Section 6.1 we used data from a major UK courier working on behalf of a meal delivery platform to simulate a set of instances, allowing us to assess in Section 6.3 the performance of both the ILP model presented in Section 4 and the VNS algorithm presented in Section 5. Finally, in Section 6.4 we compare the BAU against the VNS algorithm in a larger set of instances proposed in Reyes et al. (2018).

6.1. Courier data

Data from a major courier operating on behalf of a meal delivery company in Greater London were obtained for a 3-week period (10-31 July 2017). The data comprised individual meal pickup and delivery times and locations, courier ID and their mode of transport (bike or motorbike) for a total of 7,917 meal deliveries. These took place in scattered locations across Greater London (Figure 3). Delivery trip distances were estimated using Google's Distance Matrix API service and were found to have a mean of 2019m and a standard deviation of 1000m. Based on this, and to generate more instances, we sampled delivery distances from a Normal distribution, $X \sim N(2019, 1000)$. We first randomly decided the pickup location of a new order from the map and then we used the X distribution to sample the distance of the delivery location. Then we randomly selected the delivery point from the circumference whose centre is the pick-up location and the radius is the distance. Therefore, we guaranteed that the distribution of our instances have, on average, 2019m from pick-up point to delivery point.

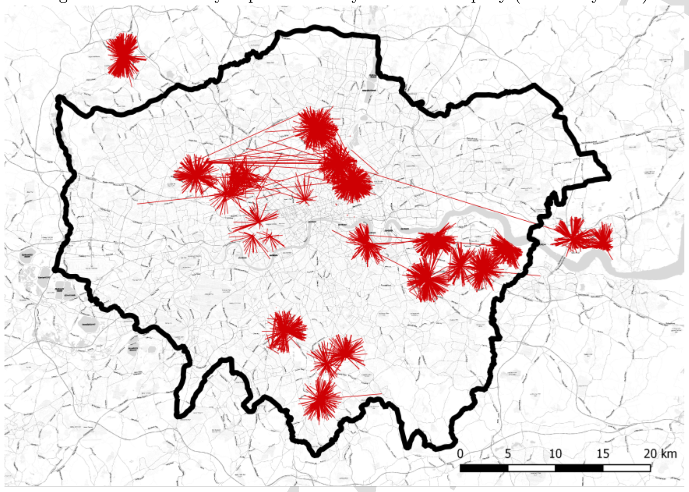
We generated 50 instances with 20, 30, 40, 50 and 60 orders (10 instances for each number of orders). In the first 20 instances, with 20 and 30 orders we have solved the multi-objective problem by using the AUGMECON2 method described in Section 4.1. For the instances with 40, and 50 and 60 orders we have solved the induced problem by considering a lexicographic approach on the four objectives, i.e., in the weighted approach described in equation (5) we considered $w_1 > w_2 > w_3 > w_4$ to show the complexity of the ILP models being solved and the efficiency of the VNS algorithm presented in 5 in the problem which consider the objectives in a lexicographic order. In all the experiments the shifts for the gig economy couriers were fixed to 4 hours from the starting work time (first delivery).

All of the ILP models were solved by using GUROBI (version 9.1.1) on an Intel (R) Core(TM) i9-9980E CPU with 3.00 GHz and 18 Cores desktop. A time limit of 2 hours was imposed in the computational experiments for the lexicographic problems, on instances with 40, 50 and 60 orders, and no time limit for the AUGMECON2 approach on the smaller instances. The same machine was used to run the VNS.

6.2. VNS and exact algorithm - Multiobjective

One of the most widely used measures to compare the quality of the Pareto set obtained by different algorithms is to compute the hyper-volume covered by the Pareto set. In order to do that, we need to have a reference solution or point (generally a much worse solution, or even a nadir point) to compute the

Figure 3: Food delivery trips recorded by a courier company (10-31 July 2017).



hyper-volume defined between that point and the Pareto set.

Let n_v (n_a) be the number of solutions in the Pareto set obtained by VNS (AUGMENCON2) algorithm. Let $S_v \in \mathbb{R}^4 \times \mathbb{R}^{n_v}$ ($S_a \in \mathbb{R}^4 \times \mathbb{R}^{n_a}$) be the matrix with four columns (linked to the four objectives) and n_v (n_a) rows mapped to all the solutions obtained by the VNS (AUGMENCON2) algorithm.

We denote by $x \in \mathbb{R}, x = (x_1, x_2, x_3, x_4)$ to the reference point to compute the hyper-volume, which is computed as follows.

$$x_i = 10 * \max\{\max_j S_v(i,j), \max_j S_a(i,j)\}, \quad \forall i \in \{1, \ldots, 4\} \tag{17}$$

In Table 2 we show the hyper-volume obtained for each instances with 20 orders and 30 orders. In this table we can observe that, on average, in instances with 20 orders the average of the VNS is 1.38% worse than the exact solution, and this difference increases to 3.05% in the instances with 30 orders. In all the cases, that difference is never higher than 5%. The CPU time required to compute the Pareto set is not comparable, since the AUGMECON2 algorithm required over 10 hours to compute the approximated Pareto set on the instances with 30 orders, and the VNS needs less than one minute for all the problems.

Table 2: Hypervolume AUGMEN2 vs VNS

	20 orders		30 orders	
	Exact	VNS	Exact	VNS
Test 1	89.46	87.93	92.90	89.27
Test 2	91.83	90.60	89.90	86.02
Test3	89.92	88.72	90.37	87.76
Test4	91.39	88.68	90.78	87.97
Test5	91.69	90.47	89.66	87.82
Test6	93.13	92.17	93.84	90.77
Test7	91.83	90.62	91.18	88.57
Test8	93.06	92.13	93.61	90.60
Test9	88.70	86.96	93.40	88.44
Test10	90.67	89.59	92.19	90.12
Average	91.17	89.79	91.78	88.73

In Tables 3 and 4 we show the specific values obtained for each instance in the problem with a lexicographic approach for all the problems with 20 and 30 orders. These solutions are obtained from the Pareto set obtained by each method, and the last column shows the number of solutions obtained in the Pareto set. If we take into account the average, the difference starts with the second objective, with a difference of 0.01 in Table 3 and a difference of 0.07 in Table 4. In 7 out of 10 instances with 20 orders the best lexicographic solution in the Pareto set is the same, and the same solutions are obtained in 5 out of 10 instances with 30 orders. It is also worth highlighting that the solution obtained by the VNS algorithm in the problem Test8 has a slightly better performance than the one obtained by the AUGMECON2 algorithm (second objective 3.335 vs 3.367 obtained by AUGMECON2). Finally, the size of the Pareto set obtained is generally higher and more erratic when applying the VNS, which is expected since it is an heuristic approach.

Table 3: Best lexicographic solutions on instances with 20-orders.

	AUGMEN2					VNS				
	obj1	obj2	obj3	obj4	#sol	obj1	obj2	obj3	obj4	#sol
Test 1	1	3.142	10.697	2.096	36	1	3.318	11.972	1.147	45
Test 2	1	3.171	14.098	1.388	35	1	3.171	14.098	1.388	115
Test3	1	4.114	10.812	2.308	31	1	4.114	10.812	2.308	9
Test4	1	3.022	13.900	2.859	21	1	3.047	9.172	2.817	8
Test5	1	3.717	15.389	2.381	50	1	3.717	15.389	2.381	50
Test6	1	2.951	12.822	1.524	53	1	2.951	12.822	1.524	144
Test7	1	2.968	12.933	3.055	25	1	2.968	12.933	3.055	45
Test8	1	3.367	13.646	2.038	37	1	3.335	12.973	2.074	150
Test9	1	3.603	10.718	2.305	24	1	3.603	10.718	2.305	8
Test10	1	3.418	9.070	1.811	22	1	3.418	9.070	1.811	55
Average	1	3.35	12.41	2.18	33.4	1	3.36	11.99	2.08	62.9

6.3. VNS and exact algorithm (Lexicographic problem)

In this section we compare the VNS against the construction heuristic presented in 5.1 (BAU) and the ILP model presented in Section 4 using the problem instances generated in Section 6.1 with 40, 50 and 60 orders, where the relevance of the objectives is considered to be in lexicographic order, being objective 1 the most relevant one and objective 4 the least relevant. The main purpose of considering this problem is to be able to compare single solutions against the BAU solutions.

In Figures 4, 5 and 6 it can be observed that the total amount of travel within the orders (from pickup to delivery) is the same for the three methods, which is expected as it is a fixed value for each order. The travel time between orders (from delivery to next pickup) depends on how the allocation of orders is done, and it can be observed here that the BAU is the worst in this respect.

Table 4: Best lexicographic solutions on instances with 30-orders

	AUGMEN2					VNS				
	obj1	obj2	obj3	obj4	#sol	obj1	obj2	obj3	obj4	#sol
Test 1	0	5.453	17.830	2.566	53	0	5.561	21.399	2.277	90
Test 2	1	5.102	17.556	2.651	57	1	5.304	16.973	2.765	173
Test3	1	4.743	12.619	0.719	52	1	4.868	14.045	1.605	74
Test4	1	4.954	18.750	1.981	60	1	5.004	17.345	2.003	155
Test5	1	3.692	20.169	2.853	56	1	3.692	20.169	2.853	49
Test6	0	4.728	18.397	2.555	60	0	4.728	18.397	2.555	121
Test7	1	4.350	17.163	2.113	71	1	4.350	17.163	2.113	294
Test8	1	3.272	24.012	2.850	53	1	3.272	24.012	2.850	184
Test9	0	3.727	22.537	2.107	70	0	3.872	23.116	2.107	159
Test10	1	4.104	22.878	2.911	56	1	4.104	22.878	2.911	235
Average	0.70	4.41	19.19	2.33	58.80	0.70	4.48	19.55	2.40	153.40

The VNS managed to reduce the travel time between orders by approximately 25%, and the ILP by almost 50% compared to the BAU. The waiting time is lower in the BAU because there are couriers with only one order assigned in a shift in some of the instances, which implies no waiting time, however, the unused time is higher (time left after the last delivery until the end of the shift).

In Table 5 we present the results obtained by the construction heuristic presented in Section 5.1, named as BAU, the VNS heuristic and the ILP model. The first three columns show the average of the 10 problem instances of the minimum, mean and maximum waiting times of all the gig economy couriers. The average minimum waiting time (second column) of the BAU is only 1%, and the maximum waiting time (third column) is 61% of the couriers' time, which indicates that there is an unbalanced set of shifts being used in the solution. The average waiting time between orders (second column) is lower in the BAU because there is a gig economy courier with only one order assigned with zero waiting time, which skews the mean.

The last three columns in Table 5 are related to the average of the minimum, average and maximum number of orders assigned to the couriers. We can observe that the VNS finds the most balanced solution, matching the ILP. The range of the orders assigned per courier ranges between 2.8 and 3.8. However, the BAU is rather unbalanced, with couriers doing 6 orders and other couriers with only one order assigned. Similar behaviour can be seen in Table 6 and Table 7, where the instances with 50 and 60 orders are considered. The results presented in Table 7 consider only the 9 out of 10 instances in which the ILP managed to compute a feasible solution within two hours of CPU time (see Table 9 in the appendix), and four instances out of ten were optimally solved. For the 5 remaining instances where GUROBI did not prove optimality, the average gap obtained was 61.6%.

Table 5: Exploring fairness on instances with 40 orders

	minwait	avwait	maxwait	minorders	avorders	maxorders
BAU	1%	27%	61%	1.1	3.25	6.1
VNS	29%	59%	78%	2.8	3.25	3.8
ILP	27%	56%	78%	2.8	3.25	3.8

Table 6: Fairness comparison on instances with 50 orders

	minwait	avwait	maxwait	minorders	avorders	maxorders
BAU	0%	28%	67%	1	3.3	6.5
VNS	20%	55%	81%	2.8	3.3	3.8
ILP	22%	53%	79%	2.8	3.3	3.8

Table 7: Fairness comparison on instances with 60 orders

	minwait	avwait	maxwait	minorders	avorders	maxorders
BAU	0%	28%	71%	1	3.29	6.89
VNS	24%	60%	81%	2.89	3.29	3.78
ILP	24%	56%	81%	2.89	3.29	3.78

Figure 4: Fairness comparison on instances with 40 orders (Average number of couriers used = 12.4)

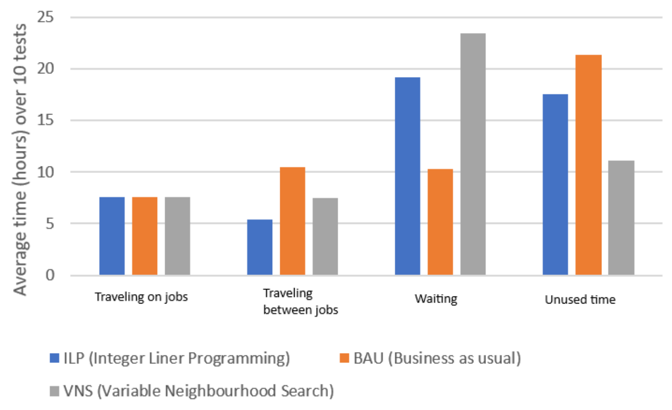


Figure 5: Fairness comparison on instances with 50 orders (Average number of couriers used = 15.3)

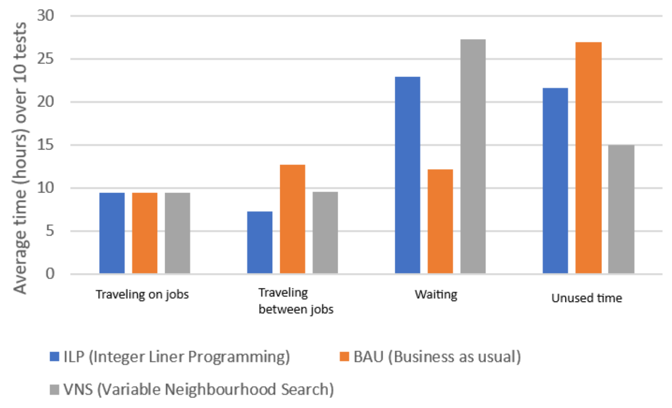
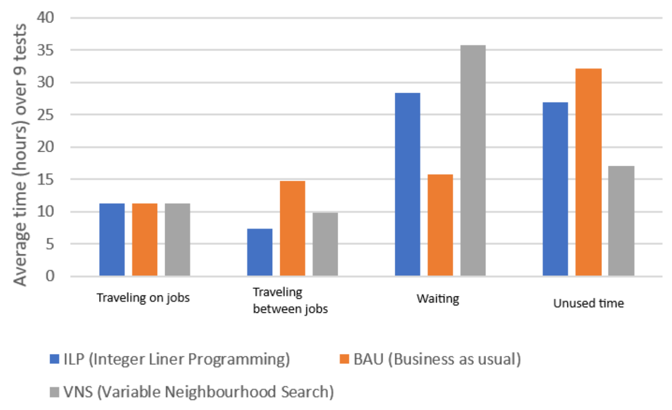


Figure 6: Fairness comparison on instances with 60 orders (Average number of couriers used = 18.4)



6.4. Previous work in MDP data

In this section we present results from using the 240 problem instances generated by Reyes et al. (2018), available at `https://github.com/grubhub/mdrplib`. The number of orders in these instances ranged from 242 to 3212 and the number of restaurants ranged from 54 to 323. While in Reyes et al. (2018) the number of couriers and the total number of working hours was assumed known and fixed and the orders might not be delivered on time, in this work we assumed that all the orders must be delivered as soon as ready. To ensure feasibility we permitted as many extra couriers as needed, but always minimising the total number of couriers used in the final plan. In Reyes et al. (2018), different orders from the same restaurant could be bundled together and picked up by the same courier in one visit. However, in our case study we assumed that each order had to be picked up and delivered separately. For the sake of clarity, service times at all restaurants and at all delivery locations were not considered, although could easily be added if known. The target in this study is to minimise the objective functions in lexicographic order, which is aligned with minimising (5) with weights such that $w_1 >> w_2 >> w_3 >> w_4$. It is worth highlighting again that we are addressing the deterministic version of the problem where all the information about orders is known beforehand.

Table 8: Solutions obtained in the MDP instances

Set	$WT(\%)$	$WTmax(hours)$	$WTmin$(hours)	range(%)	$DD(\%)$	$DDtotal(\%)$	$DifmaxD(hours)$	$DifminD(hours)$
0	-67.09	0.08	1.24	69.37	41.60	21.47	-1.52	0.52
1	-94.98	-0.28	1.31	68.88	39.72	20.70	-1.30	0.57
2	-73.62	-0.26	1.38	74.29	40.40	20.44	-1.33	0.74
3	-80.73	-0.11	1.48	71.41	37.56	19.01	-1.17	0.84
4	-108.65	0.23	1.28	82.12	36.40	17.34	-1.09	0.96
5	-75.14	0.41	0.76	67.71	30.01	14.63	-0.58	0.87
6	-87.55	0.12	1.18	74.31	35.25	17.45	-0.99	0.87
7	-99.89	-0.50	0.53	63.67	25.02	11.97	-0.35	1.10
8	-67.98	-0.09	0.48	34.77	28.37	14.30	-0.36	0.56
9	-125.20	-0.88	0.59	50.39	30.03	15.41	-0.43	0.66
Av.	-88.08	-0.13	1.02	65.69	34.44	17.27	-0.91	0.77

In Table 8 we present the summary of the results obtained for the 240 MDP instances (10 sets with 24 instances). The first column, WT (%), shows the average reduction in the total waiting time when comparing the BAU (construction heuristic presented in 5.1) solution against the VNS algorithm presented in 5.3, computed as

$$\frac{W_{VNS} - W_{BAU}}{W_{VNS}} * 100,$$

where W_{VNS} and W_{BAU} represent the total waiting time of all the couriers in the VNS and BAU solutions respectively. It can be seen that the overall average reduction in waiting time is 88.08%.

The second and third columns show the time reduction for the courier with maximum and minimum waiting times, respectively. If we represent by C_{BAU} the set of all couriers in any given instance and by w_c the waiting time of courier $c \in C_{BAU}$, then

$$WTmax = \max_{c \in C_{VNS}} w_c - \max_{c \in C_{BAU}} w_c.$$

Similarly,

$$WTmin = \min_{c \in C_{VNS}} w_c - \min_{c \in C_{BAU}} w_c.$$

Therefore, the courier with more waiting time from the VNS solution versus the BAU solution is reduced by 0.13 hours on average, but in sets 1,4,5 and 6 the maximum waiting time is slightly higher in the VNS solution. On the other hand, the courier with a shift with less waiting time in the VNS solutions has, on average, 1.02 hours more waiting ($WTmin$). Both these results suggest that the waiting times in the VNS solution are more balanced than the BAU across the couriers.

The fourth column (range) shows the percentage of reduction of the range (maximum difference of the number of orders assigned to any two couriers). We can observe that the average reduction is 65.69%, which is significantly better but it is slightly lower than in the previous section (see Tables 5,6 and 7), suggesting that this problem with higher numbers of orders becomes more challenging to be solved efficiently.

The fifth column (DD) shows the percentage travel time difference between the VNS and BAU solutions for travel between orders (from delivery point of one order to pickup point of the next), and in column $DDtotal$ we present the percentage difference of the total travel time in the solution (i.e. travel during and between orders). The VNS solution reduced the travel time between orders by 34.44% on average and the total travel time by around half of that, which is expected as travel during orders was constant and approximately equal to the amount of travel between orders.

Column $DifmaxD$ ($DifminD$) shows the mean difference between the courier with maximum (minimum) travel times from the VNS and the BAU solutions. These are defined as follows.

$$DifmaxD = \max_{c \in C_{VNS}} TT_c - \max_{c \in C_{BAU}} TT_c$$

and

$$DifminD = \min_{c \in C_{VNS}} TT_c - \min_{c \in C_{BAU}} TT_c,$$

where TT_c represents the total travelling time o courier c.

The results indicate that the 'most travelled' courier travels 0.91 hours less, on average, in the VNS solution and the 'least travelled' courier travels 0.77 hours more, both indicating better balancing of travel between couriers.

These results shows that fairness can be achieved alongside efficiency: as well as balancing the work the total distance travelled by all the couriers in the plan obtained by the VNS is on average 17.27% lower than the BAU plan.

7. Conclusions

This paper proposes a new mathematical model to address the meal delivery problem where it is assumed that orders must be picked up and delivered as soon as they are ready. The model takes into account the efficiency of the routes as well as the fairness among the couriers, introducing four objectives into the problem, where different modes of transport might be used. We show that the model is able to optimally solve a reasonable size of problem instances, up to 30 orders to compute the full Pareto set, and up to 60 orders in a reasonable amount of time for the single objective problem (if some weights are specified by the user). For larger problems we propose a VNS algorithm with simple and efficient neighbourhoods in which fitness used changes during the search. We show that it is possible to produce fair solutions from the courier perspective as well as reduce the travel times, making an efficient use of courier time and producing more robust shifts.

The potential application for new solutions to the meal delivery problem, or similar pickup and delivery tasks, is great, given the considerable demand for home delivery, accelerated by behavioural changes resulting from the Covid-19 pandemic. Coupled to this are the increasing restrictions resulting from the imposition of low emissions zones and schemes to promote walking and cycling which limit access for motorised vehicles. Under such circumstances, logistics providers are increasingly looking for environmentally friendly ways of meeting consignee delivery requirements and the use of sustainable last-mile delivery systems are being trialled by several carriers. The use of cycle and scooter couriers through last-mile delivery platforms such as Deliveroo and UberEats offer alternative ways of making deliveries in dense urban centres. In most cities there are pools of riders available, largely serving the takeaway restaurant market, that could provide an additional delivery fleet. Key questions have been raised however regarding the efficacy of such operations and how riders can gain an equitable supply of work.

The results obtained from the ILP model show that existing policies and practice, as considered here as the BAU case, can be improved in terms of both fairness and route efficiency. In addition, the VNS has been shown to be competitive in the small-scale problems solved by the ILP and, on the larger instances, provides a considerable improvement over the plans derived from the BAU strategies.

References

Arslan, A., Agatz, N., Kroon, L., & Zuidwijk, R. (2018). Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers. *Transportation Science*, (pp. 1–318).

Aziez, I., Côté, J.-F., & Coelho, L. (2020). Exact algorithms for the multi-pickup and delivery problem with time windows. *European Journal of Operational Research*, (pp. 906–919).

Baldacci, R., Bartolini, E., & Mingozzi, A. (2011). An exact algorithm for the pickup and delivery problem with time windows. *Operations Research*, *59*, 414–426. doi:10.1287/opre.1100.0881.

Bektaş, T., Gouveia, L., Martínez-Sykora, A., & Salazar-González, J.-J. (2019). Balanced vehicle routing : Polyhedral analysis and branch-and-cut algorithm. *European Journal of Operations Research*, *273*, 452–463. doi:10.1016/j.ejor.2018.08.034.

Bektaş, T., & Letchford, A. N. (2020). Using lp-norms for fairness in combinatorial optimisation. *Computers and Operations Research*, *120*, 1–24. doi:10.1016/j.cor.2020.104975.

Broughton, A., Gloster, R., Marvell, R., Green, M., Langley, J., & Martin, A. (2018). The experiences of individuals in the gig economy. *Department for Business, Energy & Industrial Strategy (BEIS)*, (p. 108). URL: https://www.gov.uk/government/publications/gig-economy-research.

Cant, C. (2020). *Riding for Deliveroo - Resistance in the New Economy*. Cambridge, UK: Polity Press.

Chartered Institute of Environmental Health (2020). Food delivery and takeaway market in the UK. *Statista*, . URL: https://www-statista-com.proxy.lib.strath.ac.uk/study/36742/takeaways-in-the-united-kingdom-uk-statista-dossier/.

Christie, N., & Ward, H. (2018). The emerging issues for management of occupational road risk in a changing economy: a survey of gig economy drivers, riders and their managers. *London: UCL Centre for Transport Studies*, .

Cosmi, M., Oriolo, G., Piccialli, V., & Ventura, P. (2019). Single courier single restaurant meal delivery (without routing). *Operations Research Letters*, *47*, 537–541. URL: https://doi.org/10.1016/j.orl.2019.09.007. doi:10.1016/j.orl.2019.09.007.

Dablanc, L., Morganti, E., Arvidsson, N., Woxenius, J., Browne, M., & Saidi, N. (2017). The rise of on-demand 'Instant Deliveries' in European cities. *Supply Chain Forum*, *18*, 203–217. URL: https://doi.org/10.1080/16258312.2017.1375375. doi:10.1080/16258312.2017.1375375.

Dahle, L., Andersson, H., Christiansen, M., & Speranza, M. G. (2019). The pickup and delivery problem with time windows and occasional drivers. *Computers and Operations Research*, *109*, 122–133. URL: https://doi.org/10.1016/j.cor.2019.04.023. doi:10.1016/j.cor.2019.04.023.

Field, F., & Forsey, A. (2018). Delivering justice? A report on the pay and working conditions of Deliveroo riders. *Politeia*, (pp. 1–23).

Gouveia, L., & Salazar-González, J. (2010). On the Vehicle Routing Problem with lower bound capacities. *Electronic Notes in Discrete Mathematics*, *36*, 1001–1008. doi:10.1016/j.endm.2010.05.127.

Hansen, P., Mladenović, N., & Moreno Perez, J. A. (2008). Variable neighbourhood search: methods and applications. *4OR*, *6*, 319–360.

Koç, C., Laporte, G., & Tükenmez, I. (2020). A review of vehicle routing with simultaneous pickup and delivery. *Computers and Operations Research*, (pp. 104987–105001).

Kytöjoki, J., Nuortio, T., Bräysy, O., & Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers and Operations Research*, *34*, 2743–2757. doi:10.1016/j.cor.2005.10.010.

Mavrotas, G. (2009). Effective implementation of the ϵ-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, *213*, 455–465. doi:10.1016/j.amc.2009.03.037.

Mavrotas, G., & Florios, K. (2013). An improved version of the augmented e-constraint method (augmecon2) for finding the exact pareto set in multi-objective integer programming problems. *Applied Mathematics and Computation*, *219*, 9652–9669. doi:10.1016/j.amc.2013.03.002.

Nolan, P. (2018). Good Work: The Taylor Review of Modern Working Practices. *Industrial Relations Journal*, *49*, 400–402. doi:10.1111/irj.12239.

Reyes, D., Erera, A. L., Savelsbergh, M. W. P., Sahasrabudhe, S., & O 'Neil, R. J. (2018). The Meal Delivery Routing Problem. *Optimization Online*, (pp. 1–70). URL: http://www.optimization-online.org/DB{_}FILE/2018/04/6571.pdf.

Sze, J. F., Salhi, S., & Wassan, N. (2017). The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: An effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search. *Transportation Research Part B: Methodological*, *101*, 162–184. doi:10.1016/j.trb.2017.04.003.

Vidal, T., Laporte, G., & Matl, P. (2020). A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research*, (pp. 401–416).

Yildiz, B., & Savelsbergh, M. (2019). Provably high-quality solutions for the meal delivery routing problem. *Transportation Science*, (pp. 1213–1499).

Zheng, J., Wang, L., Wang, L., Wang, S., Chen, J.-F., & Wang, X. (2022). Solving stochastic online food delivery problem via iterated greedy algorithm with decomposition-based strategy. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, (pp. 1–13). doi:10.1109/TSMC.2022.3189771.

APPENDIX

Appendix

Table 9: Solutions obtained by the ILP on the instances with 60 orders from set *A*. *Avw* is the average working time of all the couriers with some orders assigned in the solution and *Avt* is the average travel time. The range of the number of orders assigned is in column *range* and the last columns show the gap obtained and total CPU time in seconds, with a time limit of 2 hours.

Instance	*Avw*	*Avt*	range	Gap	CPUtime(sec)
1	3.71	2.84	1	0.56	7200
2	3.39	2.27	1	0.00	5138
3	3.30	2.69	1	0.00	5084
4	3.60	2.52	1	0.54	7200
5	-	-	-	-	7200
6	3.26	2.38	1	0.97	7200
7	3.24	2.04	1	0.48	7200
8	3.45	2.31	1	0.53	7200
9	3.36	2.52	0	0.00	768
10	3.49	2.52	1	0.00	4712

Table 10: Comparison of construction heuristics

#orders	Heur	Av Waiting time	range orders	Total travel
40	1	0.83	5	17.97
	2	0.84	4.8	18.47
	3	0.98	4.6	16.08
50	1	0.79	5.5	22.11
	2	0.78	5.4	22.26
	3	0.92	5	19.94
60	1	0.86	5.9	26.06
	2	0.91	5.6	26.29
	3	1.06	5	22.55

Antonio Martinez

https://orcid.org/0000-0002-2435-3113

Other IDs

ResearcherID: N-9895-2013 (http://www.researcherid.com/rid/N-9895-2013)

Scopus Author ID: 56449279900 (http://www.scopus.com/authid/detail.url?authorId=56449279900)

Employment (2)

University of Southampton: Southampton, GB
Employment

Source: University of Southampton

Dr. Antonio Martinez-Sykora: Southampton, Hamsphire, GB
2016-07-01 to present | Lecturer in Business Analytics (Southampton Business School)
Employment

Source: Antonio Martinez

Funding (1)

Dynamic Pricing in the Ferry Industry
Engineering and Physical Sciences Research Council (Swindon)
2015-12-14 to 2018-06-12|Grant
GRANT_NUMBER: EP/N006461/1
URL: https://grants.uberresearch.com/501100000266/3A4F15AA-93E4-4C9F-8333-2BAD57DC9B3A/Dynamic-Pricing-in-the-Ferry-Industry (https://grants.uberresearch.com/501100000266/3A4F15AA-93E4-4C9F-8333-2BAD57DC9B3A/Dynamic-Pricing-in-the-Ferry-Industry)

Source: Antonio Martinez*via*DimensionsWizard

Works (23 of 23)

Highlights

```
- In this work we highlight the need for creating fairer shifts for Gig
economy workers
- We explore a new model to address the fairness in the meal delivery
problem
- We develop some simple and effective heuristics to solve large scale
real-based problems
- Methods are compared against state-of-the-art epsilon-constraint
algorithms in small-scale problems.
- Results are compared against the business-as-usual policies on
instances with up to 3123 orders.
```

Antonio Martinez-Sykora: Conceptualization, Methodology, Data curation, Software **Fraser McLeod.**: Conceptualization, Methodology, Data curation, Software **. Thomas Cherrett**: Conceptualization, Methodology. ***Adrian Friday**: Conceptualization, Methodology*

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: