Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 00 (2023) 000–000

Transportation Research Procedia

www.elsevier.com/locate/procedia

World Conference on Transport Research - WCTR 2023 Montreal 17-21 July 2023

Fairness and Efficiency in Social Vehicle Routing Problems

Martin Aleksandrov

Freie Universität Berlin

martin.aleksandrov@fu-berlin.de

Abstract

We define Social Vehicle Routing Problems (SVRPs), where preferences of drivers and vehicles, feasibility constraints between vehicles and requests, and network metrics, inform the management of the fleet. We propose new algorithms for SVRPs, first returning a feasible matching between drivers and customers and then an optimizing feasible plan for routing the vehicles through their matched locations. We give matching algorithms for achieving fairness (i.e. FEF1, FEQX, FEFX) for drivers, efficiency (i.e. $\text{FSW}_{\max}$) for drivers, and fairness and efficiency (i.e. $\text{FSW}_{\min}$) for customers. Finally, we also give fixed-parameter tractable routing algorithms for fleet fairness (i.e. maxTRAVEL) and fleet efficiency (i.e. totTRAVEL).

Keywords: Vehicle Routing Problems; Computational Social Choice; Fairness; Efficiency

1. Introduction

Let us consider the class of Vehicle Routing Problems (VRPs) [16], where a fleet of vehicles services a set of visit requests. A generalization considers a set of pickup-and-delivery requests [37]. We initiate a study of social aspects in VRPs. For this purpose, we propose a model for *Social VRPs (SVRPs)*. We show it in Figure 1. In our SVRPs model, *preferences* of drivers and customers can play a crucial role in fleet management. For example, let us suppose that you want to arrive at the airport in the next 20 minutes and I want to arrive at the airport in the next 15 minutes. Therefore, a taxi driver would have to decide who of us to transport first and who of us to transport second, which would affect their overall routing. Thus, we let SVRPs intersect *Computational Social Choice (COMSOC)*. In our SVRPs model, *feasibility* constraints between vehicles and requests also play an important role. For example, these could be expressed via capacities constraints, dimension constraints, etc. Thus, we let SVRPs intersect *Constraint Satisfaction Problems (CSPs)*. In our SVRPs model, *network*-dependent travel times take an integral part. For example, cars take faster lanes on roads than trucks, or cars are slower inside the city center than outside it. Thus, we let SVRPs intersect *Geographic Information Systems (GISs)*.

* Corresponding author. Tel.: +49-30-838-75101.
E-mail address: martin.aleksandrov@fu-berlin.de

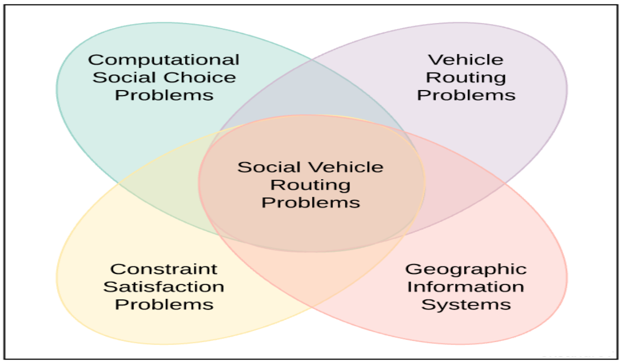

Fig. 1: Social Vehicle Routing Problems.

From a research perspective, SVRPs are multidisciplinary in nature. COMSOC problems can be from fair division, voting theory, or judgment aggregation [9]. CSPs can be from planning, loading, or searching domains [35]. GIS problems can be from navigation, mapping, or sensing domains [39]. We have therefore every confidence that this new domain lays down the blueprints of much further research in the future. However, as a first step, we draw inspiration from VRP, fair division, planning, and navigation.

From a practical perspective, SVRPs capture social preferences in emerging applications such as multi-modal intelligent transportation systems, connected and autonomous vehicles, and data-driven logistics. This aligns with the public mobility Transport Policy Flagships of the 2020 EU Strategy for Sustainable and Smart Mobility, according to which the transition to future personal mobility *must* involve the preferences of individuals. In Europe, there are several initial steps in this direction: in 2016, the German Ministry for Traffic and Digital Infrastructure granted 100 million Euros for autonomous and connected driving; in 2017, the German Ministry appointed Ethics Commission to regulate the use of such vehicles for social good; in 2020, the European Institute of Innovation and Technology in Hungary received 400 million Euros from EU for encouraging people to use electric vehicles more frequently in order to make our cities more livable places. Common objectives in these initiatives are achieving trust, social acceptance, and inclusiveness. These require that vehicles are used fairly and efficiently. In this paper, we provide an early qualitative analysis of fairness and efficiency for drivers and customers.

We study these in combination with two common *objectives* for VRPs such as the respective minimizations of the total time traveled by all vehicles (totTRAVEL) and the maximum time traveled by any vehicle (maxTRAVEL). However, in practice, drivers and customers usually have preferences over each other, and emerging VRPs demand (1) achieving fairness and efficiency for drivers and customers much more often than (2) minimizing centralized objectives such as totTRAVEL and maxTRAVEL. In fact, (1) and (2) could be incompatible in general. We next illustrate it.

Motivating example: Let us consider the straight line between 0 and 1. Also, let there be vehicle v_1 at 0, vehicle v_2 at 1, visit request r_1 at ϵ, and visit request r_2 at $2 \cdot \epsilon$. Suppose that each driver charges a cost of 1$ per visit. Minimizing totTRAVEL/maxTRAVEL dispatches v_1 to ϵ and $2 \cdot \epsilon$, and gives a value of $2 \cdot \epsilon$ which is optimal for $\epsilon < 1/4$. The outcome induces matching where the driver of v_1 receives a profit of 2$ and the driver of v_2 receives a profit of 0$. Although this matching is driver efficient, it is not driver fair because the driver of v_2 feels disadvantaged. By comparison, dispatching v_1 to ϵ and v_2 to $2 \cdot \epsilon$ gives a value for totTRAVEL/maxTRAVEL of at least $(1 - 2 \cdot \epsilon)$ which is sub-optimal because it is strictly greater than $2 \cdot \epsilon$ for $\epsilon < 1/4$. But, although this outcome is sub-optimal, it induces matching r_1 to v_1 and r_2 to v_2, where each driver gets a profit of 1$. This is driver *fair* and driver *efficient*. □

We conclude that achieving fairness for drivers requires us to solve the matching sub-problem *before* the routing sub-problem. We use this approach in the paper. By comparison, as we just observed in the motivating example, solving the routing sub-problem before the matching sub-problem may induce zero profits for some drivers, and, as a result, such drivers may decide not to participate because they might feel excluded. From this perspective, we view achieving fairness for drivers as an integral part of driving policies promoting participation and inclusiveness by making sure that drivers receive similar profits. We believe that this will improve the social acceptance of our approach. Indeed, as we will show in the paper, solving the matching sub-problem before the routing sub-problem allows us to guarantee fairness or efficiency in every instance of our model.

Template 1 A Match-1st-Route-2nd algorithm.

Input: V, R, P, F, N, O

Output: feasible matching $\mathcal{R}$, optimising feasible plan $\mathcal{P}$

1: Given V, R, P, and F, compute $\mathcal{R}$.
2: Given V, $\mathcal{R}$, N, and O, compute $\mathcal{P}$.
3: **return** $(\mathcal{R}, \mathcal{P})$

In this paper, we look at instances, where fleet V of n *vehicles* and set R of m *requests* are available in a fixed time interval, when we consider preferences P, feasibilities F, network N, and objective O. As we discussed previously, we propose a new family of algorithms (Template 1), each returning first some *feasible matching* between drivers and customers, by accounting for P and being subject to F, and then an *optimizing feasible plan* for routing the vehicles through their matched locations, by optimizing O on top of N.

Outline: We explain our contributions in Section 2 and review related literature in Section 3. We define formally the components of Template 1 in Section 4. In our model, drivers charge customers some costs for servicing their requests. We thus consider *driver-dependent costs* (i.e. request costs depend on drivers) and *driver-independent costs*. We also consider *additive profits* for drivers (i.e. the profit for some requests is the sum of the individual request costs). Additivity is a common assumption in fair division theory and fair public decision-making [13, 14, 28]. In Section 5, we give matching algorithms for achieving fairness and efficiency for drivers and customers. In Section 6, we give routing algorithms for minimizing totTRAVEL/maxTRAVEL in various practical settings. Finally, in Section 7, we give a summary and a few future directions.

2. Contributions

The matching sub-problem: In our assignment model, we let drivers (pre-)submit to the (central) planner some but possibly not all of their profit preferences and vehicle feasibilities. For example, a courier company such as Bonds Express contracts on-demand drivers, and the dispatchers often do not know all of their preferences or all of their vehicle feasibilities for requests [2]. If the planner has complete such information, then the problem is purely *centralized*. If they have no such information at all, then the problem is purely *decentralized*. Otherwise, the planner needs to decentralize some of the assignment decisions. This decentralization could be challenging simply because we often do not know how drivers might behave in practice. At this point, our model intersects behavioral game theory [11], where a key concept is the one of *rational* behavior.

In our *semi-decentralized* setting, whenever some profit preferences or vehicle feasibilities are unknown to the planner, they send some (at most m) requests to drivers, and drivers may respond with such information within some pre-specified time $T \in \Omega(m)$. In our work, we assume that if drivers respond with a request they receive then their behavior is *truthful* and *profit-maximizing*: supposing that their vehicles are truthfully feasible for at least one request they receive, drivers reveal profit information for the most profitable request among the requests they receive for which their vehicles are truthfully feasible, without changing any (*public*) preferences and feasibilities, which are known to the planner, and accounting for any (*private*) preferences and feasibilities, which are known just to them. Otherwise, we assume that drivers are *unresponsive* and cannot service requests until the next time they respond. Drivers can be unresponsive for various reasons: they depart from and arrive at the market at different times; they have made a sufficient profit in the current day and decide to go home earlier; their vehicles are truthfully infeasible.

In this setting, we first study achieving *fairness* for drivers. In centralised fair division [8], three fairness notions are "envy-freeness up to some good" (EF1) [10], "envy-freeness up to any good" (EFX) [12], and "equitability up to any good" (EQX) [20]. However, the definitions of EF1, EFX, and EQX, ignore the feasibility constraints in our setting (see Appendix A) and the semi-decentralized nature of our setting. This motivated us to define firstly new *feasible* versions of these properties: *FEF1*, *FEFX*, and *FEQX*. For any two drivers, FEF1 and FEFX assignments bound the absolute *envy* [19] for their feasible requests. For any two drivers, FEQX assignments bound the absolute *jealousy* [25] for their feasible requests. Thus, we first investigate achieving FEF1, FEFX, and FEQX. For simplicity, we say sometimes that an assignment is driver *fair* if and only if it satisfies FEF1, FEFX, or FEQX.

Then, we look at algorithms that assign requests only to responsive drivers. However, there are instances where *no* such algorithm can ever return driver fair assignments simply because some of the drivers may be unresponsive. We will give such instances. In response, we propose a *responsive version* of driver fairness, accounting for the fact that drivers can have envy/jealousy for an assigned request only if they are not unresponsive when the request is assigned. Thus, we give a polynomial-time algorithm (Algorithm 1) for returning feasible and responsive FEF1 assignments: Theorem 1 and Corollary 1. We also give a polynomial-time algorithm (Algorithm 2) for returning feasible and responsive FEQX assignments: Theorem 2 and Corollary 2. For the class P of polynomial-time solvable decision problems, we refer the reader to [22].

We continue with *efficiency* for drivers. In centralized fair division, Aziz et al. [5] investigated the complexity of computing social welfare-maximizing EF1 assignments. Similarly, in our setting, we investigate the complexity of computing an assignment, whose *feasible* social welfare (i.e. the total sum of drivers' profits within the set of feasible assignments), labeled as *FSW*, admits the maximum possible value $\text{FSW}_{\max}$. We show that this is tractable. However, assignments that are fair and give $\text{FSW}_{\max}$ may not always exist, and deciding whether such assignments give a value for FSW, which is bounded from below by $(\text{FSW}_{\max} - t)$ for some $t \in \mathbb{R}_{\geq 0}$, is NP-hard: Theorem 3. For the class NP-hard of exponential-time solvable decision problems, we refer the reader to [22].

We end with *fairness* and *efficiency* for customers. As opposed to drivers who aim at maximizing their profits, customers aim at minimizing their costs for services whenever their requests are serviced by feasible vehicles, if any. Thus, in our setting, we also investigate the complexity of computing an assignment, whose total sum of customer costs, which is also equal to the FSW, admits the minimum possible value $\text{FSW}_{\min}$ within the set of feasible assignments where drivers service as many requests as possible. We show that this takes polynomial time, but assignments that are fair and give $\text{FSW}_{\min}$ may not always exist, and deciding whether such assignments give a value for FSW, which is bounded from above by $(\text{FSW}_{\min} + t)$ for some $t \in \mathbb{R}_{\geq 0}$, is also NP-hard: Theorem 4. We summarise our matching results in Table 1.

Table 1: Key: ✓-"exist", ×-"may not exist", fair=FEF1/FEFX/FEQX.

Template 1: axiomatic results for additive profits	
	feasible &
P (Alg 1)	✓, responsive FEF1 (Thm 1 and Cor 1)
P (Alg 2)	✓, responsive FEQX (Thm 2 and Cor 2)
general complexity results for additive profits	
NP-hard	×, responsive fair & $\text{FSW}_{\max}$ (by Thm 3)
NP-hard	×, responsive fair & $\text{FSW}_{\min}$ (by Thm 4)

Our setting can be simulated on various *Internet* and *mobile* platforms. For example, in the Uber app, if drivers respond then they are available and, otherwise, they are assumed to be busy until the next time they are prompted for responses. Also, in the dispatching unit of Bonds Express, the dispatchers send request information via SMS messages to drivers and drivers respond to these whenever they become available. We believe that our concepts for responsive fairness and efficiency can promote participation on such platforms.

The routing sub-problem: Let us pick a number $f \in \mathbb{N}_{\geq 0}$. In static SVRPs, all requests are known and serviced one time. In such problems, the requests occur normally in clusters (e.g. in geographic areas), and, for this purpose, the planner shares the vehicles among the clusters. For example, they may wish to guarantee that each cluster vehicle is feasible for $O(f)$ cluster requests [26]. In repeated SVRPs, all requests are known and serviced multiple times. In such problems, suppose that each vehicle is feasible for $O(f)$ requests at each repetition. For example, waste collection induces a repeated vehicle routing problem where garbage collectors have profit preferences over the number of collection requests [6]. For such problem variants, we consider a *parameter* such as the maximum number f of feasible requests per vehicle. We prove that minimizing optimally totTRAVEL and maxTRAVEL with a "black-box" algorithm by using naive brute force may take exponential time, even when f admits a constant value: Theorem 5. We also prove that minimizing these objectives with Template 1 by using naive brute force takes polynomial time for any constant value of f: Theorem 6. In computer science terms, this means that Template 1 is *fixed-parameter tractable* (*fpt*) [17] with respect to f. As a consequence, we might prefer using Template 1 to the naive "black-box" algorithm for the routing sub-problem. We summarise our routing results in Table 2.

Table 2: Key: n vehicles, m requests, deadline T, f feasibilities.

"black-box" algorithms: computational results	
brute force	$O(2^{nf}n(2f)!)$ (Thm 5)
Template 1: computational results	
fpt (Alg 1)	$O(nm\max\{nm, T\}(2f)!)$ (Thm 1 & Thm 6)
fpt (Alg 2)	$O(nm\max\{nm, T\}(2f)!)$ (Thm 2 & Thm 6)

3. Related work

The social vehicle routing problem generalizes the fair division problem from [8] by adding feasibility constraints between agents and items, and the vehicle routing problem from [16] by adding preferences of drivers over requests. Alternatively, we can view satisfying the feasibility constraints as satisfying a second layer of preferences of drivers for requests. Multi-layer preferences were recently announced to broaden the research agenda of COMSOC [7]. Unlike existing works for EF1, EQX, and EFX, we also consider fairness and efficiency in semi-decentralized settings that intersect behavioral game theory. This opens up a future focus on settings, where drivers have bounded rationality [11]. For VRPs, Santos and Xavier [36] studied minimizing shared taxi costs among customers. Li et al. [29] considered a similar setting with people and parcels. Rheingans-Yoo et al. [34] analysed matchings with driver location (not profit) preferences. Ma et al. [30] modeled spatiotemporal settings with a focus on drivers. Xu and Xu [40] investigated trading the system efficiency for the income equality of drivers. Each of these works focuses on drivers *or* customers. By comparison, we study fairness and efficiency for both drivers *and* customers. For fair division, Dror, Feldman, and Segal-Halevi [18] considered a model, where agents have categories and upper quotas. They study F-EF1 assignments where an agent may get *any* subset of goods within each category, subject to the corresponding category quota. We prove that our notion of feasibility and FEF1 is new and stronger than F-EF1 (see Appendix B). Additionally, we are not aware of any work that considers notions similar to FEQX, FEFX, $\text{FSW}_{\max}$, and $\text{FSW}_{\min}$. For this reason, we believe that they are also new. In the settings from [23] and [27], each agent has fixed arrival and departure times. In our setting, drivers may not know when they can start servicing requests or when they will finish servicing requests and, for this reason, we do not assume to have additional information about their arrival and departure times.

4. Our fair division routing model for SVRPs

Vehicles: We let $V = \{v_1, \dots, v_n\}$ denote the driver *vehicles*, where each v_i has start/finish location $s_i \in \mathbb{R}^2/f_i \in \mathbb{R}^2$ and capacity $q_i \in \mathbb{N}_{>0}$. The locations could respectively denote depot locations, or a request location submitted in the past and a request location predicted in the future.

Requests: We let $R = \{r_1, \dots, r_m\}$ denote the customer *requests*, where each $r_j = (p_j, d_j, m_j)$ has begin/end location $p_j \in \mathbb{R}^2/d_j \in \mathbb{R}^2$ and demand $m_j \in \mathbb{N}_{>0}$. If $p_j \neq d_j$, r_j requires pickup and delivery (e.g. courier services, taxi services). Otherwise, r_j requires a visit (e.g. home services).

Preferences: We consider *driver-dependent costs*. For servicing each r_j, we let the driver of each v_i charge some cost $c_{ij} \in \mathbb{R}_{>0}$. Common examples of settings with costs are taxis, shuttles, pickup, and home services. Common examples of costs are: c_{ij} may be proportional to the average time of v_i between the locations of r_j, based on past data; c_{ij} may be the price the customer of r_j pays for the service of v_i; c_{ij} may be the Shapley value of r_j for the business if serviced by v_i; c_{ij} may include not just the marginal profit of v_i for servicing r_j, but also extra expenditures such as road tolls, driver's wage, fuel costs, etc. We also consider *driver-independent costs*. That is, $c_{ij} = c_j$ for every v_i and every r_j. We suppose that drivers have *additive profits*. That is, for $S \subseteq R$, the profit of the driver of v_i is $c_i(S) = \sum_{r_j \in S} c_{ij}$. In semi-decentralized settings, each cost c_{ij} can be public or private. For this reason, we let $\tilde{c}_{ij} = c_{ij}$ if c_{ij} is public and $\tilde{c}_{ij} = unk$ if c_{ij} is private (i.e. it is *unk*nown to the planner). The *preferences* are $P = (\tilde{c}_{ij})_{n\times m}$.

Feasibilities: We suppose that there is a set of constraints C_{ij} for each v_i and each r_j. Thus, we can define a hard feasibility indicator f_{ij}: $f_{ij} = 1$ if all constraints in C_{ij} can be satisfied; $f_{ij} = 0$ otherwise. We suppose that the capacity feasibility constraint $[(m_j \leq q_i)?1; 0]$ belongs to C_{ij}. That is, if $q_i < m_j$, then $f_{ij} = 0$ holds. However, if $f_{ij} = 0$, then $m_j \leq q_i$ might hold but other constraints in C_{ij} could be violated. common constraints relate to package dimension, location reachability, and driver shift.

For instance, suppose that vehicle v_i is feasible only for packages that can be loaded inside its trunk, subject to maximizing the total number of packages. This is known as the loading problem and it is NP-hard in general [31]. In this context, we may set f_{ij} to 1 if r_j can be loaded in v_i in some solution, and else 0. For a few packages, this can be decided with a CSP solver such as MiniZinc [32, 38]. However, in our setting, we decentralize the feasibilities of v_i and leave driver i to decide whether they can load or not packages. In our model, they can make such decisions for people as well.

Thus, as for preferences, each f_{ij} can be public or private. We let $\tilde{f}_{ij} = f_{ij}$ if f_{ij} is public, and else $\tilde{f}_{ij} = unk$. The *feasibilities* are $F = (\tilde{f}_{ij})_{n\times m}$.

Network: The vehicle and request locations form network source L. We suppose that each vehicle has travel times between pairs of locations from L. For each v_i, we write $t(i, l, l') \in [0, \infty)$ for the shortest travel time between l and l'. Thus, D_i denotes the navigation matrix $[t(i, l, l')]_{|L|\times|L|}$. The value of each $t(i, l, l')$ can account for features such as traffic velocity and volume, road closures, road constructions, intersection delays, etc. This value can be computed by querying a GIS platform such as Google Maps. Thus, each vehicle matrix D_i defines a quasi-metric in real-time environments. The *network* is $N = (L, [D_1, \dots, D_n])$.

Feasible matchings: In practice, Bonds Express cannot service all requests within a single fixed time zone (i.e. interval) and, for this reason, schedule any remaining requests for the next time zone, even though their vehicles might be feasible for such requests [2]. As a result, the current matching between drivers and customers may not give all such requests to vehicles. Similarly, we consider such matchings.

More formally, matching is $\mathcal{R} = (R_1, \dots, R_n)$, where $R_i \subseteq R$ for each v_i and $R_i \cap R_j = \emptyset$ for each (v_i, v_j) such that $i \neq j$. We say that $(R_1, \dots, R_n)$ is *feasible* if, for each v_i and each $r_j \in R_i$, $f_{ij} > 0$ holds. Feasible matchings may be incomplete (i.e. some r_j s.t. $f_{ij} > 0$ holds for some v_i may not be matched), but they are non-wasteful (i.e. no r_j s.t. $f_{ij} = 0$ holds for each v_i is matched). Feasible matchings are *complete* if they are not incomplete.

Feasible plans: For fixed $v_i \in V$ with $R_i \subseteq R$, route $\rho_i = (l_1(i), \dots, l_{2|R_i|}(i))$ is a strict sequence of the locations of the requests from R_i. We associate each $l_s(i)$ in ρ_i with some $r_j \in R_i$ and weight $cap_s(i) = +m_j$ if $l_s(i) = p_j$ and $cap_s(i) = -m_j$ if $l_s(i) = d_j$. Thus, plan $\mathcal{P} = \{\rho_1, \dots, \rho_n\}$ is a set of routes. Plan $\{\rho_1, \dots, \rho_n\}$ induces some $R_i \subseteq R$ for each v_i. For each $r_j \in R_i$, we note that $p_j = l_{k_j}(i)$ and $d_j = l_{h_j}(i)$ hold for some $k_j, h_j \in \{1, \dots, 2|R_i|\}$. We consider five types of constraints for plans:

(a) *Matching* constraints insist that the requests cannot be split across multiple vehicles: $\forall r_j \in R, \forall v_i \in V : (r_j \in R_i \Rightarrow \forall v_k \in V, k \neq i : r_j \notin R_k)$.

(b) *Feasibility* constraints ensure that each vehicle services only feasible requests: $\forall v_i \in V, \forall r_j \in R_i : f_{ij} > 0$.

(c) *Non-wastefulness* constraints require that each request, which is feasible for no vehicle, is not serviced by any vehicle: $\forall r_j \in R : (\forall v_i \in V : f_{ij} = 0) \Rightarrow (r_j \notin \bigcup_{v_i \in V} R_i)$.

(d) *Ordering* constraints ask that the pickup of each request is executed before its corresponding delivery, even if the pickup and delivery locations coincide (i.e. visit requests): $\forall v_i \in V, \forall r_j \in R_i : k_j < h_j$.

(e) *Capacity* constraints enforce that the capacity of no vehicle can be exceeded while servicing the requests: $\forall v_i \in V, \forall l_s(i) \in \rho_i : \sum_{l=1:s} cap_l(i) \leq q_i$.

If (a-c) are satisfied, then $\{\rho_1, \dots, \rho_n\}$ induces feasible $(R_1, \dots, R_n)$. However, for plan feasibility, we require that (d-e) are satisfied as well. We say that $\{\rho_1, \dots, \rho_n\}$ is a *feasible plan* if (a-e) are satisfied. A feasible plan always exists. To return one such plan, we can proceed in two steps. Firstly, for each request, pick some vehicle that is feasible for it, if any. Secondly, for each vehicle, let it service the requests assigned to it one after another. We next formalize the objectives for the non-empty set of such plans.

Feasible objectives: totTRAVEL measures *fleet efficiency* and maxTRAVEL measures *fleet fairness* [4]. More formally, let us consider feasible $\{\rho_1, \dots, \rho_n\}$. The travel time of v_i in ρ_i is $T_i(\rho_i) := t(i, s_i, l_1(i)) + t(i, l_{2|R_i|}(i), f_i) + [\sum_{j=1:(2|R_i|-1)} t(i, l_j(i), l_{j+1}(i))]$. We next define the measures totTRAVEL $:= \min_{\{\rho_1,\dots,\rho_n\}:\text{ feasible}} \sum_{v_i \in V} T_i(\rho_i)$ and maxTRAVEL $:= \min_{\{\rho_1,\dots,\rho_n\}:\text{ feasible}} \max_{v_i \in V} T_i(\rho_i)$. Thus, $\{\rho_1, \dots, \rho_n\}$ is *optimal* for totTRAVEL/maxTRAVEL if it gives the minimum possible value among all feasible plans. The *objective* is $O \in \{\text{totTRAVEL}, \text{maxTRAVEL}\}$.

5. Assignments under costs

We view the matching sub-problem as a one-sided market. For this reason, we refer to matchings as *assignments*. In our setting, we consider only *semi-decentralized* algorithms that assign requests to drivers if the algorithms have all the private request information of drivers, or else prompt drivers for some such information, and drivers reveal it.

Such an algorithm returns some $(R_1, \ldots, R_n)$ and responsiveness matrix $U = (\tilde{u}_{ij})_{n\times m}$. We can think of matrix U as a tool for capturing the responsive behavior of drivers. For each r_j, whenever r_j is assigned to some driver i, $\tilde{u}_{kj} = 1$ holds for each driver k (also i) who is not unresponsive and $\tilde{u}_{hj} = 0$ holds for each driver $h \neq k$ who is unresponsive.

5.1. Driver fairness

FEF1: For any pair of drivers i and k, the new subjective FEF1 requires eliminating any feasible envy that i might have of k by removing some request from k's bundle of i's feasible requests.

Definition 1. $(R_1, \ldots, R_n)$ *is* FEF1 *if, for each* $v_i, v_k \in V$ *where* $F_{ii} = \{r_j \in R_i | f_{ij} > 0\}$ *and* $F_{ik} = \{r_j \in R_k | f_{ij} > 0\} \neq \emptyset$, $c_i(F_{ii}) \geq c_i(F_{ik} \setminus \{r_j\})$ *holds for some* $r_j \in F_{ik}$.

In our setting, if driver 1 never responds and driver 2 always does, then *no* semi-decentralized algorithm can guarantee to return FEF1 assignments. We next present the key idea behind this observation.

Example 1. *In a decentralized SVRP, let there be* v_1, v_2 *and* r_1, r_2. *Pick* unk *public costs and* unit *public feasibilities. As the private costs are unknown, any semi-decentralized algorithm must prompt drivers for their private costs in some fixed sequence of calls (e.g.* 12, 21, 22, *etc.). Suppose that driver* 1 *never responds and driver* 2 *always does.*

If we suppose that the algorithm returns one of the assignments $(\{r_1\}, \{r_2\})$ *(FEF1),* $(\{r_2\}, \{r_1\})$ *(FEF1), or* $(\{r_1, r_2\}, \emptyset)$ *(not FEF1), then the algorithm must have prompted driver* 1 *for their private costs. But, if the algorithm has prompted driver* 1 *for their private costs and assigned some* r_j *to driver* 1 *anyway, then it must have ignored the fact that driver* 1 *never responds and, hence, the algorithm cannot be semi-decentralized. This leads to a contradiction. We conclude that the algorithm cannot assign any request to driver* 1. *Consequently, the algorithm must assign every* r_j *to agent* 1.

More specifically, the algorithm assigns r_1, r_2 *to driver* 2 *after they are prompted, they respond, and they reveal their private costs. This assignment violates FEF1 because driver* 1 *is still envious even after the removal of any request from* 2*'s bundle. Hence, the algorithm does not return a fair assignment in this instance.* □

In our semi-decentralized setting, unresponsiveness seems harmful when guaranteeing feasibility and FEF1. However, unresponsiveness takes a vital part of any semi-decentralized system. For this reason, we propose to integrate the responsiveness matrix $U = (\tilde{u}_{ij})_{n\times m}$ into the definition of FEF1.

Definition 2. $(R_1, \ldots, R_n)$ *is* responsive FEF1 *wrt responsiveness matrix* $U = (\tilde{u}_{ij})_{n\times m}$ *if, for each* $v_i, v_k \in V$ *where* $F^U_{ii} = \{r_j \in R_i | f_{ij} > 0, \tilde{u}_{ij} = 1\}$ *and* $F^U_{ik} = \{r_j \in R_k | f_{ij} > 0, \tilde{u}_{ij} = 1\} \neq \emptyset$, $c_i(F^U_{ii}) \geq c_i(F^U_{ik} \setminus \{r_j\})$ *holds for some* $r_j \in F^U_{ik}$.

In Example 1, the returned assignment is responsive FEF1 wrt $U = \left(\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}\right)$, $\left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$, or $\left(\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}\right)$. We give an algorithm for computing such assignments in every instance, that simulates a round-robin order by using a driver counter at each iteration: Algorithm 1.

In Algorithm 1, as the computation evolves, some driver i may depart from (i.e. become "unresponsive") or arrive at (i.e. become "responsive") the market. However, we do not know the iterations at which they may depart from or arrive at the market. From this perspective, we can view Algorithm 1 as a tool for achieving FEF1 between any two drivers over the total numbers of the respective assignment iterations at which they are responsive at the market. For this reason, on the one extreme, Algorithm 1 returns $(\emptyset, \ldots, \emptyset)$, which is still responsive FEF1, whenever no driver responds. On the other extreme, it returns a complete assignment whenever drivers always respond.

As we mentioned in the introduction, we consider responsive drivers who are truthful and profit-maximizing. That is, whenever some driver i respond with $r_j \in R$, c_{ij}, and f_{ij}, we assume that $\{r_h \in R | f_{ih} > 0\} \neq \emptyset$ and $r_j \in \{r_k \in R | c_{ik} = \arg\max_{r_h \in R: f_{ih} > 0} c_{ih}\}$ hold.

Theorem 1. *In semi-decentralized SVRPs, Algorithm 1 returns in* $O(m \cdot \max\{n \cdot m, T\})$ *time an assignment that is feasible and responsive FEF1 wrt the returned matrix, supposing that drivers are truthful and profit-maximizing whenever they respond (see Appendix C).*

Algorithm 1 SEMI-DECENTRALISED ROUND-ROBIN

Input: V, R, P, F, T
Output: feasible and responsive FEF1 assignment

$\forall v_i \in V : R_i \leftarrow \emptyset$, mark driver i as "responsive"
$U = (\tilde{u}_{ij})_{n\times m} \leftarrow (1)_{n\times m}$ ▷ responsiveness matrix
$i \leftarrow 1$ ▷ a driver counter
while $R \neq \emptyset$ **do**
  if all drivers are "unresponsive" **then** ▷ termination
    exit
  if $i > n$ **then** ▷ end of round
    $i \leftarrow 1$
    continue
  if $\exists r_k \in R : \tilde{c}_{ik} = unk \vee \tilde{f}_{ik} = unk$ **then**
    send R to driver i ▷ decentralisation
    if within T, driver i reply with r_j, c_{ij}, f_{ij} **then**
      mark driver i as "responsive"
      $\tilde{c}_{ij} \leftarrow c_{ij}, \tilde{f}_{ij} \leftarrow f_{ij}$
      $R_i \leftarrow R_i \cup \{r_j\}, R \leftarrow R \setminus \{r_j\}$
      $\forall k \in \{h|h$ is "unresponsive"$\} : \tilde{u}_{kj} \leftarrow 0$
    else
      mark driver i as "unresponsive"
    $i \leftarrow (i+1)$
  else
    if $\exists r_k \in R : f_{ik} > 0$ **then** ▷ centralisation
      mark driver i as "responsive"
      $r_j \leftarrow \arg\max_{r_k \in R : f_{ik} > 0} c_{ik}$
      $R_i \leftarrow R_i \cup \{r_j\}, R \leftarrow R \setminus \{r_j\}$
      $\forall k \in \{h|h$ is "unresponsive"$\} : \tilde{u}_{kj} \leftarrow 0$
    else
      mark driver i as "unresponsive"
    $i \leftarrow (i+1)$
return $[(R_1, \ldots, R_n), U]$

By comparison, if each driver always responds whenever their vehicle is feasible for some remaining requests, then Algorithm 1 returns an assignment that is not just feasible FEF1, but also complete (i.e. each request, for which some vehicle is truthfully feasible, is assigned).

Corollary 1. *In semi-decentralized SVRPs, if drivers always respond whenever their vehicles are truthfully feasible for remaining requests then Algorithm 1 returns a complete, feasible, and FEF1 assignment, supposing that drivers are truthful and profit-maximizing whenever they respond.*

Proof. By Theorem 1, the returned $(R_1, \ldots, R_n)$ is feasible and responsive FEF1 wrt U. $(\star)$ Each driver i respond whenever v_i is truthfully feasible for remaining requests. For completeness, we observe that each r_j such that $f_{ij} > 0$ holds for some v_i is assigned. For FEF1, we pick some driver i. We let p_i denote the first iteration when driver i is marked as "unresponsive" if this happens, and else the last iteration of the algorithm. At p_i, they are truthfully infeasible for any remaining requests by $(\star)$. Before/After p_i, driver i is "responsive"/"unresponsive" by $(\star)$. We next pick some driver k with $v_k \in V$. For each $r_j \in R_k$ assigned before/after p_i, we have $f_{ij} \geq 0/f_{ij} = 0$ and $\tilde{u}_{ij} = 1/\tilde{u}_{ij} = 0$. Hence, $F_{ii} = F^U_{ii}$ and $F_{ik} = F^U_{ik}$ hold. The result holds. □

By this result, it follows that complete, feasible, and FEF1 assignments always exist. Hence, Algorithm 1 returns such assignments in centralized settings as well.

FEQX: For any pair of drivers i and k, the new objective FEQX requires eliminating any feasible jealousy that i might have of k by removing some request from k's bundle of i's feasible requests.

Definition 3. $(R_1, \dots, R_n)$ *is* FEQX *if, for each* $v_i, v_k \in V$ *where* $F_{ii} = \{r_j \in R_i | f_{ij} > 0\}$ *and* $F_{ik} = \{r_j \in R_k | f_{ij} > 0\} \neq \emptyset$, $c_i(F_{ii}) \geq c_k(F_{ik} \setminus \{r_j\})$ *holds for every* $r_j \in F_{ik}$.

There are instances in our setting, where *no* semi-decentralized algorithm can guarantee to return FEQX assignments: see Example 1. For this reason, as for responsive FEF1, we integrate the responsiveness matrix $U = (\tilde{u}_{ij})_{n\times m}$ into the definition of FEQX.

Definition 4. $(R_1, \dots, R_n)$ *is* responsive FEQX *wrt responsiveness matrix* $U = (\tilde{u}_{ij})_{n\times m}$ *if, for each* $v_i, v_k \in V$ *where* $F^U_{ii} = \{r_j \in R_i | f_{ij} > 0, \tilde{u}_{ij} = 1\}$ *and* $F^U_{ik} = \{r_j \in R_k | f_{ij} > 0, \tilde{u}_{ij} = 1\} \neq \emptyset$, $c_i(F^U_{ii}) \geq c_k(F^U_{ik} \setminus \{r_j\})$ *holds for every* $r_j \in F^U_{ik}$.

In Example 1, the returned assignment is responsive FEQX wrt $U = \left(\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}\right)$, $\left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$, or $\left(\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}\right)$. We give an algorithm for computing such assignments in every instance, that simulates a greedy selection of some minimum profit driver at each iteration: Algorithm 2.

In Algorithm 2, as the computation evolves, some driver i may depart from the market (i.e. become "unresponsive") when their vehicle is removed from the fleet. However, we do not know the iteration at which they may depart from the market. From this perspective, we can view Algorithm 2 as a tool for achieving FEQX between any two drivers over the total numbers of their respective assignment iterations at which they are present in the market. For this reason, on the one extreme, Algorithm 2 returns $(\emptyset, \dots, \emptyset)$, which is still responsive FEQX, whenever no driver responds. This is natural because there are no drivers who can be assigned and service some requests. On the other extreme, we prove that the returned assignment is complete supposing that drivers always respond whenever their vehicles are truthfully feasible for remaining requests.

Theorem 2. *In semi-decentralized SVRPs, Algorithm 2 returns in* $O(m \cdot \max\{n \cdot m, T\})$ *time an assignment that is feasible and responsive FEQX wrt the returned matrix, supposing that drivers are truthful and profit-maximizing whenever they respond (see Appendix C).*

Corollary 2. *In semi-decentralized SVRPs, if drivers always respond whenever their vehicles are truthfully feasible for remaining requests then Algorithm 2 returns a complete, feasible, and FEQX assignment, supposing that drivers are truthful and profit-maximizing whenever they respond.*

Proof. By Theorem 2, the returned assignment $(R_1, \dots, R_n)$ is feasible and responsive FEQX wrt U. For completeness and FEQX, we can use absolutely the same arguments from the proof of Corollary 1. Hence, the returned assignment $(R_1, \dots, R_n)$ is also complete and FEQX. □

By this result, it follows that complete, feasible, and FEQX assignments always exist. Hence, Algorithm 2 returns such assignments in centralized settings as well.

FEFX: In many real-world applications, the planner fixes each request cost but semi-decentralize the feasibilities (e.g. DHL, GoAirlink). The cost of a given request thus does not depend on which feasible vehicle services the customer. For such applications, we compare FEFX with FEF1/FEQX.

Definition 5. $(R_1, \dots, R_n)$ *is* FEFX *if, for each* $v_i, v_k \in V$ *where* $F_{ii} = \{r_j \in R_i | f_{ij} > 0\}$ *and* $F_{ik} = \{r_j \in R_k | f_{ij} > 0\} \neq \emptyset$, $c_i(F_{ii}) \geq c_i(F_{ik} \setminus \{r_j\})$ *holds for every* $r_j \in F_{ik}$.

As for FEF1 and FEQX, there are instances such as the one in Example 1 where *no* semi-decentralized algorithm can guarantee to return FEFX assignments. We can thus define responsive FEFX wrt $U = (\tilde{u}_{ij})_{n\times m}$.

Definition 6. $(R_1, \dots, R_n)$ *is* responsive FEFX *wrt responsiveness matrix* $U = (\tilde{u}_{ij})_{n\times m}$ *if, for each* $v_i, v_k \in V$ *where* $F^U_{ii} = \{r_j \in R_i | f_{ij} > 0, \tilde{u}_{ij} = 1\}$ *and* $F^U_{ik} = \{r_j \in R_k | f_{ij} > 0, \tilde{u}_{ij} = 1\} \neq \emptyset$, $c_i(F^U_{ii}) \geq c_i(F^U_{ik} \setminus \{r_j\})$ *holds for every* $r_j \in F^U_{ik}$.

With driver-dependent additive profits, a (responsive) FEFX assignment is also (responsive) FEF1. But, there are instances where a (responsive) FEQX assignment violates even (responsive) FEF1. We next show this.

Example 2. *In a centralized SVRP, let there be* v_1, v_2 *and* r_1, r_2, r_3. *Define the costs as:* $c_{11} = c_{12} = 1, c_{13} = 6$ *and* $c_{21} = c_{22} = 3, c_{23} = 2$. *Pick unit feasibilities. Let us consider* $(\{r_1, r_2\}, \{r_3\})$. *This one is FEQX because each driver gets a profit of* 2, *but it is not FEF1 because of* $c_2(\{r_3\}) = 2 < 3 = c_2(\{r_1\}) = c_2(\{r_2\})$. □

Algorithm 2 SEMI-DECENTRALISED MIN-MAX

Input: V, R, P, F, T
Output: feasible and responsive FEQX assignment

$\forall v_i \in V : R_i \leftarrow \emptyset$, mark driver i as "responsive"
$U = (\tilde{u}_{ij})_{n\times m} \leftarrow (1)_{n\times m}$ ▷ responsiveness matrix
while $R \neq \emptyset$ **do**
    if all drivers are "unresponsive" **then** ▷ termination
        exit
    $v_i \leftarrow \arg\min_{v_h \in V} \tilde{c}_h(R_h)$ ▷ $\tilde{c}_h(R_h) = c_h(R_h)$
    if $\exists r_k \in R : \tilde{c}_{ik} = unk \vee \tilde{f}_{ik} = unk$ **then**
        send R to driver i ▷ decentralisation
        if within T, driver i reply with r_j, c_{ij}, f_{ij} **then**
            mark driver i as "responsive"
            $\tilde{c}_{ij} \leftarrow c_{ij}, \tilde{f}_{ij} \leftarrow f_{ij}$
            $R_i \leftarrow R_i \cup \{r_j\}, R \leftarrow R \setminus \{r_j\}$
            $\forall k \in \{h | h \text{ is "unresponsive"}\} : \tilde{u}_{kj} \leftarrow 0$
        else
            mark driver i as "unresponsive"
            $V \leftarrow V \setminus \{v_i\}$
    else
        if $\exists r_k \in R : f_{ik} > 0$ **then** ▷ centralisation
            mark driver i as "responsive"
            $r_j \leftarrow \arg\max_{r_k \in R : f_{ik} > 0} c_{ik}$
            $R_i \leftarrow R_i \cup \{r_j\}, R \leftarrow R \setminus \{r_j\}$
            $\forall k \in \{h | h \text{ is "unresponsive"}\} : \tilde{u}_{kj} \leftarrow 0$
        else
            mark driver i as "unresponsive"
            $V \leftarrow V \setminus \{v_i\}$
return $[(R_1, \ldots, R_n), U]$

With driver-independent additive profits, the (responsive) notions of FEFX and FEQX coincide. This follows by additivity and the fact that $c_{ij} = c_j$ holds for each v_i and each r_j. By Theorem 2 and Corollary 2, Algorithm 2 returns feasible and responsive FEFX assignments.

5.2. Driver efficiency

Let us consider the total driver cost in feasible assignment $(R_1, \ldots, R_n)$: $\sum_{v_i \in V} c_i(R_i)$. We refer to this as the *feasible social welfare*, or *FSW*, of $(R_1, \ldots, R_n)$. Thus, we study optimising $\max_{(R_1,\ldots,R_n)\text{: feasible}} \sum_{v_i \in V} c_i(R_i)$. This outcome is socially *efficient* for drivers because no other feasible assignment can give a higher welfare value.

In our semi-decentralized setting, we can compute in $O(n \cdot m)$ time a feasible assignment that maximizes the FSW, taking into account the responsive behavior of drivers. For this purpose, we give the following procedure that assigns the requests one by one in any order.

Pick r_j. Send r_j to each driver i. Consider $\{v_i \in V | i \text{ reveal } c_{ij}, f_{ij}\}$. If this set is non-empty, then we can assign r_j to $v_i = \arg\max_{v_k \in V : \tilde{f}_{ik} = f_{ik} > 0} c_{ik}$. Otherwise, remove r_j and pick another request, if any. If drivers are truthful and profit-maximizing, then this procedure gives a complete assignment of the maximum value $\text{FSW}_{\max}$.

5.3. Driver fairness and efficiency

In centralized settings, $\text{FSW}_{\max}$ assignments may not be fair in some instances, where a single driver provides each customer with a feasible service at the greatest cost. As a result, assignments that are fair for drivers (i.e. FEF1/FEQX) and efficient for drivers (i.e. $\text{FSW}_{\max}$) may *not* always exist.

Instead, we may be interested in *additive approximations*, i.e. fair assignments where the total driver cost is bounded from below by $\text{FSW}_{\max}$ minus some fixed $t \in \mathbb{R}_{\geq 0}$. We formulate a related decision problem and prove that it is NP-hard by reducing from the Partition problem [22].

Problem 1: Driver Fairness & Driver Efficiency
Data: $V, R, P, F, t \in \mathbb{R}_{\geq 0}$.
Query: Is there complete, feasible, and fair $(R_1, \ldots, R_n)$ s.t. $(\text{FSW}_{\max} - t) \leq \sum_{v_i \in V} \sum_{r_j \in R_i} c_{ij} \leq \text{FSW}_{\max}$?

Theorem 3. *In centralized SVRPs, Problem 1 belongs to the class of* NP*-hard decision problems (see Appendix C).*

Maximizing the FSW within the set of complete, feasible, and fair assignments may not give us a good approximation of $\text{FSW}_{\max}$. The reason for this result is that there are instances where one driver might overcharge consistently customers with higher costs. We demonstrate this.

Example 3. *In a centralized SVRP, let there be v_1, v_2 and r_1, r_2, r_3. For each r_j, define the costs as: $c_{1j} = \epsilon$ and $c_{2j} = c$. Here, $\epsilon \in (0, 1)$ and $c \in (1, \infty)$. Pick unit feasibilities. It is easy to see that any complete and FEF1 assignment gives one request or two requests to driver 1, and any complete and FEQX assignment gives two requests to driver 1. Pick one FEF1 assignment, say $(\{r_1\}, \{r_2, r_3\})$. Its welfare value is $FSW_1 = (2 \cdot c + \epsilon)$. Pick one FEQX assignment, say $(\{r_1, r_2\}, \{r_3\})$. Its welfare value is $FSW_2 = (c + 2 \cdot \epsilon)$. However, $(\emptyset, \{r_1, r_2, r_3\})$, which is not fair, maximises the FSW and gives $FSW_{\max} = 3 \cdot c$. For $\epsilon \to 0$ and $c \to \infty$, the differences $(FSW_{\max} - FSW_1) = (c - \epsilon)$ and $(FSW_{\max} - FSW_2) = (2 \cdot c - 2\epsilon)$ go to ∞. Their values push from below the parameter value of t in Problem 1.* □

In practice, it is unlikely that the request costs are negligibly low or extremely high. But, even if the cost of any request lies within some interval, then maximizing the FSW within the set of fair assignments might still not give us a good approximation of $\text{FSW}_{\max}$.

Example 4. *In a centralized SVRP, let there be $v_1, \ldots, v_n$ and $r_1, \ldots, r_n$. For each r_j, define the costs as: $c_{ij} = 1$ for each $v_i \in V \setminus \{v_n\}$ and $c_{nj} = 2$. Pick unit feasibilities. Any assignment that gives at least two requests to some driver and no request to another driver cannot be fair. Therefore, any complete, feasible, and fair assignment gives exactly one request to each driver. The welfare value of such an assignment is FSW$= (n + 1)$. However, giving all requests to driver n, which is not fair, gives $FSW_{\max} = 2 \cdot n$. The difference $(FSW_{\max} - FSW) = (n - 1)$ is just one unit below the worst possible value of n. Its value pushes from below the parameter value of t in Problem 1.* □

In theory, the result in Example 4 provides a lower bound on how well complete, feasible, and fair assignments approximate $\text{FSW}_{\max}$. In practice, this bound is however already large enough because some problems have tens of thousands of drivers (i.e. $n \approx 90\,000$) [3, 4].

In response, we might also want to consider *multiplicative approximations* in Problem 1: fair assignments where the total driver cost is bounded from below by $s \cdot \text{FSW}_{\max}$ for some fixed $s \in [0, 1]$. However, for additive approximations, we used $t = 0$ in the reduction in the proof of Theorem 3. Thus, for multiplicative approximations, we can use $s = 1$ in it and conclude a similar hardness result.

5.4. Customers fairness and efficiency

In centralized settings, we also study optimizing the total customer cost $\min_{(R_1,\ldots,R_n):\text{ feasible \& complete}} \sum_{v_i \in V, r_j \in R_i} c_{ij}$. This outcome is socially *efficient* for customers because if a customer can receive a cheaper service from some other feasible vehicle then we can move their request to such a vehicle and, thus, strictly decrease the total customer cost.

In our semi-decentralized setting, we can compute in $O(n \cdot m)$ time a feasible and complete assignment that minimizes the total customer cost, accounting for responsive drivers. For this purpose, we give the next procedure that assigns the requests one by one in any order.

Pick r_j. Send r_j to each driver i. Consider $\{v_i \in V | i \text{ reveal } c_{ij}, f_{ij}\}$. If this set is non-empty, then we can assign r_j to $v_i = \arg\min_{v_k \in V: \tilde{f}_{ik} = f_{ik} > 0} c_{ik}$. Otherwise, remove r_j and pick another request, if any. If drivers are truthful and profit-maximizing, then this procedure gives a complete assignment of the minimum value $\text{FSW}_{\min}$.

$\text{FSW}_{\min}$ assignments are also optimal for the maximum cost $\min_{(R_1,\ldots,R_n)\text{: feasible \& complete}} \max_{v_i \in V, r_j \in R_i} c_{ij}$ and the product cost $\min_{(R_1,\ldots,R_n)\text{: feasible \& complete}} \prod_{v_i \in V, r_j \in R_i} c_{ij}$. Thus, they are also *fair* to customers because none of them has envy for another feasible vehicle.

5.5. Driver fairness and customer efficiency

$\text{FSW}_{\min}$ assignments may violate fairness in some instances, where a single driver provides each customer with a feasible service at the lowest cost. Hence, complete and feasible assignments that are fair for drivers (i.e. FEF1/FEQX) and efficient for customers (i.e. $\text{FSW}_{\min}$) may *not* exist.

Thus, in centralized settings, we may wish to return fair assignments whose total customer cost is bounded from above by $\text{FSW}_{\min}$ plus some fixed threshold $t \in \mathbb{R}_{\geq 0}$. We formulate this as a decision problem and give another reduction to it from the Partition problem [22].

> Problem 2: Driver Fairness & Customer Efficiency
> **Data**: V, R, P, F, $t \in \mathbb{R}_{\geq 0}$.
> **Query**: Is there complete, feasible, and fair $(R_1, \ldots, R_n)$ s.t. $\text{FSW}_{\min} \leq \sum_{v_i \in V} \sum_{r_j \in R_i} c_{ij} \leq (\text{FSW}_{\min} + t)$?

Theorem 4. *In centralized SVRPs, Problem 2 belongs to the class of* NP*-hard decision problems (see Appendix C).*

Minimizing the FSW within the set of fair assignments may also not give us a reasonable approximation of $\text{FSW}_{\min}$. The reason for this result is that there are instances where one driver might undercharge consistently customers with lower costs. We illustrate this in two contexts.

Example 5. *Let us consider the centralized SVRP from Example 3. Recall,* $\epsilon \in (0, 1)$ *and* $c \in (1, \infty)$. *In this problem, pick the FEF1* $(\{r_1\}, \{r_2, r_3\})$ *and the FEQX* $(\{r_1, r_2\}, \{r_3\})$. *The welfare values of these assignments are* $FSW_1 = (2 \cdot c + \epsilon)$ *and* $FSW_2 = (c + 2 \cdot \epsilon)$, *respectively. However,* $(\{r_1, r_2, r_3\}, \emptyset)$, *which is not fair, minimizes the FSW and gives a value of* $FSW_{\min} = 3 \cdot \epsilon$. *For* ϵ *going to* 0 *and* c *going to* ∞, *the differences* $(FSW_1 - FSW_{\min}) = (2 \cdot c - 2 \cdot \epsilon)$ *and* $(FSW_2 - FSW_{\min}) = (c - \epsilon)$ *go to* ∞. *Their values push from below the parameter value of t in Problem 2.* □

Example 6. *In a centralized SVRP, let there be* $v_1, \ldots, v_n$ *and* $r_1, \ldots, r_n$. *For each* r_j, *define the costs as:* $c_{ij} = 2$ *for each* $v_i \in V \setminus \{v_n\}$ *and* $c_{jn} = 1$. *Pick unit feasibilities. In this problem, it is easy to see that any complete, feasible, and fair assignment gives exactly one request to each driver. The value of the feasible welfare of such an assignment is* $FSW = (2 \cdot n - 1)$. *However, giving all requests to driver n, which is not fair, minimizes the FSW and gives* $FSW_{\min} = n$. *The difference* $(FSW - FSW_{\min}) = (n - 1)$ *between these two assignments is just one unit below the worst possible value of n. Its value pushes from below the parameter value of t in Problem 2.* □

As for driver efficiency, we can consider *multiplicative approximations* of customer efficiency in Problem 2: fair assignments where the total customer cost is bounded from above by $s \cdot \text{FSW}_{\min}$ for some fixed $s \in [1, \infty)$. However, for additive approximations, we used $t = 0$ in the reduction in the proof of Theorem 4. For multiplicative approximations, we can use $s = 1$ in it and derive a similar hardness result.

5.6. Driver efficiency and customer efficiency

For completeness of our analysis, we also discuss achieving driver efficiency and customer efficiency. With driver-independent additive profits, a complete and feasible assignment gives $\text{FSW}_{\max}$ iff it gives $\text{FSW}_{\min}$. By Theorem 2 and Corollary 2, Algorithm 2 returns assignments that are efficient for both drivers and customers.

With driver-dependent additive profits, let us consider a customer request and two different costs for it. $\text{FSW}_{\max}$ assignments and $\text{FSW}_{\min}$ assignments will give the request to two different drivers. The $\text{FSW}_{\max}$ decision is *not* efficient for the customer, because their cost according to it is strictly higher than their cost according to the $\text{FSW}_{\min}$ decision.

6. Routing under few feasibilities

Let us consider the following parameter in a given instance: $f = \max_{v_i \in V} \sum_{r_j \in R, f_{ij} > 0} 1$. That is, f is the maximum number of feasible requests per vehicle. We let f take some constant value. Even in this case, minimizing totTRAVEL/maxTRAVEL optimally by using naive brute force may still take time, exponential in n. This is because some instances contain $O(2^{n \cdot f})$ feasible matchings. However, given some fixed feasible matching, minimizing totTRAVEL/maxTRAVEL by using naive brute force takes time, polynomial in n in each instance.

Theorem 5. *If a feasible matching is not fixed, a brute-force algorithm may take $O(2^{n \cdot f} \cdot n \cdot (2 \cdot f)!)$ time to compute an optimal plan for totTRAVEL/maxTRAVEL (see Appendix D).*

Theorem 6. *If a feasible matching is fixed, a brute-force algorithm takes $O(n \cdot (2 \cdot f)!)$ time to compute a minimizing feasible plan for totTRAVEL/maxTRAVEL (see Appendix D).*

Computing feasible matchings might take exponential time in general simply because, as we discussed earlier, deciding the vehicle feasibilities might belong to NP-hard. However, our Algorithms 1 and 2 takes polynomial time to return such matchings because they decentralize these feasibilities. Also, minimizing totTRAVEL/maxTRAVEL after fixing a feasible matching may produce sub-optimal routes, e.g. see our motivating example in the introduction.

7. Summary, future work, and acknowledgements

We defined a new class of Social Vehicle Routing Problems. We proposed new algorithms for it, which first return a feasible matching between drivers and customers, and then a minimal plan for routing the vehicles through their matched locations. We gave matching algorithms for achieving fairness notions such as FEF1, FEQX, and FEFX, and efficiency notions such as $\text{FSW}_{\max}$, and $\text{FSW}_{\min}$. We also gave lower bounds on how well driver fair assignments approximate driver and customer efficiency. Finally, we gave fixed-parameter tractable routing algorithms for minimizing maxTRAVEL and totTRAVEL. Tables 1 and 2 summarise our main results. In future work, we plan to follow multiple directions. We will look at other behavior of drivers such as bounded rationality [11]. We will transfer other COMSOC concepts such as Pareto-optimality [33] and manipulability [24], as well as add other VRP concepts such as time windows [15]. Also, we will study fairness and efficiency in general preference domains. Finally, we want to extend our model to dynamic [21] and online [1] environments. Martin Aleksandrov was supported by the DFG Individual Research Grant on "Fairness and Efficiency in Emerging Vehicle Routing Problems" (497791398).

References

[1] Aleksandrov, M., Aziz, H., Gaspers, S., Walsh, T., 2015. Online fair division: analysing a food bank problem, in: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, pp. 2540–2546. URL: `https://dl.acm.org/doi/10.5555/2832581.2832604`.

[2] Aleksandrov, M., Barahona, P., Kilby, P., Walsh, T., 2013. Heuristics and policies for online pickup and delivery problems, in: Proceedings of AAAI 2013, July 14-18, 2013, Bellevue, WA, USA. URL: `https://www.aaai.org/ocs/index.php/WS/AAAIW13/paper/view/7119`.

[3] Aleksandrov, M.D., 2021a. Fleet fairness and fleet efficiency in capacitated pickup and delivery problems, in: Proceedings of the 32nd IEEE Intelligent Vehicles Symposium (IV21) July 11-17, 2021. Nagoya, Japan, IEEE Xplore. pp. 1156–1161. doi:`10.1109/IV48863.2021.9576002`.

[4] Aleksandrov, M.D., 2021b. Minimising fleet times in capacitated pickup and dropoff problems, in: Marreiros, G., Melo, F.S., Lau, N., Cardoso, H.L., Reis, L.P. (Eds.), Progress in Artificial Intelligence - 20th EPIA Conference on Artificial Intelligence, EPIA 2021, Portugal, September 7-9, 2021, Proceedings, Springer. pp. 199–210. URL: `https://doi.org/10.1007/978-3-030-86230-5_16`.

[5] Aziz, H., Huang, X., Mattei, N., Segal-Halevi, E., 2021. Computing welfare-maximizing fair allocations of indivisible goods. `arXiv:2012.03979`.

[6] Beltrami, E.J., Bodin, L.D., 1974. Networks and vehicle routing for municipal waste collection. Networks 4, 65–94. doi:`10.1002/net.3230040106`.

[7] Boehmer, N., Niedermeier, R., 2021. Broadening the research agenda for computational social choice: Multiple preference profiles and multiple solutions, in: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC. p. 1–5. URL: `https://dl.acm.org/doi/10.5555/3463952.3463954`.

[8] Brams, S.J., Taylor, A.D., 1996. Fair Division - from Cake-cutting to Dispute Resolution. Cambridge University Press. doi:`10.1086/233796`.

[9] Brandt, F., Conitzer, V., Endriss, U., Lang, J., Procaccia, A.D., 2016. Handbook of Computational Social Choice. 1st ed., Cambridge University Press, USA. doi:`10.1017/CBO9781107446984`.

[10] Budish, E., 2011. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. Journal of Political Economy 119, 1061–1103. doi:10.1145/1807406.1807480.

[11] Camerer, C.F., 2003. Behavioral Game Theory Experiment in Strategic Interaction. volume 32 (6). doi:10.1016/j.socec.2003.10.009.

[12] Caragiannis, I., Kurokawa, D., Moulin, H., Procaccia, A.D., Shah, N., Wang, J., 2019. The unreasonable fairness of maximum nash welfare. ACM Transactions on Economics and Computation 7. doi:10.1145/3355902.

[13] Chevaleyre, Y., Endriss, U., Estivie, S., Maudet, N., 2008. Multiagent resource allocation in k-additive domains: preference representation and complexity. Annals of Operations Research 163, 49–62. doi:10.1007/s10479-008-0335-0.

[14] Conitzer, V., Freeman, R., Shah, N., 2017. Fair public decision making, in: Daskalakis, C., Babaioff, M., Moulin, H. (Eds.), Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017, ACM. pp. 629–646. doi:10.1145/3033274.3085125.

[15] Cordeau, J.F., Desaulniers, G., Desrosiers, J., Solomon, M.M., Soumis, F., 2001. VRP with Time Windows. Society for Industrial and Applied Mathematics, USA. chapter 7. pp. 157–193. doi:10.1137/1.9780898718515.ch7.

[16] Dantzig, G.B., Ramser, J.H., 1959. The truck dispatching problem. Management Science 6, 80–91. doi:10.1287/mnsc.6.1.80.

[17] Downey, R.G., Fellows, M.R., 2013. Fundamentals of Parameterized Complexity. Texts in Computer Science, Springer. doi:10.1007/978-1-4471-5559-1.

[18] Dror, A., Feldman, M., Segal-Halevi, E., 2021. On fair division under heterogeneous matroid constraints, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, AAAI Press. pp. 5312–5320. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16670.

[19] Foley, D.K., 1967. Resource allocation and the public sector. Yale Economic Essays 7, 45–98.

[20] Freeman, R., Sikdar, S., Vaish, R., Xia, L., 2019. Equitable allocations of indivisible goods, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence Organization. pp. 280–286. doi:10.24963/ijcai.2019/40.

[21] Freeman, R., Zahedi, S.M., Conitzer, V., 2017. Fair and efficient social choice in dynamic settings, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pp. 4580–4587. doi:10.24963/ijcai.2017/639.

[22] Garey, M.R., Johnson, D.S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman. URL: https://dl.acm.org/doi/book/10.5555/574848.

[23] Gerding, E.H., Perez-Diaz, A., Aziz, H., Gaspers, S., Marcu, A., Mattei, N., Walsh, T., 2019. Fair online allocation of perishable goods and its application to electric vehicle charging, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization. pp. 5569–5575. doi:10.24963/ijcai.2019/773.

[24] Gibbard, A., 1973. Manipulation of voting schemes: A general result. Econometrica 41, 587–601. doi:10.2307/1914083.

[25] Gourvès, L., Monnot, J., Tlilane, L., 2013. A matroid approach to the worst case allocation of indivisible goods, in: Rossi, F. (Ed.), IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013, IJCAI/AAAI. pp. 136–142. URL: https://dl.acm.org/doi/10.5555/2540128.2540150.

[26] Hollis, B.L., Green, P.J., 2012. Real-life vehicle routing with time windows for visual attractiveness and operational robustness. Asia-Pacific Journal of Operational Research 29. doi:10.1142/S0217595912500170.

[27] Kash, I.A., Procaccia, A.D., Shah, N., 2014. No agent left behind: Dynamic fair division of multiple resources. JAIR 51, 579–603. doi:10.1613/jair.4405.

[28] de Keijzer, B., Bouveret, S., Klos, T., Zhang, Y., 2009. On the complexity of efficiency and envy-freeness in fair division of indivisible goods with additive preferences, in: Proceedings of the 1st International ADT Conference, Venice, Italy, October 20-23., pp. 98–110. doi:10.1007/978-3-642-04428-1_9.

[29] Li, B., Krushinsky, D., Reijers, H.A., Woensel, T.V., 2014. The share-a-ride problem: People and parcels sharing taxis. European Journal of Operational Research 238, 31–40. doi:10.1016/j.ejor.2014.03.003.

[30] Ma, H., Fang, F., Parkes, D.C., 2019. Spatio-temporal pricing for ridesharing platforms, in: Proceedings of the 2019 ACM Conference on Economics and Computation, Association for Computing Machinery, New York, NY, USA. p. 583. doi:10.1145/3328526.3329556.

[31] Männel, D., Bortfeldt, A., 2018. Solving the pickup and delivery problem with three-dimensional loading constraints and reloading ban. European Journal of Operational Research 264, 119–137. doi:10.1016/j.ejor.2017.05.034.

[32] Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G., 2007. MiniZinc: Towards a standard cp modelling language, in: Proceedings of the Thirteenth International Conference on Principles and Practice of Constraint Programming, Springer-Verlag, Berlin, Heidelberg. pp. 529–543. doi:10.1007/978-3-540-74970-7_38.

[33] Plaut, B., Roughgarden, T., 2018. Almost envy-freeness with general valuations, in: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018, pp. 2584–2603. doi:10.1137/1.9781611975031.165.

[34] Rheingans-Yoo, D., Kominers, S.D., Ma, H., Parkes, D.C., 2019. Ridesharing with driver location preferences, in: Proceedings of IJCAI-19, International Joint Conferences on Artificial Intelligence Organization. pp. 557–564. doi:10.24963/ijcai.2019/79.

[35] Rossi, F., van Beek, P., Walsh, T., 2006. Handbook of Constraint Programming. Elsevier Science Inc., USA. URL: https://dl.acm.org/doi/10.5555/2843512.

[36] Santos, D.O., Xavier, E.C., 2013. Dynamic taxi and ridesharing: A framework and heuristics for the optimization problem, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, AAAI Press. pp. 2885–2891. URL: https://dl.acm.org/doi/10.5555/2540128.2540544.

[37] Savelsbergh, M.W.P., Sol, M., 1995. The general pickup and delivery problem. Transportation Science 29, 17–29. doi:10.1287/trsc.29.1.17.

[38] Stuckey, P.J., Feydy, T., Schutt, A., Tack, G., Fischer, J., 2014. The MiniZinc challenge 2008-2013. AI Magazine 35, 55–60. doi:10.1609/aimag.v35i2.2539.
[39] Wilson, J.P., Fotheringham, A.S., 2007. The Handbook of Geographic Information Science. John Wiley & Sons, Ltd. doi:10.1002/9780470690819.
[40] Xu, Y., Xu, P., 2020. Trade the system efficiency for the income equality of drivers in rideshare, in: Bessiere, C. (Ed.), Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, International Joint Conferences on Artificial Intelligence Organization. pp. 4199–4205. doi:10.24963/ijcai.2020/580.

Appendix A. Contributions

FEF1. In our setting, we can define EF1. However, as we mentioned in the main text, this notion does not account for any feasibility indicators.

Definition 7. $(R_1, \ldots, R_n)$ *is* EF1 *if, for each* $v_i, v_k \in V$ *s.t.* $R_k \neq \emptyset$, $c_i(R_i) \geq c_i(R_k \setminus \{r_j\})$ *holds for some* $r_j \in R_k$.

As a consequence, there are instances where each EF1 assignment is not feasible FEF1 and each feasible FEF1 assignment falsifies EF1. We next illustrate this.

Example 7. *In a centralized SVRP, let there be vehicles* v_1, v_2 *and requests* r_1, r_2. *Also, define any strictly positive costs, and* $f_{11} = f_{12} = 1$ *but* $f_{21} = f_{22} = 0$. *For example, the costs could be driver-independent, i.e. all equal to* 1. *We make two observations.*

First, each EF1 assignment gives exactly one request to each vehicle. Hence, each such assignment gives some infeasible request to v_2. *Such an outcome cannot be feasible.*

Second, the unique feasible assignment gives both requests to v_1. *However, this outcome violates EF1 because the costs are strictly positive, and removing any single request from the bundle of the driver of* v_1 *does not eliminate the envy of the driver of* v_2.

In centralized settings, it follows by Example 7 that the classical EF1 round-robin rule from [12] may not return a feasible FEF1 assignment. By Corollary 1, it follows that Algorithm 1 does it.

FEQX. In our setting, we can also define EQX. However, as we mentioned in the main text, this notion as the one of EF1 does not account for any feasibility indicators.

Definition 8. $(R_1, \ldots, R_n)$ *is* EQX *if, for each* $v_i, v_k \in V$ *s.t.* $R_k \neq \emptyset$, $c_i(R_i) \geq c_k(R_k \setminus \{r_j\})$ *holds for any* $r_j \in R_k$.

In Example 7, each EQX assignment gives one request to each vehicle and also is not feasible FEQX, whereas the unique feasible FEQX assignment falsifies EQX.

In centralized settings, it also follows by Example 7 that the EQX rule from [20] is not guaranteed to return a feasible FEQX assignment. By Corollary 2, it follows that Algorithm 2 does it.

FEFX. In our setting, we can also define EFX. However, as we mentioned in the main text, this notion as EF1 and EQX does not account for any feasibilities.

Definition 9. $(R_1, \ldots, R_n)$ *is* EFX *if, for each* $v_i, v_k \in V$ *s.t.* $R_k \neq \emptyset$, $c_i(R_i) \geq c_i(R_k \setminus \{r_j\})$ *holds for any* $r_j \in R_k$.

In Example 7, each EFX assignment gives one request to each vehicle and, moreover, is not feasible FEFX, whilst the unique feasible FEFX assignment violates EFX.

In centralized settings with driver-independent additive profits, it also follows by Example 7 that the popular EFX leximin rule from [33] may not return a feasible FEFX assignment. By Corollary 2 and the fact that the notions of FEQX and FEFX coincide in this domain, it follows that Algorithm 2 does it.

Appendix B. Related work

B.1. FEF1, FEQX, and FEFX: well-definedness

We note that FEF1, FEQX, and FEFX are well-defined. This is because they are defined for any assignment between drivers and customers, including infeasible assignments. Indeed, if any notion were defined only for feasible assignments, then such a notion would not be well-defined for infeasible assignments and comparable with existing notions such as EF1, EQX, and EFX.

To see this, let us consider Example 7. Any EF1/EQX/ EFX assignment in its instance is infeasible. If FEF1/ FEQX/FEFX were defined only for any feasible assignment, then we would not be able to judge whether any EF1/EQX/ EFX assignment is FEF1/FEQX/FEFX or not. From this perspective, FEF1/FEQX/FEFX and EF1/EQX/EFX would be incomparable.

B.2. FEF1 vs F-EF1: a comparison

F-EF1 was defined for a model, where each agent (driver) i partition the set of items (requests) R into categories $C_i^1, \dots, C_i^{l_i}$ for some $l_i \leq |R|$ and have category upper quotas $k_i^1, \dots, k_i^{l_i}$, respectively [18]. The authors define collection of independent sets as $\mathcal{I}_i = \{S \subseteq R | \forall h \in [l_i] : |S \cap C_i^h| \leq k_i^h\}$. They also define feasible valuation (cost) of each agent i for set S as $\hat{c}_i(S) := c_i(\text{BEST}_i(S))$, where $\text{BEST}_i(S) := \arg\max_{T \subseteq S, T \in \mathcal{I}_i} c_i(T)$. Thus, we can next define well the notion of F-EF1 in our setting.

Definition 10. $(R_1, \dots, R_n)$ *is* F-EF1 *iff for every* $v_i, v_j \in V$*, there is* $Y \subseteq R_j$ *with* $|Y| \leq 1$*, s.t.* $\hat{c}_i(R_i) \geq \hat{c}_i(R_j \setminus Y)$*.*

In our setting, we can think of the set of feasible and the set of infeasible requests per vehicle as categories. Thus, for driver i, $C_i^1 = \{r_j \in R | f_{ij} = 1\}$ and $C_i^2 = \{r_j \in R | f_{ij} = 0\}$. We note that $k_i^1 < |C_i^1|$ and $k_i^2 > 0$ do not make sense in our setting. To see this, let firstly $k_i^1 < |C_i^1|$ hold. If there is one vehicle feasible for two requests and $k_i^1 = 1$, then the driver cannot service one of the customers. Let secondly $k_i^2 > 0$ hold. If there is one vehicle that is infeasible for one request and $k_i^2 = 1$, then the driver cannot service the customer. Thus, we let $k_i^1 = |C_i^1|$ and $k_i^2 = 0$. We can now compare F-EF1 and FEF1.

Lemma 1. *An assignment that is feasible FEF1 also satisfies F-EF1.*

Proof. Let us consider a feasible FEF1 assignment $(R_1, \dots, R_n)$. Feasibility says that, for each $v_i \in V$, $F_{ii} = \{r_k \in R_i | f_{ik} > 0\} = R_i$. FEF1 requires that, for each $v_i, v_j \in V$ where $F_{ii} = \{r_k \in R_i | f_{ik} > 0\}$ and $F_{ij} = \{r_k \in R_j | f_{ik} > 0\} \neq \emptyset$, $c_i(F_{ii}) \geq c_i(F_{ij} \setminus \{r_k\})$ holds for some $r_k \in F_{ij}$. We consider two cases.

In the first case, for each $v_i, v_j \in V$ where $F_{ij} \neq \emptyset$, there is $\{r_k\} \subseteq F_{ij} \subseteq R_j$ with $|\{r_k\}| = 1$, s.t. $c_i(F_{ii}) \geq c_i(F_{ij} \setminus \{r_k\})$ holds. The set F_{ii} is maximal s.t. $F_{ii} \subseteq R_i$ and $F_{ii} \in \mathcal{I}_i$. The set $F_{ij} \setminus \{r_k\}$ is maximal s.t. $F_{ij} \setminus \{r_k\} \subseteq R_j \setminus \{r_k\}$ and $F_{ij} \setminus \{r_k\} \in \mathcal{I}_i$. Hence, $c_i(F_{ii}) = \hat{c}_i(R_i)$ and $c_i(F_{ij} \setminus \{r_k\}) = \hat{c}_i(R_j \setminus \{r_k\})$ hold.

In the second case, for each $v_i, v_j \in V$ where $F_{ij} = \emptyset$, there is $Y = \emptyset$ with $|Y| = 0 < 1$ such that $\hat{c}_i(R_i) \geq \hat{c}_i(R_j \setminus Y)$ holds. To see this, we note that $\hat{c}_i(R_i) \geq 0$ holds. The set F_{ij} is maximal s.t. $F_{ij} \subseteq R_j$ and $F_{ij} \in \mathcal{I}_i$. We derive $\hat{c}_i(R_j \setminus Y) = \hat{c}_i(R_j) = c_i(F_{ij}) = 0$ and $\hat{c}_i(R_i) \geq \hat{c}_i(R_j \setminus Y)$. It follows that $(R_1, \dots, R_n)$ is indeed F-EF1. □

Lemma 2. *An assignment that satisfies F-EF1 may not be feasible FEF1.*

Proof. Let us consider v_1, v_2 and r_1, r_2. Define the costs and feasibilities as: $c_{11} = 1, c_{12} = 2, c_{21} = 2, c_{22} = 1$ and $f_{11} = 1, f_{12} = 0, f_{21} = 0, f_{22} = 1$. Further, pick the round-robin assignment (R_1, R_2) with $R_1 = \{r_2\}$ and $R_2 = \{r_1\}$. This is not feasible FEF1 because each request is assigned to a vehicle that is not feasible for it. However, (R_1, R_2) satisfies F-EF1. Indeed, it is easy to observe that the inequalities $\hat{c}_1(R_1) \geq 0 = \hat{c}_1(\emptyset) = \hat{c}_1(R_2 \setminus \{r_1\})$ and $\hat{c}_2(R_2) \geq 0 = \hat{c}_2(\emptyset) = \hat{c}_2(R_1 \setminus \{r_2\})$ hold. □

It follows that the set of feasible FEF1 assignments could be a strict subset of the set of F-EF1 assignments. Hence, it could be harder to return assignments from the former set than from the latter set. However, the assignment of Algorithm 1 does it. By Corollary 1 and Lemma 1, it follows that this assignment satisfies further F-EF1 in centralized settings.

Appendix C. Assignments under costs

C.1. Driver fairness

Theorem 1. *In semi-decentralized SVRPs, Algorithm 1 returns in $O(m \cdot \max\{n \cdot m, T\})$ time an assignment that is feasible and responsive FEF1 wrt the returned matrix, supposing that drivers are truthful and profit-maximizing whenever they respond.*

Proof. The complexity is dominated by at most m iterations. At each of these, the current driver i enters either the centralized phase or the decentralized phase in $O(n \cdot m)$ time. The former phase takes time $O(\max\{n, m\})$. The latter phase takes time $O(\{n, T\})$, during which driver i receives the set R of unassigned requests and they may reveal some $r_j \in R$. In this case, they are truthful and profit-maximizing. Hence, $r_j \in \{r_k \in R | c_{ik} = \max_{r_h \in R : f_{ih} > 0} c_{ih}\} \neq \emptyset$ holds. Thus, they may need to consider at most m candidate feasible requests but, as $T \in \Omega(m)$, it follows that the overall time is $O(m \cdot \max\{n \cdot m, T\})$.

The drivers pick up feasible requests one-by-one until termination. Hence, the returned $(R_1, \ldots, R_n)$ is feasible. Pick two drivers i and k such that $i \neq k$. We note that $i \leq n, k \leq n$ hold. We let $r_i(1), \ldots, r_i(h_i)$ denote the 1st, …, h_ith requests picked up by driver i, and $r_k(1), \ldots, r_k(h_k)$ denote the 1st, …, h_kth requests picked up by driver k. Here $h_i, h_k \in \mathbb{N}_{\geq 0}$. We prove the result in two cases.

Case 1: We let $i < k$ hold. That is, driver i pick up before driver k. We also let l_i denote the number of requests in $r_k(1), \ldots, r_k(h_k)$ for which v_i is feasible and not unresponsive. Wlog, for $l_i \geq 1$, let these be $r_k(j_1), \ldots, r_k(j_{l_i})$ where $1 \leq j_1 \leq \ldots \leq j_{l_i} \leq h_k$ and be picked up at rounds $s_1, \ldots, s_{l_i}$, respectively.

For any fixed l between 1 and l_i, we have that $f_{ij_l} > 0, \tilde{u}_{ij_l} = 1$ hold. Furthermore, driver i must have picked up a request in round s_l. Otherwise, they would be infeasible or unresponsive for $r_k(j_l)$ and, hence, $f_{ij_l} = 0$ or $\tilde{u}_{ij_l} = 0$ would hold. This would lead to a contradiction because $f_{ij_l} > 0, \tilde{u}_{ij_l} = 1$ hold. It follows that $l_i \leq h_i$ holds.

As drivers are profit-maximizing, driver i picks feasible requests in some non-increasing cost sequence. Thus, as $i < k$, $c_i(r_i(l)) \geq c_i(r_k(l))$ follow for each $l = 1 : l_i$. Also, as $j_1 \geq 1, \ldots, j_{l_i} \geq l_i$, $c_i(r_k(l)) \geq c_i(r_k(j_l))$ follow for each $l = 1 : l_i$. These inequalities imply $c_i(r_i(l)) \geq c_i(r_k(j_l))$ for each $l = 1 : l_i$.

We let $F^U_{ii} = \{r_i(1), \ldots, r_i(h_i)\}$ and $F^U_{ik} = \{r_k(j_1), \ldots, r_k(j_{l_i})\}$. We thus derive $c_i(F^U_{ii}) = \sum_{l=1:h_i} c_i(r_i(l)) \geq \sum_{l=1:l_i} c_i(r_i(l)) \geq \sum_{l=1:l_i} c_i(r_k(j_l)) = c_i(F^U_{ik})$. If $l_i = 0$, then $F^U_{ik} = \emptyset$ and $c_i(F^U_{ii}) \geq c_i(F^U_{ik}) = c_i(\emptyset) = 0$ hold. Otherwise, $F^U_{ik} \neq \emptyset$ and $c_i(F^U_{ii}) \geq c_i(F^U_{ik}) \geq c_i(F^U_{ik} \setminus \{r\})$ for each $r \in F^U_{ik}$ hold. Hence, the notion holds for driver i.

We next show that the notion also holds for driver k. We note that driver k picks up after driver i. We let l_k denote the number of requests in $r_i(1), \ldots, r_i(h_i)$ for which v_k is feasible and not unresponsive. Wlog, for $l_k \geq 1$, let these be $r_i(j_1), \ldots, r_i(j_{l_k})$ with $1 \leq j_1 \leq \ldots \leq j_{l_k} \leq h_i$ and be picked up at rounds $s_1, \ldots, s_{l_k}$, respectively.

For any l between 2 and l_k, $f_{kj_l} > 0, \tilde{u}_{kj_l} = 1$ hold. Also, driver k must have picked up a request in round s_{l-1}. Otherwise, they would be infeasible or unresponsive for $r_i(j_l)$ and, hence, $f_{kj_l} = 0$ or $\tilde{u}_{kj_l} = 0$ would hold. This would lead to a contradiction because $f_{kj_l} > 0, \tilde{u}_{kj_l} = 1$ hold. It follows that $(l_k - 1) \leq h_k$ holds.

As $i < k$ and driver k pick feasible requests in some non-increasing cost sequence, $c_k(r_k(l)) \geq c_k(r_i(l+1))$ follow for each $l = 1 : (l_k - 1)$. Also, as $j_1 \geq 1, \ldots, j_{l_k} \geq l_k$, $c_k(r_i(l+1)) \geq c_k(r_i(j_{l+1}))$ follow for each $l = 1 : (l_k - 1)$. These inequalities imply $c_k(r_k(l)) \geq c_k(r_i(j_{l+1}))$ for each $l = 1 : (l_k - 1)$.

We let $F^U_{kk} = \{r_k(1), \ldots, r_k(h_k)\}$ and $F^U_{ki} = \{r_i(j_1), \ldots, r_i(j_{l_k})\}$. We derive $c_k(F^U_{kk}) = \sum_{l=1:h_k} c_k(r_k(l)) \geq \sum_{l=1:(l_k-1)} c_k(r_k(l)) \geq \sum_{l=1:(l_k-1)} c_k(r_i(j_{l+1}))$. The latter sum is $c_k(F^U_{ki} \setminus \{r_i(j_1)\})$. If $l_k = 0$, then $F^U_{ki} = \emptyset$ and $c_k(F^U_{kk}) \geq c_k(\emptyset) = 0$ hold. Otherwise, $F^U_{ki} \neq \emptyset$ and $c_k(F^U_{kk}) \geq c_k(F^U_{ki} \setminus \{r_i(j_1)\})$ for $r_i(j_1) \in F^U_{ki}$ hold.

Case 2: We let $i > k$ hold. We now go to Case 1 with $i = k$ and $k = i$, and follow the proof of Case 1. □

Theorem 2. *In semi-decentralized SVRPs, Algorithm 2 returns in $O(m \cdot \max\{n \cdot m, T\})$ time an assignment that is feasible and responsive FEQX wrt the returned matrix, supposing that drivers are truthful and profit-maximizing whenever they respond.*

Proof. For each iteration, after $O(n \cdot m)$ time, the centralisation/decentralisation phase takes $O(\max\{n, m\})$/ $O(\{n, T\})$ time. For each driver k, we note that $\tilde{c}_k(R_k) = c_k(R_k)$ holds because if they are assigned some r_j then $\tilde{c}_{kj} = c_{kj}$ holds. Hence, the choice of the current least bundle profit driver i is well-defined. They may respond within $T \in \Omega(m)$ with some $r_j \in R$. In this case, driver i is truthful and profit-maximizing. Hence, $r_j \in \{r_k \in R | c_{ik} = \max_{r_h \in R : f_{ih} > 0} c_{ih}\} \neq \emptyset$ holds. Thus, they may need to consider at most m candidate requests. Hence, the overall time is $O(m \cdot \max\{n \cdot m, T\})$.

The drivers pick up feasible requests one-by-one until termination. Hence, the returned assignment is feasible. Wlog, we let $(r_1, \ldots, r_s)$ denote the picking order. We note that $s \leq m$ holds. We also let h denote the iteration number at termination. We note that $h \geq (s+1)$ holds. We next let $M(p) = (R_1(p), \ldots, R_n(p))$ denote the partial assignment at the start of iteration p. We prove by induction on $p \in \{1, \ldots, h\}$ that $M(p)$ is responsive FEQX wrt the returned responsiveness matrix $U = (\tilde{u}_{ij})_{n\times m}$. Hence, the returned $M(h) = (R_1, \ldots, R_n)$ as well.

In the base case, we let $p = 1$. $M(1) = (\emptyset, \ldots, \emptyset)$ is responsive FEQX wrt U by definition. In the hypothesis, we let $p < h$ and suppose that $M(p)$ is responsive FEQX wrt U. We note that $R \neq \emptyset$ hold and some drivers are not "unresponsive". In the step case, if no request is assigned at iteration p then $M(p+1) = (R_1(p), \ldots, R_n(p))$ is responsive FEQX wrt U by the hypothesis. For this purpose, we suppose that r_j is the request assigned to v_i at iteration p. Let us look at $M(p+1) = (R_1(p), \ldots, R_i(p+1), \ldots, R_n(p))$, where $R_i(p+1) = R_i(p) \cup \{r_j\}$.

For the drivers of $v_k, v_l \in V$ with $k \neq i, l \neq i$, $c_k(F^U_{kk}(p)) \geq c_l(F^U_{kl}(p)) - c_{lh}$ holds for each $r_h \in F^U_{kl}(p)$ by the hypothesis and the fact that the bundles of v_k, v_l do not change between $M(p)$ and $M(p+1)$.

We note that $V \neq \emptyset$ holds because not all drivers are marked as "unresponsive". Otherwise, the algorithm would have terminated and this would imply $p = h$ and lead to a contradiction with $p < h$. We note that $V \subset \{v_1 \ldots, v_n\}$ might hold because some drivers might have been "unresponsive" prior to iteration p and, for this reason, their vehicles might have been removed from $\{v_1 \ldots, v_n\}$ by the algorithm. Wlog, suppose that $V \subset \{v_1 \ldots, v_n\}$ holds. We consider two cases.

Case 1: Pick some $v_k \in \{v_1 \ldots, v_n\} \setminus V$. We note that $v_i \in V$ and, hence, $k \neq i$ holds.

Sub-case 1.1: For the driver of v_i, it follows by the induction hypothesis and the fact that driver i has strictly higher profit in $M(p+1)$ than in $M(p)$ that driver i is responsive FEQX wrt U of driver k. Indeed, $c_i(F^U_{ii}(p+1)) = c_{ij} + c_i(F^U_{ii}(p)) \geq c_k(F^U_{ik}(p)) - c_{kh}$ holds for each $r_h \in F^U_{ik}(p)$ because of $f_{ij} > 0$ and $\tilde{u}_{ij} = 1$.

Sub-case 1.2: For the driver of v_k, it follows by the induction hypothesis and the fact that driver k is "unresponsive" and, therefore, $\tilde{u}_{kj} = 0$ holds at iteration p that driver k is responsive FEQX wrt U of driver i. Indeed, $c_k(F^U_{kk}(p)) \geq c_i(F^U_{ki}(p)) - c_{ih} = c_i(F^U_{ki}(p+1)) - c_{ih}$ holds for each $r_h \in F^U_{ki}(p+1)$ because of $F^U_{ki}(p) = F^U_{ki}(p+1)$.

Case 2: Pick some $v_k \in V$ with $k \neq i$. We again consider two sub-cases.

Sub-case 2.1: For the driver of v_i, we have that $c_i(F_{ii}(p+1)) = c_{ij} + c_i(F^U_{ii}(p))$ holds by $f_{ij} > 0$ and $\tilde{u}_{ij} = 1$, as well as $c_i(F_{ii}(p)) \geq c_k(F_{ik}(p)) - c_{kh}$ holds for each $r_h \in F_{ik}(p)$ by the hypothesis. Hence, driver i is responsive FEQX wrt U of driver k.

Sub-case 2.2: For the driver of v_k, we observe that $c_k(R_k(p)) \geq c_i(R_i(p))$ holds by the fact that v_i is the least bundle profit vehicle within V. Therefore, $c_k(F^U_{kk}(p)) \geq c_i(F^U_{ii}(p))$ also holds because v_i and v_k are assigned only feasible requests when they are "responsive" (i.e. $F^U_{kk}(p) = R_k(p)$ and $F^U_{ii}(p) = R_i(p)$). As v_k might not be feasible for all requests in $F^U_{ii}(p)$, we derive $F^U_{ii}(p) \supseteq F^U_{ki}(p)$. By additivity, it follows that $c_i(F^U_{ii}(p)) \geq c_i(F^U_{ki}(p))$ holds. Consequently, $c_k(F^U_{kk}(p)) \geq c_i(F^U_{ki}(p))$ holds as well.

If $f_{kj} = 0$ or $\tilde{u}_{kj} = 0$, we derive $c_k(F^U_{kk}(p)) \geq c_i(F^U_{ki}(p+1))$ because of $F^U_{ki}(p+1) = F^U_{ki}(p)$. Otherwise, we derive $c_k(F^U_{kk}(p)) \geq c_i(F^U_{ki}(p+1)) - c_{ij}$ because of $F^U_{ki}(p+1) \setminus \{r_j\} = F^U_{ki}(p)$ and additivity. As driver i is profit-maximizing, they pick feasible requests in some non-increasing cost sequence. It follows that $c_{ih} \geq c_{ij}$ holds for any $r_h \in F^U_{ki}(p+1)$. This implies $c_i(F^U_{ki}(p+1)) - c_{ij} \geq c_i(F^U_{ki}(p+1)) - c_{ih}$. We thus derive $c_k(F^U_{kk}(p)) \geq c_i(F^U_{ki}(p+1)) - c_{ih}$ for each $r_h \in F^U_{ki}(p+1)$. □

C.2. Driver fairness and efficiency

Theorem 3. *In centralized SVRPs, Problem 1 belongs to the class of* NP*-hard decision problems.*

Proof. We next show that the decision Problem 1 is intractable. For this purpose, we reduce from the popular NP-hard Partition problem [22].

Problem: Partition
Data: A multi-set $\{c_1, \ldots, c_m\}$ of positive integers, whose integer sum is equal to $2C > m \geq 1$.
Query: Is there a partition of the multi-set into two multi-sets, where the integer sum of each multi-set is C?

Given an instance of Partition with m integers and sum $2C$, construct a centralised instance of Problem 1 with 3 drivers and $(m+4)$ requests: m number-requests $\{r_1, \ldots, r_m\}$ and four special requests $\{s_1, \ldots, s_4\}$. We also let $t = 0$. We next define the costs and feasibilities, respectively.

	FEF1: costs					FEQX: costs					FEF1 & FEQX: feasibilities				
Requests	$\forall j : r_j$	s_1	s_2	s_3	s_4	$\forall j : r_j$	s_1	s_2	s_3	s_4	$\forall j : r_j$	s_1	s_2	s_3	s_4
Driver 1	$\forall j : 1$	C	$(2C - m)$	$4C$	$5C$	$\forall j : 1$	C	$(2C - m)$	$3C$	$3C$	$\forall j : 0$	1	1	1	1
Driver 2	$\forall j : c_j$	$2C$	$2C$	$3C$	$3C$	$\forall j : c_j$	$2C$	$2C$	$2C$	$1C$	$\forall j : 1$	1	1	1	1
Driver 3	$\forall j : c_j$	$2C$	$2C$	$3C$	$3C$	$\forall j : c_j$	$2C$	$2C$	$2C$	$1C$	$\forall j : 1$	1	1	1	1

The costs are normalized: $12C$ for FEF1; $9C$ for FEQX. We note that $2C > (2C - m) > 0$ holds due to $2C > m \geq 1$. Furthermore, we note that if $c_j = 1$ for some r_j with $j \in \{1, \ldots, m\}$ then a feasible assignment cannot give r_j to driver 1 because their vehicle is not feasible for r_j.

Thus, as $C > 0$, a feasible assignment gives $\text{FSW}_{\max}$ iff the special-requests s_3 and s_4 are given to driver 1, and the number-requests plus s_1 and s_2 go to drivers 2 and 3. This equivalence holds for both the above customer costs for FEF1 and FEQX.

Let there be an equal partition of the integers. Then, we can assign to each of the drivers 2 and 3 a profit of C from the number-requests plus a profit of $2C$ from s_1 and s_2, for a total profit of $3C$. The other special requests s_3 and s_4 can be assigned to driver 1. We label this by (R_1, R_2, R_3).

FEF1: (R_1, R_2, R_3) is complete, feasible, and gives $\text{FSW}_{\max}$. Driver 1 does not have feasible envy at all. Drivers 2 and 3 do not have feasible envy once any special request is removed from driver 1's bundle. Hence, (R_1, R_2, R_3) is FEF1. Conversely, let there be a complete, feasible, and FEF1 assignment that gives $\text{FSW}_{\max}$. Drivers 2 and 3 have each a profit of $6C$ for driver 1's bundle. Once a feasible request (for them) is removed from this bundle, their profit for it is $3C$. Hence, each of them must get a bundle profit of at least $3C$, so each of them must get a profit of C from the number-requests and a profit of $2C$ from s_1 and s_2. There must be an equal partition of the integers.

FEQX: (R_1, R_2, R_3) is complete, feasible, and gives $\text{FSW}_{\max}$. Driver 1 does not have feasible jealousy at all. Drivers 2 and 3 do not have feasible jealousy once any special request is removed from driver 1's bundle. Hence, (R_1, R_2, R_3) is FEQX. Conversely, let there be a complete, feasible, and FEQX assignment that gives $\text{FSW}_{\max}$. Driver 1 has a profit of $6C$. Once a feasible request (for driver 2 or 3) is removed from this bundle, their profit for it is $3C$. Hence, each of drivers 2 and 3 must get a bundle profit of at least $3C$, so each of them must get a profit of C from the number-requests and a profit of $2C$ from s_1 and s_2. There must be an equal partition of the integers. □

C.3. Driver fairness and customer efficiency

Theorem 4. *In centralized SVRPs, Problem 2 belongs to the class of* NP*-hard decision problems.*

Proof. We next show that the decision Problem 2 is intractable. For this purpose, we reduce from the popular NP-hard Partition problem [22].

> Problem: Partition
> **Data**: A multi-set $\{c_1, \ldots, c_m\}$ of positive integers, whose integer sum is equal to $2C > m \geq 1$.
> **Query**: Is there a partition of the multi-set into two multi-sets, where the integer sum of each multi-set is C?

Given an instance of Partition, construct a centralised instance of Problem 2 with 3 drivers and $(m + 4)$ requests: m number-requests $\{r_1, \ldots, r_m\}$ and four special-requests $\{s_1, \ldots, s_4\}$. We also let $t = 0$. We next define the costs and feasibilities, respectively.

	FEF1: costs					FEQX: costs					FEF1 & FEQX: feasibilities				
Requests	$\forall j : r_j$	s_1	s_2	s_3	s_4	$\forall j : r_j$	s_1	s_2	s_3	s_4	$\forall j : r_j$	s_1	s_2	s_3	s_4
Driver 1	$\forall j : 1$	$4C$	$5C$	C	$(2C - m)$	$\forall j : 1$	$4C$	$(4C - m)$	$3C$	$3C$	$\forall j : 0$	1	1	1	1
Driver 2	$\forall j : c_j$	$2C$	$2C$	$3C$	$3C$	$\forall j : c_j$	$2C$	$2C$	$4C$	$4C$	$\forall j : 1$	1	1	1	1
Driver 3	$\forall j : c_j$	$2C$	$2C$	$3C$	$3C$	$\forall j : c_j$	$2C$	$2C$	$4C$	$4C$	$\forall j : c_j$	$2C$	$2C$	$4C$	$4C$

The costs are normalised: $12C$ for FEF1; $14C$ for FEQX. We note that $3C > (2C - m) > 0$ and $(4C - m) > 2C > 0$ hold due to $2C > m \geq 1$. Also, we note that if $c_j > 1$ for some r_j with $j \in \{1, \ldots, m\}$ then a feasible assignment cannot give r_j to driver 1 because $f_{1j} = 0$ holds.

Thus, as $C > 0$, a complete and feasible assignment gives $\text{FSW}_{\min}$ iff the special-requests s_3 and s_4 are given to driver 1, and the number-requests plus s_1 and s_2 go to drivers 2 and 3. This "iff" holds for both the costs for FEF1 and FEQX.

1) Let there be an equal partition of the integers. Then, it is possible to assign to each of the drivers 2 and 3 a profit of C from the number-requests plus a profit of $2C$ from s_1 and s_2, for a total profit of $3C$. The other special requests s_3 and s_4 can be assigned to driver 1. We let (R_1, R_2, R_3) denote this one: it is complete, feasible, and gives $\text{FSW}_{\min}$.

FEF1: Driver 1 does not have feasible envy up to one request because each other driver gets exactly one request for which v_1 is feasible. Drivers 2 and 3 do not have feasible envy once any special request is removed from 1's bundle. Hence, (R_1, R_2, R_3) is FEF1.

FEQX: Driver 1 do not have feasible jealousy at all because they receive the highest profit of $6C$. Drivers 2 and 3 do not have feasible jealousy once any special request is removed from driver 1's bundle. Hence, the assignment (R_1, R_2, R_3) is FEQX.

2) Conversely, let there be a complete, feasible, and FEF1/ FEQX assignment that gives $\text{FSW}_{\min}$. We denote this assignment by (R_1, R_2, R_3).

FEF1: Drivers 2 and 3 have each a profit of $6C$ for driver 1's bundle. Once a feasible request (for driver 2 or 3) is removed from this bundle, their profit for it is $3C$.

FEQX: Driver 1 have a profit of $6C$. Once a feasible request (for driver 1) is removed from this bundle, driver 1's profit for it is $3C$.

We derive that each of drivers 2 and 3 must get a bundle profit of at least $3C$, so each of them must get a profit of C from the number-requests and a profit of $2C$ from s_1 and s_2. Hence, there must be an equal partition of the integers. In fact, $R_2 \setminus \{s_1, s_2\}$ and $R_3 \setminus \{s_1, s_2\}$ form it. □

Appendix D. Routing under few feasibilities

Theorem 5. *If a feasible matching is not fixed, a brute-force algorithm may take $O(2^{n \cdot f} \cdot n \cdot (2f)!)$ time to compute an optimal plan for totTRAVEL/maxTRAVEL.*

Proof. Pick some $n, f \in \mathbb{N}_{>0}$. Suppose that n and f are some even numbers (i.e. 2, 4, 6, etc.). In a centralized instance, let there be n vehicles and $\frac{n}{2} \cdot f$ requests. Define costs as follows: each driver charges one dollar for any request. Define feasibilities as follows: v_1, v_2 are feasible for the first f requests, v_3, v_4 for the second f requests, and so on. We let the locations of the requests be arbitrary. This instance has $O(2^{n \cdot f})$ complete and feasible (and also fair) matchings. To see this, we observe that the number of such matchings for a given pair of vehicles (v_i, v_{i+1}) with $i \in \{1, 3, \ldots, n-1\}$ is $O((2^f)^2)$. The above bound follows by observing that there are $\frac{n}{2}$ such pairs and the fact that the set of feasible requests of one such pair is disjoint from the set of feasible requests of another such pair.

A brute-force algorithm first needs to visit each of these matchings, then computes a feasible plan for it that minimizes totTRAVEL/maxTRAVEL, and finally returns an optimal plan for totTRAVEL/maxTRAVEL. Pick such matching. We note that this satisfies constraints (a-c). Thus, computing a feasible plan that minimizes totTRAVEL/-maxTRAVEL takes $O(n \cdot (2 \cdot f)!)$ time. For each bundle in the matching, there are $O((2 \cdot f)!)$ ($O(f!)$) possible routes. This happens when the bundle size is f ($f/2$) and each request in the bundle has some different pickup and delivery locations. Thus, for each of the n vehicles, the algorithm computes some minimizing route that satisfies the constraints (d-e). This results in a feasible plan that minimizes totTRAVEL/maxTRAVEL. The optimal plan is one of these plans, that minimizes the objective across all such matchings. □

Theorem 6. *If a feasible matching is fixed, a brute-force algorithm takes $O(n \cdot (2 \cdot f)!)$ time to compute a minimizing feasible plan for totTRAVEL/maxTRAVEL.*

Proof. If a feasible matching is fixed, then constraints (a-c) are satisfied. In such a matching, each driver receives at most f requests. Therefore, there are $O((2 \cdot f)!)$ possible routes for each vehicle. This happens when the driver bundle has size f and each request in it has some different pickup and delivery locations. Thus, for each of the n vehicles, a brute-force algorithm computes some minimizing route that satisfies the remaining constraints (d-e). This results in a feasible plan that minimizes totTRAVEL/maxTRAVEL. Hence, the overall running time is $O(n \cdot (2 \cdot f)!)$. □