Reducing the Minimal Fleet Size by Delaying Individual Tasks

Maximilian Kronmueller*, Andres Fielbaum*,◇ and Javier Alonso-Mora*

Citation: Kronmueller, M., Fielbaum, A., & Alonso-Mora, J. (2024). Reducing the Minimal Fleet Size by Delaying Individual Tasks. *IEEE Transactions on Intelligent Transportation Systems.* doi: 10.1109/TITS.2024.3368101

***Abstract*— This work formally defines the problem of fleet sizing with delays (FSD), where the option of delaying individual tasks within fleet sizing is considered. We prove that the FSD problem is NP-hard and solve a formulation of the FSD problem as a mixed integer linear problem (MILP). We then analyze the proposed method in detail in an abstract case and validate it in a case study of taxi rides in Manhattan. We show that fleet sizes can be decreased significantly and that the trade-off space of the number of required vehicles to execution time and added delay can be enlarged.**

I. INTRODUCTION

Fleets of vehicles of various types are used to drive today's world. For example, taxi fleets offering mobility in cities, robot fleets operating entire warehouses or bike courier fleets delivering your pizza for dinner. Thereby, a wrongly sized fleet, regardless if it is too small or too large, causes inefficient operations regarding bad service or additional costs. Thus the general question of fleet sizing, i.e. "How many vehicles does an operation optimally need?", is posed. Typically this question involves a trade-off between the quality of the solution that can be provided and the posed costs. Previous works showed the great potential of fleet sizing, [1] showcase a great reduction in required taxis within Manhattan, or [2] optimizing the size of robot fleets in flexible manufacturing systems.

All works tackling the fleet sizing question assume that tasks have a fixed starting and ending time. In this work, we pose a new problem called *fleet sizing with delays* (FSD), which allows to actively delay individual tasks slightly, if beneficial. Let us exemplify through a situation of daily life: Person A wants to go from home to sports leaving at 18:00, and person B arrives at the same home at 18:01. If A cannot delay her trip, two vehicles are required to fulfil this demand. This changes if we allow person A to leave slightly delayed at 18:01. Then, the two persons could use the same vehicle sequentially.

This paper introduces and formally defines the FSD problem in a general fashion. We give a general formulation that applies to operations, which can be described as a set of tasks, with vehicles that can execute the given tasks on their own and move independently. Through such a general formulation, this work covers a wide range of operations. For example, in the case of an operation of shared taxi rides, a task can be a series of customers a single taxi fulfills, transporting a set of customers to their locations until it becomes empty again. In the case of autonomous robots in a greenhouse, a task can represent a set of crops to be harvested or pesticides to be applied. In a warehouse or factory, a task can be a single request of parts to an engineer. In the case of an on-demand delivery service, a task can represent multiple orders one bike rider can transport at the same time.

Contribution Statement: This work proposes a new problem, *fleet sizing with delays*. It introduces the option of delaying individual tasks within fleet sizing. The problem is formally defined. We prove that the FSD problem is NP-hard. Further, we propose and solve a formulation of the FSD problem as a mixed integer linear problem (MILP). A large experimental analysis is conducted, analyzing an abstract case in detail and investigating a case study of taxi rides in Manhattan. We show that introducing the option to delay tasks has two effects on obtained results. Firstly, fleet sizes can be decreased significantly. Secondly, the trade-off space of the number of required vehicles to execution time and added delay is increased.

II. RELATED WORK

A. Overview

The fleet sizing problem was originally introduced by [3] in 1984. It poses the question of the number of required vehicles (identical operating characteristics) to satisfy a given demand. Since then, the problem itself has only slightly evolved, mainly changing in the application it is asked for and the scale it can be tackled at. For example, looking at fleet sizing in the maritime context [4], [5], considering the need to charge for electrical vehicles [6], [7], fleet sizing together with service region partitioning [8], or pooled taxi rides [9]–[11]. Extending on the fleet sizing question, the question of Fleet Design emerged [12]–[14]. Fleet Design goes beyond determining the number of vehicles of one predefined type and incorporates considerations about the types of vehicles to employ from a

* Department of Cognitive Robotics at the Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands

◇ School of Civil Engineering, University of Sydney

range of options and the corresponding quantities needed. To tackle the fleet sizing problem various approaches in various areas have been developed and applied. We categorize these approaches into three categories: chaining-based approaches, simulation-based approaches, and others. As an overview, we highlight each class.

Chaining-based approaches build on the idea of sequencing tasks to build so-called chains, each chain representing the journey of a single vehicle. The obtained number of chains equals the required fleet size if done for all tasks. The proposed approach belongs to this class. As such, we have a closer look below, see Section II-B.

Simulation-based approaches implement a simulation of an operation reflecting its real counterpart as well as possible, [15], [16] (on-demand mobility services), [17], [18] (shared rides) and [2] (flexible manufacturing systems). Once this is obtained, various input parameters are changed, and the resulting Key Performance Indicators are logged. By this means, fleets of different sizes and their influence can be analyzed. The downside to these methods is that no theoretical guarantees can be given and that it might require a lot of computational resources, as the simulation needs to be repeated for each set of parameters. The upside is the capability to represent the simulation environment in more detail, for example, traffic flow or traffic lights.

Other approaches found in the literature are analytical approaches, such as [14] looking at robots in a warehouse environment. They derive an optimal analytical solution for the case of infinite pick-up stations. Further, we see approaches doing a structured search, for example, [19] applies a binary search to find the best fleet size for a fleet of mobile robots. Further, genetic algorithms have been used in fleet sizing. A comparison of a genetic algorithm and a chaining-based approach (Hopcroft-Karp algorithm for the minimum path cover problem) was made by [20] revealing that the genetic algorithm was outperformed in regards to the quality of solution and the required computational time. [21] builds an optimization model to estimate required fleet sizes based on GPS data. [22] formulates a mixed integer linear problem to solve the joint fleet sizing and charging system planning problem.

This paper is concerned with centrally controlled systems. That is, we do not review works dealing with cases in which vehicles decide themselves on their plan, such as [23] or [24], both papers modeling the behavior of ride-hailing drivers.

B. Chaining-based approaches

For the following of this literature review, we will focus on chaining-based approaches alone, as our method belongs in this category. The general idea of chaining-based approaches was first introduced by [1] to tackle the *minimum fleet problem* for taxi rides in Manhattan. [1] first create a so-called *Vehicle-shareability network*, which is a graph capturing which rides could be executed by a single vehicle

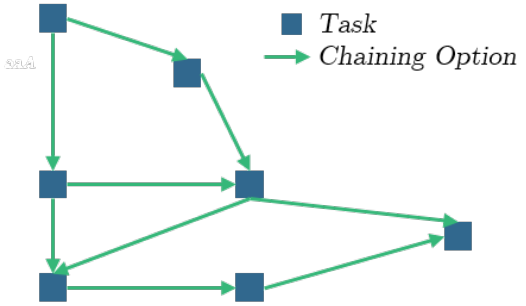

Fig. 1: Illustration of a vehicle-shareability network [1], vertices are tasks or rides, and edges represent potential chaining options, i.e. a single vehicle can serve both tasks in succession.

in succession. Rides or tasks are the graph's vertices, and edges represent potential chaining options. A vertex i is connected to another vertex j, if a vehicle can serve ride j after ride i. An example is illustrated in Figure 1. They show how through these networks, the minimum fleet sizing problem can be transformed into a minimum path cover problem. As the resulting graph is acyclic, the minimum path cover problem can be solved through a maximum-matching algorithm (Hopcroft-Karp algorithm [25]).

Since its introduction, chaining-based approaches have found great popularity, especially within the mobility of people community, and were adapted and extended. Next, we analyze adaption from a problem perspective and from a methodological one, subsequently.

Problem perspective: Finding the minimum fleet for non-shared taxi rides in the city of Manhattan by relocating taxis to serve subsequent customers, is tackled by [1]. A group of papers extend chaining-based approaches to the application of ridesharing, i.e. the option for multiple customers to share one vehicle simultaneously. A minimum fleet sizing problem with ridesharing was studied by [9], [10] and [11], while the latter focused on the inclusion of demand predictions. [26] formulates the pooling and chaining questions in a combined fashion and shows the effect of pooling on obtained fleet sizes. Fleet sizing for On-Demand Multimodal Transit Systems, combining fixed routes with on-demand vehicles, was studied by [27].

From a **methodological** point of view, the most relevant question for chaining-based approaches is how the chains are found. As a starting point, the problem is often modelled as a graph, like the vehicle-shareability-network, capturing chaining options. Transforming the graph into a bipartite graph and applying a maximum-matching algorithm (like the Hopcroft-Karp algorithm) results in a minimal fleet, like [1], [11], [28]. Other works formulate an ILP and solve it as a minimum flow problem [27], [29]. If other objectives are considered beyond the fleet size, such as the costs to chain tasks, other techniques are required. With costs, the problem changes to a minimum-weight bipartite matching problem. It can be formalized as an ILP considering costs in the objective function and solved as such or as a bipartite

matching problem with costs, approached by means of a Hungarian algorithm. Forming chains iteratively by means of solving an ILP repeatedly was done by [9]. They use the chains formed up to this point as input and prolong them by adding new tasks at each step. They start at the end of the day, and with each step, they go further back in time.

This work extends the chaining-based approaches by allowing to delay individual tasks. This changes the problem in both regards, from the tackled problem perspective and methodological. From an application-driven point of view, this enables to decrease minimal fleet sizes even further, by adding delay. From a methodological point of view, the newly posed problem can not be tackled by the existing approaches as the decision, which tasks are delayed and if, by how much, is added.

III. PROBLEM FORMULATION

In this section, we formally introduce and define the *fleet sizing problem with delays* (FSD). Intuitively described, the FSD poses a problem in which a fleet of vehicles and their operational plans need to be found to fulfil a given set of tasks while allowing to delay individual tasks. Solutions are optimized based on a given objective.

A. Formal Problem Formulation

We first introduce the inputs, then the decision variables, and last the cost function of the problem.

Inputs: The operational environment is represented by a weighted directed graph $G = (V, E)$. The set of vertices V represents all locations, and the set of weighted edges E represents connections between them. An edge exists if an vehicle can move from one vertex to another directly, and the edge's corresponding weight $c(e)$ is the time needed to traverse it.

All tasks are summarized in the set $\mathcal{T}$. Each individual task T is characterized as $T = (l_T^{start}, l_T^{end}, t_T^{start}, \alpha_T, \sigma_T)$: the first two variables representing the starting and ending location[1] (l_T^{start} and l_T^{end}), while the following two represent the starting time and duration of the task (t_T^{start} and α_T). Last, σ_T defines the maximum time a task can be delayed[2]. The resulting ending time t_T^{end} is the starting time plus the task duration, i.e. $t_T^{end} = t_T^{start} + \alpha_T$. All tasks start within $t_T^{start} \in [0, \Lambda], \quad \forall T \in \mathcal{T}$, with Λ defining the end time to request any task. An illustration of a single task, its delayed variant, and all associated points in time is shown in Figure 2.

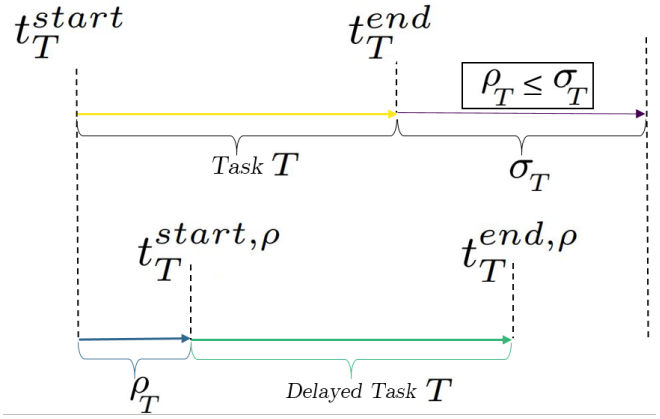

Fig. 2: Illustration of a single task, its delayed variant, and associated points in time. For clarity, we use $t_T^{start,\rho}$ and $t_T^{end,\rho}$, as the notation for the new starting and ending time of a delayed task.

Decision Variables: The decision variables of the FSD problem are the number of vehicles and their trajectories, also defining the required delay per task. A single trajectory is an operational plan for one vehicle. Formally, a trajectory is an ordered sequence of tasks, so that it is feasible for a single vehicle to serve all of them within the given maximum delays. We denote a trajectory as $g_i = (T_{i,1}, T_{i,2}, ..., T_{i,k})$, thereby $T_{i,k}$ denotes that task T is part of the trajectory g_i and is served at position k. A trajectory starts at the starting location and at the starting time of the first task it serves, after serving this task, the vehicle relocates from the ending location to the starting location of the subsequent task, if needed the vehicle waits until the task starts and serves it, subsequently the vehicle relocates again. This procedure is continued until the vehicle serves the last task, after which the vehicle stops. If the vehicle arrives at the starting location of a task later than its starting time, the trip is delayed by the difference. The time a task is actually delayed is denoted as ρ_T. A set of trajectories is summarized in the set ω. To be feasible, ω must fulfill that all tasks are served, i.e. the union of all the trajectories results in $\mathcal{T}$, mathematically $\cup_{i \in \omega} g_i = \mathcal{T}$; and that no task is delayed more than its maximum delay, i.e., $\rho_T \leq \sigma_T, \forall T \in \mathcal{T}$.

Cost Function: The cost function $C(\omega)$ assigns a cost to a set of trajectories ω. It includes costs for the number of vehicles used (fixed capital costs), their total traveling time between tasks (relocation time), and costs for the total delay added. For each vehicle used, i.e. for each executed trajectory, a capital cost of M_{fix} is charged. To ease notation, we summarize the total relocation time of a single trajectory g_i as $\phi(g_i)$. We denote the relocation time from task i to task j as τ_{T_i,T_j} or short as $\tau_{i,j}$. As such, the total relocation time of a single trajectory can also be expressed as $\phi(g_i) = \sum_{j=1}^{|g_i|-1} \tau_{T_{i,j},T_{i,j+1}}$. Note that the execution time of the vehicles during tasks is not part of the cost function, because this is a constant value. Additionally, two weights are used to weigh the influence of the total delay and the total relocation time, which we denote as M_ρ and M_ϕ. Their exact

[1]The starting and ending locations can be different or the same, there is no inherent need to return to the starting location.

[2]Without a maximum delay, a single vehicle can serve the full set of tasks by serving all subsequently.

values depend on the specific cost structure of the operation[3], which is also the case for M_{fix}. In Section V we indicate those values for our experiments. This results in the objective function as follows:

$$c(\omega) = M_{fix} \cdot |\omega| + M_\rho \cdot \sum_{T \in \mathcal{T}} \rho_T + M_\phi \cdot \sum_{g \in \omega} \phi(g) \quad (1)$$

Problem: We define Ω as the set of all sets of trajectories ω that are feasible solutions to the FSD. As such the fleet sizing problem with delays can be posed as:

$$\min_{\omega \in \Omega} c(\omega) \quad (2)$$

Note that an instance of the FSD problem is fully characterized by $G = (V, E), \mathcal{T}, M_{fix}, M_\rho, M_\phi$.

B. Problem Complexity

Here we analyze the complexity of the newly introduced FSD problem. We first compare the FSD problem on an intuitive level to its closest related problem, fleet sizing without delays. Second, we claim and prove that FSD is NP-hard (Section III-B.1).

On an intuitive level, introducing delay changes three points that are worth highlighting, when comparing FSD to fleet sizing without delays:

- First, allowing delays will result in more options to chain individual tasks. For no delays, each pair of tasks for which the starting location of the subsequent task can not be reached in time can not be chained. For two tasks i and j of the set of tasks $\mathcal{T}$, this can be expressed as
$$t_i^{end} + \tau_{i,j} \leq t_j^{start} \qquad i \in \mathcal{T}, j \in \mathcal{T}$$
This is relaxed with the option of delaying tasks, resulting in
$$t_i^{end} + \tau_{i,j} \leq t_j^{start} + \sigma_j \qquad i \in \mathcal{T}, j \in \mathcal{T}$$
- Second, the individual decisions on whether to chain a pair of tasks, are independent of each other in case of no delays. This follows as the starting and ending times are not changed by chaining. Thus, when deciding to chain task i to task j, it is irrelevant which partial trajectory the corresponding vehicle followed before task i. With delays, this is not the case, as the ending time of a task changes if the task is delayed, which in turn might modify the starting time of all the subsequent tasks assigned to the same vehicle. We remark that this fact precludes utilizing the approach based on a bipartite graph, as done by [1].
- Third, the set of all chaining options may contain a pair of tasks i and j twice, once as (i, j) and once as (j, i).

[3]For example, for taxi rides, delays are more important than for logistical operations, resulting in a higher value of M_ρ.

1) Proof: The Fleet Sizing with Delays Problem is NP-Hard: Here we prove that the FSD problem is within the complexity class of NP-Hard problems. We do so by showing that the FSD can be reduced to the problem of determining whether a Hamiltonian path exists, which is known to be NP-Complete.

A *Hamiltonian path* is defined as a path that visits each vertex of a given directed graph $G = (V, E)$ exactly once. As each vertex is exactly visited once, this implies that the path is exactly of length $|V|-1$.

Let us consider an instance of the Hamiltonian Path problem $G = (V, E)$. We **prove** that: a Hamiltonian path exists on the graph **if and only if** the FSD problem[4] $(G, \mathcal{T}, M_{fix}, M_\rho, M_\phi)$ admits a solution with a cost equal or lower to $M_{fix} + (|V|-1 \cdot |V|/2) + |V|-1$.

Reduction:

$$c(e) = 1 \quad \forall e \in E \quad (3a)$$
$$\mathcal{T} = \{T_v\}_{v \in V}, \text{ with:} \quad (3b)$$
$$l_{T=v}^{start} = l_{T=v}^{end} = v \quad \forall T \in \mathcal{T} \quad (3c)$$
$$t_T^{start} = t_T^{end} = 0 \quad \forall T \in \mathcal{T} \quad (3d)$$
$$\sigma_T = |V|-1 \quad \forall T \in \mathcal{T} \quad (3e)$$
$$M_\rho = M_\phi = 1 \quad (3f)$$
$$M_{fix} = |V|^2 + |V| > (|V|-1 \cdot |V|/2) + |V|-1 \quad (3g)$$

The cost to traverse any edge of the graph is set to one (Equation 3a). Equation 3b specifies that there is exactly one task per vertex in the graph ($|V| = |\mathcal{T}|$); this allows us to index the tasks using the vertices. For each of these tasks, the start and the end location are at the corresponding vertex (Equation 3c). Equation 3f sets the weights of added delay and relocation time equally to one. Equation 3g specifies that M_{fix} is larger than a given threshold, ensuring that reducing the fleet size is always more important than the other parts of the objective function.

To prove the above-posed claim, we need to verify the claim in both ways. First, that if the FSDW is solvable with a cost less or equal than $M_{fix} + (|V|-1 \cdot |V|/2) + |V|-1$, then a Hamiltonian path exists on G. Second, that if a Hamiltonian path exists on the graph G, then the FSD can be solved with a cost less or equal to $M_{fix} + (|V|-1 \cdot |V|/2) + |V|-1$.

Direction 1: *FSDW is solvable with a cost less or equal than* $M_{fix} + (|V|-1 \cdot |V|/2) + |V|-1 \Rightarrow$ *Hamiltonian path exists on G:*

A solution to the FSD with the target costs of $M_{fix} + (|V|-1 \cdot |V|/2) + |V|-1$ is only achievable under specific conditions. First, only a single vehicle can be used, causing costs of M_{fix}. Using an additional vehicle would add costs of M_{fix}, leading to total costs larger than $2M_{fix}$, which is, in turn, larger than the target costs (Equation 3g), $2M_{fix} > M_{fix} + (|V|-1 \cdot |V|/2) + |V|-1$. Second, a solution of FSD must serve all tasks, and therefore all vertices of the graph are

[4]One instance of the FSD is fully characterized by $G = (V, E), \mathcal{T}, M_{fix}, M_\rho, M_\phi$ (see Section III-A).

visited, as each task takes place in exactly one vertex. That is, the existence of the solution with the said costs ensures that a single vehicle is able to visit all of the nodes within the maximum allowed delays. This clearly ensures that a Hamiltonian Path must exist: otherwise, the vehicle would need to traverse at least V arcs, violating the maximum delay for the last visited task.

Direction 2: *A Hamiltonian path exists on graph $G \Rightarrow$ the FSD can be solved with a cost less or equal than $M_{fix} + (|V|-1 \cdot |V|/2) + |V|-1$:*

The desired solution is found by making a single vehicle follow exactly the Hamiltonian Path. Each vertex is visited by a Hamiltonian path; thus, each task gets served. The first task is served at $t = 0$. The next task is met a time unit later as a Hamiltonian path will traverse exactly one edge to visit the next vertex. This pattern continues till the last task is served at $t = |V|-1$, thus satisfying the constraints. As a Hamiltonian path traverses $|V|-1$ edges, each having the cost of one, this results in costs for total relocation time and delay of exactly $(|V|-1 \cdot |V|/2) + |V|-1$. As a single path is formed, a single vehicle is required, adding costs of M_{fix}. As such, A Hamiltonian path is a solution to the FSD problem and satisfies the cost cap.

Conclusion: As the FSD problem simplifies, under the above-described reduction, to the problem of whether a Hamiltonian path exists or not, which is NP-Complete, we conclude that the FSD problem is NP-Hard.

IV. METHOD: FLEET SIZING WITH DELAYS AS A MIXED INTEGER LINEAR PROBLEM

A. Mixed Integer Linear Problem

Here we formalize the FSD as a MILP based on the idea of *chaining*. We connect the individual tasks into chains, each forming a trajectory g for one vehicle. For a single vehicle to be able to serve two tasks i and j in succession, it needs to be able to relocate from the ending location of task i to the starting location of task j and arrive before its starting time plus the maximal delay. This can be expressed as:

$$t_i^{end} + \tau_{i,j} \leq t_j^{start} + \sigma_j \qquad i \in \mathcal{T}, j \in \mathcal{T} \tag{4}$$

Recall, $\tau_{i,j}$ denotes the relocation time from task i to task j. It is calculated as the sum of all costs of traversed edges if following the shortest path in between the ending location of task i and the starting location of task j.

We define the set of all pairwise feasible chaining options of tasks as $\mathcal{X}$, it contains all pairs of tasks (i, j) that fulfill Equation 4.

To formulate this idea into a tractable problem, we introduce an integer decision variable $x_{i,j}$ for each pair of tasks (i, j). $x_{i,j}$ takes the value of 1 if task i and j are served by one vehicle in succession (i before j), i.e. the two tasks are chained together. Otherwise, it takes the value of 0.

Taking the cost as defined previously (Equation 1), we award a constant value of -M_{fix} for every two tasks that are chained.[5] Further, we penalize delay and total relocation time (total driven time between tasks). This allows us to formulate the problem as a mixed integer linear problem (MILP) as follows, using the newly introduced decision variable and notation:

$$\min_{x,\rho} \sum_{(i,j)\in\mathcal{X}} x_{i,j} \cdot \Big[-M_{fix} + M_\phi \cdot \tau_{i,j} \Big] + M_\rho \cdot \sum_{i\in\mathcal{T}} \rho_i \tag{5a}$$

$$\sum_{j=1}^{\mathcal{T}} x_{i,j} \leq 1 \quad \forall i \in \mathcal{T} \tag{5b}$$

$$\sum_{i=1}^{\mathcal{T}} x_{i,j} \leq 1 \quad \forall j \in \mathcal{T} \tag{5c}$$

$$0 \leq \rho_i \leq \sigma_i \quad \forall i \in \mathcal{T} \tag{5d}$$

$$x_{i,j} \cdot \rho_i \leq x_{i,j} \cdot \Big[t_j^{start} + \rho_j - t_i^{end} - \tau_{i,j} \Big] \quad \forall i \in \mathcal{T},\ \forall j \in \mathcal{T} \tag{5e}$$

Equation 5a describes the objective function of the MILP. -M_{fix} is awarded each time two tasks are chained. Further, we add the time the vehicles need to drive between the two tasks ($\tau_{i,j}$) and the sum of the delay of all tasks ($\rho_i \; \forall i \in \mathcal{T}$). Note that the minimization problem can yield a negative value for the objective function due to awarding $-M_{fix}$, if two tasks are chained. The trivial feasible solution where all variables are zero (no chaining and no delays) yields an objective value of zero, which is not optimal. Equation 5b ensures that each task is maximally chained to one subsequent task. Equation 5c ensures that each task has a maximum of one preceding task. Note, that the Equations 5a to 5c recover the traditional ILP for fleet sizing if no delays are allowed. A constraint as Equation 5e can be omitted in the no delay case, as for no delays ($\rho = 0$), it is always trivially fulfilled because all (i, j) belong to $\mathcal{X}$, hence satisfying Equation 4 which is equivalent to Equation 5e for no delays. The set $\mathcal{X}$ does not contain any decision variable and, as such, can be calculated beforehand. Equation 5d ensures that the actual delay of a task (ρ_i) is not larger than its maximum delay (σ_i) and not smaller than zero. Equation 5e ensures that an vehicle arrives at the next task in time. To be precise, it means that if $x_{i,j} = 1$, the vehicle leaves the final location of i at its ending time plus its potential delay ($t_i^{end} + \rho_i$), drives to the new location taking time $\tau_{i,j}$, and arrives in time to the starting position of task j, including potential added delay of this task ($t_j^{start} + \rho_j$); if $x_{i,j} = 0$, this constraint has no implication. An illustration of this constraint is shown in Figure 3.

Unfortunately, the constraint formulated in Equation 5e is not linear. To linearize it, we define two new auxiliary variables $A_{i,j}$ and $B_{i,j}$ and apply a temporal scaling to ensure $\sigma \leq 1$. Intuition-wise, the new variables represent $A_{i,j} = x_{i,j} \cdot \rho_i$ and $B_{i,j} = x_{i,j} \cdot \rho_j$. As this would not be linear, these equations are represented by the four Inequalities 6a-6d and respectively 6e-6h. The temporal scaling ensures that

[5] Each chained formed means one vehicle less is required.

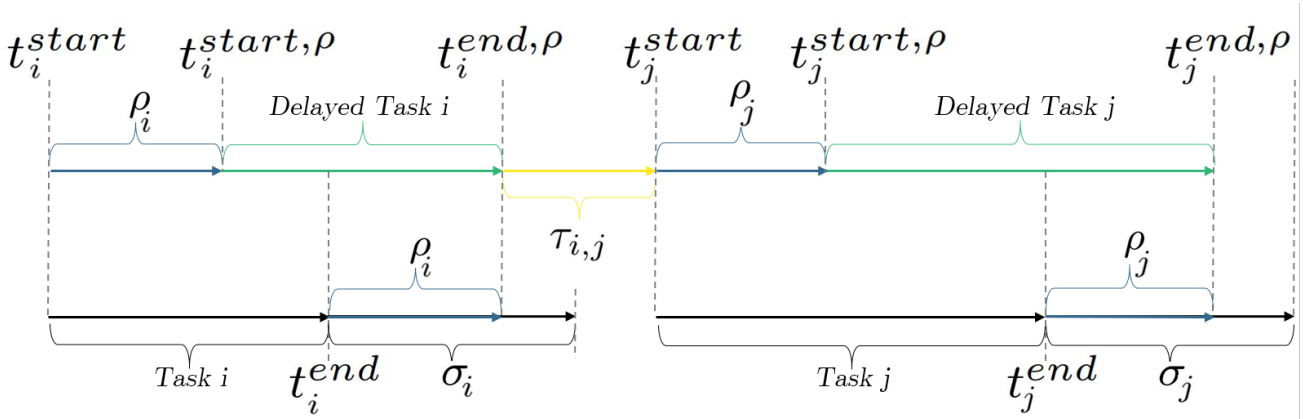

Fig. 3: Illustration clarifying the constraint of Equation 5e. Two tasks, i and j, their delayed variants, and the relocation time between are shown. An vehicle follows the top line of arrows if it serves delayed task i relocates to task j, which was also delayed by ρ_j.

equations 6c and 6g can always be satisfied also for the case of $x_{i,j} = 0$.

$$A_{i,j} \leq x_{i,j} \tag{6a}$$
$$A_{i,j} \leq \rho_i \tag{6b}$$
$$A_{i,j} \geq x_{i,j} + \rho_i - 1 \tag{6c}$$
$$A_{i,j} \geq 0 \tag{6d}$$
$$B_{i,j} \leq x_{i,j} \tag{6e}$$
$$B_{i,j} \leq \rho_j \tag{6f}$$
$$B_{i,j} \geq x_{i,j} + \rho_j - 1 \tag{6g}$$
$$B_{i,j} \geq 0 \tag{6h}$$

Two cases can occur: First, $x_{i,j} = 0$, as such Equation 6a demands $A_{i,j} = 0$. Second, if $x_{i,j} = 1$ then $A_{i,j} = \rho_i$ forced by the Equations 6b and 6c. The same holds for $B_{i,j}$ and ρ_j. Using the auxiliary variables, Equation 5e can be rewritten in linear form, see Equation 7.

$$A_{i,j} \leq B_{i,j} + x_{i,j} \cdot \left[t_j^{start} - t_i^{end} - \tau_{i,j} \right] \tag{7}$$

All used notation of this work is summarized in Table I in Appendix A.

B. Heuristics

As the size of the problem can get too large to be solvable through the above-proposed method, or the required solving time can get too vast due to its NP-Hard nature, we propose some heuristics for keeping the problem smaller. All heuristics are designed to reduce the number of potential chains, $|\mathcal{X}|$, needed to be considered by the optimizer or by tightening the given constraints. We propose the following four heuristics:

- Artificially reducing the maximally allowed delay time by introducing a new maximum delay variable ρ_{max}

$$\rho_i \leq \rho_{max} \ \forall i \in \mathcal{T}, \qquad \rho_{max} < \max_{i \in \mathcal{T}} \sigma_i$$

- Only allowing tasks to be chained if the end location of the ending task is less than a given relocation time, τ_{max}, apart from the start location of the following task.

$$\tau_{i,j} \leq \tau_{max} \ \forall i \in \mathcal{T}, \forall j \in \mathcal{T}, \qquad \tau_{max} < \max_{i,j \in \mathcal{T}} \tau_{i,j}$$

- For each starting task i, we limit the number of pairs $(i, j) \in \mathcal{X}$ to a maximum of z pairs. Tasks are ranked by cost, and the z bests are considered. Here costs are calculated as $\tau_{i,j} + \rho_{i,j}$, with $\rho_{i,j}$ being the minimal delay if the tasks are considered in isolation. z is a tuneable parameter.
- Due to allowed delays, it can happen that $(i, j) \in \mathcal{X}$ and also $(j, i) \in \mathcal{X}$. This heuristic consists of forbidding this situation, by only allowing trips to be chained if trip i ends before trip j, i.e. $t_i^{end} < t_j^{end}$. We refer to this heuristic as the "NoDuplicates"-heuristic.

C. Solving the Mixed Integer Linear Problem

For practical reasons, the MILP actually solved differs slightly from the one posed in Equation 6. Equation 5e is replaced with it's linear variant, Equation 7. Further, we add a constraint for each pair of tasks, which can not be chained. The according decision variable $x_{i,j}$ is set to zero from the beginning, i.e.

$$(i, j) \notin \mathcal{X} \rightarrow x_{i,j} = 0$$

We solve the resulting problem using the Gurobi solver version 9.5.2 [30]. If heuristics are applied the problem to solve alters slightly. If setting a new artificial delay bound (ρ_{max}), Equation 5d is adapted accordingly. The other three heuristics add an additional check while calculating the set of potential chaining options $\mathcal{X}$, which is used as described above.

V. Experiments and Results

A. Overview

This section presents our experimental results and is structured as follows: First, we analyze a theoretical example,

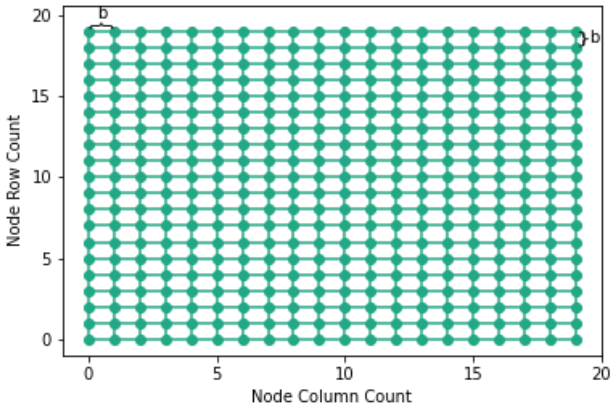

Fig. 4: A graph representing the street network of the Gridworld environment $n = 20$, thus containing 400 vertices, is depicted. For an vehicle to traverse any edge takes 10 seconds, $b = 10$.

called Gridworld, in detail. We compare runs with and without delay, analyze achievable trade-offs between fleet size, total relocation time, and delay, compare different scenarios varying the density of tasks, and analyze the introduced heuristics. Second, we investigate a case study of pooled taxi rides in Manhattan. All computations have been performed on a PC with an AMD Ryzen 5 5600X and 64GB Memory. All resulting MILPs were solved using Gurobi 9.5.2 [30].

B. Gridworld

The operational environment, which we call *Gridworld*, is a grid-like structure consisting of $n \times n$ vertices arranged in a square. Neighboring vertices are connected and equidistantly spaced with a travel time of b seconds[6]. One instance of Gridworld is fully characterized by n and b. An example of Gridworld is depicted in Figure 4. Gridworld has a maximal distance between vertices of $2 \times (n-1) \times b$ seconds. For tasks' start and end locations (l_T^{start} and l_T^{end}), we randomly (uniformly) chose two vertices from the environment graph. Start times t_T^{start} are sampled uniformly within the starting time $t = 0$ and the end of the operation Λ. In Gridworld, we define the duration of a task α_T as the time it takes to travel the shortest path from the start to the end location of the task[7]. As such the ending time t_T^{end} is determined. This results in an average task length in Gridworld of $\frac{4}{6} \times n \times b$ seconds.

Within this section, we use the following parameter settings, unless mentioned otherwise: $n = 40$; $b = 10$; $|\mathcal{T}| = 1,600$; $\sigma_T = 480$; $\Lambda = 8 \cdot 3,600$, $M_\rho = M_\phi = 1$. The parameters have been chosen to be realistic while keeping the required computational times at a reasonable scale. We sampled and solved 5 different demand scenarios. The mean and standard deviation are reported. The density analysis and the heuristics are analyzed using a single case. Unless mentioned otherwise, no heuristics are applied for all calculations within this section, and the problem at hand is solved to optimality (no time limit and an optimality gap of 0.0001).

1) Comparison against the no-delay case: First, we compare solutions to the FSD (allowing to delay individual tasks) to the traditional fleet sizing problem (not allowing delays). We do so for different values of M_{fix} (Equation 5a), as it shapes the number of vehicles used. Generally, the amount of extra delay and extra traffic to be accumulated to beneficially decrease the fleet size by one vehicle is capped at M_{fix}, as otherwise the objective value (Equation 5a) would increase and thus not chaining is better. Thus a lower absolute value of M_{fix} will lead to a less strong decrease in fleet size. Thereby, for no delays, the value of M_{fix} that equals the largest relocation time between any two vertices is of major importance. For the scenario analyzed here, this value equals $M_{fix} = 780$. A value of $M_{fix} \geq 780$ leads to a minimal fleet without delays, as it is always beneficial to chain two tasks if possible. The threshold value of M_{fix} to not decrease the fleet size at all, in other words using an individual vehicle per task, depends on the analyzed scenario. The threshold equals the minimum in the sum of of relocation time plus potentially required delay of any two tasks that can be chained.

Figure 5 shows a direct comparison of runs with and without allowing delays for $M_{fix} \in [400, 600, 800]$. All results are also listed in Table II in Appendix B. The obtained fleet sizes, total relocation times, added minor delays, as well as the difference in costs, are shown. Runs without delays are depicted in brighter colors. Allowing for delays reduces the fleet size for all values of M_{fix}. The required fleet sizes are reduced by around 1 vehicle for $M_{fix} = 800$, by around 2 vehicles for $M_{fix} = 600$, and by around 1 vehicle for $M_{fix} = 400$. This showcases that allowing delays is beneficial to decrease fleet sizes.[8] This comes at the price of increasing total relocation time and added delay. Respectively a total delay of 14:46 $\pm$ 02:51, 11:16 $\pm$ 02:02, and 05:50 $\pm$ 00:59 minutes and seconds is added. The average delay per task is less than one second for all runs. The total amount of added delay, in comparison to the total relocation time, is small. Please also note that the driving time of the vehicles during tasks is not depicted nor part of the cost function, as it is a constant. The obtained total costs, the objective of the optimization problem, are smaller (minimization problem) if delays are allowed. As such, it is beneficial to consider potential delays in fleet sizing regardless of the importance of single objectives as seen by the overall achievable lower costs.

2) Further Reduction in Fleet Size: In the previous section, we showed the potential of solving the FSD in direct comparison to not considering the option to delay tasks. Here we

[6]This is a simplifying assumption that we consider as reasonable as the proposed approach can accommodate real travel times if available. Real travel times have an effect of the specific parameters of tasks, but not on their general definition nor on the proposed method.

[7]This definition does not harm the generalization of the method and is straightforward to adapt.

[8]Larger decreases in fleet size can be obtained for larger values of M_{fix}, see below (Figure 6).

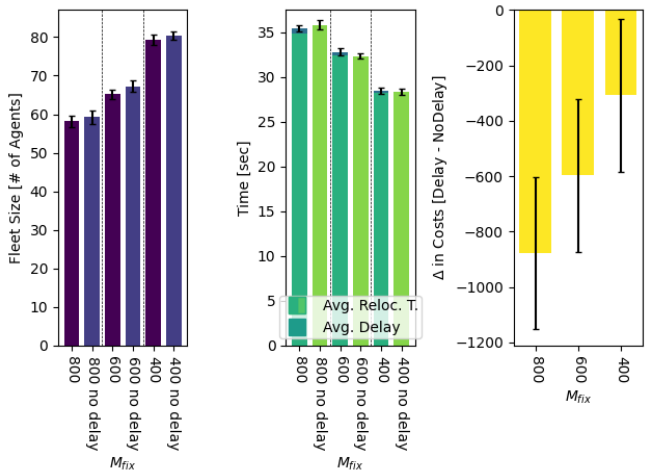

Fig. 5: Comparison of runs with and without allowing to delay tasks for 3 values of $M_{fix} \in [400, 600, 800]$, scenarios without delay are displayed in lighter colors. Allowing delays decreases the obtained fleet size (purple) by adding some minor delay (dark green). Total objective values (yellow) are lower if delays are allowed (minimization problem). The mean of 5 scenarios is shown, including their standard deviation shown using black bars.

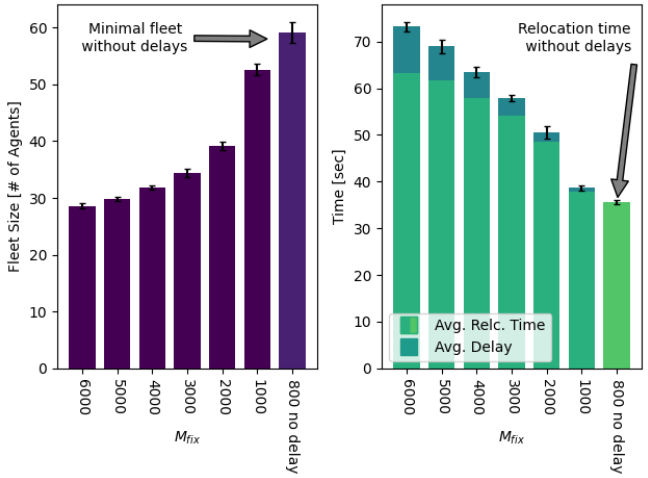

Fig. 6: Figure showing the effect of the value of M_{fix} on the obtained fleet size and the average delay and average relocation time. For comparison, a run with no delay and only optimizing on fleet size is shown. The mean of 5 scenarios is shown, including their standard deviation shown using black bars.

highlight the potential of considering delaying tasks to decrease the fleet size even more. When no delays are allowed, the threshold $M_{fix} \geq 780$ ensures that fleet size is always the primary objective. When delays are allowed, larger values of M_{fix} yield even smaller fleets, as chaining decisions are not independent of each other anymore. Consequently, we analyze $M_{fix} > 780$ within this section.

Figure 6 shows obtained results for values of M_{fix} from 1,000 to 6,000, in steps of 1,000, and for comparison the minimum achievable fleet without delays (using $M_{fix} = 800$, same as previous section). Obtained fleet sizes, total relocation time, and added delay are shown. Additionally, results are listed in Table III in Appendix B. All runs are able to achieve a smaller fleet at the price of increasing total relocation time and added delay, compared to the non-delay case. The higher M_{fix}, the smaller the obtained fleet size, accompanied by higher sums of total relocation time and added delay. This is expected as the sum of costs (delaying tasks and longer relocation times) can be more to include more tasks within each chain. Regarding fleet size, the absolute number of vehicles saved for increasing M_{fix} more and more diminishes slowly. Comparing the required fleet sizes for $M_{fix} = 6,000$ to the minimum fleet size without delays shows a decrease of needed vehicles of more than 50% (from 59.2 to 28.6 vehicles), at a cost of just 9.7 seconds of average delay per task. Average relocation time increases from 35 seconds to 1:03 minute and seconds.

In conclusion, FSD increases the potential trade-off space and, thus, the potential to decrease the number of required vehicles significantly. This potential is huge but heavily depends on the relation between costs involved, which in turn depends on the nature of the operation at hand.

3) Density analysis: This section analyses the effect if the number of potential chains is varied and shows that obtainable benefits are robust against external changes.

The number of potential tasks that a vehicle could serve after finishing a task (the number of chaining options) increases if more tasks need to be fulfilled (i.e. if we increase $|\mathcal{T}|$). The same effect occurs the smaller the environment is, as the average relocating time becomes smaller (which we analyze by changing n), and thus it is easier to reach the new starting location in time. To analyze this effect on the proposed method, we vary $|\mathcal{T}|$ and n. We compare runs with no delay, fully minimizing fleet size, to runs allowing delays with $M_{fix} = 5,000$. We vary the size of the environment n as [20,40,60]. As for the number of placed tasks, we use different numbers of total tasks to fulfill, $|\mathcal{T}| \in$ [800, 1,200, 1,600, 2,000, 2,400]. Obtained results are visualized in Figure 7. The differences in fleet size (left) and delay plus relocation time (right) for each value pair are shown for the scenarios with and without delay. Values are shown in percentage, the baseline is the corresponding run with no delays, fully minimizing fleet size.

The decrease in fleet size ranges from 15.7% to 35%. This decrease is always significant. The increase in total relocation time plus delay lies between 13.4% and 44.7%. Generally, greater changes accompany each other, seen by similar sizes of the corresponding dots in Figure 7. No other clear trends show.

4) Effect of Heuristics: To enhance the scalability of the proposed approach to solve the FSD, we introduce a variety of heuristics in Section IV-B. They reduce computational cost, while not significantly compromising the quality of the results. This capability is particularly advantageous when tackling larger scenarios. In this section, we apply these

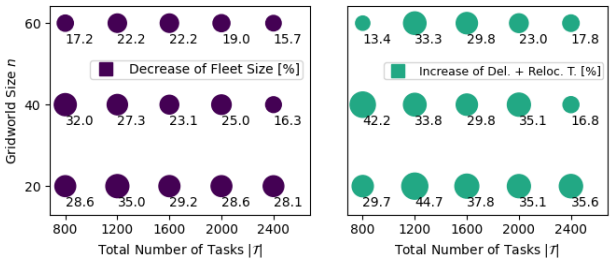

Fig. 7: The difference in fleet size and total relocation time plus delay for allowing delays to no delays is shown in percentage as the environment size and the number of placed tasks are varied.

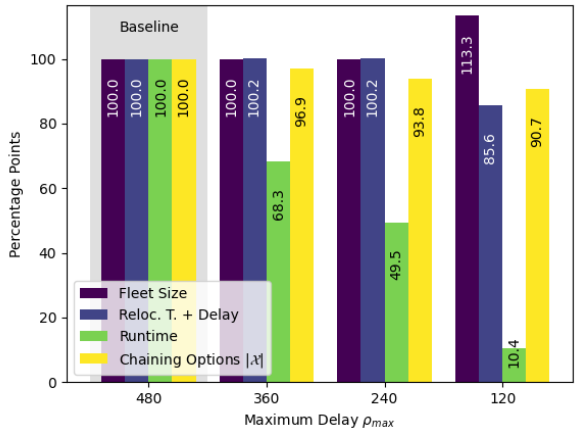

Fig. 8: Visualization of the effect of reducing the maximal allowed delay ρ_{max} on fleet size, delay plus total relocation time, run time, and the number of chaining options $\mathcal{X}$, all presented as percentages of the run without any heuristics.

heuristics and analyze their effectiveness. Additionally, we vary the strengths of the applied heuristics. For each heuristic, we compare fleet size, total relocation time plus delay, the required computation time for the optimization, and the number of chaining options, $|\mathcal{X}|$. As a baseline for comparison, we use the case with M_{fix} set to 5,000. For ease of comparison, in all figures of this section, all numbers are displayed in percentages relative to the baseline.

Restricting maximum allowed delay: We restrict the maximum allowed delay form $\sigma_i = 480 \forall i \in \mathcal{T}$ to $\rho_{max} \in [360, 240, 120]$. Results are illustrated in Figure 8. Reducing the maximally allowed delay can be effective in decreasing run times. Figure 8 shows that the heuristic only slightly increase the fleet size and relocation time plus delay, for a strong decrease in required run times. But, if the heuristic is applied strongly, i.e. for lower values of $\rho_{max} = 120$, we see a substantial increase in fleet size.

Restricting maximum allowed relocation time: We restrict the upper limit on the relocation time between two tasks to $\tau_{max} \in [600, 400, 200, 100]$. Results are displayed in Figure 9. For larger values of τ_{max} the fleet size does not increase,

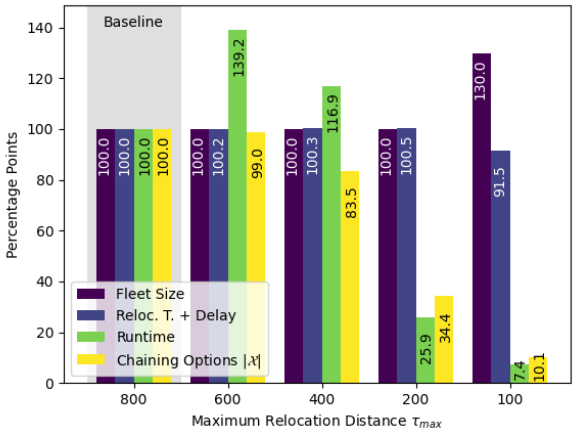

Fig. 9: Visualization of the effect of reducing the maximal allowed relocation time τ_{max} on fleet size, delay plus total relocation time, run time, and the number of chaining options $\mathcal{X}$, all presented as percentages of the run without any heuristics.

but the heuristic is also not effective in reducing required run times. Reducing the maximum relocation time strongly, $\tau_{max} \in [200, 100]$, greatly decreases the number of chaining options and a decrease in run times. For $\tau_{max} = 200$ the fleet size does not change and total relocation time and delay only increase slightly. In contrast, the fleet size increases strongly for $\tau_{max} = 100$. As such the heuristic needs to be well-tuned to be effective.

Comparing this heuristic to the previous one, we see a notable difference. Reducing the maximum delay decreases the number of potential chaining options only slightly but significantly affects the required computation time. In contrast, reducing the maximum relocating time reduces the number of potential edges in a stronger fashion but with a smaller effect on the computational time. This indicates that the complexity of the problem is within the newly allowed delay.

Restricting the maximum number of chaining options per task to z: We varied $z \in [250, 200, 150, 100]$. Most notably the run time increases for all values of z. As such, the heuristic is ineffective, and the sum of relocation times and minimum required delay is ineffective in judging the potential of chaining options. Results are shown in Figure 11 in Appendix C.

"NoDuplicates"-Heuristic: This heuristic can either be used or not (no additional tuning). The total number of chaining options only decreases very slightly (about 1.3%). The run time increases, and as such we do not recommend the use of this heuristic. Results are shown in Figure 12 in Appendix C.

C. Case Study: Manhattan

This section analyses a real-world instance to analyze the proposed method's potential. We investigate the FSD problem considering taxi rides in Manhattan.

To build the set of tasks $\mathcal{T}$, we utilize one hour of taxi rides data in Manhattan[9] [31]. 19,809 individual transportation requests are placed within this hour. The tasks we consider are not these individual requests but groups of them that are pooled to travel together. Here the pooling of requests is done following a method based on the Vehicle-Group-Assignment method by [32]. Details on this step can be found in Appendix D. The ending time of the transportation tasks is based on the task's duration, determined by the total travel time to serve all passengers following the shortest path. This pooling step leads to a total of 4,255 pooled transportation tasks or trips starting within 1 hour. Figure 13 in Appendix E shows distributions of the starting times, trip duration, ending times of the tasks, as well as the number of passengers per task. As the operational environment, Manhattan's road network is represented by a graph, and travel times are estimated for each road segment. These travel times are used for transporting passengers and empty relocation. The graph was obtained following the method described in [33]. Thereby the travel times of the graph are estimated based on the departure and arrival times of the recorded trips and averaged over a day. The average relative error of the actual travel times to the estimated travel times is minimized.

We run the above-proposed method for the Manhattan instance of the FSD problem with the following settings. For the cost function, we set $M_{fix} = 2,500$ and $M_{\phi} = M_{\rho} = 1$. Based on the heuristic analysis for Gridworld (Section V-B.4), we applied, the heuristic capping the maximum relocating time, which we set to $\tau_{max} = 800$ seconds. We do not apply the "NoDuplicates"-heuristic and the heuristic selecting chaining options based on cost. For the maximum delay of all trips, we analyze three scenarios of $\sigma \in [60, 120, 180]$ seconds, which is at the core of this work. All results are visualized in Figure 10 and listed in Table IV in Appendix B.

Each extra minute of allowed delay reduces the minimum required fleet size by about 50 vehicles (52/54/47); in percentage, this is a decrease of around 3% (3.12%/3.34%/3.01%). Added delay and relocating time increase, whereby the stronger changes are within the added delay. If 1 minute of maximum added delay is allowed an average delay of 3.2 seconds per task is added. For 2 and 3 minutes of maximal delay, this equals 10.7 and 22.37 seconds per task. This equals 8.41/29.2/62.78 seconds of delay for each vehicle over its entire day.

To conclude, introducing the option of delaying tasks is a way to reduce the number of required vehicles. A slight modification, consisting of the introduction of 1 minute of maximum delay, already allows decreasing the fleet by 3.12% at the mild cost of an average delay of 12 seconds per trip. The proposed method allows stressing this trade-off significantly further, allowing up to 3 minutes of delay, decreasing the required fleet by 9.17% compared to not allowing delays.

[9]The used data was recorded on 29.05.2013 between 1 p.m. and 2 p.m.

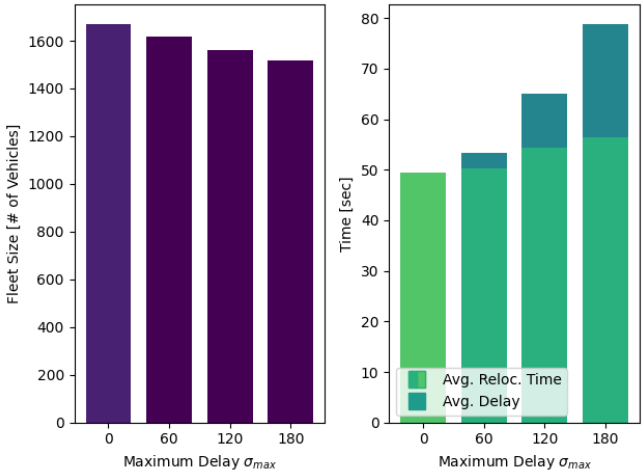

Fig. 10: Main results of fleet sizing of one hour of pooled taxi rides in Manhattan, in regards to fleet size, average relocation time, and average added delay, are displayed. For higher maximal allowed delay the required fleet sizes decrease, accompanied by an increase in total relocation time and added delay.

VI. CONCLUSION

In this work, we have posed the problem of *fleet sizing with delays*. It extends the question of "How many vehicles are needed to perform a set of tasks?" by allowing short delays before the beginning of each task, noting that such delays are commonly observed in real-life applications. We proved that the new problem is NP-Hard, and proposed a formulation of the FSD as a MILP. The formulation used is general, and as such, the methods and findings have wide applicability. First, we studied a general case (Gridworld) followed by a real-world case of shared taxis rides in Manhattan.

For Gridworld, we showed that it is beneficial to consider delays within fleet sizing, lower costs are achieved in direct comparison to not considering them. We further showed that delays allow a decrease in fleet size beyond previous limits. If a small fleet is the highest priority, we find that fleets can be decreased by 50% compared with the minimal fleet without delay, while still respecting small limits of maximal delay per task. Additionally, in this paper, various heuristics are proposed and analyzed. The work is concluded with a real-life case study of taxi rides in Manhattan. Results show the same nature and potential. By adding a maximum of 3 minutes of additional delay per ride, the required fleet size can be decreased by about 9% in comparison to the minimal fleet without delays, at the cost of fewer than 23 seconds of average delay per task.

In summary, using delays improves solutions and increases the option space for potential trade-offs. It allows a reduction of the number of vehicles used strongly.

Future work includes the introduction of delays into other types of approaches, the application of the proposed general method to specific fields and cases, other than the mobility of people, and the development of potent heuristics, such as a local search algorithm or tabu search.

ACKNOWLEDGMENT

This research was supported by Ahold Delhaize. All content represents the opinion of the author(s), which is not necessarily shared or endorsed by their respective employers and/or sponsors.

APPENDIX

A. Notation

Notation	Explanation
G	Graph, encoding the environment
V	Set of vertices of the graph, each vertex represents a location, customers order to
E	Set of edges connecting the vertices of the graph
$c(e)$	Costs (time) required to traverse edge e
$\mathcal{T}$	Set of all tasks
T	a single task $T = (l_T^{start}, l_T^{end}, t_T^{start}, \alpha_T, \sigma_T)$
t_T^{start}	Starting time of task T
t_T^{end}	Finishing time of task T
α_T	Duration of a task T
l_T^{start}	Starting location of task T
l_T^{end}	Ending location of task T
Λ	Ending time of the operation, $t_T^{start} \leq \Lambda$
σ_T	Maximum slack of task T
ρ_T	Time task T is delayed
$t_T^{start,\rho}$	Starting time of the delayed task T
$t_T^{end,\rho}$	Ending time of of the delayed task T
g_i	Trajectory i, which is defined as an ordered set of tasks $g_i = (T_{i,1}, T_{i,2}, ..., T_{i,k})$
ω	Set of trajectories
Ω	Set of all feasible sets of trajectories
$c(\omega)$	Cost function
$\mathcal{X}$	Set of all pairs of tasks (i, j), which are chainable
$x_{i,j}$	Binary decision variable, whether tasks i and j are chained
M_{fix}	Constant in the objective function awarded for chaining two tasks
M_ρ	Weight in cost function to weigh the total delay
M_c	Weight in cost function to weigh the total relocation time
$\phi(g)$	Sum of times traveled by an vehicle between tasks of trajectory g
$\tau_{i,j}$	Relocation time from task i to task j
$A_{i,j}$	Auxiliary variable to linearize Equation 7
$B_{i,j}$	Auxiliary variable to linearize Equation 7
ρ_{max}	Artificial maximum delay
τ_{max}	Artificial maximum relocation time
z	Tuneable parameter for selecting the z best edges per task
n	Side length (in number of vertices) of the squared experimental environment Gridworld
b	Distance in seconds between two connected vertices of the experimental environment Gridworld

TABLE I: Table summarizing the complete notation used throughout this work.

B. Result Tables

Have a look at Tables 2 to 4.

C. Plots of Heuristic Experiments

Have a look at Figures 11 and 12.

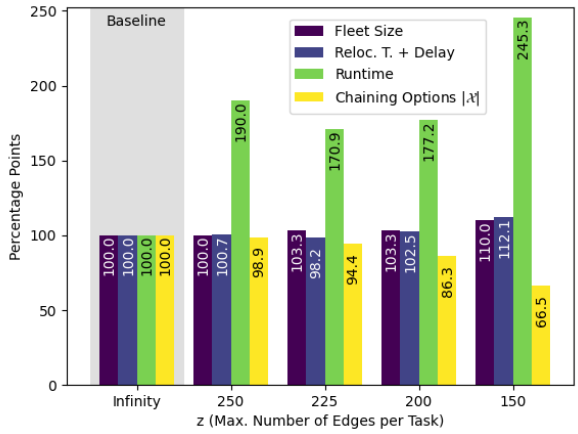

Fig. 11: Visualization of the effect of restricting the maximum number of chaining options per trip on fleet size, delay plus total relocation time, run time, and the number of chaining options $\mathcal{X}$, all presented as percentages of the run without any heuristics.

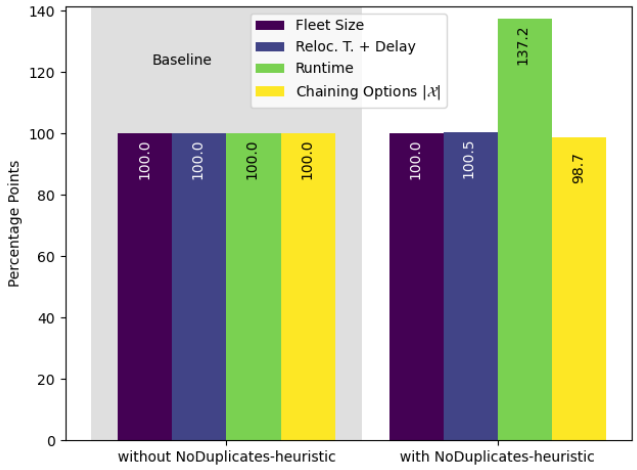

Fig. 12: Visualization of applying the "NoDuplicates"-heuristic on fleet size, delay plus total relocation time, run time, and the number of chaining options $\mathcal{X}$, all presented as percentages of the run without any heuristics.

D. Details on Vehicle Group Assignment for Pooling

To create the set of tasks or rides $\mathcal{T}$ used within the case study of Manhattan, the individual transportation requests within Manhattan have been pooled into tasks $T \in \mathcal{T}$ first. This *pooling step* was done leveraging a method close to the Vehicle Group Assignment method (VGA) [32]. VGA is a receding horizon approach. Each time step, all potential trajectories, for a set of given vehicles, are calculated. A trajectory defines which requests one vehicle serves and in which order, also determining its path. From this set of

M_{fix}	Max Delay σ [sec]	Fleet Size	Total Reloc. Time [h:min:sec]	Total Added Delay [h:min:sec]	Costs
800	480	58.2 ± 1.47	15:33:04 ± 0:05:59	0:14:46 ± 0:02:51	-1176569.4 ± 910.13
800	0	59.2 ± 1.83	15:49:08 ± 0:13:01	0:00:00 ± 0:00:00	-1175692.0 ± 932.98
600	480	65.2 ± 1.17	14:15:46 ± 0:08:16	0:11:16 ± 0:02:02	-868857.2 ± 660.19
600	0	67.2 ± 1.47	14:17:00 ± 0:06:47	0:00:00 ± 0:00:00	-868260.0 ± 666.84
400	480	79.4 ± 1.36	12:24:38 ± 0:08:36	0:05:50 ± 0:00:59	-563211.2 ± 524.8
400	0	80.4 ± 1.2	12:28:56 ± 0:10:21	0:00:00 ± 0:00:00	-562904.0 ± 474.7

TABLE II: Comparison of the fleet sizing problem allowing delays and not doing so. These results are visualized in Figure 5. We list the total delay instead of the average delay per task here as the averages are all less than one second.

M_{fix}	Max Delay σ [sec]	Fleet Size	Total Reloc. Time [h:min:sec]	Total Added Delay [h:min:sec]
6000	480	28.6 ± 0.49	1 day, 4:04:40 ± 0:07:56	4:19:54 ± 0:19:30
5000	480	29.8 ± 0.4	1 day, 3:21:56 ± 0:20:36	3:16:15 ± 0:18:07
4000	480	31.8 ± 0.4	1 day, 1:45:28 ± 0:15:27	2:31:02 ± 0:16:03
3000	480	34.4 ± 0.8	1 day, 0:05:00 ± 0:08:08	1:26:02 ± 0:07:55
2000	480	39.2 ± 0.75	21:33:02 ± 0:27:54	0:42:19 ± 0:09:07
1000	480	52.6 ± 1.02	16:49:20 ± 0:12:52	0:20:33 ± 0:03:05
800	0	59.2 ± 1.83	15:49:08 ± 0:13:01	0:00:00 ± 0:00:00

TABLE III: Results table for all experiments using values of M_{fix} greater than 800, and for comparison the scenario achieving a minimal fleet for no delays. These results are visualized in Figure 6.

M_{fix}	Max Delay ρ_{max} [sec]	Max Reloc. T. τ_{max} [sec]	z best	"NoDuplicates"	Fleet Size	Avg. Reloc. Time per Trip [min:sec]	Avg. Added Delay per Trip [min:sec]	Runtime [h:min:sec]	Chaining Options $\lvert\mathcal{X}\rvert$
2,500	0	800	x	✓	1,669	0:49	0:00	0:00:26	1,312,052
2,500	60	800	x	✓	1,617	0:50	0:03	0:01:10	1,371,178
2,500	120	800	x	✓	1,563	0:54	0:10	0:08:05	1,432,028
2,500	180	800	x	✓	1,516	0:56	0:22	47:05:04	1,494,541

TABLE IV: Results table containing all results for the scenarios analyzed in the case study of taxi rides in Manhattan.

potential trajectories, the best, according to a given objective function, are selected by means of an assignment problem, solved as an integer linear problem. For our purpose, VGA is altered to overcome the assumption of a fixed set of vehicles. When calculating all potential trajectories. It is assumed that each request has its own hypothetical vehicle available at its starting location. If a vehicle is not assigned to be used during the assignment step, it is omitted. This procedure was proposed in [34], more details can be found there. The tasks are defined as the trajectories the used vehicles take, their first request defines the task's start and the last request the task's end.

E. Details on the Manhattan Dataset

Figure 13 shows four histograms all describing the set of trips used for the Manhattan Case study. They illustrate the starting time of the trips (top left), their duration (top right), the resulting ending times (bottom left) and their total size (bottom right).

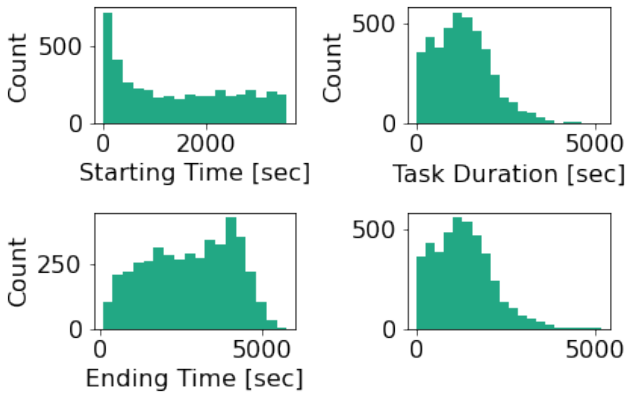

Fig. 13: This figure shows the distributions of the starting time (top left), their duration (top right), the resulting ending times (bottom left) and their total size (bottom right) of all trips used within the Manhattan case study. Size is measured as the number of individual passengers served by a trip.

REFERENCES

[1] M. Vazifeh, P. Santi, G. Resta, S. Strogatz, and C. Ratti, "Addressing the minimum fleet problem in on-demand urban mobility," *Nature*, vol. 557, 05 2018.

[2] M. Sahnoun, Y. Xu, F. B. Abdelaziz, and D. Baudry, "Optimization of transportation collaborative robots fleet size in flexible manufacturing systems," in *2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO)*, 2019, pp. 1–5.

[3] B. Golden, A. Assad, L. Levy, and F. Gheysens, "The fleet size and mix vehicle routing problem," *Computers & Operations Research*, vol. 11, no. 1, pp. 49–66, 1984.

[4] G. Pantuso, K. Fagerholt, and L. M. Hvattum, "A survey on maritime fleet size and mix problems," *European Journal of Operational Research*, vol. 235, no. 2, pp. 341–349, 2014.

[5] G. C. de Bittencourt, R. D. Seimetz Chagas, V. A. Silva, I. G. Peres Vianna, R. P. Longhi, P. C. Ribas, and V. J. M. Ferreira Filho, "A solution framework for the integrated problem of cargo assignment, fleet sizing, and delivery planning in offshore logistics," *Computers & Industrial Engineering*, vol. 161, p. 107653, 2021.

[6] H. Zhang, C. J. R. Sheppard, T. E. Lipman, and S. J. Moura, "Joint fleet sizing and charging system planning for autonomous electric vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4725–4738, 2020.

[7] M. Xu and Q. Meng, "Fleet sizing for one-way electric carsharing services considering dynamic vehicle relocation and nonlinear charging profile," *Transportation Research Part B: Methodological*, vol. 128, pp. 23–49, 2019.
[8] D. Banerjee, A. L. Erera, and A. Toriello, "Fleet sizing and service region partitioning for same-day delivery systems," *Transportation Science*, vol. 56, no. 5, pp. 1327–1347, 2022.
[9] A. Wallar, J. Alonso-Mora, and D. Rus, "Optimizing vehicle distributions and fleet sizes for shared mobility-on-demand," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3853–3859.
[10] C. Wang, Y. Song, Y. Wei, G. Fan, H. Jin, and F. Zhang, "Towards minimum fleet for ridesharing-aware mobility-on-demand systems," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.
[11] B. Qu, L. Mao, Z. Xu, J. Feng, and X. Wang, "How many vehicles do we need? fleet sizing for shared autonomous vehicles with ridesharing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 14 594–14 607, 2022.
[12] P. JONES and J. ZYDIAK, "The fleet design problem," *The Engineering Economist*, vol. 38, no. 2, pp. 83–98, 1993.
[13] A. Zhao, J. Xu, J. Salazar, W. Wang, P. Ma, D. Rus, and W. Matusik, "Graph grammar-based automatic design for heterogeneous fleets of underwater robots," *2022 International Conference on Robotics and Automation (ICRA)*, pp. 3143–3149, 2022.
[14] A. Rjeb, J.-P. Gayon, and S. Norre, "Sizing of a homogeneous fleet of robots in a logistics warehouse," *IFAC-PapersOnLine*, vol. 54, no. 1, pp. 552–557, 2021, 17th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2021.
[15] I. Markov, R. Guglielmetti, M. Laumanns, A. Fernández-Antolín, and R. de Souza, "Simulation-based design and analysis of on-demand mobility services," *Transportation Research Part A: Policy and Practice*, vol. 149, pp. 170–205, 2021.
[16] P. M. Boesch, F. Ciari, and K. W. Axhausen, "Autonomous vehicle fleet sizes required to serve different levels of demand," *Transportation Research Record*, vol. 2542, no. 1, pp. 111–119, 2016.
[17] D. Fagnant and K. Kockelman, "Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in austin, texas," *Transportation*, vol. 45, 01 2018.
[18] A. Fielbaum, A. Tirachini, and J. Alonso-Mora, "Economies and diseconomies of scale in on-demand ridepooling systems," *Economics of Transportation*, vol. 34, p. 100313, 2023.
[19] Y. Mei, Y.-H. Lu, Y. Hu, and C. Lee, "Determining the fleet size of mobile robots with energy constraints," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 2, 2004, pp. 1420–1425 vol.2.
[20] G. Wang, *Comparative Study on Solving the Minimum Fleet of Shared Autonomous Vehicles*, pp. 522–534.
[21] Y. Yang, Z. Yuan, X. Fu, Y. Wang, and D. Sun, "Optimization model of taxi fleet size based on gps tracking data," *Sustainability*, vol. 11, no. 3, p. 731, Jan 2019.
[22] H. Zhang, C. J. R. Sheppard, T. E. Lipman, and S. J. Moura, "Joint fleet sizing and charging system planning for autonomous electric vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4725–4738, 2020.
[23] J. C. Castillo, D. Knoepfle, and G. Weyl, "Surge pricing solves the wild goose chase," in *Proceedings of the 2017 ACM Conference on Economics and Computation*, ser. EC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 241–242.
[24] M. Schröder, D.-M. Storch, P. Marszal, and M. Timme, "Anomalous supply shortages from dynamic pricing in on-demand mobility," *Nature Communications*, vol. 11, 09 2020.
[25] J. E. Hopcroft and R. M. Karp, "An n^5/2 algorithm for maximum matchings in bipartite graphs," *SIAM Journal on Computing*, vol. 2, no. 4, pp. 225–231, 1973.
[26] S. Hao, X. Liu, L. Miao, W. K. V. Chan, and M. Qi, "Qualifying the benefits of ride-sharing on reducing fleet size," *Journal of Physics: Conference Series*, vol. 1903, no. 1, p. 012019, apr 2021.
[27] R. Auad-Perez and P. Van Hentenryck, "Ridesharing and fleet sizing for on-demand multimodal transit systems," *Transportation Research Part C: Emerging Technologies*, vol. 138, p. 103594, 2022.
[28] G. Wang, *Research on the Fleet Size of Shared Autonomous Vehicles in the Future City: An Example in Shanghai*, pp. 4537–4549.
[29] M. Balac, S. Hörl, and K. W. Axhausen, "Fleet sizing for pooled (automated) vehicle fleets," *Transportation Research Record*, vol. 2674, no. 9, pp. 168–176, 2020.
[30] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: https://www.gurobi.com
[31] B. Donovan and D. Work, "New york city taxi trip data (2010-2013)," 2016.
[32] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proceedings of the National Academy of Sciences*, vol. 114, no. 3, pp. 462–467, 2017.
[33] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. Strogatz, and C. Ratti, "Quantifying the benefits of vehicle pooling with shareability networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 111, 09 2014.
[34] M. Čáp and J. Alonso-Mora, "Multi-objective analysis of ridesharing in automated mobility-on-demand," in *Proceedings of Robotics: Science and Systems*, June 2018.

Maximilian Kronmueller is a Ph.D. student under the supervision of Dr. Javier Alonso-Mora and Dr. Robert Babuska. He is affiliated with the department of Cognitive Robotics at the University of Technology Delft. Previously, he obtained his Master of Science in Physics (2018) at the Technical University of Munich. His research interests lie in the following areas: online multi-task assignment, dynamic vehicle routing problems, fleet sizing, flash deliveries and learning-based methods to support routing decisions and dynamic optimization.

Andres Fielbaum is currently an Assistant Professor leading the Share research line in the TransportLab, School of Civil Engineering at the University of Sydney. He received his Ph.D. in Systems Engineering at Universidad de Chile in 2019. He was a Postdoctoral Fellow at the Department of Cognitive Robotics, Delft University of Technology, the Netherlands. Between 2015 and 2019 he was an adjunct lecturer at Universidad Federico Santa Maria, Chile. During 2019, he was a Research Fellow at Universidad de O'Higgins, Chile. His main research interests include public transport, on-demand mobility and logistics, and combinatorial optimization. Dr. Fielbaum received the first prize among young researchers worldwide to attend the World Conference on Transport Research 2019.

Javier Alonso-Mora received the Ph.D. degree in robotics on cooperative motion planning from ETH Zurich, Zurich, Switzerland, in partnership with Disney Research Studios, Zurich, Switzerland. He is currently an Associate Professor with the Delft University of Technology, Delft, The Netherlands, where he leads the Autonomous Multi-robots Lab. He was a Postdoctoral Associate with Computer Science and Artificial Intelligence Lab

(CSAIL), Massachusetts Institute of Technology, Cambridge, MA, USA. His research interests include navigation, motion planning, and control of autonomous mobile robots, with a special emphasis on multirobot systems, mobile manipulation, on-demand transportation, and robots that interact with other robots and humans in dynamic and uncertain environments.

Dr. Alonso-Mora is an Associate Editor for IEEE Transactions on Robotics and for Springer Autonomous Robots. He was the recipient of a talent scheme VENI Award from the Netherlands Organization for Scientific Research (2017), ICRA Best Paper Award on Multi-robot Systems (2019), and an ERC Starting Grant (2021).