Submodular Dispatching with Multiple Vehicles

Ignacio Erazo, Alejandro Toriello
H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology,
iien3@gatech.edu, atoriello@isye.gatech.edu

Motivated by applications in e-commerce logistics and production planning where orders (or items, or jobs) arrive at different times and must be dispatched or processed in batches, we consider a multi-vehicle dispatching problem that captures the tension between waiting for orders to arrive and the economies of scale due to batching. Our model extends the single-vehicle work in Erazo and Toriello (2024), and we focus primarily on the case of identical vehicles with submodular dispatch times. We propose four different mixed-integer programming formulations to solve this problem; we analyze the complexity of solving each formulation's linear relaxation, study the quality of the corresponding bounds, and leverage column generation to create heuristics. Moreover, we analyze solutions where all batches are intervals of consecutive orders, and identify two classes of functions for which such a solution is optimal. Finally, we computationally test our methods on applications in machine scheduling with family setups and same-day delivery.

Key words: submodular; machine scheduling; same-day delivery; column generation

1. Introduction

Retail e-commerce sales have increased significantly over the past few years, reaching a worldwide total of \$5.2 trillion in 2021, expected to grow to \$8.1 trillion by 2026 (Statista 2023b). This growth has put a spotlight on last-mile delivery, the last portion of the order fulfilment process, which can represent up to 50% of total logistics costs (Vanelslander, Deketele, and Hove 2013). Last-mile delivery systems are increasingly complex because of the scale of operations and the customers' desire to have faster deliveries, as evidenced by 22% of customers dropping online shopping sessions because shipping is too slow (Forbes 2023), and by a surging same-day delivery market that is expected to grow by over 100% in the next four years (Statista 2023a). In particular, same-day delivery (SDD) systems are difficult to design and operate because the order arrival and packaging process overlaps significantly with the dispatching and delivery process, increasing the system's dynamism and reducing opportunities to consolidate orders and decrease routing costs (Klapp, Erera, and Toriello 2020).

A common element in many processes within an e-commerce supply chain, including same-day delivery (SDD), order picking and shelf re-stocking, is the need to dispatch (i.e. deliver, process, pick or re-stock) orders or items that become available at different times, but where batching yields economies of scale in dispatching time. This also characterizes many production systems, when jobs or tasks arrive over the workday and are batched to distribute the workload between multiple machines/servers and benefit from economies of scale.

Submodular set functions are often used to model the economies of scale that arise in these and other applications; their properties have been widely studied by the combinatorial optimization community (e.g. Krause and Golovin 2014, Nemhauser and Wolsey 1999, Schrijver 2003). Submodular functions are characterized by their "discrete concavity": the marginal change in value from adding an element to a subset decreases as the subset includes more elements. In formal terms, for a ground set $N := \{1,2,\dots,n\}$ and function $f : 2^N \to \mathbb{R}$, f is submodular if

$$f(S \cup S') + f(S \cap S') \leq f(S) + f(S'), \qquad S, S' \subseteq N.$$

In many applications, submodular functions are also non-negative and monotonically non-decreasing, $f(S) \leq f(S')$ for $S \subseteq S' \subseteq N$, and the latter implies the former if $f(\varnothing) = 0$. These conditions also imply subadditivity: $f(S \cup S') \leq f(S) + f(S')$ for all $S, S' \subseteq N$ with $S \cap S' = \varnothing$.

Recently, Erazo and Toriello (2024) proposed the Subadditive Dispatching Problem (SAD) to model the tension between economies of scale due to batching and idle time due to waiting for orders. They focused on the case in which one vehicle (or picker, or server) dispatches or processes orders, and considered the class of subadditive functions to define the dispatch times. In this paper, we consider the problem with multiple vehicles, and focus on the scenario where they are identical. This problem can be studied through the lens of subadditivity; however, as we are motivated by applications in machine scheduling and same-day delivery where submodularity is a key component, we define and study the problem with submodular functions.

1.1. Problem Definition and Applications

The Multi-Vehicle Submodular Dispatching Problem (MSMD) is characterized by a finite set of orders N that must be dispatched or processed, where each order has a release time. The orders are dispatched by a set M of vehicles (servers), where each vehicle $k \in M$ has an associated non-negative, monotone, submodular set function $f_k : 2^N \to \mathbb{R}_+$. Depending on the context, e.g. delivery or production, N and f may represent different things; for clarity of exposition we adopt delivery terminology throughout the rest of the paper. Thus, N is a set of *orders*, a subset $S \subseteq N$ is a *batch* of orders, and f_k is the *dispatch time* function of vehicle k, representing the time required to deliver the orders in batch S, starting from and returning to the depot. The goal is to partition the order set N into batches that the multiple vehicles can dispatch while minimizing the makespan, the end time of the last dispatch. Finally, when $f_k = f$ for all vehicles $k \in M$, we are under the identical vehicles scenario, a common occurrence in real-world problems, and the main focus of this paper.

Suppose each order $i \in N$ is associated with a number $\tau_i > 0$; the following are some important special cases of MSMD:

- $f(S) = g\left(\sum_{i \in S} \tau_i\right)$, where $g : \mathbb{R}_+ \to \mathbb{R}_+$ is a non-decreasing concave function with $g(0) = 0$. In particular, when $\tau_i = 1$ for all $i \in N$, f is a function of the batch's cardinality. The specific case $f(S) = a + b|S| + c\sqrt{|S|}$, for $S \neq \varnothing$ and $a, b, c \geq 0$ is a continuous approximation of expected routing and delivery time, used to model average-case SDD system behavior and to perform tactical design of SDD systems with multi-vehicle fleets (Banerjee, Erera, and Toriello 2022, Banerjee et al. 2023, Stroh, Erera, and Toriello 2022). MSMD allows arbitrary and non-stationary order arrivals, unlike most previous work.

- Consider a set of nodes V with $N \subseteq V$, a depot node 0 and an undirected network $(V \cup \{0\}, E)$ with non-negative edge lengths. For $S \subseteq N$, define $f(S)$ as the optimal length of a Steiner traveling salesman problem (TSP) through $S \cup \{0\}$; a Steiner TSP tour must visit nodes $S \cup \{0\}$ but may also visit other nodes in the graph. With this function, MSMD captures operational SDD models in which same-day deliveries must be made to locations N in the network, where different orders are ready for delivery at different points in the operating day (Klapp, Erera, and Toriello 2018b). The function f is submodular for the class of *naturally* *submodular graphs* (Herer and Penn 1995), which includes paths (Klapp, Erera, and Toriello 2018a), trees and other similar topologies.

- If $f(S) = \sum_{i \in S} \tau_i$, MSMD generalizes the classical machine scheduling problem with serial batching (Graham 1969), still a subject of active study, e.g. Ghalami and Grosu (2019). Similarly, if $f(S) = \max_{i \in S}\{\tau_i\}$, MSMD generalizes the machine scheduling problem with parallel batching, which has received significant attention because of its applications in semi-conductor production; see Fowler and Mönch (2022) for a recent survey. MSMD allows for arbitrary release times in both models. Finally, suppose the order set is partitioned into Q families $F_1, \ldots, F_Q$, with each family F_q having a setup time $\tau_q \geq 0$. With $f(S) = \sum_{i \in S} \tau_i + \sum_{q: F_q \cap S \neq \varnothing} \tau_q$, MSMD generalizes serial-batching machine scheduling problems with family setups; see e.g. Kramer, Iori, and Lacomme (2021).

1.2. Contributions and Organization

This paper extends Erazo and Toriello (2024) by considering multiple vehicles, with a focus on identical vehicles. Even in special cases, this extension entails significant difficulty. For instance, for serial scheduling on a single machine, the minimum makespan can be computed in linear time, but it is NP-Hard already with two machines (Garey and Johnson 1979). We summarize our main contributions as follows:

1. We formulate the Multi-Vehicle Submodular Dispatching Problem (MSMD) and propose four different mixed-integer linear programming (MILP) formulations to solve MSMD. Two formulations extend Erazo and Toriello (2024), whereas the other two specifically reduce or eliminate symmetry in the case of multiple identical vehicles. We establish the complexity of solving the LP relaxation for each formulation.
2. We assess the quality of the lower bounds given by the LP relaxations of our formulations, and study their worst-case performance.
3. We analyze the performance of interval solutions, in which batches consist of consecutive orders. We discuss interval-solvable functions, for which there is always an optimal interval solution; MSMD with these functions can be optimized with off-the-shelf solvers. This extends the concept of First-In First-Out (FIFO) optimality (Erazo and Toriello 2024) to the multi-vehicle setting.
4. We perform a computational analysis for two applications: identical machine scheduling with serial batching, family setups and release times, and tactical design of multi-vehicle SDD systems under heterogeneous order arrival rates. The former demonstrates the empirical effectiveness of our methods for solving

large-scale instances of a known strongly NP-Hard problem, whereas the latter shows the practical insights that can be derived from efficiently solving a model with an interval-solvable function.

The paper has the following organization. Section 2 presents a brief literature review, while Section 3 defines MSMD under the heterogeneous fleet scenario, formulates it as an MILP and discusses some preliminary complexity results. Section 4 focuses on the identical vehicles scenario, presents three alternative MILP formulations, their complexity results, and discusses the quality of their LP relaxation lower bounds. This section also defines interval-solvable functions and provides examples. Section 5 summarizes our computational studies, and Section 6 concludes and outlines future avenues of work. The appendix includes proofs omitted from the main body of the paper.

2. Literature Review

2.1. Same-Day Delivery

Multiple operational models that arise in SDD systems have received significant attention, particularly models that focus on dispatching, routing and delivery of orders. These problems have been studied under different conditions, including deterministic or stochastic arrivals, single or multi-vehicle fleets, and different objectives, such as minimizing total distance driven or minimizing makespan. The literature on SDD models considers routing times in general road networks, e.g. Klapp, Erera, and Toriello (2018b, 2020), Wölck and Meisel (2022), which are not submodular except in special cases such as paths (Klapp, Erera, and Toriello 2018a, Erazo and Toriello 2024).

To study the average behavior of SDD and other last-mile distribution systems, continuous-time approximations are being increasingly used for tactical-design; see Banerjee, Erera, and Toriello (2022), Banerjee et al. (2023), Carlsson et al. (2021), Stroh, Erera, and Toriello (2022). Under reasonably mild conditions, the expected routing time when locations are sampled randomly from a geographic distribution exhibits economies of scale as the number of locations increases, growing in proportion to the square root of the number of locations (Beardwood, Halton, and Hammersley 1959); when considering discrete arrivals, this translates to submodularity. For a recent survey on applications of continuous approximations in logistics, see Franceschetti, Jabali, and Laporte (2017). This article contributes to the tactical design SDD literature by considering a multi-vehicle setting with arbitrary arrival rates, which was generally not accommodated by previous work.

2.2. Machine Scheduling

Machine scheduling problems concern the assignment and processing of jobs on a machine, often one of several, with typical objectives such as minimizing makespan, lateness, weighted completion times, etc. MSMD generalizes three machine scheduling paradigms: (i) parallel batching, (ii) serial batching, and (iii) family setups. The parallel-batching problem requires batches to be assigned to machines, with the processing time of a batch being the longest processing time among jobs in the batch. This problem has received attention for its applications in semiconductor production; see Chang, Damodaran, and Melouk (2004), Tian et al. (2009). Recently, Muter (2020) studies the problem without release dates, and Fowler and Mönch (2022) presents a survey.

The serial-batching problem also requires batches of jobs to be assigned to machines; however, the processing time of a batch is just the sum of processing times of jobs in the batch. This problem has been studied since the 1960's, with seminal heuristic work by Graham (1969), Garey and Johnson (1978), and relevant complexity results in the 70's (Garey and Johnson 1979). Work continues to this day in heuristics (Kuruvilla and Paletta 2015, Habiba et al. 2019), exact algorithms (Dell'Amico et al. 2008), and approximation algorithms (Ghalami and Grosu 2019).

Extending the serial-batching problem, researchers have studied variants where the processing time function exhibits economies of scale, such as with family setups. Family setup times plus release times were considered for the first time by Schutten, Van De Velde, and Zijm (1996). Recent work includes Pessan and Néron (2011), Schaller (2014); in particular, Kramer, Iori, and Lacomme (2021) focus on family setups, but under the weighted completion time objective and without release times. As in this article, they develop multiple MILP formulations; however, their models are of pseudo-polynomial size.

We contribute to this literature by proposing MILP formulations that generalize various machine scheduling problems, including serial batching, parallel batching, and batching with family setups. In particular, our methods allow us to study these problems through the same lens and to accommodate *mixed-batch processing* problems (combinations of serial, parallel, and family batch problems); this has been identified as a gap in the literature (Fowler and Mönch 2022).

2.3. Submodular Optimization

Submodular functions have been extensively studied in combinatorial optimization (e.g. Krause and Golovin 2014, Nemhauser and Wolsey 1999, Schrijver 2003), and can be minimized in polynomial time (Iwata, Fleischer, and Fujishige 2001, Schrijver 2000); however, most other submodular optimization problems are NP-Hard. Many applications require the ground set to be partitioned while optimizing over a different objective, and often with additional side constraints; see Bogunovic et al. (2017), Chekuri and Ene (2011), Hirayama et al. (2023), Wang et al. (2021). In particular, MSMD seeks a partition of the ground set of orders N into batches, and an assignment of those batches into vehicles to minimize the makespan, a scheduling objective; to the best of our knowledge, this is novel in the submodular optimization literature.

3. Model Formulation and Preliminaries

The Multi-Vehicle Submodular Dispatching Problem (MSMD) is characterized by an order set $N := \{1,2,\dots,n\}$, where each order $i \in N$ has a release time $r_i \geq 0$, and by a vehicle set $M := \{1,2,\dots,m\}$, where each vehicle $k \in M$ is associated with a non-decreasing, submodular set function $f_k : 2^N \to \mathbb{R}_+$ with $f_k(\varnothing) = 0$. By translating and relabeling, we may assume $0 = r_1 \leq r_2 \leq \cdots \leq r_n$. Each order must be assigned to a batch, each batch assigned to a vehicle, and a vehicle's dispatches need to be scheduled so that they do not overlap in time; the goal is to minimize the makespan. Formally, a solution is an ordered list of vectors, where vector $k \in M$ indicates the batches $S \subseteq N$ that vehicle k dispatches, and the departure time of each dispatch. Define batch collections $\mathcal{N}_i := \{S \subseteq \{1,\dots,i\} : i \in S\}$ for all $i \in N$; these collections partition the power set of N, $\bigcup_{i\in N} \mathcal{N}_i = 2^N$; vehicle k's dispatch vector has $2n$ coordinates with values $(t_{1,k}, S_{1,k}, \dots, t_{n,k}, S_{n,k})$, where $S_{i,k} \in \mathcal{N}_i \cup \{\varnothing\}$; if $S_{i,k} \neq \varnothing$, then $t_{i,k}$ represents the departure time of batch $S_{i,k}$ dispatched by vehicle k. Using this structure, we expand the MILP formulation from Erazo and Toriello (2024) to the heterogeneous multi-vehicle case. We consider the following variables:

$x_{S,k} \in \{0,1\}$: indicates if batch $S \subseteq N$ is dispatched by vehicle $k \in M$.

$t_{i,k} \geq 0$: departure time of the i-th dispatch by vehicle $k \in M$, if it occurs, for $i \in N$.

$z \geq 0$: makespan.

PROPOSITION 1. *MILP (1) solves MSMD:*

$$
\begin{aligned}
& \min z \\
& \text{s.t. } t_{i,k} \geq r_i && \forall i \in N,\ \forall k \in M && \text{(1a)} \\
& \quad t_{i+1,k} \geq t_{i,k} + \sum_{S \in \mathscr{N}_i} x_{S,k} f_k(S) && \forall i \leq n-1,\ \forall k \in M && \text{(1b)} \\
& \quad z \geq t_{n,k} + \sum_{S \in \mathscr{N}_n} x_{S,k} f_k(S) && \forall k \in M && \text{(1c)} \\
& \quad \sum_{k \in M} \sum_{\substack{S \subseteq N \\ S \ni i}} x_{S,k} = 1 && \forall i \in N && \text{(1d)} \\
& \quad z \geq 0, t \geq 0, x \in \{0,1\}.
\end{aligned}
$$

Next, we establish some complexity for MSMD.

PROPOSITION 2. *MSMD is strongly NP-Hard even if all release dates are equal, vehicles are identical and dispatch times are modular.*

Proof: If $r_1 = \cdots = r_n = 0$ and $f_k(S) = f(S) = \sum_{i \in S} \tau_i$ for all $k \in M$ and $S \subseteq N$, our problem corresponds to serial scheduling on identical machines, proved to be strongly NP-Hard by Garey and Johnson (1979). ■

PROPOSITION 3. *MSMD is strongly NP-hard even if the batches are fixed.*

Proof: This again corresponds to a serial scheduling problem on identical machines. ■

PROPOSITION 4. *Assume the batches are fixed and each batch is assigned to a vehicle. Then the makespan can be computed in $O(mn)$ time, even for heterogeneous vehicles.*

Proof: Erazo and Toriello (2024) proved that when the assignment of batches to a vehicle is given, the makespan of that vehicle can be computed in $O(n)$ time. If we perform the procedure sequentially for all vehicles we get an $O(mn)$ algorithm. ■

We next consider a simple lower bound; let $[i, j] := \{i, i+1, \ldots, j\}$ denote an *interval* batch.

PROPOSITION 5. *The value* $\max_{i \in N}\{r_i + \min_{k \in M}\{f_k([i,n])\}/\min\{m, n-i+1\}\}$ *is a lower bound for MSMD.*

Proof: At time r_i, only orders $1,\dots,i-1$ can be dispatched. From submodularity, the remaining total dispatch time across all vehicles cannot be less than $\min_{k\in M}\{f_k([i,n])\}$. Even if orders $1,\dots,i-1$ have been dispatched, there are still $n-i+1$ orders left, so the number of vehicles that can be used to dispatch them is the minimum between m and $n-i+1$. We obtain a lower bound by assuming that the workload can be perfectly divided among the maximum number of vehicles that can be used for these dispatches. ■

Formulation (1) has $O(m2^n)$ x variables Therefore, we require column generation to solve the LP relaxation of this problem.

PROPOSITION 6. *The linear relaxation of* (1) *can be solved in polynomial time.*

Proof: We relax the binary domain for each $x_{S,k}$ variable to non-negativity, and consider the dual linear program of (1). Let α be the dual variable for (1a), β for (1b) and (1c), and γ for (1d). The dual constraints corresponding to the (relaxed) x variables are

$$-\beta_{i,k} f_k(S) + \sum_{j\in S}\gamma_j \leq 0, \qquad\qquad i\in N, k\in M, S\in \mathscr{N}_i.$$

For each $i\in N, k\in M$, the separation problem of the dual linear program is then

$$\min_{S\in\mathscr{N}_i}\left\{\beta_{i,k} f_k(S) - \sum_{j\in S}\gamma_j\right\} = \min_{S\subseteq[1,i-1]}\left\{\beta_{i,k} f_k(S\cup i) - \gamma_i - \sum_{j\in S}\gamma_j\right\}.$$

As all functions f_k are submodular, each of these optimization problems is a submodular minimization problem, which can be solved in polynomial time in the oracle model (Schrijver 2003), and we need to solve a polynomial number of them. From the equivalence of separation and optimization (Grötschel, Lovász, and Schrijver 1993), it follows that the LP relaxation of (1) can be solved in polynomial time. ■

Formulation (1) has significant symmetry, because it has k variables for each subset $S\subseteq N$; that is also reflected in the separation problem for each $i\in N, k\in M$. In Section 4, we present formulations that alleviate those issues.

4. Identical Vehicles

4.1. Symmetry-Reducing Formulation

We leverage the fact that all vehicles have the same dispatch time function to create a new formulation that only has one variable for each subset of orders $S\subseteq N$. The new formulation has the following variables:

$x_S \in \{0,1\}$: indicates if batch $S \subseteq N$ is dispatched.

$t_{i,k}$: departure time of the i-th dispatch by vehicle $k \in M$, if it occurs, for $i \in N$.

z: makespan.

$w_{i,k}$: departure time of i-th dispatch assigned to machine k, if it occurs, for $i \in N$, $k \in M$.

$y_{i,k} \in \{0,1\}$: indicates if the i-th dispatch is performed by vehicle $k \in M$.

Intuitively, instead of choosing the batches and assignments simultaneously with variables $x_{S,k}$ as in (1), we make the batch decisions with variables x_S and the assignment decisions with variables $w_{i,k}$ and $y_{i,k}$.

PROPOSITION 7. *MILP (2) solves MSMD:*

$$\begin{aligned}
\min\ & z \\
\text{s.t. } & t_{i,k} \geq r_i && \forall i \in N,\ \forall k \in M && \text{(2a)}\\
& t_{i+1,k} \geq t_{i,k} + w_{i,k} && \forall i \leq n-1,\ \forall k \in M && \text{(2b)}\\
& z \geq t_{n,k} + w_{n,k} && \forall k \in M && \text{(2c)}\\
& \sum_{k=1}^{m} w_{i,k} = \sum_{S \in \mathcal{N}_i} x_S f(S) && \forall i \in N && \text{(2d)}\\
& w_{i,k} \leq y_{i,k} f([1,i]) && \forall i \in N,\ \forall k \in M && \text{(2e)}\\
& \sum_{k=1}^{m} y_{i,k} \leq 1 && \forall i \in N && \text{(2f)}\\
& \sum_{S \subseteq N : S \ni i} x_S = 1 && \forall i \in N && \text{(2g)}\\
& z \geq 0,\ t \geq 0,\ w \geq 0,\ y \in \{0,1\},\ x \in \{0,1\}.
\end{aligned}$$

Furthermore, its linear relaxation can be solved in polynomial time.

The proof can be found in Appendix A.1. Formulation (2) significantly reduces the number of batch variables x, but adds new variables w, y and new big-M constraints. Despite adding those new variables and constraints, the LP relaxation can be solved in polynomial time, just as the one from (1). We now compare the quality of the lower bounds and solutions generated by the relaxations.

THEOREM 1. *The linear relaxations of formulations (1) and (2) have equal optimal values. Moreover, given an extreme point feasible solution for one of the linear relaxations, we can obtain a feasible solution for the other linear relaxation in polynomial time.*

The proof is presented on Appendix A.2. Theorem 1 implies we can interchangeably solve the LP relaxation of either formulation. We can therefore solve the pricing problem for (2), which requires $O(n)$ submodular minimizations instead of $O(mn)$. Formulation (2) keeps the same quality for the LP bound, eliminates a significant number of variables, and also reduces the number of separation problems to be solved; however, it also introduces some symmetry with variables w, y. Next, we present a result we use as basis for two other formulations for MSMD.

PROPOSITION 8. *Consider an instance of MSMD with $n \geq m$ orders. Without loss of optimality, each vehicle performs at least one dispatch.*

Proof: Assume by contradiction that some vehicle performs no dispatches; then there must be at least one vehicle that performs two dispatches, or that has a dispatch with more than one order. In the former scenario we can assign one of the multiple dispatches to a vehicle that has no dispatches, and the makespan cannot increase. In the latter scenario we can split the batch into two sub-batches, both with smaller or equal dispatch time because of f's monotonicity. Then, by assigning one of the sub-batches to the idle vehicle, we cannot increase the makespan. ■

4.2. Flow-Based Formulation

Using Proposition 8, we propose a flow-based formulation for MSMD. We introduce a dummy source node indexed by 0 and a dummy sink node indexed by $n+1$. We minimize the makespan while sending m units of flow from the source node to the sink node; each unit of flow will go through a path that represents the dispatch schedule of a vehicle. This formulation's variables are:

$x_S \in \{0,1\}$: indicates if batch $S \subseteq N$ is dispatched.

t_i: departure time of the i-th dispatch, if it occurs, for $i \in N$.

$z = t_{n+1}$: makespan.

$y_{ij} \in \{0,1\}$: indicates if the i-th dispatch is performed immediately before the j-th dispatch by the same vehicle, for $0 \leq i < j \leq n+1$.

PROPOSITION 9. *Formulation (3) solves MSMD:*

$$\min\ z = t_{n+1}$$

$$\text{s.t. } t_i \geq r_i \qquad \forall i \in N \tag{3a}$$

$$\sum_{i \in N} y_{0,i} = m \tag{3b}$$

$$\sum_{i \in N} y_{i,n+1} = m \tag{3c}$$

$$\sum_{S \subseteq N: S \ni i} x_S = 1 \qquad \forall i \in N \tag{3d}$$

$$\sum_{S \in \mathcal{N}_i} x_S = \sum_{j=0}^{i-1} y_{j,i} \qquad \forall i \in N \tag{3e}$$

$$\sum_{j=0}^{i-1} y_{j,i} = \sum_{j=i+1}^{n+1} y_{i,j} \qquad \forall i \in N \tag{3f}$$

$$t_j \geq t_i + \sum_{S \in \mathcal{N}_i} f(S) x_S - (1 - y_{i,j}) f([1,i]) \qquad \forall (i,j):\ 1 \leq i < j \leq n+1 \tag{3g}$$

$$t \geq 0,\ y_{ij} \in \{0,1\},\ x \in \{0,1\}.$$

Furthermore, the linear relaxation can be solved in polynomial time.

The proof can be found in Appendix A.3. The presence of big-M constraints causes this formulation to have a weak LP relaxation in some instances.

4.3. Set Cover Formulation

For a set cover formulation, we consider slightly redefined variables $x_{S,k}$ that indicate the complete set of orders S dispatched by some vehicle, potentially in multiple batches. The formulation has the following variables:

$x_{S,k} \in \{0,1\}$: if orders $S \subseteq N$ are dispatched by vehicle $k \in M$, possibly in multiple dispatches.

z: makespan.

We denote the optimal makespan of a single-vehicle SMD dispatching orders $S \subseteq N$ as $\text{SMD}(S)$.

PROPOSITION 10. *Formulation (4) solves MSMD:*

$$\min \ z$$

$$\text{s.t. } z \geq \sum_{S \subseteq N} \mathrm{SMD}(S) x_{S,k} \qquad \forall k \in M \tag{4a}$$

$$\sum_{S \subseteq N} x_{S,k} = 1 \qquad k \in M \tag{4b}$$

$$\sum_{k \in M} \sum_{S \subseteq N, S \ni i} x_{S,k} = 1 \qquad \forall i \in N \tag{4c}$$

$$x \in \{0,1\}, z \geq 0.$$

Proof: Because of constraint (4c), vector x partitions the order set N. Constraint (4b) makes sure we assign one batch to each vehicle, and the correctness of the formulation comes from the fact that $\mathrm{SMD}(S)$ returns a feasible solution for a single vehicle dispatching S; therefore, each vehicle has a feasible schedule. ■

Unlike many set cover formulations, (4) includes multiple copies of each set variable, one per vehicle $k \in M$. This symmetry is unavoidable because of the makespan objective and constraints (4a); if we used only one copy of each set variable, we would need exponentially many constraints to define the makespan. Next, we discuss a simplification of the formulation.

PROPOSITION 11. *Constraints (4b) can be aggregated into a single constraint, $\sum_{k \in M} \sum_{S \subseteq N} x_{S,k} = m$, without affecting the formulation's correctness or the optimal value of the LP relaxation.*

Proof: Suppose an integer optimal solution has $x_{S_1,k} = x_{S_2,k} = 1$. Since $\sum_{k \in M} \sum_{S \subseteq N} x_{S,k} = m$, there is some $k' \in M$ with $x_{S,k'} = 0$ for all $S \subseteq N$. We can reassign either S_1 or S_2 to k' without loss of optimality. Now consider an optimal solution (x,z) for the linear relaxation with the aggregated constraint. Define a solution (x',z) with $x'_{S,k} = (1/m) \sum_{k \in M} x_{S,k}$, for all $S \subseteq N$ and $k \in M$. As x defines a fractional partition of N, x' does also, so (4c) holds. From construction, (4b) holds, and the right-hand sides of all constraints (4a) for x' are equal, hence no larger than the largest right-hand side among (4a) for x; therefore, the fractional makespan with x' cannot be larger. ■

From now on, we use (4) to refer to the formulation with the aggregated version of constraint (4b), as it simplifies the analysis. Consider the linear relaxation of (4) where we relax the binary domain for each $x_{S,k}$. Let α be the dual variable of (4a), β be the dual variable of (4b) and γ be the dual variable of (4c). The dual constraints corresponding to the (relaxed) x variables are $-\alpha_k \,\mathrm{SMD}(S) + \beta + \sum_{i\in S}\gamma_i \leq 0$, for all $S \subseteq N$ and $k \in M$. For each $k \in M$, the separation problem for these constraints is then

$$\min_{S\subseteq N}\left\{\alpha_k \,\mathrm{SMD}(S) - \beta - \sum_{i\in S}\gamma_i\right\}. \tag{5}$$

Erazo and Toriello (2024) proved that computing $SMD(N)$ is in general strongly NP-Hard; thus the separation problem is strongly NP-Hard. We next study the complexity of (5) under additional assumptions.

PROPOSITION 12. *Suppose all orders have the same release time; then (5) can be solved in polynomial time.*

Proof: We may assume that $r_i = 0$ for all $i \in N$. From subadditivity we have $\mathrm{SMD}(S) = f(S)$ for $S \subseteq N$. As f is submodular and $\alpha_k \geq 0$, the separation problem is submodular minimization. ■

Conversely, if we have arbitrary release times the problem becomes significantly harder, even for modular functions f, as we prove in the next theorem.

THEOREM 2. *Suppose each order $i \in N$ is associated with a value $\tau_i > 0$, and let $f(S) = \sum_{i\in S}\tau_i$; this corresponds to scheduling on identical serial machines with release times. The separation problem (5) is NP-Hard.*

We use a reduction from the knapsack problem; see Appendix B.1. Theorem (2) establishes that the separation problem is NP-Hard even for modular functions f; nevertheless, this special case and its extensions to serial-batch scheduling with release times have a pseudo-polynomial separation algorithm, which allows us to solve the LP relaxation in reasonable time in certain cases.

THEOREM 3. *Suppose each order $i \in N$ is associated with a value $\tau_i > 0$, and consider a fixed setup time $\tau_0 \geq 0$. For $f(S) = \tau_0 + \sum_{i\in S}\tau_i$, (5) can be solved in pseudo-polynomial time. If r_i, τ_i are integer for $i \in N$, and given an integer upper bound U for* $\mathrm{SMD}(N)$*, the complexity of solving (5) is $O(Un^2)$; moreover, if $\tau_0 = 0$, the complexity is $O(Un)$.*

The proof is in Appendix B.2. Finally, we can also extend our complexity result to the serial-batch scheduling problem with family setups.

THEOREM 4. *Suppose each order $i \in N$ is associated with a value $\tau_i > 0$; assume Q families $F_1, F_2, \ldots, F_Q$ partition order set N, and each family q has a setup time $\tau_q \geq 0$. For $f(S) = \sum_{i\in S} \tau_i + \sum_{q:F_q\cap S\neq\varnothing} \tau_q$, (5) can be solved in pseudo-polynomial time. Let U be an integer upper bound on* SMD(N)*, and define $U_q = \tau_q + \sum_{j\in F_q} \tau_j$ for $q = 1,\ldots,Q$. If vectors r,τ are integer, the complexity is $O\Big(nU\Big[\prod_{q=1}^{Q} U_q|F_q|\Big]\Big)$.*

The proof is in Appendix B.3.

4.4. Interval-Solvable Functions

We define a solution to be of *interval type*, or simply an interval solution, if its batches all have a minimum index i, a maximum index j, and the batch contains all orders in the interval $[i, j]$. A function f is *interval-solvable* if all instances defined by f have an optimal interval solution.

THEOREM 5. *Suppose each $i \in N$ is associated with a number $\tau_i > 0$. Moreover, consider some $\tau_0 \geq 0$ and a concave non-decreasing function $g : \mathbb{R} \to \mathbb{R}$ with $g(0) = 0$. The following functions are interval-solvable: (1) $f(S) = \tau_0 + \max_{i\in S}\{\tau_i\}$ and (2) $f(S) = \tau_0 + g(|S|)$. Furthermore, $f(S) = \sum_{i\in S} \tau_i$ is not only interval-solvable, it suffices to consider singleton batches.*

The proof is in appendix C. Theorem 5 verifies that some important classes of functions are interval-solvable; for instance, (1) includes parallel-batch scheduling with setups on identical machines, and (2) includes tactical design of SDD systems with identical vehicles.

4.5. Formulation Comparison

Table 1 presents a comparison of formulations (1) through (4) based on their numbers of constraints and variables, differentiating between the general case and the interval-solvable case. Depending on the function f and the number of vehicles m, the trade-offs between formulations vary significantly. In particular, when f is interval-solvable (or singleton-solvable), (1) becomes more attractive because of its relatively low number of constraints and lack of big-M coefficients.

With respect to the linear relaxation bounds, (1), (3) and (4) are incomparable; we explore this question computationally below in Section 5. Next, we study the multiplicative gap between these bounds and the optimal solution; recall that the bounds provided by (1) and (2) are equal.

Table 1 **Comparison of the four proposed formulations by number of constraints and variables.**

		MILP (1)	MILP (2)	MILP (3)	MILP (4)
	Constraints	$2nm+n$	$3nm+3n$	$0.5n^2+4.5n+2$	$m+n+1$
Variables	General problem	$m(2^n-1)+nm+1$	2^n+3nm	$2^n+0.5n^2+2.5n$	$m(2^n-1)+1$
	Interval-solvable	$0.5mn^2+1.5nm+1$	$0.5(n^2+n)+3nm+1$	n^2+3n+1	$0.5m(n^2+n)+1$
	Singleton-solvable	$2nm+1$	$n+3nm+1$	$0.5n^2+3.5n+1$	$mn+1$

PROPOSITION 13. *Let z_I^* be the optimal makespan of MSMD for an instance I with m vehicles and $n \geq m$ orders. Let $z_I^{LP(1)}$, $z_I^{LP(3)}$, $z_I^{LP(4)}$ be the optimal (fractional) makespan of the linear relaxations of (1), (3) and (4), respectively, for instance I. Even when f is modular and all release times are zero, there exists a family of instances $I_1, I_2, \dots$ such that $\lim_{h\to\infty} z_{I_h}^*/z_{I_h}^{LP(1)} = \lim_{h\to\infty} z_{I_h}^*/z_{I_h}^{LP(4)} = m$, and $\lim_{h\to\infty} z_{I_h}^*/z_{I_h}^{LP(3)} = \infty$.*

The proof is in Appendix D.1. Our next result shows that for modular functions, the lower bound provided by the linear relaxation of (1) is dominated by the lower bound from Proposition 5.

PROPOSITION 14. *Let each order $i \in N$ be associated with a value $\tau_i > 0$, and let $f(S) = \sum_{i\in S} \tau_i$. The lower bound presented in Proposition 5 is greater than or equal to the lower bound given by the linear relaxation of (1).*

The proof is in Appendix D.2. This result indicates that our formulations may sometimes be weak compared to combinatorial bounds that exploit the structure of f and other problem parameters. With this motivation, we next propose a strengthening for set cover formulation (4).

THEOREM 6. *Let LB be a lower bound for the optimal makespan. Constraint (4a) of formulation (4) can be strengthened to $z \geq \sum_{S\subseteq N} \max\{\text{SMD}(S), LB\} x_{S,k}$; furthermore, all previous complexity results on the formulation remain unchanged.*

The proof is in Appendix D.3. With this strengthening, the set cover formulation matches (or exceeds) the worst-case performance of any of our lower bounds. For instance, by using $LB = \max_{i\in N}\{\tau_i\}$, we get an optimal bound for the worst-case instances from Proposition 13.

5. Computational Study and Discussion

In this section we present two different sets of experiments. First, we verify the empirical efficacy of our methods by considering the machine scheduling problem with serial batching and family setups, known to be strongly NP-Hard. Then, we focus on tactical design in same-day delivery, as these problems consider an interval-solvable dispatch time function; the efficient solution of many related instances allows us to obtain managerial insights for these systems.

5.1. Machine Scheduling - Serial Batching with Family Setups

In this experiment we study the quality of the linear relaxation lower bounds for our MILP formulations, the quality of heuristics that rely on solving the LP relaxation with column generation, and the computational performance and scalability of our methods. We test our algorithms with instances of serial-batch scheduling on identical machines with family setups and release times, a strongly NP-Hard problem that is not interval-solvable. Similar problems have been tackled in the machine scheduling literature; for example, Kramer, Iori, and Lacomme (2021) recently considered instances with up to $n = 80$ for a similar problem without release times and with a different objective, the weighted completion time.

For instance design, we use a similar setup to Kramer, Iori, and Lacomme (2021) and previous works: the values τ_i are drawn from a uniform (integer) distribution with minimum value of 1 and maximum value of 100; family setup times τ_q are drawn from a uniform (integer) distribution with minimum value of 0 and maximum value of U_F, which we vary over our experiments. Finally, the inter-arrival times between orders are also drawn from a uniform (integer) distribution, with minimum value of 0, and maximum value of U_r.

We generate 25 instances for each parameter combination of (m, Q, U_F, U_r), and use Python 3.11, Gurobi 10.0.1 and a Windows machine with 16 GB of RAM and an Intel Core i7-12650H processor for our experiments. We use the following notation to refer to the different bounds obtained from the LP Relaxations and solutions we evaluate:

- IP: MILP formulation (1).
- LP (a): LP relaxation of MILP (a), solved with column generation, for $a = 1, 3, 4$.
- CG IP (a): MILP (a) restricted to the columns generated when computing LP (a), for $a = 1, 3, 4$.
- LBF: Lower bound based on Proposition 5, but leveraging specific aspects of this problem. LBF is detailed in Appendix E.1.
- LPS (4): LP relaxation for the strengthened set cover formulation. The lower bound considered for the strengthened constraints is LBF.

- CGS IP (4): Strong set cover formulation constrained to the columns generated when solving LPS (4).
- Interval IP: MILP (1) constrained to interval batches, which is not guaranteed to be optimal.

For column generation methods, we use the acceleration technique from Ben-Ameur and Neto (2007). For LP (1) we use the separation problem from (2), which does not differentiate by vehicle; for CG IP (4) and CGS IP (4) we use an MILP to solve the separation problem, because preliminary tests showed this to be faster than the algorithm from Theorem 4. Tables 2 and 3 show the results of our first set of instances, where $n = 15$ and the goal is to compare our bounds versus the actual optimal solution, computed using (1); a gap of 100% means the solution has the same objective as the actual optimal solution.

From Table 2 we see that LBF provides a strong bound, over 79.7% gap geometric mean for all sets of instances with three machines, and over 90.4% for five. LPS (4) leverages LBF to further improve the bound to a gap geometric mean of at least 95.9% for sets of instances with three and five machines. With respect to the other bounds, (1), (3) and (4) are incomparable, but they complement each other. When m increases, the gaps for (1) and (4) decrease, whereas the opposite happens for (3), LBF and LPS (4); the latter three exceed an 87% gap geometric mean for all sets of instances with five vehicles. When Q increases, LBF and LP (3) exhibit similar performance, whereas LP (1) and the two set-cover methods improve their bounds. Larger values of U_F translate to more savings due to batching; an increase in U_F causes LP (1) to decrease its performance; other LP methods have a worse performance if instances are dense ($U_r = 25$), but improve when arrivals are sparse. Increases in U_r significantly reduce the quality of the set cover LP (4); but other methods improve significantly. In particular, LP (3) has the highest increase, and performs better than LP (1) under these conditions. With respect to running time, methods LP (1), LP (3) and LBF require less than 0.1 seconds on average for every instance set; on the other hand, LP (4) and LPS (4) take over 100 seconds for some instances.

With respect to the heuristics, Table (3) indicates that the heuristics based on the set cover formulation have very bad performance; this is because a very small number of columns are generated when solving those LP's. CG IP (1) performs particularly well, with a gap geometric mean of 104.5% at most, and a

Table 2 **Geometric and (worst) gap percentage for our lower bounds on instances with $n = 15$, compared versus the MILP optimal solution.**

m	Q	U_F	U_r	LP (1)	LP (3)	LPS (4)	LP (4)	LBF
				Lower bounds' performance, geometric (worst) gap %				
3	2	50	25	85.8 (81.6)	67.1 (52.9)	96.9 (94.7)	88.8 (76.6)	82.2 (76.9)
			100	93.3 (87.7)	98.3 (88.0)	99.4 (97.0)	48.5 (38.9)	98.7 (93.4)
		100	25	81.3 (72.1)	69.5 (45.0)	95.9 (93.4)	87.8 (93.6)	79.7 (72.0)
			100	90.7 (85.2)	98.5 (84.0)	99.4 (93.6)	52.9 (40.1)	98.6 (86.2)
	5	50	25	89.9 (84.8)	70.9 (54.5)	97.9 (95.7)	91.8 (85.1)	87.7 (80.5)
			100	93.2 (86.8)	99.1 (89.6)	99.6 (95.9)	47.9 (37.5)	99.4 (93.8)
		100	25	87.1 (82.7)	68.0 (49.4)	96.2 (91.2)	91.4 (81.3)	83.8 (77.2)
			100	91.8 (83.0)	98.4 (91.6)	99.3 (96.2)	56.1 (47.3)	98.8 (94.3)
5	2	50	25	75.7 (68.8)	94.1 (78.1)	97.8 (92.3)	72.1 (54.5)	95.5 (84.7)
			100	92.2 (85.9)	99.0 (93.1)	99.5 (96.5)	32.0 (24.6)	99.3 (95.6)
		100	25	72.0 (64.5)	88.3 (59.0)	95.9 (90.4)	72.0 (56.0)	90.4 (73.5)
			100	88.8 (82.1)	99.3 (90.6)	99.6 (96.1)	36.8 (25.4)	99.5 (95.1)
	5	50	25	77.3 (67.7)	91.1 (58.1)	97.2 (88.7)	76.2 (52.4)	93.9 (77.8)
			100	92.7 (86.9)	99.1 (93.6)	99.4 (95.6)	33.7 (24.5)	99.2 (94.5)
		100	25	74.6 (68.9)	87.7 (47.5)	96.1 (87.8)	77.5 (65.0)	90.5 (68.7)
			100	88.4 (82.4)	99.6 (95.6)	99.8 (97.5)	39.9 (26.4)	99.7 (96.6)

worst gap (over the 400 instances) of 122.4%. The Interval IP method also provides high-quality solutions (maximum 108.0% gap geometric mean), and is better than CG IP (1) in some cases. With respect to parameters, an increase in arrival sparsity (i.e. U_r increases) improves the heuristics substantially, with many reaching optimality in almost all instances with $U_r = 100$. An increase in U_F (i.e. savings due to batching) decreases the performance of Interval IP, and also CG IP (1) though only slightly. When the number of families increases, the gaps for CG IP (1) improve, whereas the opposite happens for the Interval IP solution. Finally, when the number of machines increases, the gaps for CG IP (1) stay almost equal, whereas the gap of Interval IP increases. With respect to running time, all methods solve within 0.1 seconds, except for CG IP (3), which takes up to over 2 seconds.

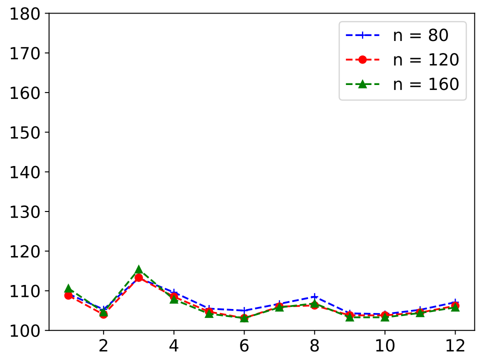
For our large experiments, we consider $n = 80, 120, 160$; $m = 5, 10, 15$; $Q = 5, 10, 15$; $U_F = 25, 100$, and $U_r = 10, 25$; we select smaller values for U_r to benchmark against harder instances. Because of their speed and empirical performance, we use LP (1) and LBF as lower bounds; in each instance we choose the maximum value between the two as our benchmark. On the heuristic side, we keep CG IP (1) and the Interval IP and report their performance. For all instances, we allow a run time of up to two minutes for both heuristic methods. We summarize results in Figure 1, where each sub-figure depicts the performance of a given heuristic, given a fixed value of m, and for $n = 80, 120, 160$. Our scenarios for (Q, U_F, U_r) are:

Table 3 **Geometric and (worst) gap percentage for our heuristics on instances with $n = 15$, compared versus the MILP optimal solution.**

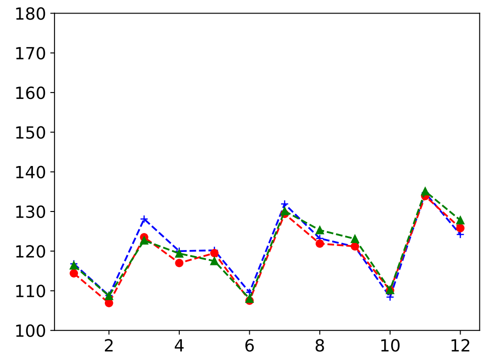
m	Q	U_F	U_r	CG IP (1)	CG IP (3)	CGS IP (4)	CG IP (4)	Interval IP
				Heuristics' performance, geometric (worst) gap %				
3	2	50	25	104.5 (113.7)	115.2 (131.6)	246.2 (267.2)	230.6 (251.2)	104.1 (111.3)
			100	100.0 (100.0)	100.0 (100.0)	130.0 (164.5)	130.0 (164.5)	100.0 (100.0)
		100	25	103.5 (110.8)	134.7 (175.7)	231.1 (265.1)	222.1 (237.8)	108.0 (117.4)
			100	100.0 (101.0)	100.9 (107.0)	135.5 (171.3)	135.5 (171.3)	100.1 (101.2)
	5	50	25	100.4 (102.8)	108.6 (120.5)	257.8 (275.1)	227.1 (253.7)	104.2 (108.6)
			100	100.0 (100.0)	100.0 (100.0)	128.8 (161.6)	128.8 (161.6)	100.0 (100.0)
		100	25	100.8 (104.8)	122.5 (137.6)	249.7 (274.3)	221.6 (244.3)	109.9 (123.1)
			100	100.0 (100.0)	100.1 (103.0)	146.4 (178.3)	146.4 (178.3)	100.0 (101.0)
5	2	50	25	100.2 (102.1)	101.3 (111.5)	279.5 (300.0)	174.9 (201.5)	100.6 (107.3)
			100	100.0 (100.0)	100.0 (100.0)	124.0 (163.7)	123.8 (158.5)	100.0 (100.0)
		100	25	103.5 (122.4)	107.1 (124.2)	265.5 (300.0)	178.4 (192.8)	101.8 (106.7)
			100	100.0 (100.0)	100.0 (100.0)	135.6 (186.8)	134.5 (178.1)	100.0 (100.0)
	5	50	25	100.1 (101.0)	100.7 (104.6)	281.4 (300.0)	174.6 (194.9)	100.5 (104.6)
			100	100.0 (100.0)	100.0 (100.0)	132.6 (169.1)	132.4 (169.1)	100.0 (100.0)
		100	25	101.3 (110.7)	106.2 (121.8)	271.7 (302.1)	176.9 (194.8)	103.4 (116.7)
			100	100.0 (100.0)	100.0 (100.0)	153.6 (187.3)	151.9 (184.2)	100.0 (100.0)

1. (5, 50, 10)
2. (5, 50, 25)
3. (5, 100, 10)
4. (5, 100, 25)
5. (10, 50, 10)
6. (10, 50, 25)
7. (10, 100, 10)
8. (10, 100, 25)
9. (15, 50, 10)
10. (15, 50, 25)
11. (15, 100, 10)
12. (15, 100, 25)

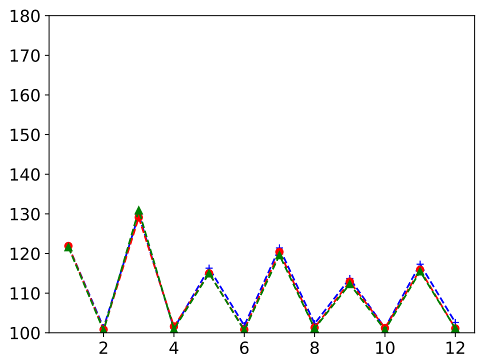
Figures 1a, 1c and 1e detail results for CG IP (1), while the other figures do the same for the Interval IP. The gap geometric means for CG IP (1) improve slightly when n increases; on the other hand, Interval IP has difficulty handling large values of n, as evidenced by gaps over 200% for $n = 160$. In particular, CG IP (1) achieves a gap geometric mean of 130% or less for all of our instance sets, and is within 1% of optimality in every single even instance set ($U_r = 25$), for $m = 10$ and $m = 15$. With respect to the effect of m, for both heuristics the worst-case and best-case gaps are amplified (i.e. worse and better, respectively) when $m = 10$; this suggests a concave structure where the values near the middle have either very low or very high gaps. With respect to Q, an increase seems to have a small positive effect for CG IP (1), and a slight negative effect for Interval IP. Finally, with respect to running times, we limited both CG IP (1) and Interval IP to run for at most two minutes; LP (1) required 44.7 seconds on average for $n = 160$, and 190 seconds for the worst instance set with $n = 160$; this shows we can obtain geometric gaps within 10% of optimality for all instance sets with $n = 160$ in less than six minutes.



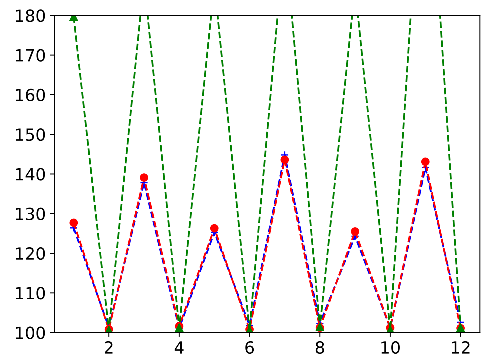
(a) CG IP (1), $m = 5$



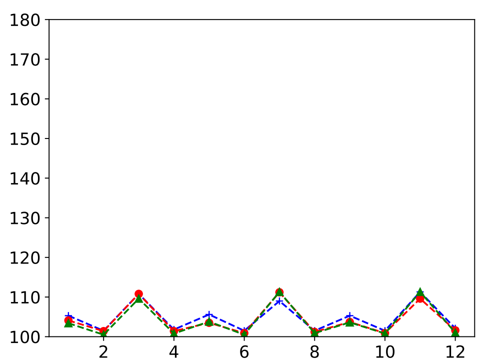
(b) Interval IP, $m = 5$



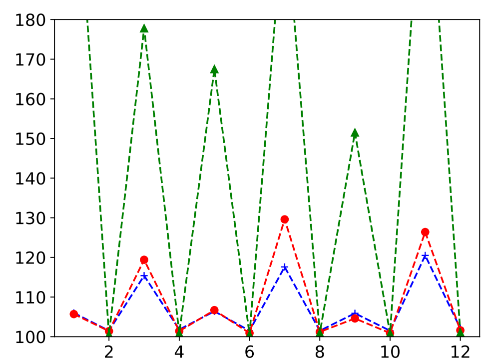
(c) CG IP (1), $m = 10$



(d) Interval IP, $m = 10$



(e) CG IP (1), $m = 15$



(f) Interval IP, $m = 15$

Figure 1 Geometric gaps (%) of CG IP (1) and Interval IP, versus the best lower bound, for instance scenarios 1-12. Each figure contains curves $n = 80,\ 120,\ 160$, given a fixed value of m.

5.2. Tactical Design for SDD

In our second set of experiments, we assess the impact on an SDD system when the size of the fleet increases. In the SDD context, the makespan objective of MSMD corresponds to the length of the delivery shift. Starting from a system with one vehicle and a shift length of ϕ^*, we investigate how many extra orders can be delivered if we increase the size of the fleet to m while ensuring the delivery shift length does not exceed ϕ^*. Moreover, we study the impact of increasing m on the structure of the optimal solution, i.e. the expected number of routes and route durations by vehicle.

We consider an SDD system studied in Stroh, Erera, and Toriello (2022) with a service area of roughly 26 square miles in northeastern metro Atlanta; it includes 22 census tracts and has a population of 92,198 as measured by the U.S. Census. The SDD system accepts orders between 9 AM and 2 PM; assuming 5% of the population in the region uses the SDD service once every two months, 50 people place orders each day on average. For this system, Stroh, Erera, and Toriello (2022) computed the dispatch-time function to be $f(S) = 10 + 1.5|S| + 24\sqrt{|S|}$ minutes. In our baseline scenarios, we set $r_1 = 9$ AM and $r_{50} = 2$ PM, a fleet of a single vehicle, and three arrival patterns for the orders: (i) constant order arrival rate, every six minutes; (ii) a U-shaped arrival rate: the first 15 orders arrive every two minutes, the next 20 orders arrive every 12 minutes, the last 15 orders arrive every two minutes; and (iii) arrivals concentrated towards the end of the order window: the first 20 orders arrive every 12 minutes, the last 30 orders arrive every two minutes. For these scenarios we use the algorithm from Erazo and Toriello (2024) to obtain the optimal solution for the single-vehicle problem.

When m increases, the SDD system has a larger delivery capacity; thus, the order deadline can be delayed, allowing more orders to be delivered while keeping a delivery shift length at or below ϕ^*. For $m > 1$, we use (1); as f is an interval-solvable function, we only need a quadratic number of variables and can use an off-the-shelf solver. For our two tests, we consider the following arrival patterns after 2 PM: (A) new orders arrive at a constant rate, every 6 minutes; and (B) new orders arrive at a constant rate, every 4 minutes. Algorithm 1 details our procedure for tests A and B; we warm-up our solver and leverage dual bounds to speed up the algorithm. Results for tests A and B under the different baseline scenarios are shown in Figure 2.

Algorithm 1 Procedure to find the number of orders that can be dispatched with m vehicles

Notation: f: dispatch time function used.

$FIFO(r, f)$: optimal polynomial-time algorithm from Erazo and Toriello (2024) that returns the optimal makespan for the single-vehicle SMD with arrival vector r and dispatch time function f.

Input: Initial arrival vector r (defined by the baseline scenario), structure of arrivals after original deadline (defined by test A or B).

```
Vehicles ← 1; Orders ← 50; Previous_Orders ← 50; Results = [50]
Solve single-vehicle SMD, return original makespan φ* ← FIFO(r, f)
Compute the maximum number of orders that can be dispatched under the particular (scenario, test) pair, denoted Max_Orders, by finding the latest order that can be dispatched by itself before φ*.
Vehicles ← Vehicles + 1
while Orders ≤ Max_Orders do
    Solve the SMD with (1), return makespan φ
    if φ ≤ φ* then
        Orders ← Orders + 1
        Update arrival r by adding new order, according to the current test (A or B)
    else
        Vehicles ← Vehicles + 1
        Append value Orders-1 to list Results
    end if
end while
Append value Max_Orders to list Results
```

Output: List of Results. The i-th entry has the maximum number of orders that can be dispatched with i vehicles while having a makespan $\phi \leq \phi^*$

As Figure 2 indicates, increasing m is more beneficial when the arrival rate for orders after the original 2 PM deadline is larger (test B). Furthermore, the benefits of increasing the fleet are also amplified when the original order arrival process is concentrated towards the end of the order window, as in baseline case (iii). This is due to two factors: first, the original solution has a larger makespan ϕ^* when the arrival process has the orders concentrated towards the end of the ordering period, and second, it is easier to get economies of scale due to batching with the new orders if the original orders are released later. Moreover, with respect to test A in all baseline scenarios, 90% of the maximum number of orders can be dispatched with just two vehicles, and over 95% with three vehicles; for test B, 80% of the maximum can be dispatched with two vehicles, and 90% with three.

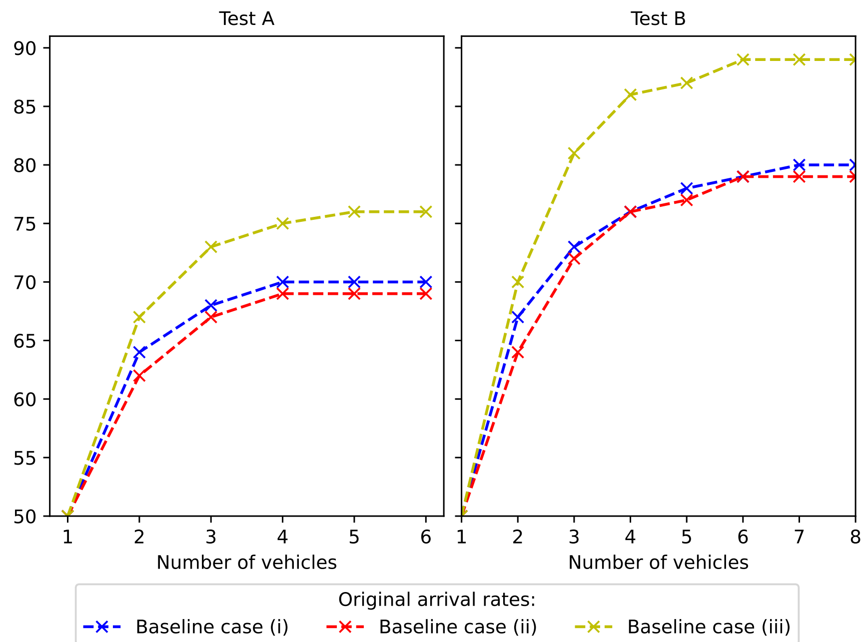
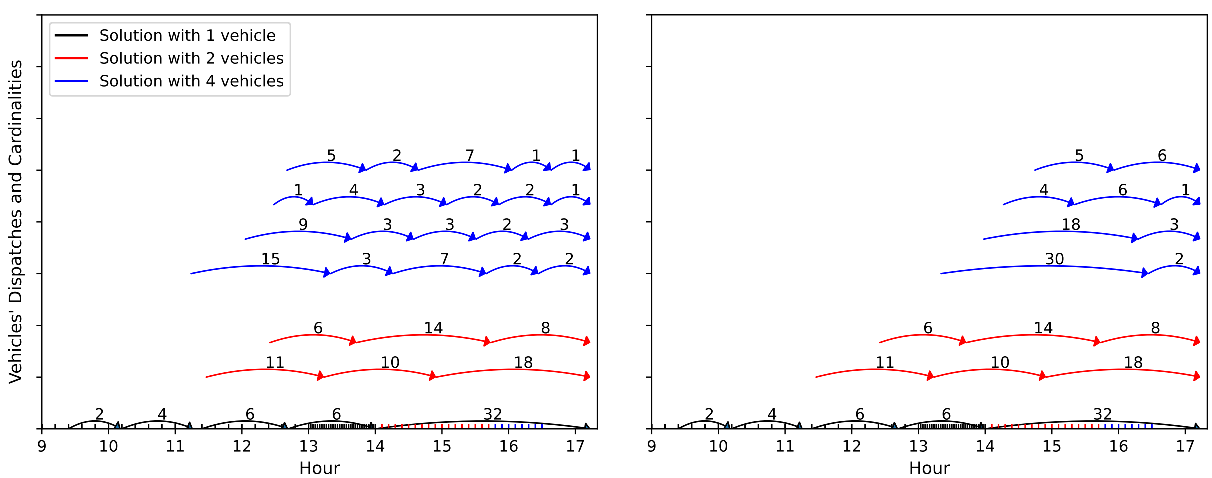


Figure 2 **Orders dispatched (y-axis) for different values of m, under tests A and B, and for all baseline cases.**



(a) Baseline case (i)

(b) Baseline case (iii)

Figure 3 **Structure of the solutions for test A, under baseline cases (i) and (iii).**

We now focus on the impact of m on the optimal solution. Figure 3a shows the changes for test A under baseline case (i), whereas Figure 3b does so for A under baseline case (iii). To obtain the structure of the optimal solutions we use a modified version of (1), shown in Appendix E.2.

The structure of the optimal solution depends heavily on the order arrival pattern before the original deadline. Figure 3a shows that for baseline scenario (i), increasing the fleet to two vehicles does not significantly affect the efficiency of dispatches in terms of their size, but the start time of the first dispatch is delayed by almost 90 minutes. When increasing from two to four vehicles, the dispatch efficiency is significantly reduced, and the first dispatch actually starts earlier. Figure 3b shows that under baseline case (iii), increasing the fleet to two vehicles improves the balance between the cardinality of different batches, and the start time of the first dispatch increases by over two hours. Increasing to four vehicles continues to improve the overall efficiency of dispatches, but does so by decreasing the cardinality balance. Moreover, the earliest departure time continues to increase, again by almost two hours. These results suggest that SDD operational efficiency benefits more from a larger fleet under some order arrival patterns. In particular, some SDD systems may have their dispatch balance improved, and may start their delivery operations later. In fact, our results suggest that if more vehicles are added to the fleet, they may not need to be available for the complete working day, and so they could be part of a shared fleet (between next-day and SDD), further improving operational efficiency for carriers.

6. Conclusions

We studied the multi-vehicle submodular dispatching problem (MSMD) and focused on the case with identical vehicles, for which we proposed four different MILP formulations. MSMD includes several important models as special cases, such as models for SDD tactical design, machine scheduling under serial-batching and parallel-batching machines, and routing models under restricted topologies. We established the difficulty of solving the LP relaxations of our formulations, and studied the quality of their bounds. In addition, we proposed a strengthened version of a set cover formulation that can leverage any known lower bound on the optimal makespan. Moreover, we characterized interval-solvable functions, which always have an optimal solution of interval type, where batches consist of consecutive orders.

A computational study on serial-batching scheduling with family setups and release dates allowed us to assess the performance of our lower bounds and heuristics. Our methods proved to be efficient from a computational standpoint, achieving results within 10% of optimality with average running times below six minutes for up to 160 jobs, an improvement over the recent literature for similar problems. Furthermore, we used our formulations and results on interval-solvable functions to computationally study SDD tactical design problems with non-stationary order arrival rates, deriving insights on fleet expansion benefits.

Our results motivate several avenues for future research, including the use of meta-heuristics to enhance our current solution methods, and new combinatorial lower bounds to leverage the structure of other dispatch time functions. More generally, the heterogeneous vehicle case presents additional challenges, as well as the case in which batches are constrained, e.g. by cardinality.

Acknowledgments

The authors' work was partially supported by the U.S. Office of Naval Research, grant N00014-18-1-2075.

References

Banerjee D, Erera A, Stroh A, Toriello A, 2023 *Who has access to e-commerce and when? Time-varying service regions in same-day delivery*. *Transportation Research Part B: Methodological* 170:148–168.

Banerjee D, Erera A, Toriello A, 2022 *Fleet Sizing and Service Region Partitioning for Same-Day Delivery Systems*. *Transportation Science* 56:1327–1347.

Beardwood J, Halton J, Hammersley J, 1959 *The shortest path through many points*. *Mathematical Proceedings of the Cambridge Philosophical Society* 55(4):299–327.

Ben-Ameur W, Neto J, 2007 *Acceleration of cutting-plane and column generation algorithms: Applications to network design*. *Networks* 49:3–17, URL http://dx.doi.org/10.1002/net.20137.

Bogunovic I, Mitrović S, Scarlett J, Cevher V, 2017 *Robust submodular maximization: A non-uniform partitioning approach*. *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 508–516, ICML'17 (JMLR.org).

Carlsson JG, Liu S, Salari N, Yu H, 2021 *Provably good region partitioning for on-time last-mile delivery*, in review, URL https://ssrn.com/abstract=3915544.

Chang PY, Damodaran P, Melouk S, 2004 *Minimizing makespan on parallel batch processing machines*. *International Journal of Production Research* 42(19):4211–4220, URL http://dx.doi.org/10.1080/00207540410001711863.

Chekuri C, Ene A, 2011 *Approximation algorithms for submodular multiway partition*. *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, 807–816, URL http://dx.doi.org/10.1109/FOCS.2011.34.

Dell'Amico M, Iori M, Martello S, Monaci M, 2008 *Heuristic and exact algorithms for the identical parallel machine scheduling problem*. *INFORMS Journal on Computing* 20(3):333–344, URL http://dx.doi.org/10.1287/ijoc.1070.0246.

Erazo I, Toriello A, 2024 *Optimizing the trade-off between batching and waiting: Subadditive dispatching*, URL https://optimization-online.org/?p=19051, *Operations Research*, minor revision.

Forbes, 2023 *E-commerce statistics*. URL https://www.forbes.com/advisor/business/ecommerce-statistics/.

Fowler JW, Mönch L, 2022 *A survey of scheduling with parallel batch (p-batch) processing*. *European Journal of Operational Research* 298(1):1–24, URL http://dx.doi.org/https://doi.org/10.1016/j.ejor.2021.06.012.

Franceschetti A, Jabali O, Laporte G, 2017 *Continuous approximation models in freight distribution management*. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research* 25(3):413–433, URL http://dx.doi.org/10.1007/s11750-017-0456-1.

Garey M, Johnson D, 1978 *"Strong" NP-Completeness Results: Motivation, Examples, and Implications*. *Journal of the ACM* 25(3):499–508, URL http://dx.doi.org/10.1145/322077.322090.

Garey M, Johnson D, 1979 *Computers and Intractability: A Guide to the Theory of NP-Completeness* (USA: W. H. Freeman & Co.), ISBN 0716710447.

Ghalami L, Grosu D, 2019 *Scheduling parallel identical machines to minimize makespan:a parallel approximation algorithm*. *Journal of Parallel and Distributed Computing* 133:221–231, URL http://dx.doi.org/https://doi.org/10.1016/j.jpdc.2018.05.008.

Graham RL, 1969 *Bounds on multiprocessing timing anomalies*. *SIAM Journal on Applied Mathematics* 17(2):416–429, URL http://www.jstor.org/stable/2099572.

Grötschel M, Lovász L, Schrijver A, 1993 *Geometric Algorithms and Combinatorial Optimization* (Springer-Verlag, Berlin).

Habiba H, Hassam A, Sari Z, Amine CM, Souad T, 2019 *Minimizing makespan on identical parallel machines*. *2019 International Conference on Applied Automation and Industrial Diagnostics (ICAAID)*, volume 1, 1–6, URL http://dx.doi.org/10.1109/ICAAID.2019.8934993.

Herer Y, Penn M, 1995 *Characterizations of naturally submodular graphs: A polynomially solvable class of the TSP*. *Proceedings of the American Mathematical Society* 123:673–679, URL http://dx.doi.org/10.1090/S0002-9939-1995-1260169-4.

Hirayama T, Liu Y, Makino K, Shi K, Xu C, 2023 *A Polynomial Time Algorithm for Finding a Minimum 4-Partition of a Submodular Function*, 1680–1691. URL http://dx.doi.org/10.1137/1.9781611977554.ch64.

Iwata S, Fleischer L, Fujishige S, 2001 *A combinatorial strongly polynomial algorithm for minimizing submodular functions*. *Journal of the Association for Computing Machinery* 48:761–777.

Klapp MA, Erera AL, Toriello A, 2018a *The one-dimensional dynamic dispatch waves problem*. *Transportation Science* 52(2):402–415, URL http://dx.doi.org/10.1287/trsc.2016.0682.

Klapp MA, Erera AL, Toriello A, 2018b *The Dynamic Dispatch Waves Problem for same-day delivery*. *European Journal of Operational Research* 271(2):519–534, URL http://dx.doi.org/10.1016/j.ejor.2018.05.03.

Klapp MA, Erera AL, Toriello A, 2020 *Request acceptance in same-day delivery*. *Transportation Research Part E: Logistics and Transportation Review* 143:102083.

Kramer A, Iori M, Lacomme P, 2021 *Mathematical formulations for scheduling jobs on identical parallel machines with family setup times and total weighted completion time minimization*. *European Journal of Operational Research* 289(3):825–840, URL http://dx.doi.org/https://doi.org/10.1016/j.ejor.2019.07.006.

Krause A, Golovin D, 2014 *Submodular Function Maximization*. Bordeaux L, Hamadi Y, Kohli P, eds., *Tractability: Practical Approaches to Hard Problems*, 71–104 (Cambridge University Press), URL http://dx.doi.org/10.1017/CBO9781139177801.004.

Kuruvilla A, Paletta G, 2015 *Minimizing makespan on identical parallel machines*. *International Journal of Operations Research and Information Systems* 6:19–29, URL http://dx.doi.org/10.4018/ijoris.2015010102.

Muter I, 2020 *Exact algorithms to minimize makespan on single and parallel batch processing machines*. *European Journal of Operational Research* 285(2):470–483, URL http://dx.doi.org/https://doi.org/10.1016/j.ejor.2020.01.065.

Nemhauser G, Wolsey L, 1999 *Integer and Combinatorial Optimization* (John Wiley & Sons, Inc.).

Pessan C, Néron E, 2011 *Setup tasks scheduling during production resettings*. *International Journal of Production Research* 49(22):6787–6811, URL http://dx.doi.org/10.1080/00207543.2010.519922.

Schaller JE, 2014 *Minimizing total tardiness for scheduling identical parallel machines with family setups*. *Computers & Industrial Engineering* 72:274–281, URL http://dx.doi.org/https://doi.org/10.1016/j.cie.2014.04.001.

Schrijver A, 2000 *A combinatorial algorithm minimizing submodular functions in strongly polynomial time*. *Journal of Combinatorial Theory, Series B* 80(2):346–355, URL http://dx.doi.org/https://doi.org/10.1006/jctb.2000.1989.

Schrijver A, 2003 *Combinatorial Optimization: Polyhedra and Efficiency* (Berlin: Springer).

Schutten M, Van De Velde S, Zijm W, 1996 *Single-machine scheduling with release dates, due dates and family setup times*. *Management Science* 42:1165–1174, URL http://dx.doi.org/10.1287/mnsc.42.8.1165.

Statista, 2023a *Global same-day delivery market size*. URL https://www.statista.com/statistics/1255427/same-day-delivery-market-size-worldwide/.

Statista, 2023b *Retail e-commerce sales worlwide*. URL https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/.

Stroh AM, Erera AL, Toriello A, 2022 *Tactical design of same-day delivery systems*. *Management Science* 68(5):3444–3463, URL http://dx.doi.org/10.1287/mnsc.2021.4041.

Tian J, Cheng TCE, Ng C, Yuan J, 2009 *Online scheduling on unbounded parallel-batch machines to minimize the makespan*. *Inf. Process. Lett.* 109:1211–1215, URL http://dx.doi.org/10.1016/j.ipl.2009.08.008.

Vanelslander T, Deketele L, Hove D, 2013 *Commonly used e-commerce supply chains for fast moving consumer goods: comparison and suggestions for improvement*. *International Journal of Logistics* 16:243–256, URL http://dx.doi.org/10.1080/13675567.2013.813444.

Wang S, Zhou T, Lavania C, Bilmes JA, 2021 *Constrained robust submodular partitioning*. Ranzato M, Beygelzimer A, Dauphin Y, Liang P, Vaughan JW, eds., *Advances in Neural Information Processing Systems*, volume 34, 2721–2732 (Curran Associates, Inc.), URL https://proceedings.neurips.cc/paper_files/paper/2021/file/161882dd2d19c716819081aee2c08b98-Paper.pdf.

Wölck M, Meisel S, 2022 *Branch-and-Price Approaches for Real-Time Vehicle Routing with Picking, Loading, and Soft Time Windows*. *INFORMS Journal on Computing* URL http://dx.doi.org/10.1287/ijoc.2021.1151, forthcoming.

Appendix A: Proofs for Formulations (2) and (3)

A.1. Formulation 2 Solves MSMD

PROPOSITION 7. *MILP (2) solves MSMD:*

$$\min z$$

$$\text{s.t. } t_{i,k} \geq r_i \qquad \forall i \in N,\ \forall k \in M \tag{2a}$$

$$t_{i+1,k} \geq t_{i,k} + w_{i,k} \qquad \forall i \leq n-1,\ \forall k \in M \tag{2b}$$

$$z \geq t_{n,k} + w_{n,k} \qquad \forall k \in M \tag{2c}$$

$$\sum_{k=1}^{m} w_{i,k} = \sum_{S \in \mathscr{N}_i} x_S f(S) \qquad \forall i \in N \tag{2d}$$

$$w_{i,k} \leq y_{i,k} f([1,i]) \qquad \forall i \in N,\ \forall k \in M \tag{2e}$$

$$\sum_{k=1}^{m} y_{i,k} \leq 1 \qquad \forall i \in N \tag{2f}$$

$$\sum_{S \subseteq N : S \ni i} x_S = 1 \qquad \forall i \in N \tag{2g}$$

$$z \geq 0,\ t \geq 0,\ w \geq 0,\ y \in \{0,1\},\ x \in \{0,1\}.$$

Furthermore, its linear relaxation can be solved in polynomial time.

Proof: We prove the equivalence between feasible solutions for MILP (1) and (2). Assume we have a solution (x^1, t^1, z^1) for MILP (1); we construct a solution (x, t, z, y, w) for MILP (2). Set x, y and w as zero vectors, then:

- For all $S \subseteq N, k \in M$ such that $x^1_{S,k} = 1$, let $i = \max\{j : j \in S\}$ and set variables $x_S = 1$, $y_{i,k} = 1$ and $w_{i,k} = f(S)$.
- For all $i \in N, k \in M$ set $t_{i,k} = t^1_{i,k}$ and $z = z^1$

As x^1 forms a partition, then x satisfies constraint (2g). As it is a partition, for each $i \in N$ there is at most one set $S \in \mathscr{N}_i$ such that $x_S = 1$; therefore (2f) $\sum_{k=1}^{m} y_i \leq 1$ holds. From construction, equality (2d) is met, and by monotonicity inequality (2e) holds. Finally, from the definition of w and feasibility of t^1, our new solution (x, t, z, y, w) satisfies constraints (2a), (2b) and (2c). From construction, the makespan is equal to the makespan of solution (x^1, t^1, z^1) for MILP (1). Now consider a feasible solution (x, t, z, y, w) for MILP (2); we construct a solution (x^1, t^1, z^1) for MILP (1):

- For all $i \in N, k \in M$ set $t^1_{i,k} = t_{i,k}$, then set $z^1 = z$ and x as a zero-valued vector.
- Find all subsets $S \subseteq N$ such that $x_S = 1$. For each of those sets S, compute its maximum index order. Because of

non-negativity of f and (2e), there is exactly one machine k such that $y_{i,k}$ is equal to one and $w_{i,k} = f(S)$. Set $x^1_{S,k} = 1$.

By definition of w, inequalities (2a), (2b) and (2c) correspond to inequalities (1a), (1b) and (1c) in MILP (1). As x is a partition, then x^1 is as well and (x^1,t^1,z^1) is feasible and has the same makespan as solution (x,t,z,y,w) for MILP (2).

Now we focus on the linear relaxation of this formulation with non-negative (instead of binary) domains for x and y. We consider the dual of the linear relaxation of (2) and let α be the dual variable of (2a), β be the dual variable of (2b) and (2c), ε be the dual variable of (2d), ϕ be the dual variable of (2e), π be the dual variable of (2f) and finally γ be the dual variable of (2g). The dual constraints corresponding to the (relaxed) x variables are $-\varepsilon_i f(S) + \sum_{j\in S}\gamma_j \le 0, i \in N, S \in \mathcal{N}_i$, and therefore the separation problem is $\min_{S\in\mathcal{N}_i}\left\{\varepsilon_i f(S) - \sum_{j\in S}\gamma_j\right\}$. Constraint (2d) can be replaced with a greater-than-or-equal constraint without loss of optimality, and thus we may assume $\varepsilon \ge 0$. Therefore, the separation problem corresponds to submodular minimization, and the linear relaxation of (2) is solvable in polynomial time. ■

A.2. Equivalence Between the LP Relaxations of Formulations 1 and 2 for MSMD

THEOREM 1. *The linear relaxations of formulations* (1) *and* (2) *have equal optimal values. Moreover, given an extreme point feasible solution for one of the linear relaxations, we can obtain a feasible solution for the other linear relaxation in polynomial time.*

Proof: Assume we have a feasible solution (x^1,t^1,z^1) for the LP relaxation of MILP (1); we construct a feasible solution (x,t,z,w,y) for the LP relaxation of MILP (2):

- Let $x_S = \sum_{k\in M} x^1_{S,k}$ for each $S \subseteq N$; which implies $x_S \ge 0$ from feasibility of x^1, but also that $x_S \le 1$.
- Set $z = z^1$, and $t_{i,k} = t^1_{i,k}$ for all $i \in N,\ k \in M$.
- Set $w_{i,k} = \sum_{S\in\mathcal{N}_i} x^1_{S,k} f(S)$ for all $i \in N,\ k \in M$; from the feasibility of x^1 and f, then $w_{i,k} \ge 0$.
- Define $y_{i,k}$ as the minimum number such that $w_{i,k} \le y_{i,k} f(\{1,\dots,i\})$; this implies $0 \le y_{i,k}$, as $w_{i,k} \ge 0$, but also that $y_{i,k} \le 1$ as $w_{i,k} = \sum_{S\in\mathcal{N}_i} x^1_{S,k} f(S) \le 1 \times f(\{1,\dots,i\})$, where the last inequality follows from no order being dispatched more than once (over all the fractional dispatches including that order).

We show that constraints (2a)-(2g) are satisfied by (x,t,z,w,y). As (1a) holds for x^1 then (2a) holds. Moreover, $\sum_{k\in M} w_{i,k} = \sum_{k\in M}\sum_{S\in\mathcal{N}_i} x^1_{S,k} f(S) = \sum_{S\in\mathcal{N}_i} f(S)\sum_{k\in M} x^1_{S,k} = \sum_{S\in\mathcal{N}_i} f(S) x_S$, and so (2d) is satisfied. From the definitions of $w_{i,k}$ and $t_{i,k}$ plus the feasibility of x^1 and t^1, constraints (2b) and (2c) hold as well. Moreover, by construction (2e) holds and from the choice of $y_{i,k}$ we have that $\frac{\sum_{k\in M} w_{i,k}}{f(\{1,\dots,i\})} = \sum_{k\in M} y_{i,k}$, and as $\sum_{k\in M} W_{i,k} \le f(\{1,\dots,i\})$, then (2f) is satisfied. Finally, (2g) follows from the choice of x. For every feasible solution (x^1,t^1,z^1) for the LP relaxation of (1), we can thus create a feasible solution (x,t,z,w,y) for the LP relaxation of MILP (2) and with the same makespan.

Assume now that we have a feasible solution (x,t,z,w,y) for the LP relaxation of MILP (2); we create a feasible solution (x^1,t^1,z^1) for the LP relaxation of MILP (2) by setting $z^1 = z$, vector x^1 as zero, and $t^1_{i,k} = t_{i,k}$ for all $i \in N,\ k \in$

M. Then fFor values $x^1_{i,k}$ do as follows: for each $i \in N$, we initialize an auxiliary vector V_i that has the same values as $w_{i,k}$ for each k, i.e. $V_{i,k} = w_{i,k}$. Then, for each S such that $x_S > 0$, we find its maximum index $i = \max\{j : j \in S\}$ and do as follows: find the first index k such that $V_{i,k} > 0$; if $V_{i,k} > x_S f(S)$, then set $x^1_{S,k} = x_S$, substract $x_S f(S)$ from $V_{i,k}$ and set x_S to zero. On the other hand, when $V_{i,k} \leq x_S f(S)$, we set $x^1_{S,k} = \frac{V_{i,k}}{f(S)} \leq x_S$, set $V_{i,k} = 0$ and reduce x_S by $x^1_{S,k}$, continuing until $x_S = 0$. Because of feasibility of (x,t,z,w,y) (constraint (2d)) the final result of this procedure is a vector V_i that is zero-valued for all $i \in N$.

We prove that (x^1,t^1,z^1) is feasible for the LP relaxation of MILP (1). From the definition of t^1 (1a) holds, and by construction of the vector x^1 so do constraints (1b) and (1c) (auxiliary vectors V_i are zero at the end of the procedure above). Furthermore, for all $S \subseteq N$ we have $\sum_{k\in M} x^1_{S,k} = x_S$, and (1d) holds. Every feasible solution (x,t,z,w,y) for the LP relaxation of MILP (2) can thus be mapped to a feasible solution (x^1,t^1,z^1) for the LP relaxation of MILP (1),

To conclude the proof, if we are given an extreme point solution for one of the LP relaxations, then that solution has a polynomially bounded number of variables x with non-zero value, and hence the number of operations needed to go from the solution of one LP relaxation to the solution for the other is polynomially bounded too. ■

A.3. Formulation 3 solves MSMD

PROPOSITION 9. *Formulation (3) solves MSMD:*

$$\min\ z = t_{n+1}$$

$$\text{s.t. } t_i \geq r_i \qquad \forall i \in N \tag{3a}$$

$$\sum_{i\in N} y_{0,i} = m \tag{3b}$$

$$\sum_{i\in N} y_{i,n+1} = m \tag{3c}$$

$$\sum_{S\subseteq N: S\ni i} x_S = 1 \qquad \forall i \in N \tag{3d}$$

$$\sum_{S\in \mathcal{N}_i} x_S = \sum_{j=0}^{i-1} y_{j,i} \qquad \forall i \in N \tag{3e}$$

$$\sum_{j=0}^{i-1} y_{j,i} = \sum_{j=i+1}^{n+1} y_{i,j} \qquad \forall i \in N \tag{3f}$$

$$t_j \geq t_i + \sum_{S\in \mathcal{N}_i} f(S)x_S - (1-y_{i,j}) f([1,i]) \qquad \forall (i,j):\ 1 \leq i < j \leq n+1 \tag{3g}$$

$$t \geq 0,\ y_{ij} \in \{0,1\},\ x \in \{0,1\}.$$

Furthermore, the linear relaxation can be solved in polynomial time.

Proof: We prove that MILP (3) solves MSMD by showing that each feasible solution for (3) is feasible for MSMD, and that each optimal solution for MSMD is feasible for MILP (3). Consider a feasible solution (x,y,t) for MILP (3); there must be m indices $i \in N$ such that $y_{0,i} = 1$. We denote those indices as $i_1^1, \dots, i_m^1$, and will map index i_k^1 to vehicle k for $k \in M$. We construct a solution $\{v_k\}_{k=1}^m$ for MSMD:

- Assign z as the makespan for our solution in MSMD, then for each vehicle $k \in M$, we find the set of indices $i_k^1, i_k^2, \dots, i_k^\ell$ such that $y_{i_k^1, i_k^2} = \dots = y_{i_k^\ell, n+1} = 1$; the set exists because of constraint (3f).
- For each vehicle $k \in M$, initialize vector v_k as $v_k = (0, \varnothing, 0, \varnothing, \dots, 0, \varnothing)$, and set $i = 1$. If i is one of the values $i_k^1, \dots, i_k^\ell$, then because of constraint (3e) there must be a subset $S_i \in \mathscr{N}_i$ with $x_{S_i} = 1$; assign $v_k[2i] = t_i$ and $v_k[2i+1] = S_i$; otherwise assign $v_k[2i] = \max\left(r_i, v_k[2(i-1)]\right)$. Increase i by one and repeat until $i = n$.

Because x is a partition (3d), our solution for MSMD is as well. Moreover, because of construction and feasibility of vector t (constraints (3a) and (3g)), our vectors v_k also have a feasible schedule for the vehicles. As the makespan z is feasible given this schedule, we just constructed a solution for MSMD, with equal makespan. Now, let $\{v_k\}_{k=1}^m$ be an optimal solution for MSMD; we construct a solution for MILP (3) by first initializing vectors x, y, t as zero; then:

- For each $k \in M$, find all the indices $i_1, i_2, \dots, i_\ell$ such that $v_k[2i_1+1], v_k[2i_2+1], \dots, v_k[2i_\ell+1]$ are non-empty sets; we denote those sets as $S_{i_1}, S_{i_2}, \dots, S_{i_\ell}$. Set $y_{0,i_1} = y_{i_1,i_2} = \dots = y_{i_\ell,n+1} = 1$ and $x_{i_1} = x_{i_2} = \dots = x_{i_\ell} = 1$. Furthermore, for $j = i_1, \dots, i_\ell$, set $t_j = v_k[2j]$.
- Let J be the set of indices that have not been modified; i.e. $j \in J$ if t_j was not modified. For $j \in J$ set $t_j = \max\{\max_{i<j}(t_i), r_j\}$, with $t_0 = 0$. Finally, set t_{n+1} as the minimum value that meets constraints (3g).

We show that (x,t,y) is feasible: from construction and feasibility of the MSMD solution we know that constraint (3a) holds. By construction of values y, we know that constraints (3b), (3c), (3e) and (3f) hold. As $\{v_k\}_{k=1}^m$ induces a partition, then constraint (3d) is also satisfied. Finally, with respect to constraint (3g) we have two cases. Consider j in J, then for all $i < j$ we have $y_{i,j} = 0$, hence $t_j \geq t_i + \sum_{S \in \mathscr{N}_i} x_S f(S) - f([1,i])$. As $f([1,i]) \geq \sum_{S \in \mathscr{N}_i} x_S f(S)$, then our choice of t_j satisfies (3g). On the other hand, every order $j \notin J$ must be associated to a dispatch done by a vehicle k:

- Consider all indices $i \in J$, with $i < j$; if $t_i = r_i$ then constraint (3g) holds; otherwise $t_i = t_k$ for some $k < i$, and because no batch in $\mathscr{N}_i$ is dispatched, then the inequality that involves indices j and k is stronger.
- Consider all indices $i < j$ that are associated to the same vehicle k; because of feasibility of the original solution for MSMD, then constraint (3g) holds for all pairs (i,j).
- Consider all indices $i_1^h, i_2^h, \dots, i_\ell^h$ associated to vehicle $h \neq k$ and batch $S_1^h, S_2^h, \dots$, such that $i_1^h < \dots < i_\ell^h < j$. Because of optimality of the MSMD solution, there exists a largest index, say i_a^h such that $t_{i_a^h} = r_{i_a^h}$; and for every index

$b : a \leq b \leq \ell$ we have that $t_{i_b^h} + f(S_b^h) = r_{i_a^h} + \sum_{p=a}^{b} f(S_p^h) \leq r_{i_b^h} + f(\cup_{p=a}^{b} S_p^h)$, otherwise the MSMD solution is not optimal. Moreover, for all $b \leq \ell$, we have $f([1, i_b^h]) \geq f(\cup_{p=a}^{b} S_p^h)$, thus for $b : a \leq b \leq \ell$,

$$
\begin{aligned}
t_j \geq r_j \geq r_{i_b^h} &\geq r_{i_b^h} + f(\cup_{v=a}^{b} S_v^h) - f([1, i_b^h]) \\
&= r_{i_b^h} + f(\cup_{v=a}^{b} S_v^h) - (1 - y_{i_b^h, j}) f([1, i_b^h]) \\
&\geq t_{i_b^h} + f(S_b^h) - (1 - y_{i_b^h, j}) f([1, i_b^h]).
\end{aligned}
$$

Therefore, constraint (3g) holds for indices $i_a^h, \dots i_\ell^h$. Because of the choice of a, $r_{i_a^h} \geq t_{i_c^h} + f(S_c^h)$ for $c < a$, so the inequality holds for indices $i_1^h, \dots, i_{a-1}^h$. The three cases include all values i such that $i < j$, hence (3g) holds for all pairs (i, j), with $j \in N$. In the case of $j = n+1$ (sink node), then the values $y_{i,n+1} = 1$ enforce the correct computation of the makespan, and constraints with $y_{i,n+1} = 0$ are redundant. It follows that solution (x, t, z) for (3) is feasible. As we already showed that every feasible solution for (3) is feasible for MSMD, and that every optimal solution for MSMD is feasible for (3); then formulation (3) solves MSMD. With respect to the linear relaxation of (3), the separation problem is very similar to that for (1) and (2), and can again be solved as a series of submodular minimization problems. ■

Appendix B: Proofs for the Set Cover Formulation

B.1. Complexity of Separation Problem

THEOREM 2. *Suppose each order $i \in N$ is associated with a value $\tau_i > 0$, and let $f(S) = \sum_{i \in S} \tau_i$; this corresponds to scheduling on identical serial machines with release times. The separation problem (5) is NP-Hard.*

Proof: For each $k \in M$ the separation problem is $\min_{S \subseteq N} \{\alpha_k \, \mathrm{SMD}(S) - \beta - \sum_{i \in S} \gamma_i\}$. As $\alpha \geq 0$, we just need to optimize the separation problem for $k^* = \mathrm{argmin}_{k \in M} \{\alpha_k\}$. As f is modular, we can reformulate the separation problem as the following maximization problem, where $z = \mathrm{SMD}(S)$:

$$\max_{t,x,z} \; -z\alpha_{k^*} + \sum_{i \in N} \gamma_i x_i \tag{6a}$$

$$\text{s.t. } t_i \geq r_i \qquad i \in N \tag{6b}$$

$$t_{i+1} \geq t_i + x_i \tau_i \qquad i \in N \setminus n \tag{6c}$$

$$z \geq t_n + x_n \tau_n \tag{6d}$$

$$x_i \in \{0, 1\} \qquad \forall i \in N \tag{6e}$$

$$t, z \geq 0 \tag{6f}$$

Constraints (6b), (6c) and (6d) are the feasibility constraints for SMD, and binary variable x_i represents the choice of adding order i to the subset S in the separation problem (5).

To complete the proof we reduce the knapsack problem to formulation (6). Consider the following knapsack problem with integer numbers $v_i, w_i > 0$ for all $i = 1, 2 \ldots, n-1$ (representing value and weight of item i, respectively) and integer knapsack capacity $W \geq w_i$; for all $i = 1, \ldots, n-1$:

$$\max_{x} \sum_{i=1}^{n-1} v_i x_i \tag{7a}$$

$$\text{s.t.} \sum_{i=1}^{n-1} v_i x_i \leq W \tag{7b}$$

$$x_i \in \{0,1\} \qquad \forall i = 1, \ldots, n-1 \tag{7c}$$

We reduce this knapsack problem to MILP (6) by using the following transformation: we set $r_i = 0$ for $i = 1, \ldots, n-1$ and $r_n = W$. Moreover, we define $\tau_i = w_i$ for $i = 1, \ldots, n-1$ and $\tau_n = 1$. Then we set $\gamma_i = v_i$ for $i = 1, \ldots, n-1$ and $\gamma_n = 1 + \sum_{i=1}^{n-1} v_i$ and finally $\alpha_{k^*} = \sum_{i=1}^{n-1} v_i$. The corresponding instance for separation problem (6) is:

$$\max_{t,x,z} \; -z\left(\sum_{i=1}^{n-1} v_i\right) + \sum_{i=1}^{n-1} v_i x_i + \left(1 + \sum_{i=1}^{n-1} v_i\right) x_n \tag{8a}$$

$$\text{s.t. } t_n \geq W \tag{8b}$$

$$t_{i+1} \geq t_i + x_i w_i \qquad \forall i = 1, \ldots n-1 \tag{8c}$$

$$z \geq t_n + x_n \tag{8d}$$

$$x_i \in \{0,1\} \qquad \forall i \in N \tag{8e}$$

$$t, z \geq 0 \tag{8f}$$

As $r_i = 0$ for $i = 1, \ldots, n-1$, by repeatedly using constraints (8c) and starting from $t_1 = 0$, we can set $t_2 = 0 + x_1 w_1$, then $t_3 = t_2 + x_2 w_2 = x_1 w_1 + x_2 w_2$ and continue until $t_{n-1} = \sum_{i=1}^{n-2} x_i w_i$. Then (8) is equivalent to:

$$\max_{t_n,x,z} \; -z\left(\sum_{i=1}^{n-1} v_i\right) + \sum_{i=1}^{n-1} v_i x_i + \left(\sum_{i=1}^{n-1} v_i\right) x_n + x_n \tag{9a}$$

$$\text{s.t. } t_n \geq W \tag{9b}$$

$$t_n \geq \sum_{i=1}^{n-1} x_i w_i \tag{9c}$$

$$z \geq t_n + x_n \tag{9d}$$

$$x_i \in \{0,1\} \qquad \forall i \in N \tag{9e}$$

$$t_n, z \geq 0 \tag{9f}$$

We prove the equivalence between solutions for (9) and solutions for the original knapsack problem. Assume we have an optimal solution for (9), it is clear that $x_n = 1$ because adding order n increases the makespan z by just 1 (constraint (9d)) and therefore increases the total objective value by 1. It follows that the optimal solution is completely described by the orders maximizing $-z\left(\sum_{i=1}^{n-1} v_i\right) + \sum_{i=1}^{n-1} v_i x_i$. An optimal solution is such that $\sum_{i=1}^{n-1} x_i w_i \leq W$, otherwise by integrality of values w_i and W we would have $\sum_{i=1}^{n-1} x_i w_i \geq W+1$, and therefore $z \geq W+2$, implying that $-z\left(\sum_{i=1}^{n-1} v_i\right) + \sum_{i=1}^{n-1} v_i x_i \leq -(W+2)\left(\sum_{i=1}^{n-1} v_i\right) + \sum_{i=1}^{n-1} v_i x_i \leq -(W+1)\left(\sum_{i=1}^{n-1} v_i\right)$, the objective when we do not dispatch to any of the $n-1$ first orders, a contradiction with optimality. Therefore, the optimal solution of (9) is such that $z = W+1$, and maximizes $\sum_{i=1}^{n-1} v_i x_i$ subject to binary variables x_i for $i = 1,\dots,n-1$ and $\sum_{i=1}^{n-1} x_i w_i \leq W$; it is precisely a feasible optimal solution for the knapsack problem. Furthermore, (9) is a feasible problem (by setting all $x_i = 0$, $t_i = 0$ for $i = 1,\dots,n$, and $t_n = z = W$) and its objective is bounded by above (by $2(\sum_{i=1}^{n-1} v_i) + 1$), thus (9) is guaranteed to have an optimal solution. The knapsack problem is NP-Hard, and we did a transformation with a linear number of steps to get MILP (9); a special case of the separation problem (5) that considers a modular function f. ■

B.2. Complexity of Separation Problem for Modular Function with Setup Time

THEOREM 3. *Suppose each order $i \in N$ is associated with a value $\tau_i > 0$, and consider a fixed setup time $\tau_0 \geq 0$. For $f(S) = \tau_0 + \sum_{i\in S} \tau_i$, (5) can be solved in pseudo-polynomial time. If r_i, τ_i are integer for $i \in N$, and given an integer upper bound U for* SMD(N)*, the complexity of solving (5) is $O(Un^2)$; moreover, if $\tau_0 = 0$, the complexity is $O(Un)$.*

Proof: We start by remembering that the separation problem (5) is: $\min_{S\subseteq N}\{\alpha_k \,\mathrm{SMD}(S) - \beta - \sum_{i\in S}\gamma_i\}$, and we assume all values τ_i are integer. The separation problem can be solved by computing the shortest path between a source and a sink node. We describe the directed acyclic network with source node T_0 and sink node T_1 as follows:

- States (Ψ,ℓ,h); for $\Psi = 0,\dots,U$, $\ell \in N$ and $h \in \{\ell+1,\dots,n\} \cup \{0\}$. Ψ denotes the partial makespan after the end of stage ℓ, stage ℓ decides if order ℓ will be dispatched or not; and h represents the order with largest index that will be a part of the current batch; if $h = 0$, then there is no current batch.

- First we describe the arcs leaving node T_0:

  —Arcs $T_0 \to (\Psi,1,0)$ with distance 0 for $\Psi \leq \tau_1 + \tau_0 - 1$ (i.e. order 1 is not dispatched to); and with distance $-\gamma_1$ for $\Psi \geq \tau_1 + \tau_0$ (order 1 is dispatched as a singleton; this is our feasibility condition).

  —Arcs $T_0 \to (\Psi,1,h)$ with distance $-\gamma_1$ for $\Psi \geq r_h + \tau_1 + \tau_0$ (i.e. order 1 is in a batch that has as largest order h; this is the feasibility condition); for all $1 < h \leq n$.

- Now we describe the arcs leaving node $(\Psi,\ell,0)$ for $\ell = 1,\dots,n-1$ and $\Psi \leq U$:

—Arcs $(\Psi,\ell,0) \to (\Psi,\ell+1,0)$ with distance 0 (order $\ell+1$ will not be dispatched).

—Arcs $(\Psi,\ell,0) \to (\Psi+\tau_{\ell+1}+\tau_0,\ell+1,0)$ with distance $-\gamma_1$ ($\ell+1$ dispatched as a singleton); if $\Psi \geq r_{\ell+1}$.

—Arcs $(\Psi,\ell,0) \to (\Psi+\tau_{\ell+1}+\tau_0,\ell+1,h)$ with distance $-\gamma_1$ (order $\ell+1$ will be dispatched in a batch that has h as its largest index order); if $\Psi \geq r_h$ (feasibility condition), and $\ell+1 < h$.

- Arcs leaving nodes (Ψ,ℓ,h) for $\ell = 1,\ldots,n-1$, $\ell < h$ and $\Psi \leq U$:

—If $\ell+1 = h$, then need to end the dispatch; i.e. arc $(\Psi,\ell,\ell+1) \to (\Psi+\tau_{\ell+1},\ell+1,0)$ with distance $-\gamma_{\ell+1}$.

—Arc $(\Psi,\ell,h) \to (\Psi,\ell+1,h)$ with distance 0; not adding order $\ell+1$ to the current batch.

—Arc $(\Psi,\ell,h) \to (\Psi+\tau_{\ell+1},\ell+1,h)$ with distance $-\gamma_{\ell+1}$; adding order $\ell+1$ to the current batch.

- Arcs $(\Psi,n,0) \to T_1$ with distance $\Psi\alpha_k - \beta$; for all $\Psi \geq \min_{i\in N}(r_i+\tau_i)+\tau_0$; which guarantees the optimal solution includes at least one order being dispatched.

The total number of arcs departing states $(\Psi,\ell,0)$ is $O(Un^2)$; and the total number of arcs departing states (Ψ,ℓ,h) is $O(Un^2)$; hence the total number of arcs in this directed acyclic graph is $O(Un^2)$ and solving the separation problem has complexity $O(Un^2)$. When $\tau_0 = 0$ we do not need the h in the states and have just $O(Un)$ arcs. ■

B.3. Complexity of Separation Problem for Family Setups

THEOREM 4. *Suppose each order $i \in N$ is associated with a value $\tau_i > 0$; assume Q families $F_1,F_2,\ldots,F_Q$ partition order set N, and each family q has a setup time $\tau_q \geq 0$. For $f(S) = \sum_{i\in S}\tau_i + \sum_{q:F_q\cap S\neq\varnothing}\tau_q$, (5) can be solved in pseudo-polynomial time. Let U be an integer upper bound on* SMD(N)*, and define $U_q = \tau_q + \sum_{j\in F_q}\tau_j$ for $q = 1,\ldots,Q$. If vectors r,τ are integer, the complexity is $O\left(nU\left[\prod_{q=1}^{Q} U_q|F_q|\right]\right)$.*

Proof: The proof in Appendix B.2 corresponds to the case where $Q = 1$; we generalize the shortest path problem needed to solve the separation problem in (5). Assume that vectors τ, r are integer, and that values U_q are known for $q = 1,\ldots,Q$. We use the notation $F_{\{i\}}$ to denote the family that contains $i \in N$. The directed acylic graph is as follows:

- States $(\Psi,\ell,\psi_1,h_1,\psi_2,h_2,\ldots,\psi_Q,h_Q)$; for all $\Psi \leq U$, $\ell \in N$, $0 \leq \psi_q \leq U_q$ and $\ell < h_q$ with $h_q \in F_q$ or $h_q = 0$ for all $q \leq Q$. Ψ represents the partial makespan, ℓ is the decision stage (whether to dispatch to order $\ell+1$ or not). For all $q \leq Q$, if $h_q > 0$, then h_q corresponds to the largest index of family q that is included in the current batch of that family and ψ_q is the accumulated dispatching time of that batch; otherwise, if $h_q = 0$ then there is no current dispatch for family q, with $\psi_q = 0$.

- We now proceed to describe the arcs for this acyclic directed graph. Between stage ℓ and $\ell+1$, the only coordinates of the states vector that can change are Ψ and the coordinates of the family of order $\ell+1$; i.e. $F_{\{\ell+1\}}$; so our notation for states will be $(\Psi,\ell,\ldots,\psi_{F_{\{\ell+1\}}},h_{F_{\{\ell+1\}}},\ldots)$. We start with the arcs departing T_0:

—Arcs $T_0 \to (\Psi, 1, \ldots, 0, 0, \ldots)$ with distance 0; for all $\Psi < \tau_1 + \tau_{F_{\{1\}}}$ (order 1 is not dispatched).

—Arcs $T_0 \to (\Psi, 1, \ldots, 0, 0, \ldots)$ with distance $-\gamma_1$; for all $\tau_1 + \tau_{F_{\{1\}}} \leq \Psi \leq U$ (order 1 dispatched as singleton).

—Arcs $T_0 \to (\Psi, 1, \ldots, \tau_1 + \tau_{F_{\{1\}}}, h_{F_{\{1\}}}, \ldots)$ with distance $-\gamma_1$; for all $\Psi \leq U$, and $1 < h_{F_{\{1\}}}$ with $h_{F_{\{1\}}} \in F_{\{1\}}$ (i.e., order 1 is dispatched in a batch that has as maximum index $h_{F_{\{1\}}}$, feasibility conditions are enforced later).

- For each $\ell = 1, \ldots, n-1$, and $\Psi \leq U$; when $h_{F_{\{\ell+1\}}} = 0$ then $\psi_{F_{\{\ell+1\}}} = 0$ and the arcs are:

—$(\Psi, \ell, \ldots, 0, 0, \ldots) \to (\Psi, \ell+1, \ldots, 0, 0, \ldots)$ with distance 0 (order $\ell+1$ is not dispatched).

—$(\Psi, \ell, \ldots, 0, 0, \ldots) \to (\Psi + \tau_{\ell+1} + \tau_{F_{\{\ell+1\}}}, \ell+1, \ldots, 0, 0, \ldots)$ with distance $-\gamma_{\ell+1}$ (order $\ell+1$ is dispatched as a singleton), if $\Psi \geq r_{\ell+1}$ (feasibility condition).

—$(\Psi, \ell, \ldots, 0, 0, \ldots) \to (\Psi, \ell+1, \ldots, \tau_{\ell+1} + \tau_{F_{\{\ell+1\}}}, h, \ldots)$ with distance $-\gamma_{\ell+1}$ for $h > \ell+1$ and $h \in F_{\{\ell+1\}}$ (order $\ell+1$ is dispatched in a batch where the largest order is h, feasibility condition is enforced later).

- For each $\ell = 1, \ldots, n-1$, and $\Psi \leq U$; when $h_{F_{\{\ell+1\}}} = \ell+1$ then the only possible arc (that represents the dispatch being finalized) is $(\Psi, \ell, \ldots, \psi_{F_{\{\ell+1\}}}, h_{F_{\{\ell+1\}}}, \ldots) \to (\Psi + \psi_{F_{\{\ell+1\}}} + \tau_{\ell+1}, \ell+1, \ldots, 0, 0, \ldots)$ and exists only if $\Psi \geq r_{\ell+1}$ (feasibility condition for the batch with largest index $\ell+1$).

- For each $\ell = 1, \ldots, n-1$, and $\Psi \leq U$, when $h_{F_{\{\ell+1\}}} > \ell+1$ then arcs are:

—$(\Psi, \ell, \ldots, \psi_{F_{\{\ell+1\}}}, h_{F_{\{\ell+1\}}}, \ldots) \to (\Psi, \ell+1, \ldots, \psi_{F_{\{\ell+1\}}}, h_{F_{\{\ell+1\}}}, \ldots)$ with distance 0 ($\ell+1$ not dispatched).

—$(\Psi, \ell, \ldots, \psi_{F_{\{\ell+1\}}}, h_{F_{\{\ell+1\}}}, \ldots) \to (\Psi, \ell, \ldots, \psi_{F_{\{\ell+1\}}} + \tau_{\ell+1}, h_{F_{\{\ell+1\}}}, \ldots)$ with distance $-\gamma_{\ell+1}$ (order $\ell+1$ is added to the current batch of family $F_{\{\ell+1\}}$; feasibility conditions are enforced at $\ell+1 = h_{F_{\{\ell+1\}}}$).

- Finally, we have the arcs $(\Psi, n, \ldots, 0, 0, \ldots) \to T_1$ with distance $\Psi\alpha_k - \beta$; for $\Psi \geq \min_{i \in N}(r_i + \tau_i + \tau_{F_{\{i\}}})$; condition that guarantees any shortest path dispatches to at least one order.

For each $\ell < n$, if $h_{F_{\{\ell+1\}}} \neq 0$, then state $(\Psi, \ell, \ldots, \psi_{F_{\{\ell+1\}}}, h_{F_{\{\ell+1\}}}, \ldots)$ has either one or two departing arcs; so there are $O\left(U\left[\Pi_{q=1}^{Q} U_q \times |F_q|\right]\right)$ arcs departing from these states. On the other hand, if $h_{F_{\{\ell+1\}}} = 0$, then there are up to $|F_{\{\ell+1\}}| + 2$ departing arcs; however as $\psi_{F_{\{\ell+1\}}} = 0$ and $|F_{\{\ell+1\}}| < U_{F_{\{\ell+1\}}}$, then the total number of arcs departing from these states is also $O\left(U\left[\Pi_{q=1}^{Q} U_q \times |F_q|\right]\right)$. By adding all arcs over $\ell \in N$, there are $O\left(nU\left[\Pi_{q=1}^{Q} U_q \times |F_q|\right]\right)$ arcs. ■

Appendix C: Interval-Solvable Functions

THEOREM 5. *Suppose each $i \in N$ is associated with a number $\tau_i > 0$. Moreover, consider some $\tau_0 \geq 0$ and a concave non-decreasing function $g : \mathbb{R} \to \mathbb{R}$ with $g(0) = 0$. The following functions are interval-solvable: (1) $f(S) = \tau_0 + \max_{i \in S}\{\tau_i\}$ and (2) $f(S) = \tau_0 + g(|S|)$. Furthermore, $f(S) = \sum_{i \in S} \tau_i$ is not only interval-solvable, it suffices to consider singleton batches.*

Proof: We prove the claim for each of the functions separately:

(1) As shown by Erazo and Toriello (2024), we may assume without loss of optimality that the τ_i are in non-increasing order. Consider a solution for the MSMD and order all batches according to their *minimum index*; let $B_1,\dots,B_h$ be the ordered batches. We set $i=1$ and construct iteratively a new solution as follows: Let l_i and u_i be the minimum and maximum indices (respectively) of B_i; we construct new interval batch $B'_i=\{l_i,l_i+1,\dots,u_i\}$ and assign it to the same vehicle to which B_i is assigned to; then update all the batches $B_{i+1},\dots,B_h$ by removing the new orders that were just assigned to B'_i, and increase i by one. If B_i is empty, then we add B'_i as empty and increase i by one; we finish this procedure when $i=h+1$. By construction, the minimum index of B'_i is greater than or equal to the minimum index of B_i for all i where B'_i is not empty, so $f(B'_i)\le f(B_i)$ because of τ being monotone non-increasing. Furthermore, the maximum index of B'_i is smaller or equal to the maximum index of B_i for all $i=1,\dots,h$ where B'_i is not empty, so each of those respective B'_i dispatches do not start later than when batch B_i starts its dispatch in the original optimal solution. Each vehicle has its makespan reduced or maintained and the interval solution is optimal.

(2) From Proposition 4, without loss of optimality we assume that each vehicle dispatches to batches according to an increasing value of the *maximum index* of the batch. From this observation, we create an interval solution as follows: order all the batches according to their maximum index and let $B_1,\dots,B_h$ be the ordered batches, with respective cardinalities $C_1,\dots,C_h$. We can construct the interval solution by doing $B'_1=\{1,\dots,C_1\}$, and $B'_{i+1}=\{\sum_{j=1}^{i}C_j+1,\dots,\sum_{j=1}^{i+1}C_j\}$ for $i=1,\dots,h-1$, such as to get batches $B'_1,\dots,B'_h$, where by construction $|B'_i|=|B_i|$ and $\max_{j\in B'_i}\{j\}\le\max_{j\in B_i}\{j\}$. By assigning each batch B'_i to the same vehicle that batch B_i was assigned to, the new solution cannot increase its makespan; thus, the interval solution is optimal as well.

The function $f(S)=\sum_{i\in S}\tau_i$ being singleton-solvable comes from the serial-batching machine scheduling literature. ∎

Appendix D: Strength of LP relaxation Lower Bounds

D.1. Worst Case for the Bound Given by the LP Relaxations of our Formulations

PROPOSITION 13. *Let z^*_I be the optimal makespan of MSMD for an instance I with m vehicles and $n\ge m$ orders. Let $z_I^{LP(1)}$, $z_I^{LP(3)}$, $z_I^{LP(4)}$ be the optimal (fractional) makespan of the linear relaxations of (1), (3) and (4), respectively, for instance I. Even when f is modular and all release times are zero, there exists a family of instances $I_1,I_2,\dots$ such that $\lim_{h\to\infty}z^*_{I_h}/z_{I_h}^{LP(1)}=\lim_{h\to\infty}z^*_{I_h}/z_{I_h}^{LP(4)}=m$, and $\lim_{h\to\infty}z^*_{I_h}/z_{I_h}^{LP(3)}=\infty$.*

Proof: Consider any arbitrary number of vehicles m and positive integer h; we design an instance I_h with $n=m+2$ orders, $r_i=0$ for all $i\in N$; $\tau_1=1$ and $\tau_i=1/h$ for all orders $i=2,\dots,m+2$. For $h\ge n-1$, the makespan of instance $z^*_{I_h}$ is equal to 1. We prove the claim for each of the LP relaxations:

- For the LP relaxation of formulation (1), set $x_{N,k} = 1/m$ for every vehicle $k \in M$; this ensures constraint (1d) holds. As all release times are zero, we set $t_1 = t_2 = \ldots = t_n = 0$ and constraints (1a), (1b) and (1c) hold. Finally, $z_{I_h}^{LP(1)} = \big((n-1)/h+1\big)/m = (m+1)/hm + 1/m$; thus $\lim_{h\to\infty} z_{I_h}^* / z_{I_h}^{LP(1)} = m$.
- For the LP relaxation of formulation (4), set $x_{\{1\},k} = x_{\{2\},k} = \ldots = x_{\{m-1\},k} = x_{[m-1,m+1],k} = 1/m$ for every vehicle $k \in M$. This choice guarantees that constraints (4b) and (4c) hold. Each constraint (4a) becomes $z \geq 1/m + (m-2)/hm + 3/hm = 1/m + (m+1)/hm$; thus $z_{I_h}^{LP(4)} = 1/m + (m+1)/hm$; and $\lim_{h\to\infty} z_{I_h}^* / z_{I_h}^{LP(1)} = m$.
- For the LP relaxation of formulation (3) set $x_i = 1$ for all $i \in N$, then (3d) holds. Moreover, set $y_{0,1} = 1,\ y_{1,2} = 2/3,\ y_{1,3} = 1/3$ (i.e. flow constraints at order 1 hold), $y_{0,2} = 1/3,\ y_{2,3} = 2/3,\ y_{2,n+1} = 1/3$ (i.e. flow constraints at order 2 hold); $y_{3,4} = 1/3,\ y_{3,n+1} = 2/3$ (i.e. flow constraints at order 3 hold), $y_{0,4} = 2/3,\ y_{4,n+1} = 1$ (i.e. flow constraints at order 4 hold) and finally, $y_{0,i} = y_{i,n+1} = 1$ for $i \in 5,\ldots,n$. It follows that constraints (3e) and (3f) hold; and as $n = m+2$; then (3b) and (3c) hold as well. Now with respect to constraints (3g), starting with orders 1, 2, 3 and 4:

—$t_1 = 0$; then $t_2 \geq t_1 + 1 - (1-2/3)\times 1 = 2/3$; thus we set $t_2 = 2/3$

—$t_3 \geq t_1 + 1 - (1-1/3)\times 1 = 1/3$ but also $t_3 \geq t_2 + 1/h - (1-2/3)\times(1+1/h) = 2/3 + 1/h - (1/3)(1+1/h) = 1/3 + 2/(3h)$; therefore we set $t_3 = 1/3 + 2/(3h)$

—$t_4 \geq t_1 + 1 - 1 = 0$; $t_4 \geq t_2 + 1/h - (1+1/h) = -1/3$ and $t_4 \geq t_3 + 1/h - (1-1/3)(1+2/h) = 1/3 + 2/(3h) - 2/3 - 4/(3h) = -1/3 - 2/(3h)$; so we set $t_4 = 0$

For t_i with $5 \leq i \leq n$ because $f([1,i]) > 1$ and the only non-zero incoming variable y is $y_{0,i} = 1$, then $t_i = 0$. For t_{n+1}:

—$t_{n+1} \geq t_1 + 1 - 1 = 0$

—$t_{n+1} \geq t_2 + 1/h - (1-1/3)(1+2/h) = 2/3 + 1/h - 2/3 - 4/(3h) < 0$

—$t_{n+1} \geq t_3 + 1/h - (1-2/3)(1+3/h) = 1/3 + 1/h - 1/3 - 1/h = 0$

—$t_{n+1} \geq t_4 + 1/h - 0 = 1/h$.

For all $5 \leq i \leq n$ we get $t_{n+1} \geq t_i + 1/h = 1/h$; thus, vector t is feasible for constraints (3a) and (3g). We conclude that $t_{n+1} = z_{I_h}^{LP(4)} = 1/h$; and $\lim_{h\to\infty} z_{I_h}^* / z_{I_h}^{LP(1)} = \infty$. By swapping the values of τ_1 and τ_n, then $z_{I_h}^{LP(1)}$ and $z_{I_h}^{LP(4)}$ remain constant, but $z_{I_h}^{LP(3)} = z_{I_h}^*$; this suggests that our formulations have a complementary structure for the lower bounds. ■

D.2. The Linear Relaxation of (1) is Dominated

PROPOSITION 14. *Let each order $i \in N$ be associated with a value $\tau_i > 0$, and let $f(S) = \sum_{i\in S} \tau_i$. The lower bound presented in Proposition 5 is greater than or equal to the lower bound given by the linear relaxation of* (1).

Proof: We start the proof by adding a constraint in the formulation, that implies we get an upper bound on the objective of the LP relaxation. We enforce $x_{S,1} = x_{S,2} = \ldots = x_{S,m}$ for all subsets $S \subseteq N$; that makes the LP relaxation to be equivalent to solving a modified instance of a single vehicle SMD, with function $f'(S) = f(S)/m = (\sum_{i\in S}\tau_i)/m$. Erazo and Toriello (2024) proved that this problem can be solved in linear time with the recursion $t_1 = r_1 = 0$ and $t_{i+1} = \max(t_i + f'(\{i\}), r_{i+1})$ for all $i \in N\backslash\{1\}$. From the recursion, we know that there exists a maximum index $j \geq 1$ such that $t_j = r_j$, and therefore we have:

$$z = t_{n+1} = t_j + \sum_{i=j}^{n} \frac{f'(\{i\})}{m} = r_j + \frac{f(\{j,\ldots,n\})}{m} \leq r_j + \frac{f(\{j,\ldots,n\})}{\min(m, n-j+1)} \leq \max_{i\in N}\left[r_i + \frac{f(\{i,\ldots,n\})}{\min(k, n-i+1)}\right].$$

The equations follow from definition, and the last expression is precisely the lower bound from Proposition 5. ■

D.3. Strong Set Cover Formulation

THEOREM 6. *Let LB be a lower bound for the optimal makespan. Constraint (4a) of formulation (4) can be strengthened to* $z \geq \sum_{S\subseteq N} \max\{\mathrm{SMD}(S), LB\} x_{S,k}$*; furthermore, all previous complexity results on the formulation remain unchanged.*

Proof: To prove that the strong formulation continues to solve the MSMD, we assume that we have an optimal solution z, x for MILP (4). Because that solution is optimal, then each vehicle $k \in M$ will have only one variable $x_{S_k,k}$ equal to one, and constraints (4a) are just equal to $z \geq \mathrm{SMD}(S_k)x_{S_k,k}$ for all $k \in M$. Moreover, LB is a lower bound on the makespan; therefore, $z \geq LB$, which implies $z \geq \max(\mathrm{SMD}(S_k), LB)x_{S_k,k}$ for all $k \in M$, thus z, x is also optimal for the strong set cover formulation. The separation problem of the strong set cover MILP is $\min_{S\subseteq N}\left(\alpha_k \max(LB, SMD(S)) - \beta - \sum_{i\in S}\gamma_i\right)$ for all $k \in M$. The proof for Theorem 2 continues to hold for any knapsack problem with $W \geq LB$; thus, the complexity result holds. For Theorems 3 and 4, the same dynamic programs work; but for the arcs arriving into the terminal state T_1 we modify the distance to be $\max(LB, \Psi)\alpha_k - \beta$ instead of $\Psi\alpha_k - \beta$. ■

Appendix E: Experiment Details

E.1. Lower Bound for Serial-Batch Scheduling with Family Setups and Release Times

PROPOSITION 15. *Consider an instance of MSMD with m vehicles and Q families $F_1, \ldots, F_Q$ that partition order set N. Each family q has a setup time $\tau_q \geq 0$, each order $i \in N$ is associated to a positive number τ_i, and* $f(S) = \sum_{i\in S}\tau_i + \sum_{q:F_q\cap S\neq\varnothing}\tau_q$*. For all $i \in N$, define $G_i := \min(m, n-i+1)$, $L_i := \left|\{q : F_q \cap [i,n] \neq \varnothing\}\right|$ (number of families intersecting batch $[1,n]$), and $P_i := \{q : |F_q \cap [i,n]| > 1\}$. Furthermore, define V_i as the increasing vector with the values τ_q for each $q \in P_i$, each value repeated exactly $|F_q \cap [i,n]| - 1$ times. Finally, let $P_i(a)$ be the sum of the first a components of vector P_i, with $P_i(0) = 0$ for all $i \in N$. Then,*

$$\max_{i\in N}\left\{r_i + \left[f([i,n]) + P_i\big(\max\{G_i - L_i, 0\}\big)\right]/G_i\right\} \qquad \textit{is a lower bound for the MSMD instance.}$$

Proof: For all $i \in N$, at time r_i we still have to dispatch to orders $[i,n]$, so at least $f([i,n])$ units of time need to be divided into at most m vehicles if $m \geq n-i+1$ and $n-i+1$ vehicles otherwise (one order per vehicle). Thus, $r_i + f([i,n])/G_i$ is a lower bound. For any i, if $G_i \leq L_i$, then the expression within the principal maximum function is exactly that lower bound. On the other hand, if $G_i > L_i$ then the total dispatching load of the lower bound is divided into more vehicles than the total number of families intersecting with $[i,n]$. Therefore each vehicle needs to incur at least one family setup time, we can add the $G_i - L_i$ smallest setup times among families that intersect batch $[i,n]$ to $f(S)$ to get a lower bound on the total dispatch time done by the G_i vehicles. As $f(S)$ considers the setup of each family intersecting the batch, then only the families in P_i are eligible for their setup time to be selected; and if family q intersects the batch k times, then that setup can be added up to $k-1$ extra times. Thus, we add $P_i(G_i - L_i)$ to $f(S)$ to get a lower bound $r_i + \big(f([i,n]) + P_i(G_i - L_i)\big)/G_i$. By maximizing over $i \in N$, we find the desired lower bound. ■

E.2. Details on Dispatch Structure for $m \geq 2$

We have as input the number of vehicles m, the vector of arrival times r (with $n_1 > n$ orders, determined by Algorithm 1), the dispatch time function f and the original makespan ϕ^*. From Algorithm 1, m vehicles can dispatch to those n_1 orders within the desired makespan; therefore we modify MILP (1) to find the most efficient solution given those constraints. We also leverage the fact that f is interval-solvable. Considering the same variables as MILP (1):

$$\min \sum_S x_{S,k} f(S)$$

$$s.t.\ t_{i,k} \geq r_i \qquad \forall i = 1,\dots,n_1,\ \forall k = 1,\dots,m \tag{10a}$$

$$t_{i+1,k} \geq t_{i,k} + \sum_{S \in \mathcal{N}_i} x_{S,k} f_k(S) \qquad \forall i = 1,\dots,n_1-1,\ \forall k = 1,\dots,m \tag{10b}$$

$$\phi^* \geq t_{n_1,k} + \sum_{S \in \mathcal{N}_{n_1}} x_{S,k} f_k(S) \qquad \forall k = 1,\dots,m \tag{10c}$$

$$\sum_{k=1}^{m} \sum_{S: S \ni i} x_{S,k} = 1 \qquad \forall i = 1,\dots,n_1 \tag{10d}$$

$$t \geq 0, x \in \{0,1\}$$