Submodular Dispatching

Ignacio Erazo Alejandro Toriello
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology, Atlanta, Georgia 30332
ierazo@gatech.edu, atoriello@gatech.edu

October 30, 2022

Abstract

Motivated by applications in e-commerce logistics where orders or items arrive at different times and must be dispatched or processed in batches, we propose the submodular dispatching problem (SMD), a strongly NP-hard model defined by a set of orders with release times and a non-decreasing submodular dispatch time function. A single uncapacitated vehicle must dispatch orders in batches to minimize the makespan, the time at which all orders have been dispatched. We analyze FIFO solutions, discuss cases when they are optimal and provide a dynamic program to obtain such solutions. Furthermore, we give a polynomial-time 1.5-approximation algorithm based on the linear relaxation of an MILP formulation; we show that this approximation guarantee is best possible among a certain class of heuristics, and give additional results on the problem's approximability. We computationally test our methods on applications in same-day delivery and warehousing.

1 Introduction

Over the 2012-2021 period, the share of retail sales in the U.S. stemming from e-commerce grew from 8.0% to 19.1%, and year-over-year growth in online sales exceeded 12% every year; the COVID-19 pandemic accelerated this trend, as evidenced by 31.8% growth during the 2019-2020 period and 14.2% in 2020-2021 (Digital Commerce 360, 2022). Business-to-business e-commerce sales are also expected to expand by 17.5% annually from 2020 to 2027 (Caldwell, 2021). The rapid growth in e-commerce underscores the importance of last-mile delivery, the last portion of the order fulfilment process, when items ordered are delivered to the end customer. Last-mile delivery systems are increasingly complex and research in this space is critical, as the last mile may incur up to 50% of a supply chain's total logistics costs (Vanelslander et al., 2013). Several efforts have been made to study these systems from an integrated point of view (Juhász and Bányai, 2018), and also to optimize many of the system's components, such as the vehicle routing aspect (Desai et al., 2020; Tilk et al., 2021; Özarık et al., 2021). A comprehensive survey on the last mile from an operations research perspective can be found in Boysen et al. (2021).

Within the last-mile context, same-day delivery (SDD) systems are particularly difficult to design and operate, because the order arrival, picking 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 et al., 2020). SDD systems are increasingly relevant, as 51% of U.S. retailers currently offer SDD and 65% plan to offer it within the next two years (Saleh, 2022); the SDD market also just had its greatest ever year-on-year growth in 2021 (Technavio, 2021). More than half of consumers between the ages of 18 and 34 expect an SDD option in e-commerce and are willing to pay for the service (Saleh, 2022); therefore, there is a growing need to study SDD systems. Operational aspects and day-to-day decisions in SDD systems, such as how to solve related underlying vehicle routing problems (VRP) to design delivery routes, have received the most attention from the research community, e.g. Klapp et al. (2018b,a, 2020); Reyes et al. (2018); Shelbourne et al. (2017); Sun et al. (2021); Voccia et al. (2019); Wölck and Meisel (2022); Yu and Liu (2014). Recently, some studies have focused on the design of SDD systems, e.g. Banerjee et al. (2022); Carlsson et al. (2021); Stroh et al. (2022), focusing on fleet sizing, region partitioning and other tactical questions. Several of these works study the tension between order arrival times and the economies of scale obtained by batching more orders in a route, a distinctive feature of SDD that differentiates it from other last-mile delivery systems, and one of the motivations for our present work.

Furthermore, the growth of the e-commerce sector and SDD has also spawned new trends in warehousing, such as "urban warehouses" (Krug, 2020), and has played a part in an increase of around 13% in the total number of warehouses in the U.S. in the last five years (Statista, 2022). Two important processes in a warehouse are order picking, the collection of items from their storage locations to fulfill a customer order, and stocking, the replenishment of new items in storage. The former process may account for up to 55% of total warehouse operating costs (Bartholdi and Hackman, 2019) and has received significant attention from the research community (de Koster et al., 2007). In particular, the single-block, single-picker problem has been widely studied (Gademann et al., 2001; Goeke and Schneider, 2021; Ratliff and Rosenthal, 1983; Scholz et al., 2016), along with recent extensions that integrate picking and storage with other processes, such as combining order-picking and vehicle routing (Schubert et al., 2021), order-picking and packing (Zhong et al., 2021), and storage and order-picking (Jiang et al., 2021; Wang et al., 2020). In contrast, stocking (also known as put-away) has not received as much attention from academia, even though it may represent up to 15% of total warehouse operating costs (Bartholdi and Hackman, 2019). Both order picking and stocking involve deciding what orders or items to batch together into pick/stow routes, and in modern e-commerce settings this occurs in a potentially dynamic environment where orders/items have different arrival times. There is an important trade-off between waiting and having more potential batches to choose from, but potentially delaying the process by doing so; this constitutes a second motivation for our work.

We propose the submodular dispatching problem (SMD) as a generalization of these and similar problems in e-commerce logistics, intended to study the interaction between release times and economies of scale derived from batching. The problem is defined by a set of orders (or items, jobs, etc.) that must be dispatched (or picked, stored, processed) in a single vehicle (or by a single picker, server, machine). Each order has a release time, and an arbitrary non-negative, monotone, submodular function describes the time required to dispatch any subset, with the objective of minimizing the makespan, the time at which all orders have been dispatched. This model generalizes and subsumes various specific applications, including (1) tactical design models for SDD (Banerjee et al., 2022; Stroh et al., 2022), (2) routing models with release times on special network topologies where optimal traveling salesman problem (TSP) routing times are submodular, including paths (Klapp et al., 2018b) and single-block warehouses (Ratliff and Rosenthal, 1983), and (3) single-machine scheduling problems with release times and batching (Goemans et al., 2002; Schutten et al., 1996; Yuan et al., 2006). We summarize our main contributions as follows:

1. We propose the submodular dispatching problem (SMD) and demonstrate various specific applications. We show that it is NP-hard even in the case of routing on generalized star graphs. We propose a mixed-integer linear programming (MILP) formulation for SMD whose linear relaxation can be solved in polynomial time.

2. We analyze the performance of First-In First-Out (FIFO) solutions, in which orders are dispatched according to release times. The best FIFO solution can be computed efficiently via dynamic programming, and is a 2-approximation for SMD. We discuss FIFO-optimal func-

tions, for which some FIFO solution is optimal.

3. We design a heuristic based on the optimal solution of the MILP's linear relaxation, and show that this heuristic is a 1.5-approximation. The heuristic uses at most two dispatches, and we show that this approximation guarantee is best possible for such heuristics. More generally, we show that a heuristic that uses at most d dispatches cannot have an approximation guarantee better than $1 + 1/d$.

4. We perform a computational analysis for two applications: tactical design of SDD systems under heterogeneous order arrival rates, and the single-block, single-picker problem with release times. The former model can be optimized with our FIFO algorithm, while our results for the latter demonstrate the empirical effectiveness of our heuristic methods.

The remainder of the paper has the following organization. Section 2 presents a brief literature review, while Section 3 defines the SMD, formulates it as an MILP, and discusses applications and preliminary results. Section 4 discusses FIFO solutions, their approximation guarantee, FIFO-optimality, and a generalization we denote Q-FIFO-optimality. Section 5 presents our 1.5-approximation algorithm based on the linear relaxation of the MILP formulation, and also discusses a set of valid inequalities that can strengthen the bound given by the relaxation. Section 6 summarizes our computational studies, and Section 7 concludes and outlines future avenues of work. The appendix includes all proofs omitted from the main body of the paper.

2 Literature Review

2.1 Same-Day Delivery and Related Work

Different problems that arise in SDD systems have received recent attention; in particular, models that study dispatching, routing and delivery of orders in SDD are a particular case of vehicle routing problems (VRP) and have been studied under different conditions, including deterministic or stochastic order arrivals, and single- or multi-vehicle fleets, and also with varying objectives, such as minimizing makespan, minimizing total routing distance or maximizing the expected number of served orders. The common element is the presence of release times that prevent a route from including an order if its departure occurs before the order is released. These models also usually have one common deadline for all deliveries (the end of the service day) rather than order-specific deadlines more common in food delivery (Reyes et al., 2018; Shelbourne et al., 2017). SMD is a generalization of deterministic models of this kind when routing times are submodular; this is not the case in general road networks but does apply in restricted topologies, such as in Klapp et al. (2018b). Other works that study SDD routing models where routing times are not necessarily submodular include Klapp et al. (2018a, 2020); Sun et al. (2021); Voccia et al. (2019); Wölck and Meisel (2022).

Recently, tactical SDD design models have used continuous approximations to study the average behavior of an SDD system (Banerjee et al., 2022; Stroh et al., 2022). Whereas the routing time for subsets of fixed locations is not necessarily submodular, 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; see Beardwood et al. (1959) and its many extensions. This translates to submodularity when discretizing arrivals, and SMD can thus generalize SDD tactical design models while allowing arbitrary arrival rates, which previous works do not accommodate. For a recent survey on applications of continuous approximations in logistics, see Franceschetti et al. (2017); other recent applications of these techniques in the last mile include Carlsson et al. (2021); Carlsson and Song (2017); Liu et al. (2020).

2.2 Warehouse Order Picking

In warehouse logistics, order picking problems concern the scheduling and routing of pickers to retrieve items in an order or set of orders while minimizing some metric of interest, such as distance traveled, makespan, etc. There are many variants of the problem, for example considering the number of items in an order and their multiplicity ("1-SKU-1-item" versus "multiple-SKUs-multiple-items", and other combinations), the number of pickers, and also the warehouse topology that the picker(s) traverse; see de Koster et al. (2007) for a comprehensive review. One of the fundamental models is the single-block, single-picker problem, which has a single picker traversing a rectangular warehouse with multiple parallel aisles and two cross-aisles; see Figures 1a and 1b. In the most widely studied variant, the dedicated storage model, each SKU is stored in only one location, and Ratliff and Rosenthal (1983) proposed a dynamic program to solve this model with running time proportional to the number of aisles; many other methods and extensions have been studied since, e.g. Gademann et al. (2001); Goeke and Schneider (2021); Löffler et al. (2021); Qin et al. (2022); Schiffer et al. (2022); Scholz et al. (2016). All such picking problems involve solving one or many TSP's on a warehouse topology; Herer and Penn (1995) showed that a single-block warehouse's topology is *naturally submodular*, meaning TSP routing time is a submodular function of the set of visited locations. Thus, SMD generalizes single-block, single-picker problems and several extensions, including scattered storage (items may be stored in multiple locations), multiple-SKUs-multiple-items variants, the integration of packing times (Zhong et al., 2021), and others.

With respect to stocking or put-away, to the best of our knowledge there is no academic work on how to optimize the routing and batching decisions of this process, and most of the literature focuses on other concerns, such as minimizing space over storage policies (Fumi et al., 2013), storage location strategies (Yang et al., 2021; Zhang et al., 2020; Zhou et al., 2020), or integrating storage allocation with the subsequent order-picking process (Mirzaei et al., 2021). Our model is suited to study batching and routing decisions in the stocking process, particularly when items to stock have different arrival times, and as such may help to fill this gap in the literature.

2.3 Machine Scheduling

Machine scheduling problems concern the assignment of jobs to one or several machines, with typical objectives including minimizing makespan, lateness, weighted completion times, etc. Within the vast body of scheduling research, our current work is related most to scheduling models that consider release times, batching or both, starting with Hariri and Potts (1983), among the first to consider release times. Batch setup times first appeared in Monma and Potts (1989), and then Schutten et al. (1996) considered both release times and *family setup times*, a submodular generalization of batch setup times. Related work includes Ahmadi et al. (1992); Brucker et al. (1998); Dobson and Nambimadom (2001); Li et al. (2005); Yuan et al. (2006); the surveys Allahverdi (2015); Webster and Baker (1995) cover relevant literature, the former particularly focusing on setup times or costs, while the latter, older review covers batching. The results in Yuan et al. (2006) are particularly relevant, as they consider family setup times, release times, and the makespan objective, and show that this problem is strongly NP-hard, implying that SMD is strongly NP-hard as well.

3 Model Formulation and Preliminaries

The submodular dispatching problem (SMD) is characterized by a finite set $N := \{1, 2, \ldots, n\}$, where each $i \in N$ has a release time $r_i \in \mathbb{R}_+$, and by a set function $f : 2^N \to \mathbb{R}$ satisfying the following conditions:

1. Monotone: $f(S) \leq f(S')$ for all $S \subseteq S' \subseteq N$.
2. Submodular: $f(S \cup S') + f(S \cap S') \leq f(S) + f(S')$ for all $S, S' \subseteq N$.
3. Non-Negative: $f(\emptyset) = 0$, which together with monotonicity implies non-negativity.

By relabeling and translating if necessary, we may assume $0 = r_1 \leq r_2 \leq \ldots \leq r_n$. Depending on the context, e.g. delivery, warehousing, 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 is the *dispatch time* function, representing the time required for a vehicle to be dispatched from a depot to deliver the orders in batch S and return to the depot. A single delivery vehicle (or more generally, a server) is available to execute dispatches.

Informally, the goal is a partition of the order set N into batches that the vehicle can dispatch while minimizing the makespan, the end time of the last dispatch. The dispatches may not overlap in time, as they are performed by a single vehicle, and if a batch contains an order i, its dispatch cannot begin before i's release time. Formally, a solution is an ordered set of dispatches (t_j, S_j) for $j \in N$, where the batches $S_j \subseteq N$ partition N, and t_j is the dispatch time of the j-th batch. The solution is feasible if

$$t_j \geq \max_{i \in S_j}\{r_i\}, \quad j \in N$$

$$t_{j+1} \geq t_j + f(S_j), \quad j \in N \setminus n.$$

The makespan of a solution is then $t_n + f(S_n)$. This definition assumes exactly n dispatches occur; when the context is clear we may also refer to a smaller number of dispatches and omit empty batches.

We can model SMD as a mixed-integer linear program (MILP) with the following variables:

x_S: Indicates if batch $S \subseteq N$ is dispatched.

t_i: Departure time for dispatch $i \in N$.

z: Makespan.

We also use the set $N_i := \{1, \ldots, i\}$ to denote orders that may be dispatched at time r_i.

Proposition 1. *The MILP* (1) *solves the submodular dispatching problem:*

$$\min_{t,x,z \geq 0} \; z \tag{1a}$$

$$\text{s.t. } t_i \geq r_i \qquad i \in N \tag{1b}$$

$$t_{i+1} \geq t_i + \sum_{S \subseteq N_i, S \ni i} x_S f(S) \qquad i \in N \setminus n \tag{1c}$$

$$z \geq t_n + \sum_{S \subseteq N, S \ni n} x_S f(S) \tag{1d}$$

$$\sum_{S \subseteq N, S \ni i} x_S = 1 \qquad i \in N \tag{1e}$$

$$x_S \in \mathbb{Z} \qquad \emptyset \neq S \subseteq N. \tag{1f}$$

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

Proof. We start by showing the equivalence between feasible solutions for the MILP and feasible solutions for SMD. Assume we have a solution for the MILP, and construct a solution for SMD as follows. For each $j \in N$ there can be at most one subset $S_j \subseteq N_j$ with $S_j \ni j$ such that $x_{S_j} = 1$, because of constraint (1e). We define the j-th dispatch as (t_j, S_j) if such a subset exists, and as $(t_j, \emptyset)$ otherwise. This ordered set of dispatches is a feasible solution for SMD. From (1e), the batches partition N; by construction, $S_j \in N_j$ and thus $t_j \geq \max_{i \in S_j}\{r_i\}$; finally, $t_{j+1} \geq t_j + f(S_j)$ follows from (1c).

Now assume we have feasible solution for SMD, $(\hat{t}_j, S_j)$ for $j = 1, \dots, k$ and some $k \leq n$, where we index only non-empty dispatches. Define $i_j = \max_{i \in S_j}\{i\}$ as the index corresponding to the latest order in batch S_j. Consider two consecutive batches S_j, S_{j+1}; if $i_j > i_{j+1}$, define (t'_j, S_{j+1}) and (t'_{j+1}, S_j) with $t'_j = \hat{t}_j$ and $t'_{j+1} = \hat{t}_j + f(S_{j+1})$, leaving all other dispatches unchanged. Then

$$t'_j = \hat{t}_j \geq r_{i_j} \geq r_{i_{j+1}},$$

$$t'_{j+1} \geq \hat{t}_j \geq r_{i_j},$$

$$t'_{j+1} + f(S_j) = \hat{t}_j + f(S_j) + f(S_{j+1}) \leq \hat{t}_{j+1} + f(S_{j+1}).$$

Thus, the modified solution is also feasible and either leaves the makespan unchanged or possibly decreases it if $j + 1 = n$. We may therefore assume without loss of optimality that for batches $S_j, S_{j'}$, $j < j'$ implies $i_j < i_{j'}$, i.e. batches are sequenced according to their respective latest orders. Define $x_{S_j} = 1$ for $j = 1, \dots, k$, setting all other x variables to zero; by definition this satisfies (1e). For dispatch times, we let $t_{i_j} = \hat{t}_j$ for $j = 1, \dots, k$. This defines k of the dispatch time variables in (1), including t_n. For any remaining undefined variable $t_{\hat{\imath}}$, we set $t_{\hat{\imath}} = t_{i_j}$ for the smallest j with $\hat{\imath} < i_j$. Finally, we set $z = t_n + f(S_n)$; the solution's feasibility for (1) then follows from the feasibility of the original solution for SMD.

Now consider the linear relaxation of (1), where we relax the binary domain for each x_S variable to non-negativity. Let $\beta \geq 0$ be the dual variables corresponding to (1c) and (1d), and γ to (1e). The dual constraints corresponding to the (relaxed) x variables are

$$-\beta_i f(S) + \sum_{j \in S} \gamma_j \leq 0, \qquad\qquad i \in N, S \subseteq N_i \text{ and } S \ni i.$$

For each $i \in N$, the separation problem for these constraints is then

$$\min\left\{\beta_i f(S) - \sum_{j \in S} \gamma_j : S \subseteq N_i \text{ and } S \ni i\right\} = \min\left\{\beta_i f(S \cup i) - \gamma_i - \sum_{j \in S} \gamma_j : S \subseteq N_{i-1}\right\},$$

which can be solved in polynomial time because f is submodular; see e.g. Schrijver (2003). It therefore follows that the pricing problem for variables x is solvable in polynomial time, and thus the linear relaxation of (1) is also solvable in polynomial time via the ellipsoid algorithm. $\square$

3.1 Submodular Functions and Applications

We present some examples of functions f that can be used in SMD, and their respective applications.

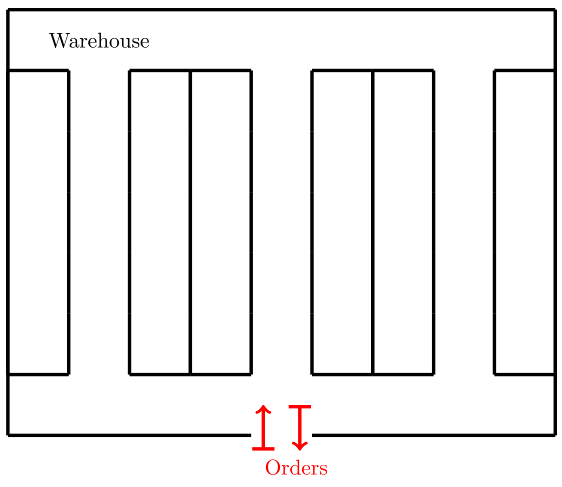
1. $f(S) = g\left(\sum_{i \in S} \tau_i\right)$, where $g : \mathbb{R} \to \mathbb{R}$ is a non-decreasing concave function with $g(0) = 0$, and $\tau_i > 0$ for all $i \in N$. In particular, when $\tau_i = 1$ for all $i \in N$, g is a function of the batch's cardinality. A specific application is $f(S) = a + b|S| + c\sqrt{|S|}$ for $S \neq \emptyset$, with $a, b, c \geq 0$; this represents a continuous approximation of expected dispatch time in SDD systems, used for tactical design in Banerjee et al. (2022); Stroh et al. (2022). Our model is similar but allows arbitrary, non-stationary (discretized) arrivals, while the analysis in these works requires additional assumptions on arrival rates.

2. Suppose the order set N is a node set in an undirected graph, the graph's edges have associated travel times, and $f(S)$ is the Steiner TSP function, which computes the minimum-duration tour through nodes S and possibly also a depot node, traversing other nodes if necessary. The Steiner TSP function is not necessarily submodular for arbitrary graphs, but is submodular for certain graph classes, such as *naturally submodular* graphs (Herer and Penn, 1995). In such graphs, any two nodes have at most two internally node-disjoint paths connecting them. This includes trees, and also graphs representing single-block warehouse layouts, as depicted in Figures 1a and 1b. Our model can thus be applied to single-block, single-picker problems with release times. Furthermore, the submodularity of f in the single-block topology is preserved for certain generalizations, such as when items are stored in multiple locations or when orders have multiple SKU's (see Appendix A for details).

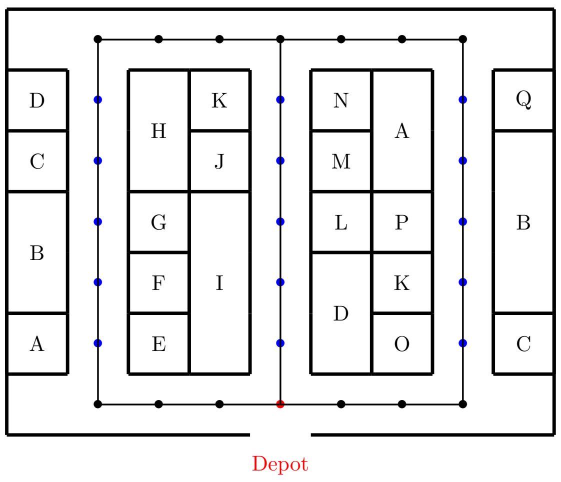
3. Let $\mathcal{F} \subseteq 2^N$ be a collection of subsets of N called families, and define for each family $F \in \mathcal{F}$ a number $\tau_F > 0$. Then

$$f(S) = \sum_{F \in \mathcal{F}, F \cap S \neq \emptyset} \tau_F$$

is submodular and non-decreasing, and can be used for instance to model single-machine scheduling problems with batching and family setups (Yuan et al., 2006), or the more general class of weighted coverage functions (Karimi et al., 2017).



(a) Single-block warehouse with three aisles.



(b) Underlying graph.

Figure 1: Single-block warehouse with three aisles, multiple SKU's (letters) and multiple storage locations.

3.2 Preliminary Results

We present some preliminary results that we use in the rest of the paper.

Proposition 2. *Let $S_1, \dots, S_k$ be any partition of the order set N into batches, and let $i_j = \max\{S_j\}$ denote each batch's latest order, where we relabel batches if necessary so that $i_1 < \cdots < i_k$. It is optimal to dispatch the batches in increasing order of i_j, and the resulting dispatch times and makespan can be computed in $O(n \log n)$ time.*

Proof. The proof of Proposition 1 establishes that if we dispatch the batches in any other order, we can make pairwise swaps to order the solution as claimed without loss of optimality. Sorting the batches by their latest order index i_j takes $O(n \log n)$ time. The resulting solution can be computed by the recursion

$$t_1 = r_{i_1}; \qquad t_j = \max\{t_{j-1} + f(S_{j-1}), r_{i_j}\}, \quad j = 2, \dots, k.$$

This takes linear time and produces the dispatches (t_j, S_j) for $j = 1, \dots, k$, with makespan $t_k + f(S_k)$. The dispatches are feasible by construction; furthermore, they form an optimal solution to

$$\min_{t,z} \; z$$

$$
\begin{aligned}
\text{s.t. } & t_j \geq r_{i_j} && j = 1, \ldots, k \\
& t_{j+1} \geq t_j + f(S_j) && j = 1, \ldots, k-1 \\
& z \geq t_k + f(S_k),
\end{aligned}
$$

showing that if the batches are dispatched in this order, the solution yields the best possible makespan. □

Proposition 2 shows that for a given partition of the order set into batches, the corresponding best possible dispatch times and makespan can be computed relatively easily. Thus, SMD essentially reduces to choosing the batches.

Corollary 3. *When f is modular, $f(S) = \sum_{i \in S} \tau_i$ with $\tau \in \mathbb{R}^N_+$, it is optimal to dispatch orders individually by release time, and this solution can be computed in linear time.*

Proof. The partition of orders into singletons is optimal, as batching is equivalent to dispatching single orders sequentially. Because the orders are already sorted, it is only necessary to compute the makespan, which can be done in linear time. □

Furthermore, when there are no release times ($r_1 = \cdots = r_n = 0$), the problem is trivial, as all orders can be dispatched in a single batch. Therefore, the complexity of SMD is driven by the interaction between release times and the batch economies of scale captured in the submodularity of the dispatch function f. Proposition 4 establishes the complexity of SMD.

Proposition 4. *SMD is strongly NP-Hard, even in the special case where f is a Steiner TSP in a generalized "star" graph, a tree where a depot node has arbitrary degree but all other nodes have degree one or two.*

This result is in addition to Yuan et al. (2006), who show that SMD is strongly NP-Hard when f represents a batch scheduling function with family setups. The complete proof can be found in Appendix B.1; the proof relies on a reduction from the 3-partition problem.

Nevertheless, SMD has a straightforward 2-approximation algorithm.

Proposition 5. *A heuristic that groups all orders into a single batch, resulting in a solution with makespan $r_n + f(N)$, is a 2-approximation for SMD. This approximation guarantee is tight.*

Proof. Let z^* be the optimal makespan; then $r_n \leq t_n \leq z^*$. Similarly, $f(N) \leq z^*$, because $f(N)$ is the optimal makespan if all release times are zero. Thus, $r_n + f(N) \leq 2z^*$. It is simple to construct an instance with $n = 2$ in which the heuristic solution has a makespan that is twice the optimum: Take a modular f with $\tau_1 = r_2 = 1$ and $r_1 = \tau_2 = 0$. $\square$

This heuristic gives the simplest possible solution, as it uses only one dispatch. In general, solutions with a small number of dispatches are appealing, as they may be easier to compute and offer operational simplicity. The next proposition gives a lower bound on the approximation ratio of such solutions.

Proposition 6. *Consider a heuristic for SMD that, for any instance, generates a solution with at most $d < n$ dispatches. The heuristic's approximation factor cannot be smaller than $1 + 1/d$.*

The family of instances proving this proposition is given in Appendix B.2. In Section 5 below, we present a heuristic based on the linear relaxation of (1) that achieves this approximation guarantee for $d = 2$ dispatches.

4 First-In First-Out (FIFO) Algorithms and Functions

A natural operating rule that arises in many applications is first-in, first-out (FIFO), the idea that orders should be dispatched in the sequence in which they are released. FIFO is appealing from an operational perspective, as it simplifies dispatching decisions, and may also offer customer service benefits. When combined with batching, FIFO implies that an order j cannot be dispatched before another order $i < j$; therefore, FIFO solutions only dispatch "interval" batches of the form $[i, j] := \{i, i+1, \ldots, j\}$ for $i \leq j$.

4.1 FIFO Algorithm

We next propose a dynamic program to compute the best FIFO solution for SMD. Introduce the notation $f_{i,j} := f([i, j])$, and compute $z_{i,j}$ for all $i, j \in N$ with $i \leq j$ using the following recursion:

$$z_{1,j} = r_j + f_{1,j} \qquad j \in N \tag{2a}$$

$$z_{i,j} = \max\left\{r_j, \min_{k<i}\{z_{k,i-1}\}\right\} + f_{i,j} \qquad 2 \leq i \leq j \leq n \tag{2b}$$

$$z^{\text{FIFO}} = \min_{i \in N}\{z_{i,n}\}. \tag{2c}$$

Intuitively, $z_{i,j}$ is the minimum makespan required to serve N_j in FIFO order when the last dispatched batch is $[i, j]$.

Proposition 7. *Let κ be the number of operations needed to compute all values $f_{i,j}$, for $1 \le i \le j \le n$. For an SMD instance, the best makespan among FIFO solutions is given by z^{FIFO}, and can be computed in $\Theta(n^2 + \kappa)$ time.*

This proof is in Appendix C.1. As the result points out, for FIFO solutions the complexity bottleneck is actually κ, the complexity of calculating the $f_{i,j}$ values, which is $\Omega(n^2)$ (one computation per value) and may be larger for some functions.

Although it may perform well in many practical settings, in the worst case the FIFO algorithm cannot improve on the simple single-batch solution; the next theorem formalizes this result.

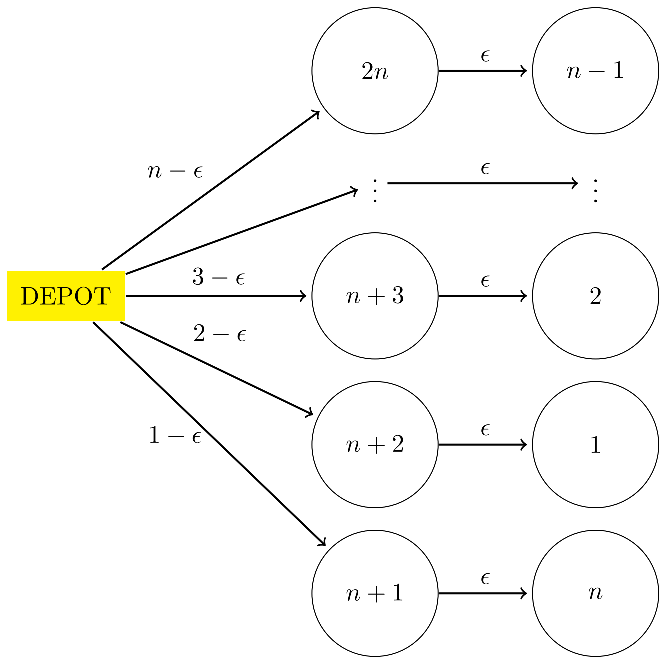


Figure 2: Family of instances in which the best FIFO makespan approaches twice the optimum. In the instance, $r_1 = r_2 = \ldots = r_n = r_{n+1} = 0$, and $r_{n+i} = \sum_{j=1}^{i-1} j = \frac{i(i-1)}{2}$ for $i = 2, 3, \ldots, n$.

Theorem 8. *The FIFO Algorithm is a 2-approximation for SMD, and this guarantee is tight.*

Proof. The single-batch solution is a FIFO solution, so the FIFO algorithm inherits the guarantee from Proposition 5. To show that the approximation ratio is tight, we construct a family of instances as depicted in Figure 2. In the figure, the nodes represent orders, the dispatch times correspond to Steiner TSP routing times in the graph and orders have the following release times: $r_1 = r_2 = \ldots = r_n = r_{n+1} = 0$ and $r_{n+i} = \sum_{j=1}^{i-1} j = \frac{i(i-1)}{2}$ for $i = 2, 3, \ldots, n$. We define path 1 as nodes $\{n,\ n+1\}$ and for $i \geq 2$ path i is the set of nodes $\{i-1,\ n+i\}$. It is clear that the optimal solution dispatches to each path only once and so the optimal solution is given by batches $\{n, n+1\}, \{1, n+2\}, \ldots, \{n-1, 2n\}$, dispatched consecutively with no idle times, and has a makespan of $\sum_{i=1}^{n} i = \frac{(n+1)n}{2} = \frac{n^2+n}{2}$.

When constructing a FIFO solution, only one dispatch is needed for path i if and only if the dispatch for both orders in the path leaves the depot at least at time r_{n+i} and includes all orders in between $i-1$ and $n+i$. Because all orders in between are included, FIFO solutions may only have one batch that has both nodes of a path being dispatched together (resulting in only one visit to that respective path). Therefore, to find an optimal FIFO solution we must select the subset of paths that will be visited only once in that solution, and clearly path 1 is always included, as nodes n and $n+1$ would always be between the nodes of other paths. For illustration purposes, consider the solution where only path 1 is served once, which dispatches the batches $[1, n+1]$ and $[n+2, 2n]$ and has a makespan that is only $(1-\epsilon)$ units of time less than the single-batch makespan. We claim that the optimal FIFO solution is described by waiting until time 1, then consecutively dispatching batches $[1, n-2]$ and $[n-1, 2n]$ (with no idle time in between), with a makespan of $1 + \sum_{i=2}^{n-1} i + \sum_{i=1}^{n} i - \epsilon(n-2) = \frac{n(n-1)}{2} + \frac{(n+1)n}{2} - \epsilon(n-2) = n^2 - \epsilon(n-2)$. Note that this solution visits paths n and 1 only once and all other paths twice.

To prove the claim, consider adding path $1 < i < n$ to the set of paths visited only once. In order to accommodate that requirement there are two options:

i) *Path n continues to be visited only once:* Under this scenario the departure time of the last dispatch (that must include all paths served only once) is still $n(n+1)/2$, and now as path i is visited only once, then the batches that are previously dispatched cannot include the node $i-1$ (the node farthest away in path i), so the vehicle will incur extra idle time to leave at the required time for the last dispatch, and as the dispatch time of the last batch will have its value increased by ϵ, the makespan increases.

ii) *Path n is now visited twice in the solution:* By letting $i \geq 2$ be the largest index of a path visited only once, then paths $2, \dots, i-1$ are the only other paths that may be visited only once. Note that the batch that includes all paths that are visited only once must leave the depot at least at time r_i, but for that to happen the vehicle will have an idle time of at least one unit even if all the orders $1, \dots, i-2$ are served previously (recalling that node $i-1$ is in the batch that includes path i). This implies that for each path $2, \dots, i-1$ we can actually reduce the makespan by ϵ if we decide to visit that path twice. It follows that in this case the best solution is the one that only has 1 and i as the subset of paths visited once, and is given by the vehicle departing at time 1 and consecutively dispatching the batches $[1, i-2], [i-1, n+i], [n+i+1, 2n]$, with a cost of $r_i + \sum_{j=1}^{n} j - \epsilon(n-2) + \sum_{j=i+1}^{n} j - \epsilon(n-i) = \frac{(n+1)n}{2} + \frac{(n+1)n}{2} - i - \epsilon(2n-2-i) = n^2 + n - i - \epsilon(2n-2-i)$, which is minimized at $i = n$.

Therefore, the makespan of the best FIFO solution is $n^2 - \epsilon(n-2)$, whereas the makespan of the optimal solution is $(n^2+n)/2$, which gives the tight 2-approximation ratio when n goes to infinity. □

4.2 FIFO-Optimal Functions and Generalizations

We say that f is FIFO-optimal if any instance of SMD with dispatch time function f has an optimal FIFO solution. Knowing that a function is FIFO-optimal allows us to optimize SMD in polynomial time, and also guarantees the optimal solution will have the simple FIFO structure. Several important function classes turn out to be FIFO-optimal, as the next proposition explains.

Proposition 9. *Let $\tau_0 \geq 0$, $\tau_i > 0$ for $i \in N$, and let $g : \mathbb{R} \to \mathbb{R}$ be a concave non-decreasing function with $g(0) = 0$. The following functions are FIFO-optimal,*

1. $f(S) = \tau_0 + \sum_{i \in S} \tau_i$,
2. $f(S) = \tau_0 + \max_{i \in S}\{\tau_i\}$,
3. $f(S) = \tau_0 + g(|S|)$,

where in each case the function is defined for $S \neq \emptyset$, letting $f(\emptyset) = 0$.

Proof. It is clear that the three functions are monotone, non-negative and submodular. We present the rest of the proof for each function:

1. Consider a non-FIFO optimal solution having batches $S_k = \{i_1, \dots, i_h\}$ and $S_{k+1} = \{j_1, \dots, j_k\}$ with a pair (a, b) such that $i_a > j_b$. Move all such orders from batch $k+1$ to k. Notice that batch k continues to start at the same time as in the previous solution, while requiring a larger dispatching time, implying that any idle time before batch $k+1$ cannot increase. Also, the sum of the dispatch times of both batches is equal, and so the makespan cannot be increased by performing this transformation. By executing this procedure sequentially for all such pairs of batches, starting with the smallest indices, we obtain a FIFO solution, and as it cannot have a greater makespan than the optimal solution, it is also optimal.

2. We prove $f(S) = \max_{i \in N}\{\tau_i\}$ is FIFO-optimal, which implies the claim, as we can add the constant τ_0 to all terms τ_i to put the function in this form.

   First, we may assume without loss of optimality that the first dispatched batch includes the order $i = \max\{\text{argmax}_{j \in N}\{\tau_i\}\}$. Suppose this order appears instead in batch $k > 1$; then simply add all orders from preceding batches to this batch. The change cannot increase k's dispatch time, and it can still be dispatched at the same time or earlier.

   Now construct a subset $N' \subseteq N$ as follows. Initialize $N' = \emptyset$, and auxiliary set $C = N$. While C is not empty, add order $i = \max\{\text{argmax}_{j \in C}\{\tau_j\}\}$ to N', then redefine $C \leftarrow [i+1, n]$; N' always includes n. By applying the preceding argument inductively, it follows that we can construct an optimal solution for N by solving SMD for N'. Specifically, suppose $N' = \{i_1, \dots, i_h\}$, with indexes following the ordering in N; then if a batch in a solution for set N' contains i_a, the corresponding batch in the solution for N contains $[i_{a-1}+1, i_a]$, or $[1, i_a]$ if $a = 1$.

   It follows that we may focus on optimizing SMD for the set N', where orders i in this set are indexed in non-decreasing order of r_i and decreasing order of τ_i. Suppose we have an optimal non-FIFO solution; then in two consecutive batches S_k and S_{k+1} we have $j \in S_k$ and $i \in S_{k+1}$ with $i < j$. As before, we can move j to S_{k+1} without changing the start time of either dispatch or increasing either dispatch time; by applying this operation as many times as necessary, we can convert the solution into a FIFO solution with an equal makespan.

3. Suppose we have a non-FIFO optimal solution, with batches S_k, S_{k+1} including orders $j \in S_k$

and $i \in S_{k+1}$, where $i < j$. By swapping these orders, placing i in the k-th batch and vice versa, we maintain the cardinality of both batches and thus their dispatch times. Furthermore, the starting times of the dispatches are still feasible, because $r_i \leq r_j \leq t_k$ and $t_{k+1} \geq t_k \geq r_j$. Applying this operation as many times as necessary, we obtain a FIFO optimal solution. $\square$

The function classes encompassed by the proposition include machine scheduling with batch setup times and either serial processing (1) or parallel processing (2), routing times along a single path (2), and continuous approximations of dispatch time (3), e.g. $f(S) = a + b|S| + c\sqrt{|S|}$.

Unfortunately, FIFO-optimality is not necessarily preserved by addition. In fact, even adding a modular function to a FIFO-optimal function may remove the FIFO-optimality property. To name one example, Figure 2 shows that the worst-case family of instances for the FIFO algorithm – in terms of approximation guarantee – has a dispatch time function f that represents routing times on a generalized "star" graph, a union of paths with a common depot. At the same time, $f(S) = \max_{i \in S}\{\tau_i\}$ represents routing time on a single path, and thus routing time on a generalized "star" graph is the sum of FIFO-optimal functions. Furthermore, Proposition 4 shows that SMD is strongly NP-Hard precisely for these dispatch time functions representing routing times on a generalized "star" graph. However, this proof also relies on a number of paths that grows with n. We next address the case in which the number of paths remains constant as n grows, and its generalization.

As a motivating example, consider an SMD instance in which dispatch times are given by routing on two disjoint paths with only a depot node in common. Even this function is not FIFO-optimal, but each path is individually "FIFO-optimal," i.e. should be dispatched in FIFO order, and an optimal solution can restrict itself to only dispatching batches containing orders from one path or the other. Intuitively, we can generalize the FIFO recursion (2) by tracking the last batch dispatched on each path; this then leads to an optimal algorithm for the two-path instance with running time $\Theta(n^4 + \kappa)$, where κ now represents the time required to compute all values $f_{i,j}$ for interval batches from each path.

Formally, suppose the order set N is partitioned into Q subsets, $S_1, \ldots, S_Q$, and the dispatch function is given by

$$f(S) = \sum_{q=1}^{Q} f_q(S \cap S_q),$$

where each f_q is a FIFO-optimal function; we call such a function Q-FIFO-optimal.

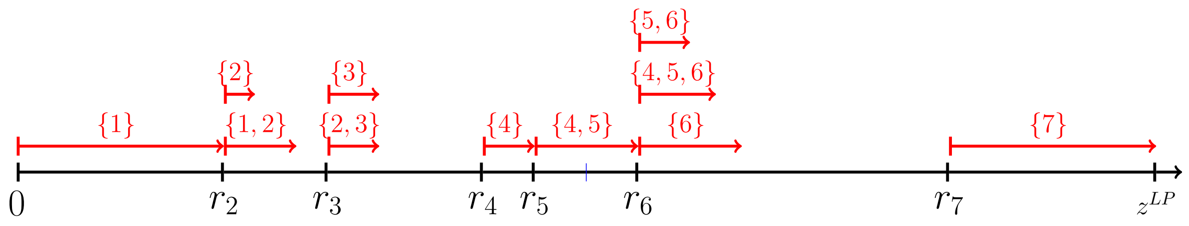
Theorem 10. *Let Q be a fixed integer. For any Q-FIFO-optimal dispatch function, SMD can be optimized in $\Theta(n^{2Q} + \kappa)$ time, where κ is the time required to compute all interval dispatch times for each set S_q that partitions N.*

The proof can be found in Appendix C.2.

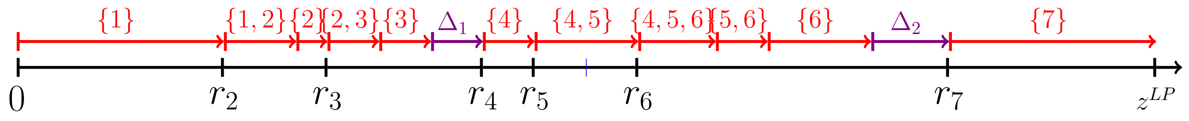
5 Linear Relaxation-Based Analysis

From Proposition 1 we know that the linear relaxation of SMD can be solved efficiently; the pricing problem corresponds to a series of submodular function minimizations. We leverage this fact to create an approximation algorithm for SMD that runs in $O(n^2)$ time, taking as input an extreme point optimal solution of the linear relaxation of (1), presented in Algorithm 1.

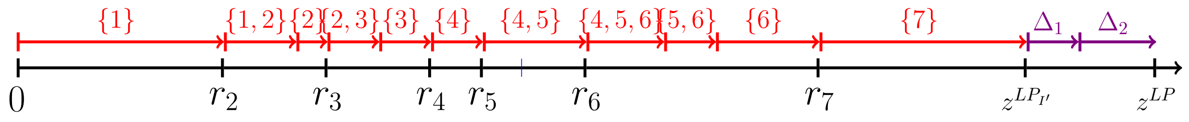
The intuition behind the heuristic is as follows: There are at most $2n$ variables x^{LP} with non-zero value, and we can group those variables according to the highest index among the orders in the respective batch. For each $i \in N$ we get a (possibly empty) set of batches B_i, where each batch has a respective non-zero x^{LP} variable and where the highest index in the batch is precisely i. A visual representation is given in Figure 3a, where the set of batches B_i is depicted just above the respective release time r_i, and the length of each arrow corresponds to the value $x_S^{LP} f(S)$ for that batch. Furthermore, for each $i \in N$ we can order the batches in B_i according to their cardinality in non-increasing fashion, leading to an ordering (and indexing) of the fractional dispatches; this sorting requires $O(n \log(n))$ time. For the same example in Figure 3, the ordering of the fractional dispatches can be seen in Figure 3b. If there exists a gap (idle time) between the fractional dispatches (Δ_1, Δ_2 in Figure 3b), this implies that the orders that were released before the gap have been completely dispatched by that time, so we can construct a modified instance I' with no idle time by reducing the release times r_i for the orders appearing after the gaps in the original instance I, as shown in Figure 3c; this step requires $O(n^2)$ time in total. The fractional dispatches from the original instance I are still optimal, and new instance I' has a fractional makespan $z^{LP_{I'}}$



(a) Optimal fractional solution for instance I



(b) Optimal fractional solution for instance I with ordered fractional batches and idle times



(c) Modified instance I' with the same fractional dispatches

Figure 3: Illustration on how to construct batch D using Algorithm 1.

equal to exactly z^{LP} minus the sum of idle times of instance I. Because I' has no idle time, we can find a fractional dispatch D^h that starts before $z^{LP_{I'}}/2$ and ends at $z^{LP_{I'}}/2$ or afterwards (batch $\{4,5\}$ in Figure 3c). By defining D as the union of all the batches fractionally served up to D^h, we can construct a simple two-dispatch heuristic that batches D and $N\backslash D$, for which computing the makespan is trivial. In the example, D is the union of batches $\{1\}, \{1,2\}, \ldots, \{4\}, \{4,5\}$.

Theorem 11. *Let (t^{LP}, x^{LP}, z^{LP}) be an extreme point optimal solution of the LP relaxation of* (1)*, and define $\Delta := z^{LP} - \sum_{S\subseteq N} x_S^{LP} f(S) \geq 0$. The two-dispatch solution obtained by applying Algorithm 1 has a tight $3/2 - \Delta/2z^{LP}$ approximation ratio and can be computed in $O(n^2)$ time.*

The complete proof can be found in Appendix D.1. When the LP relaxation has no idle times ($\Delta = 0$), the solution obtained from Algorithm 1 has an approximation guarantee of $3/2$, and when $\Delta > 0$, the slightly refined guarantee follows from the transformation into the new instance I' that has no idle times. The tightness of the approximation ratio follows from Proposition 6, as we cannot do better with at most two dispatches.

The theorem's proof uses the non-negativity and monotonicity of f, but not submodularity.

Algorithm 1 Approximation algorithm based on the optimal solution of the LP relaxation of (1)

Notation: i_S: index of batch S in the ordered fractional solution
$\hat{t}_i$: dispatch time of batch with index i
$\hat{e}_i$: ending time of batch dispatch with index i
Δ: vector of tuples (j, Δ_j), where $j \in N$ and Δ_j is the gap before index j.

Input: Optimal solution of the LP relaxation (t^{LP}, x^{LP}, z^{LP}), order set N and release time vector r

```
1:  for $j \in N$ do
2:      Initialize $\Delta$ as empty list
3:      $B_j \leftarrow \emptyset$    [$B_j$ will be an ordered set of batches]
4:  end for
5:  for all subsets $S \subseteq N$ with $x_S > 0$ do
6:      $j \leftarrow \max_{k \in S} k$
7:      Insert $S$ in $B_j$ in the earliest position such that next batches have smaller or equal cardinality
8:  end for
9:  $\ell \leftarrow 1, \hat{e}_0 \leftarrow 0$
10: for $j = 1, 2, \ldots, n$ do
11:     for $S \in B_j$ do
12:         $i_S \leftarrow \ell$
13:         $\hat{t}_\ell \leftarrow \max\{r_j, \hat{e}_{\ell-1}\}$
14:         if $\hat{t}_\ell - \hat{e}_{\ell-1} > 0$ then
15:             Append to $\Delta$ the tuple $(j, \hat{t}_\ell - \hat{e}_{\ell-1})$
16:         end if
17:         $\hat{e}_\ell \leftarrow \hat{t}_\ell + x_S f(S)$
18:         $\ell \leftarrow \ell + 1$
19:     end for
20: end for
21: for tuples $(j, \Delta_j)$ in $\Delta$, starting from the last one (highest index) do
22:     Reduce all $r_k$ for $k \geq j$ by $\Delta_j$
23:     For all $i$ such that $\hat{t}_i \geq r_j$, reduce $\hat{t}_i$ and $\hat{e}_i$ by $\Delta_j$
24: end for
25: $z \leftarrow \max_i\{\hat{e}_i\}$
26: By looking sequentially, find $\hat{\imath}$ such that $\hat{t}_{\hat{\imath}} < \frac{z}{2}$ and $\hat{e}_{\hat{\imath}} \geq \frac{z}{2}$
27: $D = \bigcup_{S: i_S \leq \hat{\imath}} \{S\}$
```

Output: D and $N \backslash D$

We instead use a weaker property, *fractional subadditivity*, which any submodular function possesses. Thus, our result implies a polynomial-time 1.5-approximation as long as f is fractionally subadditive and the linear relaxation of (1) can be solved in polynomial time. We formally define fractional subadditivity and provide more details in Appendix D.1.

The next result establishes a lower bound on the best approximation ratio we can hope for when using the optimal solution of the LP relaxation.

Proposition 12. *Let $z^*(I)$ be the optimal makespan of SMD for instance I, and $z^{LP}(I)$ be the optimal (fractional) makespan of the LP relaxation of* (1) *for instance I. There exists a family of instances $I_1, I_2, \dots$ such that $\lim_{m\to\infty} z^*(I_m)/z^{LP}(I_m) = 4/3$. Therefore, when $\zeta < 4/3$ it is not possible to create a ζ-approximation algorithm for SMD based solely on the optimal solution of the LP relaxation of* (1).

Proof. Consider the family of instances shown in Figure 4, where $r_i = i - 1$, $\tau_i = n + 1 - i$ and $f(S) = \max_{i\in S}\{\tau_i\}$. It is easy to see that the optimal solution for the LP relaxation of this

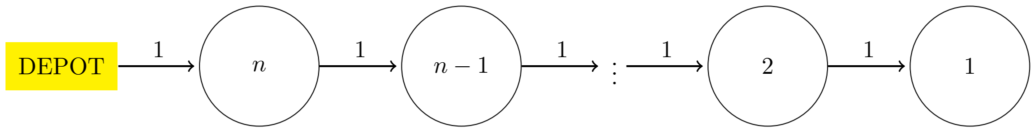


Figure 4: Family of instances where $\frac{4}{3}z^{LP}$ approaches z^*.

family of instances is given by partially dispatching batch $\{1\}$ for 1 unit of time ($x_{\{1\}} = 1/n$), then batch $\{1,2\}$ for one unit of time ($x_{\{1,2\}} = 1/n$), and so on until partially dispatching batch $\{1,2,\dots,n-1\}$ for one unit of time ($x_{\{1,\dots,n-1\}} = 1/n$). Then when the order n is released, the batches $\{1,\dots,n\}$, then $\{2,\dots,n\}$, then $\{3,\dots,n\}$ and so on, until $\{n\}$ are fractionally dispatched, each with respective value $1/n$. This optimal solution leads to a makespan of $z^{LP} = n - 1 + \frac{1}{n}\sum_{i=1}^{n} i = n - 1 + \frac{1}{n}\frac{n(n+1)}{2} = n + \frac{n-1}{2} = \frac{3}{2}n - 0.5$.

Regarding the optimal solution, we know it has a FIFO structure (Proposition 9), and that the first dispatch includes order 1. Furthermore, as $f(\{1\}) = n > r_n$, any optimal solution has at most two dispatches. In fact, all two-dispatch solutions that start the first dispatch at r_i and have as first batch $[1,i]$ are optimal solutions and have the same makespan of $z^* = r_i + \tau_1 + \tau_{i+1} = i - 1 + n + n - i = 2n - 1$. When n goes to infinity, $\frac{4}{3}z^{LP} \to z^*$. $\square$

5.1 Valid Inequalities

We now discuss improving the lower bound given by the linear relaxation of (1) using valid inequalities.

Theorem 13. *Inequalities (1b) in (1) can be strengthened to*

$$t_i \geq r_i + \sum_{j<i} \sum_{S \subseteq N_j, S \ni j} \max\{0, r_j - r_i + f(S)\} x_S, \quad i = 2, 3, \ldots, n. \tag{3}$$

The proof can be found in Appendix D.2. Consider the linear relaxation of (1) where we replace (1b) with (3). As before, let $\beta \geq 0$ be the dual variables corresponding to (1c) and (1d), and γ to (1e). Let $\alpha \geq 0$ be the dual variables corresponding to (3). The dual constraints corresponding to the (relaxed) x variables are

$$-\beta_i f(S) + \sum_{j \in S} \gamma_j - \sum_{j>i} \max(0, r_i - r_j + f(S)) \alpha_j \leq 0, \qquad i \in N, S \subseteq N_i \text{ and } S \ni i.$$

For each $i \in N$, the separation problem for these constraints is then

$$\min \Big\{ \beta_i f(S) - \sum_{j \in S} \gamma_j + \sum_{j>i} \max(0, r_i - r_j + f(S)) \alpha_j : S \subseteq N_i \text{ and } S \ni i \Big\},$$

which unfortunately is no longer submodular. Nonetheless, it can be solved with integer programming, and its efficiency depends on the structure of the function f. We explore this question computationally in the next section.

6 Computational Study

6.1 SDD System with Non-Homogeneous Arrival Rates

We consider an SDD design problem, using the same data as in Stroh et al. (2022) for a service region in northeast Atlanta, Georgia, where the dispatch time approximation is given by $f(S) = 10 + 1.5|S| + 24\sqrt{|S|}$ minutes. The SDD service day begins at 9 AM, with a constant order arrival rate of one per six minutes from 9 AM to 2 PM, for a total of 50 orders. In this experiment, our goal is to observe how time-varying order arrival rates affect the planned dispatches when the decision

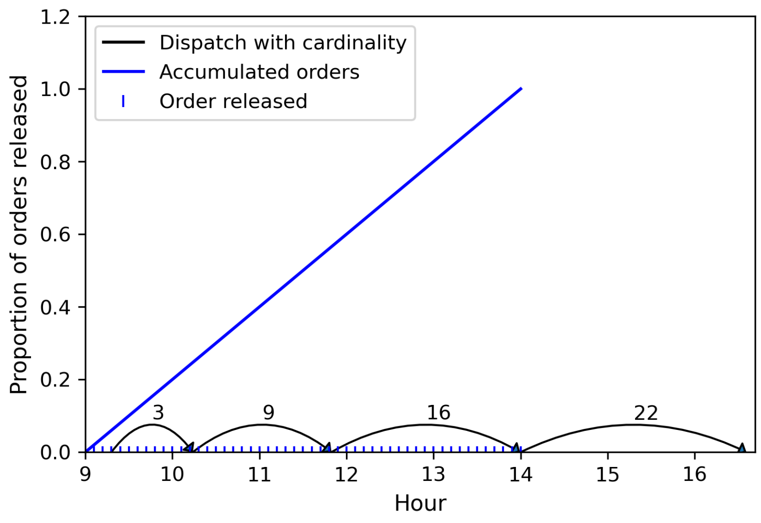
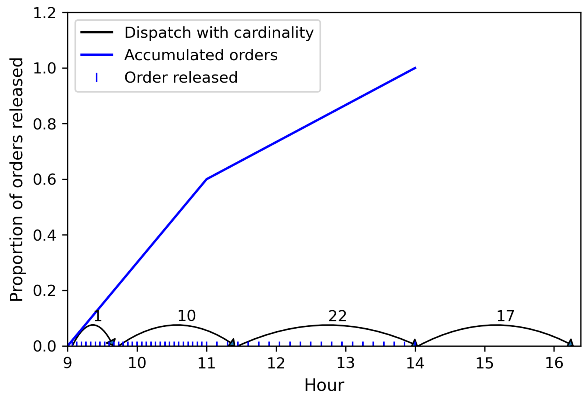


Figure 5: SDD tactical design results for uniform arrivals every six minutes.

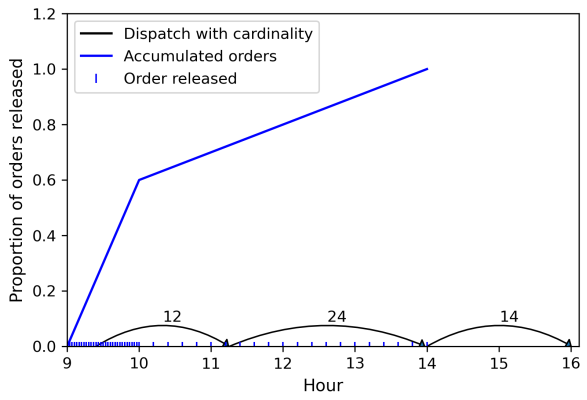
maker seeks to minimize the makespan, which here corresponds to the delivery time guarantee that can be quoted to customers. From Proposition 9, the instance is FIFO-optimal, so we use the DP formulation (2).

Fixing $r_1 = 9$ AM and $r_{50} = 2$ PM, we tested the following seven scenarios, illustrated in Figures 5 and 6a-6f: (i) constant order arrival rate, every six minutes; (ii) the first 30 orders arrive every four minutes, the next 20 orders arrive every nine minutes; (iii) the first 30 orders arrive every two minutes, the next 20 orders arrive every 12 minutes; (iv) the first 20 orders arrive every nine minutes, the last 30 orders arrive every four minutes; (v) the first 20 orders arrive every 12 minutes, the last 30 orders arrive every two minutes; (vi) the first 15 orders arrive every four minutes, the next 20 orders arrive every nine minutes, the last 15 orders arrive every four minutes; and (vii) the first 15 orders arrive every two minutes, the next 20 orders arrive every 12 minutes, the last 15 orders arrive every two minutes.

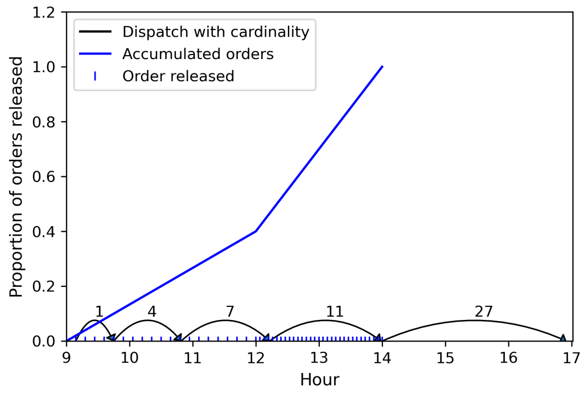
Assuming constant order arrivals from 9 AM until 2 PM (Figure 5), the solution has four planned dispatches with increasing quantities, and the last dispatch returns at roughly 4:30; this time corresponds to the delivery time guarantee that the SDD provider can quote to its customers. We next explore how varying arrival rates impact these conclusions. Figures 6a and 6b show that when more orders are expected at the start of the day, we can quote an earlier delivery guarantee and may see fewer dispatches; this suggests incentivizing SDD orders early in the day, perhaps



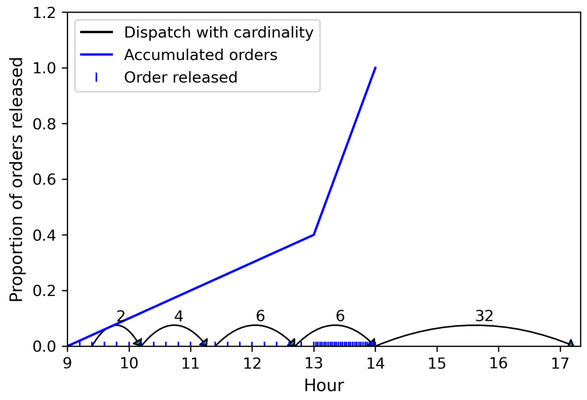
(a)



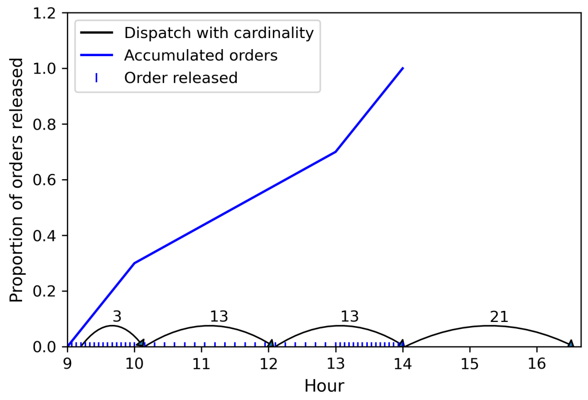
(b)



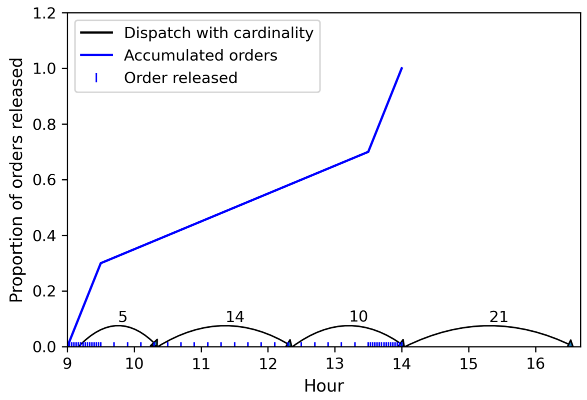
(c)



(d)



(e)



(f)

Figure 6: Dispatches for different order arrival patterns.

via discounts or promotions. Conversely, if the system expects a surge in orders at the end of the window (Figures 6c and 6d), the delivery guarantee must be delayed and the vehicle may even make additional, inefficient dispatches; this suggests more conservative planning is merited if many SDD customers make last-minute orders at the end of the service window. Finally, Figures 6e and 6f depict mixed scenarios with two order peaks at the start and end of the window. In this case, the two peaks somewhat cancel out and the final return time and number of dispatches are both similar to the case with constant arrivals. However, we do observe non-monotone order quantities in the dispatches when the peaks are pronounced (Figure 6f).

6.2 Single-Block, Single-Picker Routing with Release Times

We next study the computational performance of our proposed bounds and heuristics on a single-block, single-picker routing problem with release times. To describe the experiments, we use the notation (A, P, n), where A represents the number of aisles, P the number of positions per aisle, and n is as before the total number of orders. We assume single storage, i.e. a SKU is stored in only one location, and all orders are 1-SKU-1-item. Positions are one unit of time away from their closest neighbors, and the depot (the picker's start and end position) is the bottom of the middle aisle. We generate order release times using a Poisson process with an arrival rate of two per time unit, and order locations are sampled from the uniform distribution without replacement. We generate 20 instances for each parameter combination of (A, P, n), and use Python 3.8, Gurobi 9.1.1 and a Windows machine with 32 GB of RAM and an AMD Ryzen 7 4800H processor for our experiments.

We first focus on relatively smaller instances, with five aisles, up to 10 potential positions per aisle and up to 30 orders. For the first four instance classes with $n \leq 20$, we are able to solve the MILP (1) to optimality and compare all other methods against this benchmark. For larger instances, we use the linear relaxation of (1) strengthened with inequalities (3) as reference point. We use the following notation:

- IP: MILP formulation (1).
- CG: Column generation for the linear relaxation of (1).
- CGS: Column generation for the linear relaxation of (1) strengthened with (3).

- 2-DISP: Two-dispatch solution from Algorithm 1.
- CG IP: Heuristic solution of (1) restricted to columns generated by CG plus the two columns of 2-DISP.
- FIFO: Best FIFO solution computed via (2).

For column generation, we used the acceleration technique from Ben-Ameur and Neto (2007). For the pricing problem in CG, we adapted the algorithm from Ratliff and Rosenthal (1983) into a prize-collecting version, essentially a prize-collecting Steiner TSP in a single-block warehouse; for the pricing problem in CGS we used an MILP formulation solved with Gurobi. We describe the pricing problems and acceleration scheme in more detail in Appendix E.

(A, P, n)		IP	CG	CGS	2-DISP	CG IP	FIFO
(5,10,10)	Gap geometric mean (%)	100	97.07	99.95	102.10	100.52	110.96
	Worst gap (%)	100	93.41	99.13	108.99	103.84	118.86
	Best gap (%)	100	100.00	100.00	100.00	100.00	106.34
	Same Obj. as IP (%)	-	5	90	45	85	0
	Beats FIFO (%)	-	-	-	100	100	-
	Time per instance (s)	0.11	0.10	5.10	0.00	0.01	0.00
(5,10,15)	Gap geometric mean (%)	100	96.52	99.61	103.74	101.46	109.82
	Worst gap (%)	100	93.49	95.28	116.75	108.52	115.63
	Best gap (%)	100	100.00	100.00	100.00	100.00	106.40
	Same Obj. as IP (%)	-	5	75	5	45	0
	Beats FIFO (%)	-	-	-	95	100	-
	Time per instance (s)	4.55	0.28	32.72	0.00	0.01	0.01
(5,10,18)	Gap geometric mean (%)	100	96.09	99.85	104.51	100.61	108.88
	Worst gap (%)	100	93.73	98.91	122.42	103.54	119.12
	Best gap (%)	100	99.70	100.00	100.44	100.00	101.83
	Same Obj. as IP (%)	-	0	65	0	60	0
	Beats FIFO (%)	-	-	-	90	100	-
	Time per instance (s)	42.76	0.45	62.84	0.00	0.01	0.02
(5,10,20)	Gap geometric mean (%)	100	96.47	99.61	108.58	100.45	108.67
	Worst gap (%)	100	93.41	97.08	134.00	102.23	116.46
	Best gap (%)	100	99.70	100.00	100.00	100.00	101.34
	Same Obj. as IP (%)	-	0	60	5	60	0
	Beats FIFO (%)	-	-	-	75	100	-
	Time per instance (s)	189.38	0.61	93.14	0.00	0.01	0.02
(5,10,30)	Gap geometric mean (%)	-	95.78	100	116.84	103.33	109.83
	Worst gap (%)	-	92.12	100	138.30	107.48	118.74
	Best gap (%)	-	98.56	100	102.00	100.00	104.20
	Same Obj. as CGS (%)	-	0	-	0	15	0
	Beats FIFO (%)	-	-	-	30	100	-
	Time per instance (s)	-	1.74	663.23	0.00	0.02	0.05

Table 1: Results for smaller single-block, single-picker instances.

Table 1 summarizes our experiment results. We observe that the bound obtained by the linear relaxation of (1) is quite strong, with an average gap of over 97%. Furthermore, the strengthened inequalities (3) allow us to improve the gaps significantly so that the geometric mean is around 99.6%, albeit at the expense of a more computationally expensive pricing problem. On the primal side, the best FIFO solution has a fairly stable performance, with gaps of roughly 9% to 10%; the performance of Algorithm 1 varies more, with average gaps that grow from 2% to over 16% as n grows. Using Algorithm 1's solution and the linear relaxation as the basis for an IP-based heuristic produces the best solutions, with geometric gaps below 3.5%, while still only requiring small additional computing times.

Next, we evaluate the performance of the heuristic methods 2-DISP, CG IP and FIFO on significantly larger instances, comparing the gap against the lower bound obtained by solving CG. We use $A \in \{5, 15, 25\}$, $P = 45$ and values of n up to 300, similar to the parameters recently used by Goeke and Schneider (2021) in their experiments for the single-block, single-picker problem.

Table 2 summarizes the results. If the number of aisles A remains constant but the number of orders n increases, the gap between the LP relaxation and the three heuristics increases; the magnitude of the increase is smaller for the FIFO Algorithm, and also is smaller (for all three heuristics) when the number of aisles is bigger. Furthermore, when the order density per aisle n/A is kept constant, the gaps improve when the number of aisles is bigger. From both observations we conclude that the methods are most effective for warehouses with a larger number of aisles and a relatively low number of orders for the picker.

When comparing the heuristics, the CG IP heuristic almost always beats the FIFO algorithm, and even 2-DISP outperforms FIFO under some conditions; CG IP has average gaps of around 5% or less in all cases except (5, 45, 60). From a computational standpoint, for instances beyond $n = 300$ and $A = 25$ the average time to compute CG (and thus to obtain a heuristic solution via CG IP) may exceed an hour, and thus it may be more practical to compute the FIFO solution, since it has average gaps of 10% or less in all our experiments and its computing time scales well as n grows.

The previous results indicate that solutions with relatively few dispatches perform well in our experiments; in particular, even 2-DISP performs quite well under many parameter combinations despite generating at most two dispatches. In our final set of experiments, we examine whether a smaller arrival rate may impact this behavior. Specifically, we modify the $(25, 45, 150)$ instances

(A, P, n)		CG	2-DISP	CG IP	FIFO
(5, 45, 15)	Gap geometric mean (%)	0	3.94	1.93	5.39
	Worst (Best) gap (%)	0	19.24 (0.93)	5.73 (0.19)	8.82 (2.78)
	Beats FIFO (%)	-	90	100	-
	Time per instance (s)	0.45	0.00	0.01	0.01
(15, 45, 45)	Gap geometric mean (%)	0	3.08	2.06	3.70
	Worst (Best) gap (%)	0	4.88 (1.97)	4.02 (0.27)	5.26 (2.53)
	Beats FIFO (%)	-	95	100	-
	Time per instance (s)	14.36	0.00	0.13	0.25
(25, 45, 75)	Gap geometric mean (%)	0	3.13	1.79	3.38
	Worst (Best) gap (%)	0	3.82 (2.48)	3.48 (0.53)	4.25 (1.96)
	Beats FIFO (%)	-	90	95	-
	Time per instance (s)	82.79	0.00	0.60	1.13
(5, 45, 30)	Gap geometric mean (%)	0	11.47	3.38	5.42
	Worst (Best) gap (%)	0	45.8 (1.14)	8.45 (0.23)	9.55 (2.12)
	Beats FIFO (%)	-	45	95	-
	Time per instance (s)	2.96	0.00	0.03	0.06
(15, 45, 90)	Gap geometric mean (%)	0	4.48	2.64	4.70
	Worst (Best) gap (%)	0	7.7 (2.65)	6.21 (0.77)	6.78 (3.21)
	Beats FIFO (%)	-	90	100	-
	Time per instance (s)	109.97	0.00	0.59	1.40
(25, 45, 150)	Gap geometric mean (%)	0	4.78	2.37	4.31
	Worst (Best) gap (%)	0	8.98 (3.46)	3.92 (0.76)	6.07 (3.45)
	Beats FIFO (%)	-	75	100	-
	Time per instance (s)	667.33	0.01	3.35	6.35
(5, 45, 45)	Gap geometric mean (%)	0	15.62	5.66	7.70
	Worst (Best) gap (%)	0	44.09 (3.37)	10.95 (2.11)	12.35 (3.72)
	Beats FIFO (%)	-	30	100	-
	Time per instance (s)	7.74	0.00	0.05	0.16
(15, 45, 135)	Gap geometric mean (%)	0	4.86	3.71	4.80
	Worst (Best) gap (%)	0	7.07 (3.40)	6.16 (1.72)	6.80 (3.68)
	Beats FIFO (%)	-	85	100	-
	Time per instance (s)	263.90	0.00	1.11	3.87
(25, 45, 225)	Gap geometric mean (%)	0	4.90	2.89	4.74
	Worst (Best) gap (%)	0	8.01 (3.61)	4.41 (1.37)	6.53 (3.61)
	Beats FIFO (%)	-	90	100	-
	Time per instance (s)	1671.38	0.01	6.63	17.47
(5, 45, 60)	Gap geometric mean (%)	0	26.42	9.21	10.61
	Worst (Best) gap (%)	0	47.32 (11.03)	12.60 (5.01)	15.73 (6.35)
	Beats FIFO (%)	-	10	90	-
	Time per instance (s)	15.08	0.00	0.09	0.32
(15, 45, 180)	Gap geometric mean (%)	0	5.88	3.84	5.95
	Worst (Best) gap (%)	0	8.35 (3.27)	6.20 (1.74)	8.35 (3.56)
	Beats FIFO (%)	-	95	100	-
	Time per instance (s)	566.59	0.01	2.21	8.04
(25, 45, 300)	Gap geometric mean (%)	0	7.88	3.04	5.45
	Worst (Best) gap (%)	0	23.92 (4.21)	4.44 (1.67)	7.05 (4.21)
	Beats FIFO (%)	-	50	100	-
	Time per instance (s)	3325.64	0.01	11.49	36.58

Table 2: Results for larger single-block, single-picker instances.

by reducing the arrival rate from 2 per time unit to $1, 1/2$ and $1/4$, keeping all other parameters constant, which translates to multiplying the original arrival times by $2, 4$ and 8. These modified instances can equivalently be interpreted as having sparser arrivals over time or a slower picker. Table 3 summarizes the results and includes the average number of dispatches generated by each heuristic; the results demonstrate the adaptability of the CG IP solution, which maintains its good performance and low gaps (under 5% on average in all cases), whereas both 2-DISP and FIFO see gaps increase significantly as the arrival rate drops. Both FIFO and especially CG IP use more dispatches as the rate drops; the smaller increase in the FIFO solution may be explained because it becomes more desirable to violate FIFO under a lower arrival density in order to create more efficient batches.

Arrival Rate		CG	2-DISP	CG IP	FIFO
2	Gap geometric mean (%)	0	4.78	2.37	4.31
	≤ 2 Dispatches (%)	-	100	35	100
	Average dispatch count	-	1.2	3.1	1.75
	Beats FIFO (%)	-	75	100	-
1	Gap geometric mean (%)	0	8.47	3.27	7.83
	≤ 2 Dispatches (%)	-	100	10	100
	Average dispatch count	-	1.15	4.45	2
	Beats FIFO (%)	-	55	95	-
1/2	Gap geometric mean (%)	0	15.63	3.89	14.12
	≤ 2 Dispatches (%)	-	100	0	80
	Average dispatch count	-	1.25	6.15	2.2
	Beats FIFO (%)	-	25	100	-
1/4	Gap geometric mean (%)	0	26.38	4.19	22.41
	≤ 2 Dispatches (%)	-	100	0	20
	Average dispatch count	-	1.5	9.05	3.05
	Beats FIFO (%)	-	0	100	-

Table 3: Impact of arrival rate on heuristic performance for $(25, 45, 150)$ instances.

7 Conclusions

We introduced the submodular dispatching problem (SMD), studied its complexity and devised different heuristic methods to obtain efficient solutions. We first analyzed FIFO solutions, proposing a dynamic program to compute the best FIFO solution, which is a 2-approximation for SMD in general and is also optimal for FIFO-optimal dispatch functions. We used this method to computationally study same-day delivery tactical design problems with non-stationary order arrival rates, which could not be accommodated by previous work. We also used the linear relaxation of MILP formulation (1) to provide a polynomial-time 1.5-approximation algorithm for SMD. The solution produced by this algorithm has at most two dispatches, and the 1.5 approximation guarantee is best possible for such heuristics. A second computational study on the single-block, single-picker problem with release times shows that the linear relaxation bound and our various heuristic solutions have reasonable gaps, and that they can be obtained in reasonable computational time even for large instances with up to 300 orders.

Our results motivate several avenues for future research and SMD can be naturally generalized in several ways. We are currently studying the multi-vehicle version of SMD, which has many of the same applications as (single-vehicle) SMD. Even for modular dispatch times, the two-vehicle version of SMD is already NP-hard without release dates, as it generalizes the partition problem, and therefore presents significant additional challenges. Another interesting variant is a capacitated SMD, where batches are limited by cardinality, or more generally by a knapsack constraint. This variant is also already NP-hard even in very simple cases (Poon and Zhang, 2004). Other generalizations could include order deadlines or different objectives. In general, the study of integrated logistics processes in the presence of varying order arrivals and economies of scale in dispatching or processing have significant application potential in e-commerce, and pose many interesting challenges for the research community.

Acknowledgments

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

References

J. H. Ahmadi, R. H. Ahmadi, S. Dasu, and C. S. Tang. Batching and Scheduling Jobs on Batch and Discrete Processors. *Operations Research*, 40:750–763, 1992.

A. Allahverdi. The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246(2):345–378, 2015. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2015.04.004. URL https://www.sciencedirect.com/science/article/pii/S0377221715002763.

D. Banerjee, A. Erera, and A. Toriello. Fleet Sizing and Service Region Partitioning for Same-Day Delivery Systems. *Transportation Science*, 2022. Forthcoming.

J. Bartholdi and S. Hackman. Warehouse and Distribution Science, Release 0.98.1, 2019. URL https://www.warehouse-science.com/book/editions/wh-sci-0.98.1.pdf.

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

W. Ben-Ameur and J. Neto. Acceleration of cutting-plane and column generation algorithms: Applications to network design. *Networks*, 49:3–17, 01 2007. doi: 10.1002/net.20137.

O. Bondareva. Some applications of linear programming methods to the theory of cooperative games (in Russian). *Problemy Kybernetiki*, 10:119–139, 1963.

N. Boysen, S. Fedtke, and S. Schwerdfeger. Last-mile delivery concepts: a survey from an operational research perspective. *OR Spectrum*, 43:1–58, 2021. doi: https://doi.org/10.1007/s00291-021-00633-0.

P. Brucker, A. Gladky, H. Hoogeveen, M. Y. Kovalyov, C. N. Potts, T. Tautenhahn, and S. L. van de Velde. Scheduling a batching machine. *Journal of Scheduling*, 1(1):31–54, 1998.

A. Caldwell. 67 e-commerce stats and facts to know in 2021, 2021. URL https://www.netsuite.com/portal/resource/articles/ecommerce/ecommerce-statistics.shtml.

J. G. Carlsson and S. Song. Coordinated logistics with a truck and a drone. *Management Science*, 64(9):1–31, 2017.

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

R. de Koster, T. Le-Duc, and K. J. Roodbergen. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2):481–501, 2007. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2006.07.009. URL https://www.sciencedirect.com/science/article/pii/S0377221706006473.

H. Desai, B. Remer, B. Chalaki, L. Zhao, A. Malikopoulos, and J. Rios-Torres. Vehicle routing for the last-mile logistics problem, 2020. In review, URL https://arxiv.org/abs/1904.05464.

Digital Commerce 360. E-commerce growth, 2022. URL https://www.digitalcommerce360.com/article/us-ecommerce-sales/.

G. Dobson and R. S. Nambimadom. The Batch Loading and Scheduling Problem. *Operations Research*, 49:52–65, 2001.

J. Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Structures and their Applications, Proceedings Calgary International Conference*, pages 69–87, 1970.

A. Franceschetti, O. Jabali, and G. Laporte. Continuous approximation models in freight distribution management. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 25(3):413–433, October 2017. doi: 10.1007/s11750-017-0456-1. URL https://ideas.repec.org/a/spr/topjnl/v25y2017i3d10.1007_s11750-017-0456-1.html.

A. Fumi, L. Scarabotti, and M. M. Schiraldi. Minimizing warehouse space with a dedicated storage policy. *International Journal of Engineering Business Management*, 5:21, 2013. doi: 10.5772/56756. URL https://doi.org/10.5772/56756.

A. Gademann, J. van den Berg, and H. van der Hoff. An order batching algorithm for wave picking in a parallel-aisle warehouse. *IIE Transactions*, 33(5):385–398, 2001. doi: 10.1080/07408170108936837. URL https://doi.org/10.1080/07408170108936837.

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

D. Goeke and M. Schneider. Modeling single picker routing problems in classical and modern warehouses. *INFORMS Journal on Computing*, 33:436–451, 2021.

M. X. Goemans, M. Queyranne, A. S. Schulz, M. Skutella, and Y. Wang. Single machine scheduling with release dates. *SIAM J. Discret. Math.*, 15(2):165–192, Feb. 2002. ISSN 0895-4801. doi: 10.1137/S089548019936223X. URL https://doi.org/10.1137/S089548019936223X.

A. Hariri and C. Potts. An algorithm for single machine sequencing with release dates to minimize total weighted completion time. *Discrete Applied Mathematics*, 5(1):99 – 109, 1983. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(83)90019-7. URL http://www.sciencedirect.com/science/article/pii/0166218X83900197.

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

W. Jiang, J. Liu, Y. Dong, and L. Wang. Assignment of duplicate storage locations in distribution centres to minimise walking distance in order picking. *International Journal of Production Research*, 59(15):4457–4471, 2021. doi: 10.1080/00207543.2020.1766714. URL https://doi.org/10.1080/00207543.2020.1766714.

J. Juhász and T. Bányai. Last mile logistics: an integrated view. *IOP Conference Series: Materials Science and Engineering*, 448:012026, 11 2018. doi: 10.1088/1757-899X/448/1/012026.

M. R. Karimi, M. Lucic, H. Hassani, and A. Krause. Stochastic submodular maximization: The case of coverage functions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6856–6866, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

M. A. Klapp, A. L. Erera, and A. Toriello. The Dynamic Dispatch Waves Problem for same-day delivery. *European Journal of Operational Research*, 271(2):519–534, 2018a. doi: 10.1016/j.ejor.2018.05.03. URL https://ideas.repec.org/a/eee/ejores/v271y2018i2p519-534.html.

M. A. Klapp, A. L. Erera, and A. Toriello. The one-dimensional dynamic dispatch waves problem. *Transportation Science*, 52(2):402–415, Mar. 2018b. ISSN 1526-5447. doi: 10.1287/trsc.2016.0682. URL https://doi.org/10.1287/trsc.2016.0682.

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

K. Krug. 7 last mile delivery trends you won't want to miss, 2020. URL https://legacyscs.com/last-mile-delivery/.

S. Li, G. Li, X. Wang, and Q. Liu. Minimizing makespan on a single batching machine with release times and non-identical job sizes. *Operations Research Letters*, 33:157–164, 03 2005. doi: 10.1016/j.orl.2004.04.009.

S. Liu, L. He, and Z.-J. M. Shen. On-Time Last-Mile Delivery: Order Assignment with Travel-Time Predictors. *Management Science*, 67:4095–4119, 2020.

M. Löffler, N. Boysen, and M. Schneider. Picker Routing in AGV-Assisted Order Picking Systems. *INFORMS Journal on Computing*, 34:440–462, 2021.

M. Mirzaei, N. Zaerpour, and R. de Koster. The impact of integrated cluster-based storage allocation on parts-to-picker warehouse performance. *Transportation Research Part E: Logistics and Transportation Review*, 146:102207, 2021. ISSN 1366-5545. doi: https://doi.org/10.1016/j.tre.2020.102207. URL https://www.sciencedirect.com/science/article/pii/S1366554520308498.

C. L. Monma and C. N. Potts. On the complexity of scheduling with batch setup times. *Operations Research*, 37(5):798–804, Oct. 1989. ISSN 0030-364X. doi: 10.1287/opre.37.5.798. URL https://doi.org/10.1287/opre.37.5.798.

S. S. Özarık, L. P. Veelenturf, T. van Woensel, and G. Laporte. Optimizing e-commerce last-mile vehicle routing and scheduling under uncertain customer presence. *Transportation Research Part E: Logistics and Transportation Review*, 148:102263, 2021. ISSN 1366-5545. doi: https://doi.org/10.1016/j.tre.2021.102263. URL https://www.sciencedirect.com/science/article/pii/S1366554521000399.

C. K. Poon and P. Zhang. Minimizing makespan in batch machine scheduling. *Algorithmica*, 39(2): 155–174, feb 2004. ISSN 0178-4617. doi: 10.1007/s00453-004-1083-4. URL https://doi.org/10.1007/s00453-004-1083-4.

H. Qin, J. Xiao, D. Ge, L. Xin, J. Gao, S. He, H. Hu, and J. G. Carlsson. JD.com: Operations Research Algorithms Drive Intelligent Warehouse Robots to Work. *INFORMS Journal on Applied Analytics*, 52: 42–55, 2022.

H. Ratliff and A. Rosenthal. Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, 31:507–521, 06 1983. doi: 10.1287/opre.31.3.507.

D. Reyes, A. L. Erera, and M. W. Savelsbergh. Complexity of routing problems with release dates and deadlines. *European Journal of Operational Research*, 266(1):29–34, 2018. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2017.09.020. URL https://www.sciencedirect.com/science/article/pii/S037722171730841X.

K. Saleh. The importance of same day delivery – statistics and trends, 2022. URL https://www.invespcro.com/blog/same-day-delivery/.

M. Schiffer, N. Boysen, P. S. Klein, G. Laporte, and M. Pavone. Optimal Picking Policies in E-Commerce Warehouses. *Management Science*, 2022. doi: 10.1287/mnsc.2021.4275. Forthcoming.

A. Scholz, S. Henn, M. Stuhlmann, and G. Wäscher. A new mathematical programming formulation for the single-picker routing problem. *European Journal of Operational Research*, 253(1):68–84, 2016. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2016.02.018. URL https://www.sciencedirect.com/science/article/pii/S0377221716300388.

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

D. Schubert, H. Kuhn, and A. Holzapfel. Same-day deliveries in omnichannel retail: Integrated order picking and vehicle routing with vehicle-site dependencies. *Naval Research Logistics (NRL)*, 68(6):721–744, 2021. doi: https://doi.org/10.1002/nav.21954. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.21954.

M. Schutten, S. Van De Velde, and W. Zijm. Single-machine scheduling with release dates, due dates and family setup times. *Management Science*, 42:1165–1174, 11 1996. doi: 10.1287/mnsc.42.8.1165.

L. Shapley. On balanced sets and cores. *Naval Research Logistics Quarterly*, 14(4):453–460, 1967. doi: https://doi.org/10.1002/nav.3800140404. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800140404.

L. Shapley. Cores of convex games. *International Journal of Game Theory*, 1:11–26, 1971.

B. C. Shelbourne, M. Battarra, and C. N. Potts. The vehicle routing problem with release and due dates. *INFORMS Journal on Computing*, 29(4):705–723, 2017. doi: 10.1287/ijoc.2017.0756. URL https://doi.org/10.1287/ijoc.2017.0756.

Statista. Total number of warehouses in the us from 2007 to 2020, 2022. URL https://www.statista.com/statistics/873492/total-number-of-warehouses-united-states/.

A. Stroh, A. Erera, and A. Toriello. Tactical design of same-day delivery systems. *Management Science*, 68: 3444–3463, 2022.

X. Sun, K. Li, and W. Li. The vehicle routing problem with release dates and flexible time windows. *Engineering Optimization*, 2021. doi: 10.1080/0305215X.2021.1974853. Forthcoming.

Technavio. Same-day delivery market in us, forecast and analysis, 2021. URL https://www.technavio.com/report/same-day-delivery-market-in-the-us-industry-analysis .

C. Tilk, K. Olkis, and S. Irnich. The last-mile vehicle routing problem with delivery options. *OR Spectrum*, 43:877–904, 2021. doi: https://doi.org/10.1007/s00291-021-00633-0.

T. Vanelslander, L. Deketele, and D. Hove. Commonly used e-commerce supply chains for fast moving consumer goods: comparison and suggestions for improvement. *International Journal of Logistics*, 16: 243–256, 06 2013. doi: 10.1080/13675567.2013.813444.

S. Voccia, A. Campbell, and B. Thomas. The same-day delivery problem for online purchases. *Transportation Science*, 53(1):167–184, 2019.

M. Wang, R.-Q. Zhang, and K. Fan. Improving order-picking operation through efficient storage location assignment: A new approach. *Computers & Industrial Engineering*, 139:106186, 2020. ISSN 0360-8352. doi: https://doi.org/10.1016/j.cie.2019.106186. URL https://www.sciencedirect.com/science/article/pii/S0360835219306552.

S. Webster and K. R. Baker. Scheduling Groups of Jobs on a Single Machine. *Operations Research*, 43:692–703, 1995.

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

D. Yang, Y. Wu, and W. Ma. Optimization of storage location assignment in automated warehouse. *Microprocessors and Microsystems*, 80:103356, 2021.

W. Yu and Z. Liu. Vehicle routing problems with regular objective functions on a path. *Naval Research Logistics (NRL)*, 61(1):34–43, 2014. doi: https://doi.org/10.1002/nav.21564. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.21564.

J. Yuan, Z. Liu, C.-T. Ng, and T. Cheng. Single machine batch scheduling problem with family setup times and release dates to minimize makespan. *Journal of Scheduling*, 9:499–513, 12 2006. doi: 10.1007/s10951-006-8776-2.

J. Zhang, S. Onal, and S. K. Das. The dynamic stocking location problem – dispersing inventory in fulfillment warehouses with explosive storage. *International Journal of Production Economics*, page 107550, 2020.

S. Zhong, V. Giannikas, J. Merino, D. McFarlane, J. Cheng, and W. Shao. Evaluating the benefits of picking and packing planning integration in e-commerce warehouses. *European Journal of Operational Research*, pages 67–81, 2021. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2021.09.031. URL https://www.sciencedirect.com/science/article/pii/S0377221721008067.

L. Zhou, L. Sun, Z. Li, W. Li, N. Cao, and R. J. Higgs. Study on a storage location strategy based on clustering and association algorithms. *Soft Computing*, 24:5499–5516, 2020.

A Submodularity of Routing Times on Single-Block Warehouses

The single-block layout in warehouses has an underlying naturally submodular graph (Herer and Penn, 1995), thus if all edges have non-negative weights, the Steiner TSP cost of visiting any set of nodes is submodular. It follows that dispatch times are submodular for "1-SKU-multiple-items" models. We claim that the dispatch times remain submodular for "multiple-SKU-multiple-items" models with multiple storage locations, and prove this via two propositions.

Proposition 14. *If all orders have at most one SKU (but may have many items) and SKUs have multiple possible storage locations, the dispatch times remain submodular.*

Proof. Let $A = \{a_1, a_2, \ldots, a_i\}$ and $B = \{b_1, b_2, \ldots, b_j\}$ be two subsets of "1-SKU-1-item" orders with P_{a_h} denoting the set of possible locations for a_h, $h = 1, \ldots, i$, and the same definition for P_{b_h} with $h = 1, \ldots, j$. Then there exist vectors $p_a^* = (p_{a_1}^*, \ldots, p_{a_i}^*) \in P_{a_1} \times P_{a_2} \times \ldots \times P_{a_i}$ and $p_b^* = (p_{b_1}^*, \ldots, p_{b_j}^*) \in P_{b_1} \times P_{b_2} \times \ldots \times P_{b_j}$ satisfying

$$f(A) = f(p_{a_1}^*, \ldots, p_{a_i}^*) = \min_{P_{a_1} \times P_{a_2} \times \ldots \times P_{a_i}} f(p_{a_1}, \ldots, p_{a_i})$$
$$f(B) = f(p_{b_1}^*, \ldots, p_{b_j}^*) = \min_{P_{B_1} \times P_{B_2} \times \ldots \times P_{B_j}} f(p_{b_1}, \ldots, p_{b_j}),$$

where we abuse notation and use f to denote optimal routing time through locations.

Furthermore, from submodularity we know that $f(p_a^* \cap p_b^*) + f(p_a^* \cup p_b^*) \leq f(p_a^*) + f(p_b^*) = f(A) + f(B)$. Let $A \cap B = \{c_1, \ldots, c_r\}$ and $A \cup B = \{d_1, \ldots, d_q\}$, with respective definitions P_{c_h} for $h = 1, \ldots, r$; and P_{d_h} for $h = 1, \ldots, q$; then

$$f(A \cap B) = \min_{P_{c_1} \times \ldots \times P_{c_r}} f(p_c) \leq f(p_a^* \cap p_b^*)$$
$$f(A \cup B) = \min_{P_{d_1} \times \ldots \times P_{d_q}} f(p_d) \leq f(p_a^* \cup p_b^*).$$

It follows that $f(A \cap B) + f(A \cup B) \leq f(A) + f(B)$, so the dispatching times remain submodular under multiple storage locations. Note that the order-picking literature usually assumes that the dispatch times do not vary if multiple items (of one SKU) are included (instead of one), so the proof applies for "1-SKU-multiple-items" orders. □

Now, we can extend the result to multiple SKUs.

Proposition 15. *Dispatch times remain submodular under "multiple-SKUs-multiple-items" orders with multiple storage locations.*

Proof. For any order with multiple SKUs, we can consider each SKU as an independent order, and simply define the batch as being the union of these "orders". The result follows from Proposition 14. □

B Proofs from Section 3

B.1 Strong NP-Hardness Proof

Proposition 4. *SMD is strongly NP-Hard, even in the special case where f is a Steiner TSP in a generalized "star" graph, a tree where a depot node has arbitrary degree but all other nodes have degree one or two.*

Proof. Consider the 3-partition problem (3-PP), which is known to be strongly NP-hard (Garey and Johnson, 1979). An instance of 3-PP is described by two positive integers a and h and a subset $S = \{a_1, a_2, \ldots, a_{3h}\}$, where each a_i is a positive integer with $a/4 < a_i < a/2$ for all $i = 1, \ldots, 3h$. The question is whether it is possible to find a partition $S_1, S_2, \ldots, S_h$ such that $\sum_{i:a_i \in S_\ell} a_i = a$, for all $\ell = 1 \ldots, h$.

We reduce 3-PP to the decision version of SMD with dispatch times given by a generalized "star" graph. Consider an arbitrary instance of 3-PP and create the following instance for SMD. For each a_i number, we create a path with exactly h nodes. We denote nodes as $v_{i,j}$ where the first index represents the path of the node and the second index is its respective position in the path; in particular, $v_{i,h}$ is the farthest node in path i from the depot and $v_{i,1}$ is closest to the depot in that path. Each node $v_{i,h}$ has a release time of 0 and a round-trip from the depot takes a_i time units; node $v_{i,j}$ has a release time $(h-j)a$ and the round-trip from the depot takes $a_i j/h$. The dispatch time to nodes in different paths is just the sum of the times incurred by visiting each path individually. We claim that a 3-PP instance is "Yes" if and only if the corresponding SMD instance has makespan smaller or equal to $a(h + \frac{h-1}{2})$.

First, consider a solution for the 3-PP problem, given by the partition $S_1, \ldots, S_h$. We construct a solution for SMD as follows. The first dispatch leaves the depot at time 0 and includes all the available nodes from paths in set S_1. This dispatch finishes at time a, and the second then covers all available nodes from paths in S_2. This finishes at time $2a$ and the third dispatch covers available nodes in paths corresponding to S_3. We repeat this until the h-th dispatch, which includes all nodes in the three paths corresponding to S_h and finishes at time ah. At this point, all remaining uncovered nodes are available to be dispatched, and furthermore, dispatching uncovered nodes in paths corresponding to S_i takes $a(h-i)/h$ time. Adding a dispatch that includes all uncovered nodes, the makespan is $ah + \sum_{i=1}^{h-1} \frac{h-1}{h} a = a(h + \frac{h(h-1)}{2h}) = a(h + \frac{h-1}{2})$. It follows that if the 3-PP problem is feasible, there exists a solution with the desired makespan for SMD.

Consider now a solution for SMD with makespan less than or equal to $a(h + \frac{h-1}{2})$. We claim that this makespan can only be achieved by instances corresponding to feasible 3-PP instances, and that it is actually the minimum makespan for such instances. Notice that it is always feasible to wait until time $a(h-1)$ and then include all nodes in a single dispatch, inducing a makespan of $a(h-1) + ah$. It follows that any solution having a makespan of $a(h + \frac{h-1}{2})$ must have dispatches "saving" $a\frac{h-1}{2}$ time between time 0 and $a(h-1)$.

Any dispatch starting during interval $[0, a)$ contains only nodes of the type $v_{i,h}$; therefore, after the dispatch is finished, the paths i that were visited have as farthest uncovered node $v_{i,h-1}$, requiring a round-trip of $\frac{h-1}{h}a_i$ time. Hence, the total time saved (with respect to the previously described feasible solution) by that dispatch is equal to its dispatch time multiplied by $1/h$. Similarly, any dispatch starting during interval $[a, 2a)$ can save at most its dispatch time multiplied by $2/h$. In general any dispatch starting in the interval $[ai, a(i+1))$ can save at most its dispatch time multiplied by $(i+1)/h$, for $i = 0, \ldots, (h-2)$, and that saving is obtained only if the paths visited by that dispatch are visited for the first time.

The maximum total time saved by these possible dispatches is obtained by saving the maximum possible time in each interval, which is exactly $a\sum_{i=1}^{h-1}\frac{i}{h} = a\frac{h-1}{2}$. This implies that a solution with makespan $a(h+\frac{h-1}{2})$ has no idle time, and has a dispatch finishing at each of the respective times ai, for $i = 1, \ldots, h-1$, and each of these dispatches serves its corresponding paths for the first time. We can thus describe each interval $[ai, a(i+1))$ by the paths the dispatch visits during this time, and the dispatch time defined by visiting the last nodes of those paths is exactly equal to a. As each path is visited only once during the interval $[0, a(h-1))$, if we describe the paths served by each interval $[ai, a(i+1)]$ as subsets S_i, then those subsets represent a partition of the subset S for the 3-PP problem, satisfying the desired conditions. Hence if there exists a solution for SMD with makespan equal to $a(h+\frac{h-1}{2})$, the 3-PP instance has the desired partition.

We proved that the 3-PP problem is feasible if and only if there exists a solution with makespan $a(h+\frac{h-1}{2})$ for SMD, and the reduction from the 3-PP problem to SMD is polynomial: we create $3h^2$ nodes with polynomially bounded dispatch times and release times. We conclude that our formulation is strongly NP-Hard for a generalized "star" network, and hence for arbitrary submodular functions. □

For illustration, we present an example where the 3-PP instance has $h = 2$. Figure 7 presents the graph used for the SMD instance, where all nodes in the outer layer have their orders released at time 0 and the orders corresponding to the nodes in the inner layer are released at time a. The goal is a set of order batches with corresponding makespan equal to $2.5a$. The makespan when batching all orders together is equal to $3a$, with the dispatch leaving at time a; to get the desired solution at least one dispatch must start earlier, and during the $[0, a)$ time interval $0.5a$ units of time must be "saved" compared to the one-dispatch solution.

B.2 Approximation Guarantee with Limited Dispatches

Proposition 6. *Consider a heuristic for SMD that, for any instance, generates a solution with at most $d < n$ dispatches. The heuristic's approximation factor cannot be smaller than $1 + 1/d$.*

Proof. Consider an instance with n orders, dispatch time function $f(S) = |S|$ and release times $r_i = i - 1$ for $i = 1, \ldots, n$. We may assume without loss of generality that the solution has no idle time after the first dispatch leaves the depot. Because of this fact and the modular dispatch time function, the best solution with $d < n$ dispatches is equivalent to finding the minimum time t such that we can do d consecutive dispatches that cover all orders.

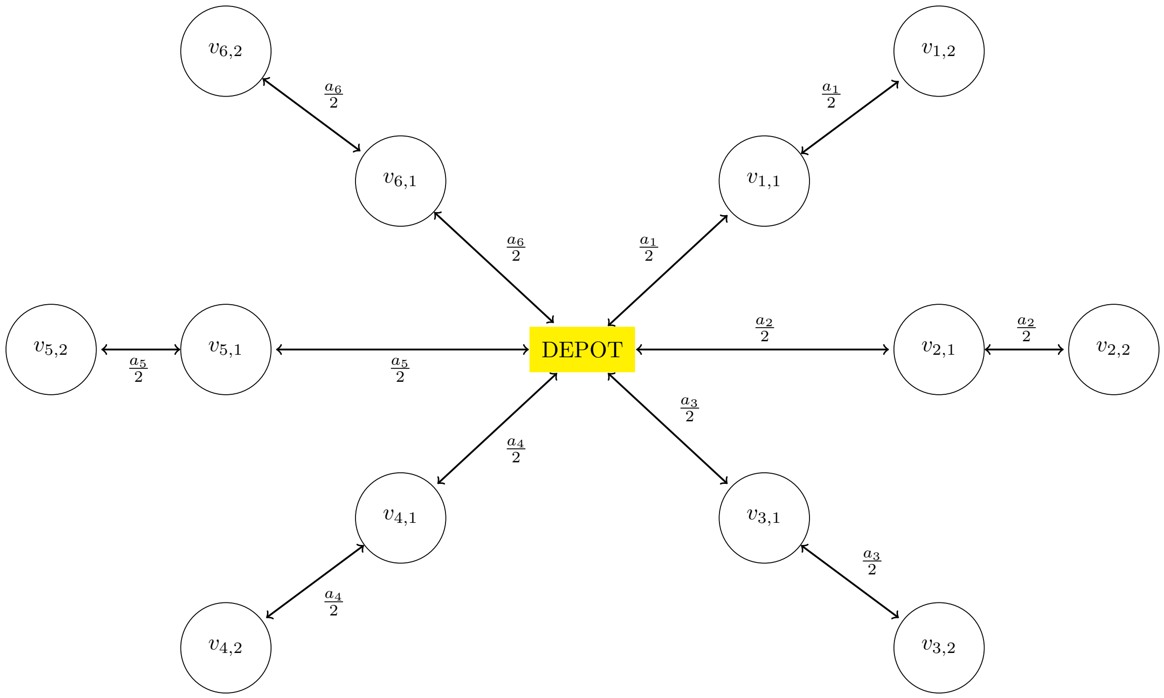


Figure 7: Example of SMD instance derived from 3-PP with $h = 2$.

At any time t before the first dispatch, the number of released orders is $\lfloor t + 1 \rfloor$. Furthermore, after dispatching $|S|$ orders, the number of orders that were released during the dispatch is exactly $|S|$, as long as the dispatch returns by time $n - 1$. It follows that we can get a lower bound on the solution's makespan by assuming that for any t before the first dispatch, the number of orders that have been released is $t + 1$ instead of $\lfloor t + 1 \rfloor$, and by then precisely arranging the d dispatches so that they incur no idle time: the departure time of each dispatch is equal to the release time of the last order that has been released up to that point. This translates to the equation $t + (d - 1)(t + 1) = n - 1$, where t is the start time of the first dispatch, then $d - 1$ dispatches each with dispatch time $(t + 1)$ are executed (as that is the number of orders released between dispatches), and the last dispatch starts when the last order is released. This yields $t = n/d - 1$, and the lower bound on the makespan is $n + n/d - 1$ because n is the total dispatch time of all dispatches.

The optimal solution uses n consecutive dispatches, starting at time zero and with no idle time between dispatches, giving an optimal makespan of n. Therefore, the approximation ratio of a heuristic that executes at most d dispatches cannot be better than $1 + 1/d - 1/n$, which goes to $1 + 1/d$ as n goes to infinity. $\square$

C Proofs from Section 4

C.1 FIFO Algorithm

Proposition 7. *Let κ be the number of operations needed to compute all values $f_{i,j}$, for $1 \leq i \leq j \leq n$. For an SMD instance, the best makespan among FIFO solutions is given by z^{FIFO}, and can be computed in $\Theta(n^2 + \kappa)$ time.*

Proof. First, we claim that $z_{i,j}$ is the minimum makespan required to dispatch orders in N_j in FIFO fashion when the last batch dispatched is $[i, j]$. We prove the claim by induction, and the base case is straightforward as all $z_{1,j}$ are the optimal makespan of dispatching orders $\{1, 2 \ldots, j\}$ in one batch. As induction hypothesis, suppose there exists some $i_0 \in N$ such that for all $i \leq i_0$ and all $j \geq i$, $z_{i,j}$ represents the minimum makespan to dispatch orders in N_j in FIFO fashion when the last batch dispatched is $[i, j]$.

Now we consider the values $z_{i_0+1,j}$ for $j \geq i_0 + 1$. The dispatch time of the batch $[i_0 + 1, j]$ is given by $f_{i_0+1,j}$, and the dispatch cannot leave the depot earlier than both r_j and the optimal makespan needed to dispatch orders in N_{i_0}; the latter is $\min_{k \leq i_0}\{z_{k,i_0}\}$ because of the induction hypothesis and the fact that orders need to be dispatched in FIFO fashion. Thus $z_{i_0+1,j} = \max\{r_j, \min_{k \leq i_0}\{z_{k,i_0}\}\} + f_{i_0+1,j}$ is the makespan to dispatch orders in N_j in FIFO fashion when the last batch is $[i_0 + 1, j]$. It follows that $z_{i,n}$ represents the minimum makespan to dispatch all n orders in a FIFO manner when the last batch dispatched is $[i, n]$, and then the makespan for the best FIFO solution is given by $\min_{i \leq n}\{z_{i,n}\}$.

Assuming we perform κ operations to compute, the $f_{i,j}$ values, the recursion's complexity is dominated by equations (2b), which take quadratic time (for each $i \geq 2$ there are roughly $i - 1$ operations), yielding the complexity. □

C.2 Q-FIFO-Optimal Algorithm

Theorem 10. *Let Q be a fixed integer. For any Q-FIFO-optimal dispatch function, SMD can be optimized in $\Theta(n^{2Q} + \kappa)$ time, where κ is the time required to compute all interval dispatch times for each set S_q that partitions N.*

Proof. We denote the Q non-intersecting sets of orders defining the Q-FIFO-optimal function as $S_1, S_2, \ldots, S_Q$, with respective release time vectors $r^1, r^2, \ldots, r^Q$. Furthermore, we let $n_1, n_2, \ldots, n_Q$ be the number of orders in each of the Q sets, and we denote the pre-computed dispatch times within the Q sets as $f^1, \ldots, f^Q$, with $f^q_{i,j}$ being the dispatch time of batch $[i, j]$ from set S_q, where $1 \leq i \leq j \leq n_q$. To solve this problem, we use an array z with $2Q + 1$ dimensions, where the first index indicates the last set S_q that had a dispatch, whereas the other index pairs represent the order intervals dispatched last for each of the sets; as an example, $z(2, i_1, j_1, i_2, j_2, i_3, j_3)$ represents the minimum makespan when the last dispatch corresponding to set S_q was $[i_q, j_q]$ for $q = 1, 2, 3$, and the very last dispatch was $[i_2, j_2]$. We also use zeroes to indicate no dispatch for that set has yet occurred. The recursion defining z is then

$$z\Big(\sigma,(i_q,j_q)_{q=1}^{Q}\Big) = f^{\sigma}_{i_\sigma,j_\sigma} + \max\begin{cases} r^{\sigma}_{j_\sigma}, \\ \min\{z(\sigma',(i_q,j_q)_{q\neq\sigma}\cup(k_\sigma,i_\sigma-1)) : k_\sigma\in[1,i_\sigma-1]; \sigma'\in\{q : i_q\geq 1\}\}, \end{cases} \tag{4a}$$

if $i_\sigma \geq 2$;

$$z\Big(\sigma,(i_q,j_q)_{q=1}^{Q}\Big) = f^{\sigma}_{i_\sigma,j_\sigma} + \max\{r^{\sigma}_{j_\sigma}, \min\{z(\sigma',(i_q,j_q)_{q\neq\sigma}\cup(0,0)) : \sigma'\in\{q : i_q\geq 1\}\setminus\{\sigma\}\}\}, \quad \text{if } i_\sigma = 1. \tag{4b}$$

The boundary condition is simply $z(0,\ldots,0)=0$. The notation $(i_q,j_q)_{q\neq\sigma}\cup(k_\sigma,i_\sigma-1)$ means the index pair $(k_\sigma,i_\sigma-1)$ is appended to the previous vector in the σ-th position. Intuitively, the current state identifies that the last dispatch had orders from set S_σ, and thus the recursion minimizes over which previous set had the next-to-last dispatch, and what the previous dispatch to S_σ contained. If the dispatch to S_σ was the first one, in which case $i_\sigma = 1$, there is no previous dispatch to this set, indicated by the previous state having the zero indexes. The optimal makespan of SMD is then

$$\min\Big\{z\Big(\sigma,(i_q,n_q)_{q=1}^{Q}\Big) : \quad i_q = 1,\ldots,n_q,\ q=1,\ldots Q; \quad \sigma=1,\ldots,Q\Big\}. \tag{4c}$$

Generalizing the argument from the proof of Proposition 7, the array z tracks the optimal makespan for each partial state. There are $O(n^{2Q})$ states, each with $O(n)$ terms in its recursion, yielding a running time of $O(n^{2Q+1})$.

We can reduce the running time by an order of magnitude by noting that the minimization in (4a) can be computed beforehand. Specifically, for $\sigma\in[1,Q]$, $i_\sigma\geq 2$ and $(i_q,j_q)_{q\neq\sigma}$, define

$$\check{z}(\sigma,i_\sigma,(i_q,j_q)_{q\neq\sigma}) = \min\{z(\sigma',(i_q,j_q)_{q\neq\sigma}\cup(k_\sigma,i_\sigma-1)) : k_\sigma\in[1,i_\sigma-1], \sigma'\in\{q : i_q\geq 1\}\}. \tag{4d}$$

We can then replace the minimization in (4a) with the appropriate term from the auxiliary array $\check{z}$. This reduces the number of terms in the original recursions to a constant number (either two or Q), and adds $O(n^{2Q-1})$ auxiliary states with $O(n)$ terms in each minimization, yielding the desired complexity.

□

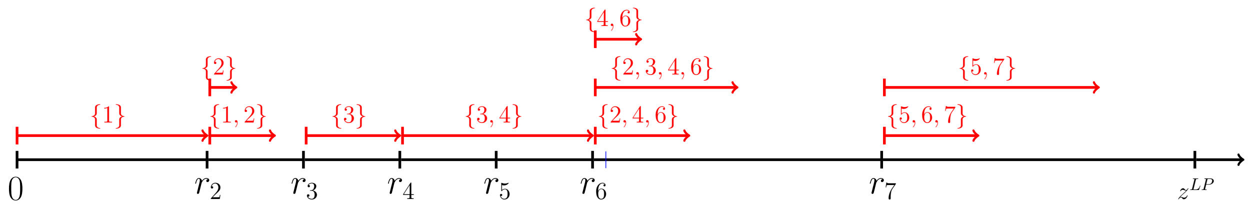
D Proofs from Section 5

D.1 Approximation Ratio of Algorithm 1

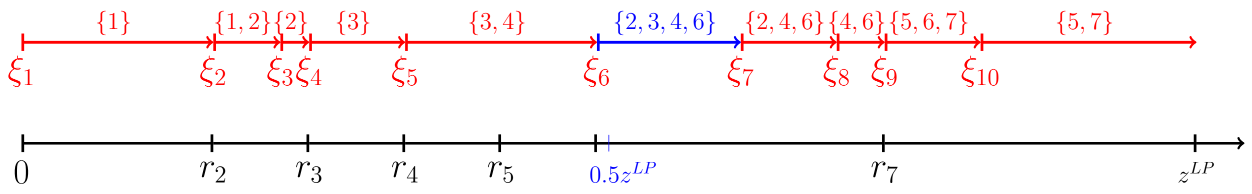
Theorem 11. *Let (t^{LP},x^{LP},z^{LP}) be an extreme point optimal solution of the LP relaxation of (1), and define $\Delta := z^{LP} - \sum_{S\subseteq N} x^{LP}_S f(S) \geq 0$. The two-dispatch solution obtained by applying Algorithm 1 has a tight $3/2 - \Delta/2z^{LP}$ approximation ratio and can be computed in $O(n^2)$ time.*

Proof. Consider an arbitrary instance of SMD, and let (t^{LP}, x^{LP}, z^{LP}) be an extreme point optimal solution for the linear relaxation of (1). We focus first on the case where the solution has no idle time, $\Delta = 0$, and afterwards extend the result to $\Delta > 0$.

There are at most $2n$ positive variables x_S^{LP}, and we can group them according to the highest index in the batch, which takes $O(n)$ time. For each $i \in N$ we get a (possibly empty) set of batches B_i, where each $S \in B_i$ has a corresponding positive x_S^{LP} variable and the highest index in S is exactly i; see Figure 8a for a visual representation where the sets of batches B_i are stacked just above the respective release time r_i, and the length of the arrows correspond to the respective value $x_S^{LP} f(S)$. Furthermore, for each $i \in N$ we can sequence the batches in B_i according to their cardinality (in non-increasing fashion) leading to a sequencing (and indexing) of all fractional dispatches; this sorting requires $O(n \log(n))$ time. For the example in Figure 8, Figure 8b shows the fractional dispatches after the indexing. For the rest of the proof, we refer to the first batch according to this sequence as S_1, the second batch is S_2, and so on. Because we assume the solution has no idle time, we can define the fractional starting time of each batch as ξ_j, where ξ_1 is 0 and $\xi_j = \sum_{k<j} x_{S_k}^{LP} f(S_k)$. Furthermore, there is some batch S_h satisfying $\xi_h < z^{LP}/2 \leq \xi_{h+1}$; in Figure 8b it is the blue dispatch.



(a) Fractional dispatches grouped according to highest index.



(b) Sequenced fractional dispatches; the blue dispatch corresponds to D_h.

Figure 8: Visual representation of a fractional solution.

We define $D = \bigcup_{k=1}^{h} S_k$ and claim that the two-dispatch solution defined by batches D and $N \setminus D$ has makespan bounded above by $3z^{LP}/2$. Let $r_D = \max_{i \in D}\{r_i\}$ be the release time of batch D; from the feasibility of the LP's solution we get $r_D \leq \xi_h$. To prove the claim, we use the non-negativity and monotonicity of f, but not submodularity; instead, we use the weaker property of *fractional subadditivity*: for any non-empty $S \subseteq N$ and any $\alpha \in \mathbb{R}_+^{2^S \setminus \emptyset}$ satisfying

$$\sum_{S' \subseteq S, S' \ni i} \alpha_{S'} = 1, \quad i \in S,$$

we have

$$\sum_{\emptyset \neq S' \subseteq S} \alpha_{S'} f(S') \geq f(S).$$

Observe that if we apply the condition only to binary α we obtain (non-fractional) subadditivity. A submodular function is always fractionally subadditive; this follows from combining the fact that base polyhedra of submodular functions are non-empty (Edmonds, 1970; Shapley, 1971), or equivalently, that convex cost-sharing games have a non-empty core, with the Bondareva-Shapley theorem (Bondareva, 1963; Shapley, 1967), which states that a game is *balanced* if and only if it has a non-empty core. A submodular function defines a convex cost-sharing game; the fact that the core of the game and any sub-game is non-empty means the game is *totally balanced*, and this property precisely corresponds to f being fractionally subadditive.

We prove the claim in two separate cases:

1. Suppose $r_D + f(D) \leq z^{LP}$. By construction, all orders in $N \setminus D$ have not been fractionally dispatched before $z^{LP}/2$. From the fractional subadditivity and monotonicity of f, we obtain $f(N \setminus D) \leq \sum_{k>h} x_{S_k}^{LP} f(S_k) \leq z^{LP}/2$, and thus the solution's makespan is bounded above by $3z^{LP}/2$.

2. Now assume $r_D + f(D) > z^{LP}$, so the solution defined by batches D and $N \setminus D$ has makespan of $r_D + f(D) + f(N \setminus D)$. For contradiction purposes, suppose $r_D + f(D) + f(N \setminus D) > 3z^{LP}/2$, which implies

$$\begin{aligned} f(D) > \frac{3}{2}z^{LP} - r_D - f(N \backslash D) &\geq \frac{3}{2}z^{LP} - r_D - \left(z^{LP} - \sum_{k \leq h} x_{S_k}^{LP} f(S_k)\right) \\ = \frac{1}{2}z^{LP} + \sum_{k \leq h} x_{S_k}^{LP} f(S_k) - r_D &> \sum_{k \leq h} x_{S_k}^{LP} f(S_k) = \xi_{h+1}. \end{aligned}$$

The second inequality is a consequence of fractional subadditivity, monotonicity, and the fact that $\Delta = 0$. The last inequality follows from the fact that $r_D \leq \xi_h < z^{LP}/2$. We claim that a solution that fractionally dispatches batch D from time 0 up to time ξ_{h+1} (with $x_D = \xi_{h+1}/f(D) < 1$), then fractionally dispatches all orders with $x_N = 1 - x_D$ and finally fractionally dispatches $N \setminus D$ with $x_{N \setminus D} = x_D$ yields a lower bound for z^{LP}. We prove the claim as follows: let I_1 be the original instance, and now consider the instance I_2 where all orders with release time earlier than ξ_h that are not included in D have their release time moved to ξ_{h+1}. This new instance cannot have a lower makespan because its release times are equal or later than those in the original instance, and so the dispatches from the original instance are also an optimal solution for this new modified instance. We depict an example in Figures 9a, 9b and 9c. Now, we consider instance I_3, where the release times from orders in D are set to 0, and the release times of orders in $N \setminus D$ are set to ξ_{h+1}, as shown in Figure 9d. All the release times are smaller or equal to the ones in instance I_2, which means an optimal solution for I_3 provides a lower bound for I_2 (and thus also I_1). It is not hard to see that our proposed solution is optimal for I_3; Figure 9e shows this for our example. Therefore,

$$\begin{aligned} z^{LP} &\geq \frac{\xi_{h+1}}{f(D)}(f(D)+f(N\setminus D)) + \frac{1-\xi_{h+1}}{f(D)} f(N) \\ &\geq \frac{\xi_{h+1}}{f(D)}(f(D)+f(N\setminus D)) + \frac{1-\xi_{h+1}}{f(D)} f(D) = f(D) + \frac{\xi_{h+1}}{f(D)} f(N\setminus D), \end{aligned} \tag{5}$$

where the second inequality follows from monotonicity. For fixed $f(N \setminus D)$ and ξ_{h+1}, the right-hand side of (5) is minimized at $f(D) = \sqrt{\xi_{h+1} f(N\setminus D)}$; however, recalling that $f(D) > \xi_{h+1} > f(N\setminus D)$, we can actually lower-bound the right-hand side by assuming $f(D) = \xi_{h+1}$, and thus $z^{LP} \geq f(D) + f(N \setminus D)$. Finally, as $r_D \leq \xi_h < \frac{1}{2} z^{LP}$, $r_D + f(D) + f(N\setminus D) \leq \frac{3}{2} z^{LP}$, which is our desired contradiction.

The argument above shows that when $\Delta = 0$, the two-dispatch solution given by D and $N \setminus D$ has makespan bounded above by $3z^{LP}/2$. Proposition 6 verifies that this approximation ratio is tight, as we cannot do better with a two-dispatch solution. We now extend the result to when $\Delta > 0$. As before, we can sequence the batches from the optimal solution, ordered first by latest order in the batch and then by cardinality. The solution may now have idle time, gaps between the end of one fractional dispatch and the start of the next; whenever the solution has idle time, it means all orders with smaller release time have been completely dispatched before this point. We denote these idle times by $\Delta_1, \Delta_2, \ldots, \Delta_p$, with $\sum_{i=1}^{p} \Delta_i = \Delta$. We can construct a new instance I' by performing the following procedure: starting with $i = p$, for all orders with release time greater or equal to the end of the period corresponding to Δ_i, subtract Δ_i to their release times, then reduce i by one and repeat until $i = 0$. Then x^{LP} remains optimal for I' (with a makespan of $z^{LP} - \Delta$) because it is still feasible by construction, and the solution has no idle time in this instance. Applying the previous argument, the two-dispatch solution has a makespan bounded above by $3(z^{LP} - \Delta)/2$ for I'. Returning to the original instance, the solution's makespan is therefore at most $3z^{LP}/2 - \Delta/2$, because in I' we moved up release times by at most Δ. The $O(n^2)$ complexity follows from the transformation to instance I', which requires $O(n^2)$ operations.

□

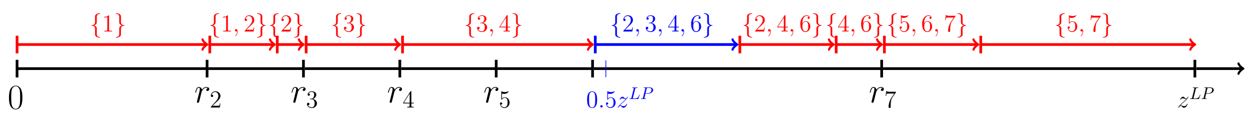
D.2 Strengthened Inequalities

Theorem 13. *Inequalities (1b) in (1) can be strengthened to*

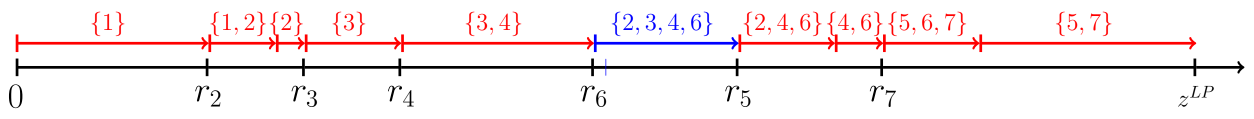
$$t_i \geq r_i + \sum_{j<i} \sum_{S \subseteq N_j, S \ni j} \max\{0, r_j - r_i + f(S)\} x_S, \quad i = 2, 3, \ldots, n. \tag{3}$$

Proof. We prove the theorem by induction on i. Let (t, x, z) be any feasible solution to (1). When $i = 2$, we have

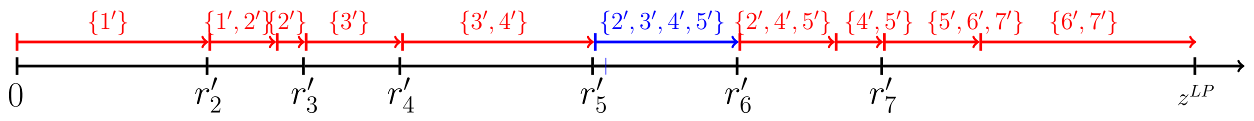
$$t_2 \geq \max\{r_2, r_1 + x_{\{1\}} f(\{1\})\} = r_2 + x_{\{1\}} \max\{0, r_1 - r_2 + f(\{1\})\}.$$



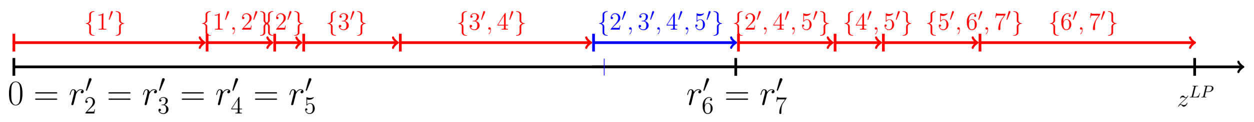
(a) Optimal fractional solution for instance I_1.



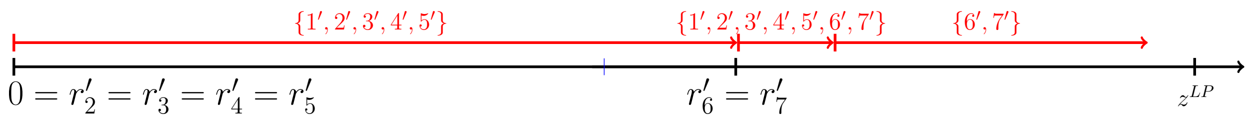
(b) Moving the arrival of order 5 to ξ_{h+1}.



(c) Optimal solution for instance I_2 (after adjusting the indices).



(d) Instance I_3.



(e) Optimal solution for instance I_3.

Figure 9: Visual representation of the lower bound for z^{LP}.

Now suppose $i \geq 3$. If no dispatch occurs before i is released, the inequality is valid trivially. Similarly, if the coefficient in the inequality evaluates to zero for all previous dispatches, the inequality is again trivially valid. Assume this is not the case, and let $k = \max\Big\{j < i : \sum_{S \subseteq N_j, S \ni j} x_S \max\{0, r_j - r_i + f(S)\} > 0\Big\}$ be the largest index before i for which there is a dispatch with positive coefficient in the inequality. By feasibility, we have

$$t_i \geq t_k + \sum_{S \subseteq N_k, S \ni k} x_S f(S) = r_i + \Bigg(-r_i + t_k + \sum_{S \subseteq N_k, S \ni k} x_S f(S)\Bigg).$$

The term inside the parenthesis can be further expanded and bounded below as follows:

$$\begin{aligned} -r_i + t_k + \sum_{S \subseteq N_k, S \ni k} x_S f(S) &\geq -r_i + r_k + \sum_{S \subseteq N_k, S \ni k} x_S f(S) + \sum_{j<k} \sum_{S \subseteq N_j, S \ni j} x_S \max\{0, r_j - r_k + f(S)\} \\ &\geq -r_i + r_k + \sum_{S \subseteq N_k, S \ni k} x_S f(S) + \sum_{j<k} \sum_{S \subseteq N_j, S \ni j} x_S \max\{0, r_j - r_i + f(S)\} \\ &= -r_i + r_k + \sum_{S \subseteq N_k, S \ni k} x_S f(S) + \sum_{j<i, j \neq k} \sum_{S \subseteq N_j, S \ni j} x_S \max\{0, r_j - r_i + f(S)\} \\ &= \sum_{j<i} \sum_{S \subseteq N_j, S \ni j} x_S \max\{0, r_j - r_i + f(S)\}. \end{aligned}$$

The first inequality follows from induction, the second because $r_i \geq r_k$, while the first equation follows from the assumption that any dispatch between k and i has a zero coefficient; the last equation simply groups terms.

□

E Implementation Details for Single-Block, Single-Picker Experiments

E.1 CG pricing problem

Recall the pricing problem for the linear relaxation of (1),

$$\min\left\{ \beta_i f(S) - \sum_{j \in S} \gamma_j : S \subseteq N_i, S \ni i \right\},$$

for each $i \in N$. When f represents optimal routing times through a single-block warehouse, this corresponds to a prize-collecting Steiner TSP over the set N_i, where $\beta_i f(S)$ is the routing time and γ_j is a prize obtained from visiting $j \in N_i$. For m aisles and n single-SKU orders scattered over a single-block warehouse with single storage locations, Ratliff and Rosenthal (1983) proposed an $O(m+n)$ algorithm to solve the Steiner TSP; we generalize that algorithm for the prize-collecting version.

The algorithm uses dynamic programming and builds a solution one aisle at a time. Starting from the left-most aisle, the algorithm considers all possible traversals of each aisle, as depicted in Figure 10. Unlike the original Ratliff and Rosenthal (1983) algorithm, we must consider traversals that do not visit all positions, including the possibility of not traversing the aisle at all (Figure 10(vi)). Furthermore, we must consider more of these traversals than the original algorithm; specifically, if an aisle has k positions we may visit, there are k possible traversals of type (ii) and (iii) from Figure 10, and there are $O(k^2)$ possible traversals of type (iv). Other aspects of the algorithm remain similar to the original, yielding a complexity of $O(m+n^2)$ for a single $i \in N$ and $O(mn+n^3)$ overall.

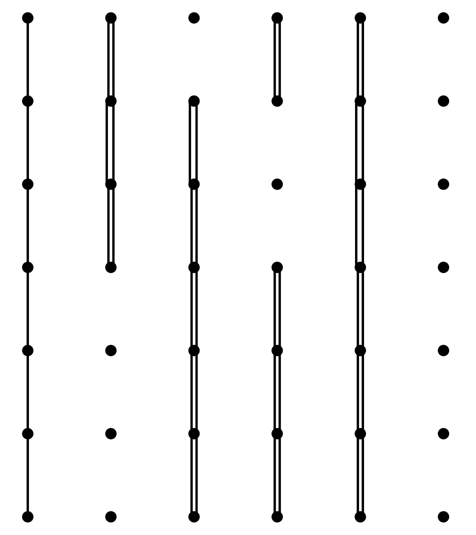


Figure 10: Possible aisle traversal configurations for the prize-collecting Steiner TSP in a single-block warehouse.

E.2 CGS pricing problem

For the linear relaxation of (1) strengthened with (3), the pricing problem is

$$\min\left\{\beta_i f(S) - \sum_{j\in S}\gamma_i + \sum_{j>i}\max\{0, r_i - r_j + f(S)\}\alpha_j : S \subseteq N_i, S \in i\right\},$$

for each $i \in N$. We use the following integer program:

$$\begin{aligned}
\min_{s,w,y\geq 0}\ & \sum_{j,k\in N_i\cup 0} c_{jk}w_{jk} - \sum_{j\in N_i}\gamma_j y_j + \sum_{j=i+1}^{n}\alpha_j s_j \\
\text{s.t. } & \sum_{k\neq j} w_{jk} = 2y_j, && j \in N_i \cup 0 \\
& \sum_{k\in S,\ell\notin S} w_{k\ell} \geq 2y_j, && S \subseteq N_i, j \in S \\
& s_j \geq r_i - r_j + \sum_{k,\ell\in N_i\cup 0} c_{k\ell}w_{k\ell} && j > i \\
& y_0 = y_i = 1; \quad y_j \in \{0,1\}, j \in N_{i-1} \\
& w_{0,j}, \in \{0,1,2\}, j \in N_i; \quad w_{jk} \in \{0,1\}, j,k \in N_i.
\end{aligned}$$

In this formulation, the y variables indicate a position is visited, the w variables indicate consecutive picking locations, and the s variables linearize the term $\max\{0, r_i - r_j + f(S)\}$. We use index 0 for the depot. The objective coefficients c are shortest path distances between locations, scaled by β_i. In our implementation, we separate the subtour elimination constraints only at integer solutions in the branch-and-bound tree.

E.3 Column Generation Acceleration

We implement the acceleration procedure from Ben-Ameur and Neto (2007) when solving the linear relaxations of (1) and its strengthening with (3). At every iteration, this procedure uses an incumbent feasible dual solution, denoted δ_f here for convenience, which can be initialized with any trivially feasible solution such as $\delta_f = 0$, and the dual solution obtained by the master solve, denoted δ_m. Instead of attempting to separate δ_m, we solve the dual separation problem for the convex combination $\hat{\delta} = \lambda\delta_f + (1-\lambda)\delta_m$. Intuitively, this solution is likelier to be feasible or at least closer to the dual feasible region. If $\hat{\delta}$ is dual feasible, it necessarily has a larger objective value than δ_f, so it becomes the new feasible incumbent. Otherwise, it is dual infeasible, so we add dual cutting planes (i.e. columns in the primal) and perform another master solve. After preliminary calibration, we used $\lambda = 0.6$ in our experiments.