ONLINE DELIVERY AND DISPATCH WITH CAPACITY CONSTRAINTS

by

Naga Sitaram Guduri

December 2025

A dissertation submitted to the

Faculty of the Graduate School of

the University at Buffalo, State University of New York

in partial fulfilment of the requirements for the

degree of

Master of Science

Department of Computer Science and Engineering

Acknowledgments

I would like to express my sincere gratitude to everyone who supported and encouraged me throughout my academic journey and the completion of this thesis.

First and foremost, I am deeply thankful to my advisor, Professor Kelin Luo, for her constant guidance, patience, and unwavering support. Her insights, encouragement, and constructive feedback played a crucial role in shaping this work and in my growth as a researcher. I am especially grateful for the time she invested in mentoring me and for believing in my abilities throughout this journey.

I would also like to thank my friends for their encouragement, discussions, and moral support, which made this journey both enjoyable and meaningful. Their presence helped me stay motivated during challenging times.

Finally, I am immensely grateful to my family for their unconditional love, understanding, and support. Their constant encouragement and belief in me provided the foundation and strength needed to pursue and complete this work.

This thesis would not have been possible without the support of all these individuals, and I am truly grateful to each of them.

Table of Contents

List of Tables

List of Figures

Abstract

We study the online food delivery problem with capacity constraints, where requests arrive over time in a metric space and a single server must deliver them while minimizing the maximum flow time. Flow time, defined as the delay between a request's release and completion, is a fundamental but relatively underexplored objective in online vehicle routing. We focus on structured metric spaces, specifically path graphs, which enable precise competitive analysis under capacity limitations.

We distinguish between myopic algorithms, which dispatch requests immediately, and far-sighted algorithms, which delay service to batch future requests. For path graphs with unit capacity, we analyze two myopic strategies and show that Earliest Released First is 2-competitive while Earliest Anticipation First is 3-competitive. In contrast, we prove that for capacity 2, no myopic FIFO-based algorithm admits a bounded competitive ratio.

To handle general capacity K, we introduce a far-sighted batching algorithm based on time-interval partitioning and show that it achieves a constant competitive ratio. These results demonstrate inherent limitations of myopic dispatch under increasing capacity and highlight the necessity of anticipation and batching for flow-time optimality in online delivery systems.

Chapter 1

Introduction

Online decision-making problems arise in many real-world systems where inputs arrive over time and decisions must be made without knowledge of future events. Such problems are naturally modeled using online algorithms, which process requests incrementally and are evaluated by comparing their performance against an optimal offline algorithm with full future information. Applications of online algorithms are particularly prominent in modern logistics and mobility systems, including food delivery platforms, autonomous delivery robots, and automated warehouse dispatch systems.

A central performance metric in these systems is *flow time*, defined as the delay between a request's release time and its completion. Minimizing maximum flow time is crucial for ensuring fairness and responsiveness, as it prevents individual requests from experiencing excessive delays. Despite its importance, flow-time minimization has received comparatively less attention than objectives such as total travel distance or total latency, especially in online routing and delivery settings.

1.1 Online Food Delivery with Capacity Constraints

The online food delivery problem models a setting in which delivery requests arrive over time at different locations in a metric space, and a server starting from a depot must deliver these requests. Each request specifies a release time and a delivery location, and the server must decide when to depart from the depot and which subset of pending requests to serve on each trip. The challenge lies in making these decisions online while minimizing the maximum flow time over all requests.

In practice, delivery vehicles are subject to *capacity constraints*, limiting the number of requests that can be served in a single trip. Capacity fundamentally changes the structure of the problem by introducing batching decisions and tradeoffs between waiting to accumulate requests and dispatching early to reduce delay. Understanding how capacity affects the performance of online algorithms is therefore essential for both theoretical analysis and practical system design.

1.1.1 Motivation and Challenges

Capacity constraints introduce several challenges that do not arise in uncapacitated settings. Myopic algorithms that immediately serve available requests may perform well when capacity is limited to one, but can become highly inefficient as capacity increases. Conversely, delaying service to batch requests can reduce travel overhead but risks increasing flow time for early arrivals.

Another key difficulty is that optimal offline solutions can coordinate batching decisions using future knowledge, while online algorithms must operate without such information. This asymmetry makes it nontrivial to design online strategies with provable guarantees, particularly for minimizing maximum flow time. As a result, it is unclear whether simple dispatch policies can remain competitive as capacity grows, or whether more sophisticated, far-sighted strategies are required.

1.2 Algorithmic Perspectives

Online algorithms for delivery problems can be broadly classified based on how they react to arriving requests. A natural class of strategies consists of *myopic algorithms*, which dispatch the server whenever pending requests are available and the vehicle is at the depot, without intentionally waiting for future arrivals. These algorithms are simple, intuitive, and easy to implement, making them attractive in real-time systems.

In contrast, *far-sighted algorithms* intentionally delay service in order to batch requests, trading off increased waiting time for improved utilization of capacity. While such strategies can potentially reduce travel overhead, they also risk increasing the flow time of early requests. Analyzing this tradeoff is particularly important in capacitated settings, where batching decisions directly affect performance.

1.2.1 Myopic vs. Far-Sighted Strategies

The distinction between myopic and far-sighted strategies plays a central role in this thesis. Myopic algorithms tend to prioritize immediacy and fairness for early requests, while far-sighted algorithms exploit future arrivals to improve overall efficiency. Capacity amplifies the tension between these approaches, as higher capacity increases the potential benefits of batching but also increases the risk of excessive delay.

A key question we address is whether myopic strategies can achieve bounded competitive ratios under capacity constraints, or whether anticipation is fundamentally necessary once capacity exceeds one. This question is explored through a detailed analysis of specific myopic policies and the design of batching-based far-sighted algorithms.

1.3 Our Contributions

This thesis presents a systematic study of the online food delivery problem under capacity constraints on structured metric spaces. Our main contributions can be summarized as follows:

- We analyze myopic algorithms on path graphs and show that for unit capacity, simple policies achieve constant competitive ratios for minimizing maximum flow time.
- We prove that for capacity two, no myopic FIFO-based algorithm admits a bounded competitive ratio on path graphs.
- We introduce and analyze a far-sighted batching algorithm for general capacity K, and show that it achieves a constant competitive ratio for path graphs.

These results reveal sharp separations between algorithmic strategies as capacity increases and provide insight into the structural role of batching and anticipation in online delivery systems.

1.4 Organization of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 introduces the necessary preliminaries and reviews related work. Chapter 3 formally defines the online food delivery model and problem setting. Chapter 4 analyzes myopic algorithms under capacity constraints. Chapter 5 presents a far-sighted batching algorithm and its theoretical analysis. Chapter 6 concludes with a discussion of implications, limitations, and directions for future work.

Chapter 2

Literature Review

This chapter reviews prior work related to online delivery and routing problems, with a focus on flow-time minimization and capacity constraints. We first discuss classical online scheduling and routing models, and then survey work specific to online food delivery and related vehicle routing problems. Finally, we highlight how existing results motivate the questions studied in this thesis.

2.1 Online Algorithms and Flow-Time Objectives

Online algorithms have been extensively studied in settings where decisions must be made without knowledge of future inputs. A standard framework for analyzing online algorithms is competitive analysis, in which the performance of an online algorithm is compared to that of an optimal offline algorithm that has full knowledge of the request sequence.

Flow time is a fundamental objective in online scheduling and routing problems. In classical scheduling settings, minimizing maximum flow time has been studied on single and multiple machines, often under release times and processing constraints. These problems highlight the inherent tension between fairness and efficiency, as algorithms must balance early service against waiting for better schedules.

In routing and mobility problems, flow-time objectives are more challenging due to

travel times and spatial constraints. While objectives such as total distance or total latency have received significant attention, maximum flow time has been comparatively less explored, particularly in online metric routing settings.

Beyond classical competitive analysis, flow-time objectives have been extensively studied in online scheduling under various machine models. Hall et al. (1997) studied minimizing maximum flow time on parallel machines and highlighted the inherent difficulty of achieving fairness without delaying service. Chekuri et al. (2001) further investigated flow-time minimization with release times and showed that even small relaxations in model assumptions can drastically change achievable guarantees. These results emphasize that flow time behaves fundamentally differently from completion time or makespan.

More recent work has explored resource augmentation as a means to overcome impossibility results for flow-time objectives. Phillips et al. (1997) and Kalyanasundaram and Pruhs (2000) demonstrated that modest speed augmentation can yield strong guarantees for maximum flow time, suggesting that exact competitiveness may be unattainable without additional power. These insights strongly influence later work in routing and delivery, where spatial constraints further complicate flow-time optimization.

2.2 Online Routing and Vehicle Routing Problems

Vehicle routing problems have a long history in operations research and theoretical computer science. In offline settings, classical problems such as the Traveling Salesman Problem (TSP), the Vehicle Routing Problem (VRP), and the Capacitated Vehicle Routing Problem (CVRP) have been studied extensively, leading to a rich body of approximation algorithms and heuristics. Foundational surveys and algorithmic treatments of these problems can be found in the work of Toth and Vigo (2002) and Laporte (2009), which highlight the algorithmic challenges introduced by capacity constraints, routing costs, and combinatorial structure.

Online variants of routing problems introduce additional complexity due to the lack of future information. Early work on online routing focused primarily on minimizing total distance, makespan, or completion time. Ausiello et al. (2001) studied online versions of the Traveling Salesman Problem and established competitive bounds under various assumptions on the metric space. Similarly, Irani et al. (2001) analyzed online routing problems with time-dependent requests, showing that constant competitive ratios are achievable for distance-based objectives in restricted settings.

The k-server problem provides a closely related abstraction for online routing in metric spaces. The seminal work of Manasse, McGeoch, and Sleator (1990) introduced the k-server problem and demonstrated its fundamental role in online algorithm design. Subsequent work by Koutsoupias and Papadimitriou (1995) established tight competitive bounds for general metrics, while Chrobak and Larmore (1991) and Hammar and Nilsson (2002) showed that restricting the metric to trees or paths enables significantly stronger guarantees. These results underscore the importance of structured metric spaces in obtaining meaningful competitive analysis for online routing problems.

Despite these advances, most online routing results focus on objectives such as total distance traveled or completion time. Flow-time objectives, particularly minimizing the maximum flow time, are considerably more challenging due to their sensitivity to waiting decisions and batching effects. In scheduling settings, flow time has been studied extensively as a fairness-oriented objective, but incorporating spatial constraints and routing decisions substantially complicates the analysis. As shown by Bansal and Pruhs (2010), flow-time minimization often requires fundamentally different algorithmic techniques compared to distance-based objectives.

The online food delivery problem can be viewed as a special case of the Dial-a-Ride Problem, where all requests originate from a common depot and only delivery locations vary. This structural simplification eliminates pickup-routing interactions and allows the analysis to focus on the interaction between waiting, routing, and capacity constraints.

Dial-a-ride problems have been studied extensively in offline settings, but their online counterparts remain relatively underexplored, especially under flow-time objectives.

Recent work has begun to address these gaps by studying online food delivery and related problems on structured metric spaces. Guo, Kesselheim, and Radke (2021) showed that minimizing maximum flow time is intractable in general metrics, even for a single uncapacitated vehicle, and identified tree metrics as a tractable class. In subsequent work, Guo, Kesselheim, and Radke (2023) analyzed the problem on star graphs and demonstrated that simple FIFO-based strategies are suboptimal, motivating anticipation-based algorithms such as Earliest Anticipation First. These results highlight both the necessity of restricting the metric space and the limitations of purely myopic routing strategies.

Overall, prior work indicates that while online routing with distance-based objectives is well understood, online routing under flow-time objectives—particularly in the presence of capacity constraints—remains poorly characterized. Existing results either assume unit capacity, uncapacitated vehicles, or focus on specific metric structures. This motivates a systematic study of how capacity constraints affect the competitiveness of online algorithms for minimizing maximum flow time, particularly on structured metric spaces such as path graphs.

In addition to TSP-style formulations, online routing has been studied through variants of the Dial-a-Ride and Pickup-and-Delivery problems. Psaraftis (1988) provided early formulations of dynamic dial-a-ride problems, identifying the tradeoff between waiting for future requests and serving current demand. Mitrovic-Minic and Laporte (2004) later proposed dynamic routing strategies for real-time transportation systems, though without worst-case guarantees on flow time.

Theoretical work on online vehicle routing has also examined lower bounds and structural limitations. Blum et al. (1994) studied online routing under adversarial arrivals and showed that unrestricted metrics lead to poor competitive performance. As a result, subsequent work increasingly focused on tree, line, and star metrics, where geometry can be

exploited algorithmically. These findings reinforce the importance of structured metric spaces in enabling meaningful competitive analysis.

Additional work on online routing has explored alternative problem formulations and competitive limits under adversarial arrivals. Bartal et al. (2001) introduced probabilistic metric embeddings to analyze online routing problems, showing that general metrics can be reduced to tree metrics at the cost of logarithmic distortion. While this technique enables stronger guarantees for distance-based objectives, it remains insufficient for controlling worst-case flow time, reinforcing the need for problem-specific structural assumptions.

Online routing has also been studied under deadline and time-window constraints. Bansal et al. (2004) analyzed online scheduling and routing with deadlines, demonstrating that feasibility and delay guarantees often require rejecting or postponing requests. Such results highlight the intrinsic tension between immediacy and fairness in online routing systems.

Several works investigate online routing with multiple vehicles or servers. Feuerstein and Stougie (2001) studied online vehicle routing with multiple servers and showed that coordination between vehicles is necessary to avoid poor competitive performance. Similarly, Fagin and Raghavan (1996) analyzed online routing with parallel servers and established lower bounds that persist even when multiple vehicles are available.

More recent work has examined online routing under resource augmentation and speed scaling. Chan et al. (2012) studied online routing with speed augmentation and showed that modest increases in speed can significantly improve competitive ratios. These results parallel similar findings in scheduling and suggest that augmentation-based models may be necessary to overcome impossibility results in capacitated routing.

Finally, online routing problems have been connected to real-time transportation systems and dynamic logistics. Ichoua et al. (2006) studied dynamic vehicle routing with real-time information, identifying batching and delayed dispatch as essential mechanisms for stabilizing system performance. Although these models focus primarily on empirical

and heuristic performance, they provide further evidence that immediate service is often suboptimal in dynamic routing environments.

2.3 Online Food Delivery on Structured Graphs

Recent work has studied the online food delivery problem on structured metric spaces like star graphs. These settings capture important geometric constraints while remaining analytically tractable. Prior results show that for uncapacitated or unit-capacity servers, simple online algorithms can achieve constant competitive ratios for minimizing maximum flow time.

A representative example of this line of work is the study by Guo et al. [21], which formalizes the Online Food Delivery Problem (OFDP) with the objective of minimizing the maximum flow time under online request arrivals. The authors study the problem on star graphs, where all delivery locations are leaves connected to a central depot, capturing a restaurant-centric delivery structure while remaining analytically tractable. Under this setting, they show that simple FIFO-based strategies are suboptimal and instead analyze anticipation-based algorithms that prioritize requests using the sum of release time and travel distance.

In particular, they prove that the Earliest Anticipation First (EAF) algorithm achieves a tight competitive ratio of 3 among all myopic algorithms. By allowing limited waiting through a far-sighted strategy, they further propose the Postpone and Anticipation First (PAF) algorithm, which improves the competitive ratio to $8/3$. These results demonstrate that even limited anticipation and controlled waiting can significantly improve worst-case flow-time performance in structured metric spaces such as star graphs.

Guo et al. [20] significantly extend the study of the Online Food Delivery Problem by considering general tree metrics, multiple vehicles, and both capacitated and uncapacitated settings. The paper shows that minimizing maximum flow time is inherently hard in general

metrics: even for a single uncapacitated vehicle, the offline problem is NP-hard to approximate and no $o(n)$-competitive online algorithm exists. These strong lower bounds highlight that the maximum flow-time objective is substantially more challenging than makespan or total travel distance objectives commonly studied in vehicle routing.

To overcome these negative results, the authors identify tree metrics as a tractable class and design an $O(1)$-competitive online algorithm for the uncapacitated problem on trees. Their approach groups requests arriving within bounded time windows into carefully constructed bundles, which are then served using subtree traversals with controlled backlog. The paper further shows that finite vehicle capacity fundamentally alters the problem: even on small trees, no constant-competitive algorithm exists without additional assumptions. To address this, the authors analyze a speed-augmentation model and prove near-tight tradeoffs between vehicle speed and competitive ratio. These results establish a sharp boundary between tractable and intractable regimes for online food delivery under maximum flow-time objectives and motivate the focus on structured metrics and relaxed models in subsequent work.

Structured metrics such as lines, cycles, and trees have received particular attention due to their relevance in transportation and logistics. Sitters (2004) studied online routing on the line and demonstrated that batching decisions are unavoidable for fairness-oriented objectives. Englert and Röglin (2017) further examined online routing on paths and trees, showing that delaying service can be provably necessary to control maximum delay.

Recent work has also explored stochastic and hybrid models. Jaillet and Wagner (2008) studied online vehicle routing under stochastic arrivals, highlighting qualitative similarities with adversarial flow-time models. Although these results focus on expected performance rather than worst-case guarantees, they provide additional motivation for understanding how anticipation and batching affect delay-sensitive objectives.

2.4 Capacity Constraints and Batching

Capacity constraints fundamentally alter the behavior of online delivery systems. While batching requests can reduce travel overhead, it can also increase waiting times for individual requests. In offline settings, batching under capacity constraints is well understood through CVRP formulations, but online batching remains much less explored.

Capacity constraints introduce an additional layer of complexity by coupling routing decisions across requests. As shown by Ascheuer et al. (2000), batching decisions in capacitated routing can dramatically alter optimal schedules even in offline settings. In online models, these effects are amplified: serving requests greedily may lead to unbounded delays when capacity exceeds one.

Related insights arise from online load balancing and batching problems. Azar et al. (1999) studied online batching under capacity constraints and showed that myopic strategies can fail catastrophically under adversarial arrivals. These results suggest that explicit waiting and coordinated batching are often necessary to control worst-case flow time.

Beyond routing, similar phenomena have been observed in online scheduling with batch processing. Coffman et al. (1996) studied batch scheduling models in which jobs can be processed together, showing that batching introduces a fundamental tradeoff between throughput and delay. Their results highlight that naive batching rules may significantly increase flow time, motivating more carefully controlled delay mechanisms.

Impossibility results for greedy service also appear in queueing-inspired online models. Bertsimas and van Ryzin (1991) analyzed dynamic vehicle routing with stochastic arrivals and showed that immediate dispatch policies can be suboptimal even under probabilistic assumptions, as delaying service allows the system to exploit spatial clustering. Although their analysis is stochastic rather than adversarial, it reinforces the importance of batching for stabilizing delay.

From a worst-case perspective, Megow et al. (2012) studied online scheduling with

commitment and rejection and showed that delaying acceptance decisions is often unavoidable to achieve bounded flow-time guarantees. These insights parallel the need for delayed dispatch in capacitated routing, where committing too early to serve requests can block future batching opportunities.

Additional lower bounds arise from online admission control and congestion models. Andrews et al. (2001) showed that without controlled waiting or rejection, online algorithms with capacity constraints suffer unbounded delay under adversarial arrivals. While their work focuses on network congestion, the underlying mechanism—capacity-induced coupling across requests—closely mirrors challenges in online delivery.

Recent work has also examined batching through the lens of competitive queueing theory. Leonardi and Raz (2014) studied online service systems with batching and demonstrated that bounded waiting often requires coordinated service policies that intentionally delay individual jobs. These results provide further evidence that immediate service is incompatible with fairness-oriented objectives under capacity constraints.

Finally, online matching with delays provides a related abstraction. Akbarpour et al. (2020) studied dynamic matching markets and showed that allowing limited waiting dramatically improves performance and stability. Although their setting differs from routing, the core insight—that strategic delay enables better global outcomes under capacity constraints—directly informs batching strategies in online delivery.

Together, these results indicate that capacity constraints fundamentally change the nature of online decision-making. Simple greedy or myopic strategies are insufficient once capacity exceeds one, and far-sighted algorithms that explicitly delay service and coordinate batching become necessary. These observations motivate the focus of this thesis on capacity-aware online algorithms with provable guarantees on maximum flow time.

Some prior work considers delayed dispatch or batching heuristics, but often without rigorous guarantees on maximum flow time. In particular, it is unclear whether simple myopic strategies remain effective once capacity exceeds one, or whether far-sighted strategies

Table 2.1: Summary of prior work related to online routing, food delivery, and flow-time objectives

Reference	Problem Setting	Metric / Model	Main Result
Ausiello et al. (2001)	Online TSP	General metrics	Constant competitive ratios for distance-based objectives
Manasse et al. (1990)	k-server problem	General metrics	Introduced competitive framework for online routing
Koutsoupias and Papadimitriou (1995)	k-server problem	General metrics	Tight competitive bounds
Hammar and Nilsson (2002)	k-server problem	Tree metrics	Improved competitiveness on structured metrics
Bansal and Pruhs (2010)	Online scheduling	Machine models	Flow-time minimization requires different techniques
Guo et al. (2021)	Online food delivery	Tree metrics	$O(1)$-competitive algorithm; hardness in general metrics
Guo et al. (2022)	Online food delivery	Star graphs	FIFO suboptimal; EAF achieves tight competitive ratio
Guo et al. (2023)	Online food delivery	Star graphs	Far-sighted strategies improve flow-time bounds

that explicitly delay service are necessary.

These questions motivate a closer examination of capacity-aware online algorithms and their limitations.

2.5 Positioning of This Work

This thesis builds on prior work in online food delivery and flow-time minimization, while addressing gaps related to capacity constraints. Unlike earlier studies that focus primarily on unit-capacity servers, we analyze how competitiveness changes as capacity increases. We provide both impossibility results for myopic algorithms and constructive guarantees

for far-sighted strategies.

By focusing on structured metric spaces, we obtain sharp competitive bounds that clarify when anticipation and batching are unavoidable. Our results complement existing work on online routing and contribute new insights into the design of real-time delivery algorithms under capacity limitations.

Chapter 3

Problem Formulation and Algorithm Framework

In this chapter, we formally describe the online food delivery problem studied in this thesis. We introduce the underlying metric space, define requests, flow time, capacity constraints, and feasible schedules, and specify the algorithmic framework used throughout the analysis. These definitions form the basis for the results presented in the subsequent chapters.

3.1 Problem Notation

Let (V, d) be a metric space, where V denotes the set of delivery locations and $d(u, v)$ represents the distance between locations u and v. We assume $|V| = n$, where n is the number of delivery locations. A distinguished node $o \in V$ represents the depot from which the server originates and to which it must return after completing deliveries

We consider a single server that operates at unit speed. The server has a fixed capacity $K \in \mathbb{Z}_{>0}$, meaning it can carry at most K requests in a single trip. This thesis considers only one server and it has a unit speed.

3.2 Online and Offline Solutions

A delivery request is denoted by $\rho = (r_\rho, v_\rho)$, where $r_\rho \in \mathbb{Z}_{>0}$ is the release time of the request and $v_\rho \in V$ is the delivery location. Requests arrive online over time, and the server must make dispatch decisions without knowledge of future arrivals.

If request ρ is delivered at time t_ρ, then its *flow time* is defined as

$$F_\rho = t_\rho - r_\rho.$$

The objective of the online food delivery problem is to minimize the maximum flow time over all requests.

3.3 Algorithm Framework

The output of an online algorithm is a sequence of *trips*. A trip is defined as a tuple

$$(t, (o, v_1, v_2, \ldots, o), R'),$$

where t is the departure time of the trip, $(o, v_1, v_2, \ldots, o)$ is the route taken by the server starting and ending at the depot, and $R' \subseteq R$ is the set of requests served in the trip.

Each trip must satisfy:

- For every request $\rho \in R'$, $t > r_\rho$.
- $|R'| \leq K$.
- Once a trip begins, it must be completed without interruption.

Trips are executed sequentially. If C_{k-1} is the completion time of trip $k-1$, then the departure time t_k of trip k must satisfy $t_k > C_{k-1}$.

If a request ρ is served in a trip departing at time O_ρ, then its completion time is

$$t_\rho = O_\rho + d(o, v_\rho),$$

and its flow time is

$$F_\rho = O_\rho + d(o, v_\rho) - r_\rho.$$

3.4 Structured Graphs

This thesis focuses on structured metric spaces that allow precise competitive analysis.

3.4.1 Path Graphs

In the path graph setting, the metric space forms a path with the depot located at left end. The server must travel along the path to serve requests and return to the depot.

3.5 Online and Offline Solutions

An *offline optimal solution* has full knowledge of the entire request sequence in advance and can schedule trips to minimize the maximum flow time. Let F^* denote the maximum flow time achieved by such an optimal offline algorithm.

An online algorithm produces a schedule without future knowledge and achieves maximum flow time F. Performance is measured using competitive analysis. An online algorithm is α-competitive if

$$F \leq \alpha \cdot F^*,$$

for all input instances.

3.6 Algorithmic Framework

We study two broad classes of online algorithms.

3.6.1 Myopic Algorithms

Myopic algorithms dispatch the server immediately whenever there are pending requests at the depot, without waiting for future arrivals. We consider the following myopic dispatch policies:

- *First-In-First-Out (FIFO)*, which serves the request with the *earliest release time*. That is, among all available requests, FIFO prioritizes the oldest request.
- *Earliest Release First (ERF)*, which serves the request with the *latest release time*. In contrast to FIFO, ERF prioritizes the most recently released request and is therefore the exact opposite of FIFO.
- *Earliest Anticipation First (EAF)*, which serves the request with the smallest anticipation time.

For a request $\rho = (r_\rho, v_\rho)$, the anticipation time is defined as

$$r_\rho + d(o, v_\rho),$$

where r_ρ is the release time of the request and $d(o, v_\rho)$ denotes the distance from the depot o to the delivery location v_ρ.

3.6.2 Far-Sighted Algorithms

Far-sighted algorithms intentionally delay service in order to batch requests and better utilize capacity. We guess the optimal maximum flow time (F^* and divide the time into intervals of F^* and group requests from consecutive intervals.

Chapter 4

Algorithm Analysis

This chapter presents the main theoretical results of the thesis. We analyze online algorithms for minimizing the maximum flow time under capacity constraints on structured metric spaces. We begin with myopic algorithms on path graphs for small capacities, establish impossibility results, and then introduce a far-sighted algorithm that achieves bounded competitive ratios for general capacity.

4.1 Path Graphs with Capacity One

We begin by analyzing myopic algorithms on path graphs when the server has unit capacity.

4.1.1 FIFO Algorithm

We first establish a structural lemma for the offline optimal solution.

Lemma 1. *If request ρ_L has the maximum flow time in the optimal offline solution, then all requests released before ρ_L are served before it.*

Proof. Assume for contradiction that ρ_L has the maximum flow time in the optimal solution, but there exists a request ρ_i with $i < L$ that is not served before ρ_L.

Let r_i, d_i denote the release time and distance of request ρ_i, and let O_i^* denote the departure time of the trip serving ρ_i in the optimal solution. Since ρ_i is served after ρ_L, its flow time satisfies

$$F_i^* \geq O_i^* + 2d_L + d_i - r_i.$$

On the other hand, the flow time of ρ_L is

$$F_L^* = O_L^* + d_L - r_L.$$

Since $r_L \geq r_i$, we obtain

$$F_i^* \geq F_L^* + d_L + d_i + r_L - r_i > F_L^*,$$

which contradicts the assumption that ρ_L has the maximum flow time. Therefore, all requests released before ρ_L must be served before ρ_L in the optimal solution. □

Theorem 1. *For path graphs with capacity one, the FIFO algorithm is 2-competitive.*

Proof. In order to simplify the analysis, we restrict attention to work-conserving instances. That is, we assume that whenever the vehicle returns to the depot, there exists at least one pending request, and hence the vehicle departs immediately without idling. We divide the proof into 4 subparts.

Let ‘l‘ be the maximum flow time in **OPT**.

Let ‘i‘ be the maximum flow time in **ALG**.

If there are a total of K requests, then we can divide the proof into 4 cases and use mathematical induction to show that:

$$\frac{F}{F^*} < 2$$

The four cases are:

- $l = K,\ i = K$
- $l = K,\ i < K$
- $l < K,\ i = K$
- $l < K,\ i < K$

We will use mathematical induction to prove this. We will assume that our algorithm is true for $K - 1$ and try to prove it for K.

1. For $l < K, i < K$: We can use induction to prove that our theorem holds.

2. $l < K, i = K$:

$$F_l^* = O_l^* + d_l - t_l$$

$$F_l^* = O_l^* + d_l - t_l \quad \text{(all requests that came before } l \text{ are served)}$$

$$F_l^* = O_l^* + 2\sum_{i=1}^{l-1} d_i + d_l - t_l$$

Now, if there was no request that was served whose $t_i > l$, then the above F_l^* is the same for myopic. But our assumption is that f_k is the max and not F_l^*. So, $\exists$ at least one request such that $t_{i'} > t_l$ has been served.

$$F_l^* \geq t_{l+1} + 2\sum_{i=1}^{l-1} d_i + d_l - t_l$$

$$F_l^* \geq 2\sum_{i=1}^{l-1} d_i + d_l + (t_{l+1} - t_l) \quad \text{(+ve)}$$

$$F_l^* \geq 2\sum_{i=1}^{l-1} d_i + d_l \geq 2\sum_{i=1}^{l} d_i \tag{1}$$

Below is the flow time for our myopic algorithm for case 2

$$F_k = O_k + d_k - t_k$$

$$F_k < O_k + d_k$$

$$F_k < 2\sum_{i=1}^{k-1} d_i + d_k$$

$$F_k < 2\sum_{i=1}^{k} d_i \tag{3}$$

Suppose x^*, is the last req that is served in the optimal.

$$F_x^* = O_1^* + \sum_{i=1}^{l-1} 2d_i + d_l + d_l + \sum_{i=l+1, i\neq x}^{k} 2d_i + d_x - t_x$$

$$F_l^* = O_1^* + \sum_{i=1}^{l-1} 2d_i + d_l - t_l$$

$$F_x^* = F_l^* + d_l + \sum_{i=l+1, i\neq n}^{k} 2d_i + d_x - t_x + t_l$$

We know that

$$F_x^* < F_l^*, F_x^* \geq 0$$

$$F_l \geq d_l + 2\sum_{l+1, i\neq x} d_i + d_x + t_l - t_x$$

In the above $t_l - t_x$ is negative

$$F_l^* \geq 2\sum_{l}^{k} d_i \tag{2}$$

Using the above 1,2, 3 we get

$$F_k < 2\sum_{1}^{l} d_i + 2\sum_{l+1}^{k} d_i$$

$$F_k < F_l^* + F_l^*$$

$$F_k < 2F_l^*$$

Case 3 :l = K, $i < K$

Because of the assumption we know that

$$\frac{F_i}{F_l^*} < 2$$

The above is true for all $l < K$ and $i < K$. So it will also be true for $l = k$ and $i < K$ because $F_K^* \geq F_i^*$ where $i < K$. Case 4: l = K, $i = K$ We now compare both sides against the same quantity.

- **ALG:**

$$F_K \leq 2\sum_{i=1}^{K} d_i.$$

- **OPT:**

$$F_l^* \geq 2\sum_{i=1}^{K} d_i.$$

Therefore,

$$F_K \leq 2F_K^*.$$

□

4.1.2 EAF Algorithm

For EAF and ERF we use standard simplification rules to make the proof easier.

Instance Simplification

We make the following assumptions on the input instance to simplify the proof. Without loss of generality we only need to focus on these kinds of input instance.

- The maximum flow time is obtained at the last served request: $\max F_\rho = F_{\rho_n}$
- EAF serves one request at each time.

We use the legal definition and the properties from paper[1].The definitions and properites are listed below

Definition*(Legal Pair)* We say an ordered request pair (a, b), where $O_a \leq O_b$, is legal if:

$$r_a + d_a \leq r_b + d_b$$

Properties of Legal Pairs:

- In EAF, if $O_a < O_b$ and $r_b < O_a$, then $r_a + d_a \leq r_b + d_b$ and (a, b) is legal.
- **(Reward Property):** If (a, b) is legal, then we have:

$$F_b \leq F_a^* + C_b^* - C_a^*$$

Let C andC^* be the completion time of the online algorithm and the optimal algorithm. Then we have the following

Lemma: $C - C^* \leq F^*$

The OPT starts at the same time or later than the departure time of ALG.

Assume (t, t') be the last idle time made by EAF before its completion time.

After t', EAF finds at least one request waiting when it returns to the depot, so it departs immediately.

EAF after t' serves requests $\rho_{k+1}, \rho_{k+2}, \ldots, \rho_n$

Let $S := \{\rho_i : i > k\}$

$$S := \{\rho_i : r_i \geq t'\}$$

$$C = t' + \sum_{\rho \in S} (C_i - C_{i+1})$$

$$C = t' + 2 \cdot \sum_{\rho \in S} d_i + d_{\rho_n}$$

For OPT, we cannot start before t'

$$C^* \geq t' + 2 \cdot \sum_{\rho \in \mathrm{S}} d_i + d_{\rho^*}$$

$$-C^* \leq -\left(t' + 2 \cdot \sum_{\rho \in \mathrm{S}} d_i + d_{\rho^*} \right)$$

$$C - C^* \leq d_{\rho_n} - d_{\rho^*} < F^*$$

Theorem 2. *For path graphs with single capacity the EAF algorithm is 3 competitive*

For EAF we will divide the pairs into legal and non-legal pair and find the competitive ratio for both the legal and non-legal pair.

Legal Pair

Lemma 2. *For legal pairs in EAF, the competitive ratio is* $2F^*$.

Proof. We use the reward property to prove this.

Reward property is : $F_b \leq F_a^* + C_b^* - C_a^*$

$$F \leq F^* + F^*$$

$$F \leq 2F^*$$

□

Non-Legal Pairs

Lemma 3. *For non-legal pairs, the competitive ratio is 3.*

Proof. (ρ^*, ρ) is not a legal pair. So $r_{\rho^*} + d_{\rho^*} > r_{\rho_n} + d_{\rho_n}$.

$O_{\rho^*} < O_{\rho_n}$ because ρ_n is the last request served in EAF.

Claim: $r_{\rho^*} < r_{\rho_n}$ and $O_{\rho^*} < r_{\rho_n}$

Reason: If not, then because of EAF, ρ_n should have been served first.

We have:

$$C_{\rho_n} - C_{\rho^*} \leq F^*$$

$$C_{\rho_n} \leq C_{\rho^*} + F^*$$

$$C_{\rho_n} \leq 2F^* + d_{\rho^*} + r_{\rho^*}$$

Now, using:

$$F_{\rho_n} = C_{\rho_n} - d_{\rho_n} - r_{\rho_n}$$

(substituting this above)

$$F_{\rho_n} \leq 2F_{\rho^*} + d_{\rho^*} + r_{\rho^*} - d_{\rho_n} - r_{\rho_n}$$

$$F_{\rho_n} \leq 2F_{\rho^*} + F_{\rho^*} \quad \text{(because } r_{\rho^*} > r_{\rho_n} \text{ and } d_i < F_i\text{)}$$

$$F \leq 3F^*$$

4.1.3 ERF Algorithm

Lemma 4. $C - C^* \leq F^*$

We can use this lemma for all the cases with capacity 1 as in the above proof we don't use the algorithm but only the condition that the capacity is 1. □

Theorem 3. *For path graphs with single capacity ERF algorithm is 4 competitive*

Proof. **(a) Maximum Flow Time:**

The bound on the completion time doesn't depend on the algorithm:

$$C - C^* \leq F^* \quad \text{for cap = 1}$$

After t_1

$$C = O_\alpha + 2d_\alpha + 2\sum_{\rho \in \alpha} d_\rho$$

$$C^* \geq O_\alpha + 2\sum_{\alpha'} d_\rho$$

$$F^*_\rho = C^*_\rho - d_\rho - r^*_\rho$$

$$C^* - r_{\rho *} = F^*_\rho + d_{\rho^*} \leq 2F^*$$

r_{ρ_*} is at most r_{ρ_n} because ρ_n is lastly served in ERF.

$$C^* - r_{\rho_n} \leq C^* - r_{\rho^*} \leq 2F^*$$

$$F_{\rho_n} = C_{\rho_n} - d_{\rho_n} - r_{\rho_n} \leq 2F^* + C^*_{\rho} - d_{\rho_n} - r_{\rho_n} \leq 2F^* + 2F^* - d_{\rho} \leq 4F^*$$

$$F \leq 4F^*$$

□

4.2 Path Graphs with Capacity Two

We now show that increasing capacity fundamentally changes the performance of myopic algorithms.

Theorem 4. *For path graphs with capacity two, no myopic FIFO-based algorithm admits a bounded competitive ratio.*

Proof. We prove this by giving example of an instance where the competitive ratio is unbounded.

$$[0, T][0, 1][T, T][1, T] \ldots \text{(n such requests)}$$

FIFO: When we follow the FIFO algorithm the total completion time it takes to complete the requests is $2nT$ Total completion time = $2nT$

$$\text{Flow Time}_{\max} = 2nT - T - (n-1)T = nT$$

OPT: In optimal case we can wait for time T and then club the same requests together. Then the total completion time will be $n + nT + T$

Total completion time = $T + \left\lceil \frac{n}{2} \cdot T + \frac{n}{2} \right\rceil \cdot 2 = T + nT + n$

$$\text{Flow Time}_{\max} = T + nT + n - T - (n-1)T = n + T$$

$$\text{Competitive ratio} = \frac{nT}{n+T}$$

If $n = T$, then

$$\text{Competitive ratio} = \frac{n}{2} \rightarrow \textbf{unbounded}$$

□

This result demonstrates that myopic strategies that perform well for unit capacity fail once capacity exceeds one as capacity is not properly utilised as they cannot wait and group requests together.

4.3 Far-Sighted Algorithm for General Capacity

To handle larger capacities, we introduce a far-sighted batching algorithm. The above ERF and EAF algorithms work for single capacity and are not generalizable when we increase capacity, and as such we use a far-sighted algorithm to find a bounded competitive ratio. We propose a far sighted algoritm in which we assume that we know the optimal maximum flow time which is F^*. We will divide our time range into intervals of F like $[(i-1)F, iF]$ where $i \in [0, \infty]$ and we call the interval $[(i-1)F, iF]$ as iF. The requests that arrive in the interval iF will be defined as R_i and will be served at the beginning of $(i+2)F$. To better use the capacity that the server has we will try to batch the requests in the interval with some of the requests in the adjacent intervals we will call them R_i^{left} and R_i^{right}.

Let's define $R_i = \{\rho \in R : (i-1)F \leq r_\rho \leq iF\}$. We bundle request together and

generate $R_i{}'$ for R_i which bundles requests R_i, R_i^{left} and R_i^{right}. We will generate the bundles for odd $i's$. The cost function that we will use for this algorithm is defined as

$$C_K(S) = \sum_{j=0} d_{n-jK}$$

We will sort the elements in S in decreasing order and we will use the cost function to partition the requests in R_{i-1} and R_{i+1} and to bundle them and create $R_i^{'}$.

Below figure illustrates the high-level structure of the far-sighted batching algorithm. Requests arriving in interval I_i are not necessarily served immediately; instead, they may be bundled with carefully chosen requests from neighboring intervals. This controlled delay enables more efficient use of server capacity while ensuring that the maximum flow time remains bounded.

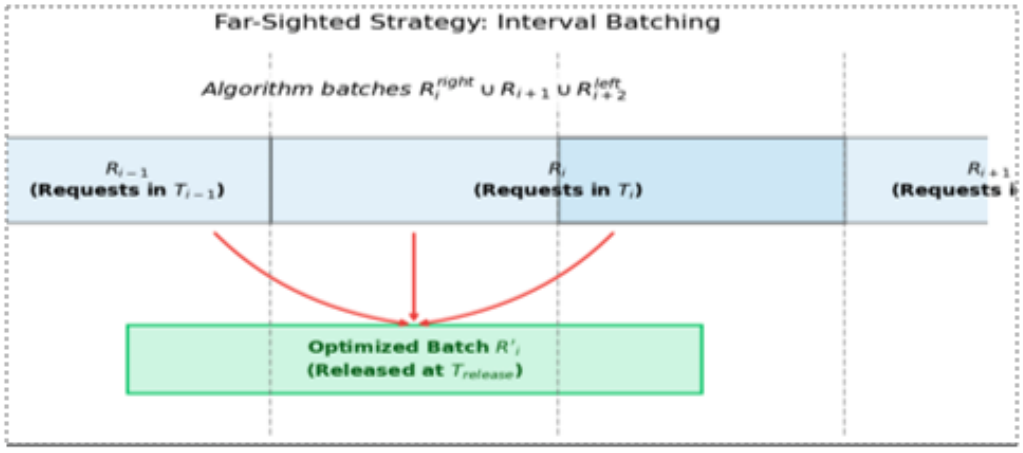

Figure 4.1: Far-sighted interval batching strategy for general capacity. Requests arriving in interval I_i are batched together with selected requests from adjacent intervals to better utilize server capacity while controlling maximum flow time.

We partition R_i into $(R_i^{left}, R_i^{right})$ such that we minimise $C_k(R_i^{\text{left}} \cup R_{i-1}) + C_k(R_i^{\text{right}} \cup R_{i+1})$. For every request in $\rho \in R_i$ the algorithm will check if $C_k(R_{i-1} \cup \rho) < C_k(R_{i+1} \cup \rho)$ and if it is true then the request $\rho \in R_i^{left}$ else $\rho \in R_i^{right}$.

For every trip the server will need to deliver all the requests and for this it will travel to

Algorithm 1 Partition of Requests into Bundles

1: **for** $i = 2, 4, 6, 8, \ldots$ **do**
2: wait until time $(i+1)F$
3: let $R_i = \{\rho \in R : r_\rho \in [(i-1)F, iF)\}$ as defined in the text
4: partition R_i into R_i^{left} and R_i^{right} so as to minimize $C_k(R_i^{\text{left}} \cup R_{i-1}) + C_k(R_i^{\text{right}} \cup R_{i+1})$
5: **end for**
6: release the bundle $R'_{i-1} := R_{i-2}^{\text{right}} \cup R_{i-1} \cup R_i^{\text{left}}$
7: at time $(i+1)F$

the furthest distance among the batch of request and then it will also need to return to the depot so it travels a distance of $2d_{\rho_{max}}$.

We will first prove the theorem for capacity 2 and then generalise it for capacity K. We need the following lemmas to prove our theorem.

Lemma 5.

$$C_2(S) = \int_0^\infty \left\lceil \frac{n_S(t)}{2} \right\rceil \, dt$$

S is sorted in non-increasing order

$$S = \{s_1 \geq s_2 \geq \ldots\}$$

Proof. Let's define

$$k = n_S(t) = \#\{x \in S : x \geq t\}.$$

for a threshold t.

Since the sequence is non-increasing,

$$x_j \geq t \quad \Rightarrow \quad j \leq k.$$

Thus,

$$\sum_{\substack{j \\ \text{odd}}} \mathbf{1}\{x_j \geq t\} = \sum_{\substack{j \\ \text{odd}}} \mathbf{1}\{j \leq k\} = \#\{\text{odd } j \in \{1, 2, \ldots, k\}\}.$$

Now, how many odd integers are $\leq k$?

$$\begin{cases} k = 2q \quad \Rightarrow \quad \#\{\text{odd } j \leq k\} = q = \frac{k}{2}, \\ k = 2q + 1 \quad \Rightarrow \quad \#\{\text{odd } j \leq k\} = q + 1 = \frac{k+1}{2}. \end{cases}$$

Therefore, in both cases,

$$\#\{\text{odd } j \leq k\} = \left\lceil \frac{k}{2} \right\rceil.$$

Hence,

$$\sum_{\substack{j \\ \text{odd}}} \mathbf{1}\{x_j \geq t\} = \left\lceil \frac{n_S(t)}{2} \right\rceil.$$

$$\Longrightarrow \; C_2(S) = \int_0^\infty \left\lceil \frac{n_S(t)}{2} \right\rceil \, dt$$

□

Lemma 6. *For any subsets A, B, C:*

$$C_2(A \cup B \cup C) \leq C_2(A \cup B) + C_2(B \cup C) - C_K(B) + \Delta_K$$

where $\Delta_K = \min(\max A, \max B, \max C)$

Proof. For any $t \geq 0$, let

$$a = n_A(t), \quad b = n_B(t), \quad c = n_C(t).$$

We want to prove:

$$\left\lceil \frac{a+b+c}{2} \right\rceil \leq \left\lceil \frac{a+b}{2} \right\rceil + \left\lceil \frac{b+c}{2} \right\rceil - \left\lceil \frac{b}{2} \right\rceil + \mathbf{1}\{t \leq \Delta\}.$$

Let

$$a = 2x + \alpha, \quad b = 2y + \beta, \quad c = 2z + \gamma, \quad (\alpha, \beta, \gamma) \in \{0, 1\}.$$

Then we will have

$$\left\lceil \frac{2k+\epsilon}{2} \right\rceil = k + \epsilon.$$

Let's define $q(t)$ as below

$$q(t) := \left\lceil \frac{a+b+c}{2} \right\rceil - \left\lceil \frac{a+b}{2} \right\rceil - \left\lceil \frac{b+c}{2} \right\rceil + \left\lceil \frac{b}{2} \right\rceil.$$

We have

$$q(t) = \left\lceil \frac{\alpha+\beta+\gamma}{2} \right\rceil - \left\lceil \frac{\alpha+\beta}{2} \right\rceil - \left\lceil \frac{\beta+\gamma}{2} \right\rceil + \left\lceil \frac{\beta}{2} \right\rceil.$$

If $\beta = 0$ (i.e. b is even), then

$$q(t) = \left\lceil \frac{\alpha+\gamma}{2} \right\rceil - \left\lceil \frac{\alpha}{2} \right\rceil - \left\lceil \frac{\gamma}{2} \right\rceil.$$

Thus,

$$q(t) \in \{0, -1\} \leq 0$$

If $\beta = 1$ (i.e. b is odd), then

$$q(t) \in \{0, 1\}.$$

Explicitly,

$$q(t) = \left\lceil \frac{\alpha + \gamma + 1}{2} \right\rceil - \left\lceil \frac{\alpha + 1}{2} \right\rceil - \left\lceil \frac{\gamma + 1}{2} \right\rceil + 1.$$

If $\alpha = \gamma = 1$ (so a, b, c are all odd), then

$$q(t) = 1.$$

Therefore:

$$q(t) \leq 0 \quad \text{if } b \text{ is even,}$$

and

$$q(t) = 1 \quad \text{if } a, b, c \text{ are odd.}$$

So,

$$\left\lceil \frac{a + b + c}{2} \right\rceil \leq \left\lceil \frac{a + b}{2} \right\rceil + \left\lceil \frac{b + c}{2} \right\rceil - \left\lceil \frac{b}{2} \right\rceil$$

The above holds if for every $t > 0$ there does not exist a t such that $b(t) = n_B(t)$ is odd, $a(t) = n_A(t)$ is odd, and $c(t) = n_C(t)$ is also odd.

$\implies$ when all three are odd, $q(t)$ requires one extra trip.

So when a, b, c are odd, how do we bound the extra trip?

We know

$$q(t) = 1 \quad \text{when all } a, b, c \text{ are odd.}$$

The threshold *t* is

$$t \leq \max(A), \quad t \leq \max(B), \quad t \leq \max(C).$$

Therefore,

$$t \leq \min\big(\max A, \max B, \max C\big).$$

We consider

$$\int_0^\infty \mathbf{1}\{t \leq \min(\max A, \max B, \max C)\}\, dt = \min(\max A, \max B, \max C).$$

Thus,

$$\left\lceil \frac{a+b+c}{2} \right\rceil - \left\lceil \frac{a+b}{2} \right\rceil - \left\lceil \frac{b+c}{2} \right\rceil + \left\lceil \frac{b}{2} \right\rceil - \mathbf{1}\{t \leq \min(\max A, \max B, \max C)\} \leq 0.$$

Integrating over t, we obtain

$$C_2(A \cup B \cup C) - C_2(A \cup B) - C_2(B \cup C) + C_2(B) - \min(\max A, \max B, \max C) \leq 0.$$

Hence,

$$C_2(A \cup B \cup C) \leq C_2(A \cup B) + C_2(B \cup C) - C_2(B) + \min(\max A, \max B, \max C).$$

and we have the last term only when there exists a 't' such that $b(t) = n_B(t)$ is odd, $a(t) = n_A(t)$ is odd, and $c(t) = n_C(t)$ is also odd.

□

Let's assume the below assumptions and let's now prove the bound for the time interval

$$A = R_{i-1}^{\text{right}}, \quad B = R_i, \quad C = R_{i+1}^{\text{left}}, \quad R_i^T = R_{i-1}^{\text{right}} \cup R_i \cup R_{i+1}^{\text{left}}$$

$$\sum_{i \text{ odd}} C_2(R_i^T) = \sum_{i \text{ odd}} C_2(R_{i-1}^{\text{right}} \cup R_i \cup R_{i+1}^{\text{left}})$$

$$\sum_{i \text{ odd}} C_2(R_i^T) \leq \sum_{i \text{ odd}} C_2(R_{i-1}^{\text{right}} \cup R_i) + C_2(R_i \cup R_{i+1}^{\text{left}}) - C_2(R_i)$$

Let's change the indexing from odd to even

$$\sum_{i \text{ odd}} \Big[C_2(R_{i-1}^{\text{right}} \cup R_i) + C_2(R_i \cup R_{i+1}^{\text{left}}) \Big]$$

$$= \sum_{\substack{i \text{ even} \\ [a,b]}} \Big[C_2(R_{i-1} \cup R_i^{\text{left}}) + C_2(R_i^{\text{right}} \cup R_{i+1}) \Big] + C_2(R_{a-1}^{\text{right}} \cup R_a) + C_2(R_b \cup R_{b+1}^{\text{left}}) - \sum_i C_2(R_i)$$

Let $A \subseteq R_i$ and Let $f_i(A)$ be

$$f_i(A) = C_2(R_{i-1} \cup A) + C_2((R_i \setminus A) \cup R_{i+1})$$

$$R_i^{\text{left}} \in \arg\min_A f_i(A), \quad A \subseteq R_i$$

$$f_i(R_i^{\text{left}}) \leq f_i(A)$$

Let R' be the new request after we make the request transfer to remove the all parity

Let S be the partition of requests that depends on the offline optimum trip.

$$S_i = R'_i \cup R'_{i+1}$$

$$R'_i = (S_i \cap R'_i) \cup (S_i \cap R'_{i+1})$$

$$A_i^{\text{opt}} := S_{i-1} \cap R'_i$$

$$f(R_i'^{\text{left}}) \leq f(A_i^{\text{opt}})$$

$$C_2(R'_{i-1} \cup R_i'^{\text{left}}) + C_2(R_i'^{\text{right}} \cup R'_{i+1}) \leq C_2((S_{i-1} \cap R'_i) \cup R'_{i-1}) + C_2((S_i \cap R'_i) \cup R'_{i+1})$$

$$C_2(R'_{i-1} \cup R_i'^{\text{left}}) + C_2(R_i'^{\text{right}} \cup R'_{i+1}) \leq C_2(S_{i-1} \cup R'_i) + C_2(S_i \cup R'_{i+1})$$

Because after the transfer of request triple odd doesn't exist the below condition still holds

$$\therefore \quad \sum_{i \text{ odd}} C_2(R_i^{T_i}) \leq \sum_{i \text{ even}} C_2(R'_{i-1} \cup R_i'^{left}) + C_2(R_i'^{\text{right}} \cup R'_{i+1}) + C_2(R'_{a-1}{}^{right} \cup R'_a) + C_2(R'_b \cup R'_{b+1}{}^{\text{left}}) - \sum_{i \text{ odd}}$$

$$\sum_{i \text{ odd}} C_2(R_i'^{T_i}) \leq \sum_{i \text{ even}} \Big[C_2((S_{i-1} \cap R'_i) \cup R'_{i-1}) + C_2((S_i \cap R'_i) \cup R'_{i+1})\Big] + C_2(R'_{a-1}{}^{\text{right}} \cup R'_a) + C_2(R'_b \cup R'_{b+1}{}^{\text{left}}) -$$

$$C_2(R'^{\text{right}}_{a-1} \cup R'_a) \leq C_2(S_{a-2} \cap R'_{a-1}) + C_2(S_{a-1} \cup R'_a)$$

$$R'_{a-1} = (S_{a-2} \cap R'_{a-1}) \cup (S_{a-1} \cap R'_{a-1})$$

$$C_2(R'_b \cup R'^{\text{left}}_{b+1}) \leq C_2(R'_b \cup S_b) + C_2(S_{b+1} \cup R'_{b+1})$$

$$\sum_{i \text{ odd}} C_2({R'}_i^{T_i}) \leq \sum_{\substack{i \text{ even} \\ [a,b]}} \Big[C_2(S_{i-1} \cup R'_{i-1}) + C_2(S_i \cup R'_{i+1})\Big]$$

$$+ C_2(S_{a-2} \cap R'_{a-1}) + C_2(S_{a-1} \cup R'_a) + C_2(S_b \cup R'_b) + C_2(S_{b+1} \cup R'_{b+1})$$

We will reindex from even to odd again. We get

$$\sum_{i \text{ odd}} C_2({R'}_i^{T_i}) \leq \sum_{i \text{ odd}} \Big[C_2(S_i \cup {R'}_i) + C_2(S_{i-1} \cup {R'}_i)\Big] + C_2(S_{a-2} \cap {R'}_{a-1}) + C_2(S_{a-1} \cup {R'}_a) + C_2(S_{b+1} \cup {R'}_{b+1}) -$$

For reindexing we use the following

Let's use

$$F_j = C_2(S_j \cup R'_j), \quad G_j = C_2(S_{j-1} \cup R'_j)$$

Then

$$\sum_{i \text{ odd}} C_2({R'}_i^{T_i}) \leq \sum_{i \text{ even}} (F_{i-1} + G_{i+1})$$

After reindexing, we get the expression below:

$$\sum_{i \text{ odd}} C_2({R'}_i^{T_i}) \leq \sum_{i \text{ odd}} \Big[C_2(S_i \cup {R'}_i) + C_2(S_{i-1} \cup {R'}_i)\Big] + C_2(S_{a-2} \cap R'_{a-1}) + C_2(S_{b+1} \cup R'_{b+1}) - \sum_{i \text{ odd}} C_2(R'_i)$$

Now because of our initial assumption we know that R_i^T doesn't have a triple odd so we don't need the extra trip. So our theorem becomes

$$C_2(A \cup B \cup C) \leq C_2(A \cup B) + C_2(B \cup C) - C_2(B)$$

When $(i, i+1)$ is not a triple odd, we have(proved below)

$$C_2(S_{i-1} \cup R'_i) + C_2(S_i \cup R'_i) \leq C_2(S_{i-1}) + C_2(S_i) + C_2(R'_i)$$

$$\sum_{i \text{ odd}} C_2(R_i^{T_i}) \leq \sum_{i \text{ odd}} \Big[C_2(S_{i-1}) + C_2(S_i)\Big] + C_2(S_{a-2} \cap R'_{a-1}) + C_2(S_{b+1} \cap R'_{b+1})$$

$$\leq (b - a + 4)F$$

This will add an additional F to the flow time as the request might start one F early because of the extra trip.

So for the total size of requests released in [t,t'] the most time it takes is (t-t') + 4F for any $t \leq t'$. Therefore for an odd i, R' has the earliest request belonging to (i-3)F, and thus all requests in R' are served by time (i+6)F as they are released at time (i+2)F. So the total flow time would be 9F.

Generalisation

For a vehicle with capacity K, the cost function is

$$C_K(S) = \sum_{j=0} d_{n-jK}$$

where $S = \{x_1 \geq x_2 \geq \ldots\}$.

Each trip carries K requests, so you pick every K^{th} element starting from index 1.

For any nonnegative x_j:

$$x_j = \int_0^\infty \mathbf{1}(x_j \geq t)\, dt$$

so

$$C_K(S) = \int_0^\infty \sum_{j=1 \,(\mathrm{mod}\, K)} \mathbf{1}(x_j \geq t)\, dt.$$

Let $n_S(t) = \#\{x \in S : x \geq t\}$. The number of indices $j \equiv 1 \,(\mathrm{mod}\, K)$ with $j \leq n_S(t)$ is

$$\#\{j \leq n_S(t) : j \equiv 1 \,(\mathrm{mod}\, K)\} = \left\lceil \frac{n_S(t)}{K} \right\rceil.$$

Hence

$$C_K(S) = \int_0^\infty \left\lceil \frac{n_S(t)}{K} \right\rceil \, dt.$$

For capacity K we can generalise this as

$$a = Kx + \alpha, \quad b = Ky + \beta, \quad c = Kz + \gamma, \qquad (\alpha, \beta, \gamma) \in \{0, 1, \ldots, K-1\}.$$

The cost identity to prove becomes

$$\left\lceil \frac{a+b+c}{K} \right\rceil \leq \left\lceil \frac{a+b}{K} \right\rceil + \left\lceil \frac{b+c}{K} \right\rceil - \left\lceil \frac{b}{K} \right\rceil + \mathbf{1}\{t \leq \Delta_K\},$$

where Δ_K is the threshold correction analogous to

$$\Delta = \min(\max A, \max B, \max C).$$

Definition of $q_K(t)$

$$q_K(t) := \left\lceil \frac{a+b+c}{K} \right\rceil - \left\lceil \frac{a+b}{K} \right\rceil - \left\lceil \frac{b+c}{K} \right\rceil + \left\lceil \frac{b}{K} \right\rceil.$$

Substitute $a = Kx + \alpha$, $b = Ky + \beta$, $c = Kz + \gamma$. Then

$$q_K(t) = \left\lceil \frac{\alpha + \beta + \gamma}{K} \right\rceil - \left\lceil \frac{\alpha + \beta}{K} \right\rceil - \left\lceil \frac{\beta + \gamma}{K} \right\rceil + \left\lceil \frac{\beta}{K} \right\rceil,$$

where $(\alpha, \beta, \gamma) \in \{0, 1, \ldots, K-1\}$.

Proof. Let's analyze the possible values of $q_K(t)$ for integer triples (α, β, γ).

[label=(c)]

1. For most residue triples (α, β, γ), the integer parts of each term cancel, and hence

$$q_K(t) \leq 0.$$

2. The value $q_K(t) = 1$ can occur only under specific overflow conditions. This happens precisely when

$$\alpha + \beta + \gamma \geq K \quad \text{and} \quad \alpha + \beta < K, \quad \beta + \gamma < K.$$

The term $\left\lceil \frac{\alpha+\beta+\gamma}{K} \right\rceil$ increases by 1 whenever the total sum exceeds a multiple of K. However, if both partial sums $\alpha + \beta$ and $\beta + \gamma$ remain below K, neither of their ceiling terms increases individually. Thus, the full sum "overflows" one multiple of K even though no pair does.

This case represents a similar *triple-overflow* condition, a generalization of the classical "triple-odd" case for $K = 2$.

Conclusion: The "extra trip" arises exactly when all three segments together exceed one

additional multiple of K, while each pair separately does not. Therefore,

$$q_K(t) = \begin{cases} 1, & \text{if } \alpha + \beta + \gamma \geq K \text{ and } \alpha + \beta < K,\ \beta + \gamma < K, \\ 0, & \text{otherwise.} \end{cases}$$

□

Proof. This "overflow" condition corresponds to the case where the combined contribution of the three request sets A, B, C exceeds one additional multiple of K, while each pair individually does not.

The extra trip is bounded by

$$t \leq \min(\max A, \max B, \max C) =: \Delta_K.$$

Hence, $q_K(t)$ will have an extra trip when the above condition is met.

Integrating both sides of the identity

$$C_K(A \cup B \cup C) - C_K(A \cup B) - C_K(B \cup C) + C_K(B) = \int_0^\infty q_K(t)\, dt,$$

we obtain

$$C_K(A \cup B \cup C) \leq C_K(A \cup B) + C_K(B \cup C) - C_K(B) + \Delta_K.$$

The additive term Δ_K represents the contribution from the limited range of $t \leq \Delta_K$ where overflow occurs, corresponding to at most one additional capacity-K trip.

Conclusion. The inequality formally captures the bounded deviation from submodularity introduced by vehicle capacity K:

$$C_K(A \cup B \cup C) \leq C_K(A \cup B) + C_K(B \cup C) - C_K(B) + \Delta_K,$$

cddwhere $\Delta_K = \min(\max A, \max B, \max C)$ quantifies the threshold of the final joint overflow.

□

We can use the above lemmas to prove the same bound on the time interval [a,b] and the flow time will be the same as for capacity 2 which is 9F.

We start by defining:

$$q_k = \left\lceil \frac{\alpha+\beta+\gamma}{k} \right\rceil - \left\lceil \frac{\alpha+\beta}{k} \right\rceil - \left\lceil \frac{\beta+\gamma}{k} \right\rceil + \left\lceil \frac{\beta}{k} \right\rceil$$

Use the inequality:

$$\lceil x+y \rceil \leq \lceil x \rceil + \lceil y \rceil$$

Let:

$$x = \frac{\alpha+\beta}{k}, \quad y = \frac{\gamma}{k}$$

Then:

$$\left\lceil \frac{\alpha+\beta+\gamma}{k} \right\rceil - \left\lceil \frac{\alpha+\beta}{k} \right\rceil \leq \left\lceil \frac{\gamma}{k} \right\rceil$$

Hence:

$$\left\lceil \frac{\alpha+\beta+\gamma}{k} \right\rceil - \left\lceil \frac{\alpha+\beta}{k} \right\rceil \leq \left\lceil \frac{\gamma}{k} \right\rceil$$

Further Bounding on q_k

We begin with:

$$q_k \leq \left\lceil \frac{\gamma}{k} \right\rceil - \left\lceil \frac{\beta+\gamma}{k} \right\rceil + \left\lceil \frac{\beta}{k} \right\rceil$$

This simplifies to:

$$q_k \leq \left\lceil \frac{\beta}{k} \right\rceil + \left\lceil \frac{\gamma}{k} \right\rceil - \left\lceil \frac{\beta+\gamma}{k} \right\rceil$$

Lower Bounding Inequality

We use the ceiling addition lower bound:

$$\lceil x + y \rceil \geq \lceil x \rceil + \lceil y \rceil - 1$$

Let:

$$x = \left\lceil \frac{\beta}{k} \right\rceil, \quad y = \left\lceil \frac{\gamma}{k} \right\rceil$$

From the ceiling addition lower bound:

$$\left\lceil \frac{\beta + \gamma}{k} \right\rceil \geq \left\lceil \frac{\beta}{k} \right\rceil + \left\lceil \frac{\gamma}{k} \right\rceil - 1$$

Rearranging gives:

$$q_k \leq \left(\left\lceil \frac{\gamma}{k} \right\rceil + \left\lceil \frac{\beta}{k} \right\rceil \right) - \left(\left\lceil \frac{\beta + \gamma}{k} \right\rceil \right) \leq 1$$

Thus,

$$q_k \leq 1$$

This implies:

$$q_k \text{ cannot exceed } 1$$

So we will have at most 1 extra trip and we can propogate it and get the above bound.

Theorem 5. *For path graphs with capacity K there is a far-sighted algorithm that has a competitive ratio of 9F.*

Proof:

Let

$$A = R_{i-1}^{\text{right}}, \quad B = R_i, \quad C = R_{i+1}^{\text{left}}, \quad R_i^T = R_{i-1}^{\text{right}} \cup R_i \cup R_{i+1}^{\text{left}}$$

$$\sum_{i \text{ odd}} C_K(R_i^T) = \sum_{i \text{ odd}} C_K(R_{i-1}^{\text{right}} \cup R_i \cup R_{i+1}^{\text{left}})$$

$$\sum_{i \text{ odd}} C_K(R_i^T) \leq \sum_{i \text{ odd}} C_K(R_{i-1}^{\text{right}} \cup R_i) + C_K(R_i \cup R_{i+1}^{\text{left}}) - C_K(R_i)$$

Let's change the indexing from odd to even

$$\sum_{i \text{ odd}} \Big[C_K(R_{i-1}^{\text{right}} \cup R_i) + C_K(R_i \cup R_{i+1}^{\text{left}}) \Big]$$

$$= \sum_{\substack{i \text{ even} \\ [a,b]}} \Big[C_K(R_{i-1} \cup R_i^{\text{left}}) + C_K(R_i^{\text{right}} \cup R_{i+1}) \Big] + C_K(R_{a-1}^{\text{right}} \cup R_a) + C_K(R_b \cup R_{b+1}^{\text{left}}) - \sum_i C_K(R_i)$$

Let $A \subseteq R_i$ and Let $f_i(A)$ be

$$f_i(A) = C_K(R_{i-1} \cup A) + C_K((R_i \setminus A) \cup R_{i+1})$$

$$R_i^{\text{left}} \in \arg\min_A f_i(A), \quad A \subseteq R_i$$

$$f_i(R_i^{\text{left}}) \leq f_i(A)$$

Let R' be the new request after we make the request transfer to remove the all parity

Let S be the partition of requests that depends on the offline optimum trip.

$$S_i = R'_i \cup R'_{i+1}$$

$$R'_i = (S_i \cap R'_i) \cup (S_i \cap R'_{i+1})$$

$$A_i^{\text{opt}} := S_{i-1} \cap R'_i$$

$$f(R_i^{'\text{left}}) \leq f(A_i^{\text{opt}})$$

$$C_K(R'_{i-1} \cup R_i^{'\text{left}}) + C_K(R_i^{'\text{right}} \cup R'_{i+1}) \leq C_K((S_{i-1} \cap R'_i) \cup R'_{i-1}) + C_K((S_i \cap R'_i) \cup R'_{i+1})$$

$$C_K(R'_{i-1} \cup R_i^{'\text{left}}) + C_K(R_i^{'\text{right}} \cup R'_{i+1}) \leq C_K(S_{i-1} \cup R'_i) + C_K(S_i \cup R'_{i+1})$$

Because after the transfer of request triple odd doesn't exist the below condition still holds

$$\therefore \quad \sum_{i \text{ odd}} C_K(R_i^{T_i}) \leq \sum_{i \text{ even}} C_K(R'_{i-1} \cup R_i^{'\text{left}}) + C_K(R_i^{'\text{right}} \cup R'_{i+1}) + C_K(R_{a-1}^{'\text{right}} \cup R'_a) + C_K(R'_b \cup R_{b+1}^{'\text{left}}) - \sum_{i \text{ odd}} C_K$$

$$\sum_{i \text{ odd}} C_K(R_i^{'T_i}) \leq \sum_{i \text{ even}} \Big[C_K((S_{i-1} \cap R'_i) \cup R'_{i-1}) + C_K((S_i \cap R'_i) \cup R'_{i+1}) \Big] + C_K(R_{a-1}^{'\text{right}} \cup R'_a) + C_K(R'_b \cup R_{b+1}^{'\text{left}}) - \sum_{i}$$

$$C_K(R'^{\text{right}}_{a-1} \cup R'_a) \leq C_K(S_{a-2} \cap R'_{a-1}) + C_K(S_{a-1} \cup R'_a)$$

$$R'_{a-1} = (S_{a-2} \cap R'_{a-1}) \cup (S_{a-1} \cap R'_{a-1})$$

$$C_K(R'_b \cup R'^{\text{left}}_{b+1}) \leq C_K(R'_b \cup S_b) + C_K(S_{b+1} \cup R'_{b+1})$$

$$\sum_{i \text{ odd}} C_K(R'^{T_i}_i) \leq \sum_{\substack{i \text{ even} \\ [a,b]}} \Big[C_K(S_{i-1} \cup R'_{i-1}) + C_K(S_i \cup R'_{i+1})\Big]$$
$$+ C_K(S_{a-2} \cap R'_{a-1}) + C_K(S_{a-1} \cup R'_a) + C_K(S_b \cup R'_b) + C_K(S_{b+1} \cup R'_{b+1})$$

We will reindex from even to odd again. We get

$$\sum_{i \text{ odd}} C_K(R'^{T_i}_i) \leq \sum_{i \text{ odd}} \Big[C_K(S_i \cup R'_i) + C_K(S_{i-1} \cup R'_i)\Big] + C_K(S_{a-2} \cap R'_{a-1}) + C_K(S_{a-1} \cup R'_a) + C_K(S_{b+1} \cup R'_b$$

For reindexing we use the following

Let's use

$$F_j = C_K(S_j \cup R'_j), \quad G_j = C_K(S_{j-1} \cup R'_j)$$

Then

$$\sum_{i \text{ odd}} C_K(R'^{T_i}_i) \leq \sum_{i \text{ even}} (F_{i-1} + G_{i+1})$$

After reindexing, we get the expression below:

$$\sum_{i \text{ odd}} C_K(R'^{T_i}_i) \leq \sum_{i \text{ odd}} \Big[C_K(S_i \cup R'_i) + C_K(S_{i-1} \cup R'_i)\Big] + C_K(S_{a-2} \cap R'_{a-1}) + C_K(S_{b+1} \cup R'_{b+1}) - \sum_{i \text{ odd}} C_K(R'_i)$$

Now because of our initial assumption we know that R_i^T doesn't have a triple odd so we don't need the extra trip. So our theorem becomes

$$C_K(A \cup B \cup C) \leq C_K(A \cup B) + C_K(B \cup C) - C_K(B)$$

When $(i, i+1)$ is not a triple odd, we have(proved below)

$$C_K(S_{i-1} \cup R'_i) + C_K(S_i \cup R'_i) \leq C_K(S_{i-1}) + C_K(S_i) + C_K(R'_i)$$

$$\sum_{i \text{ odd}} C_K(R_i^{T_i}) \leq \sum_{i \text{ odd}} \Big[C_K(S_{i-1}) + C_K(S_i)\Big] + C_K(S_{a-2} \cap R'_{a-1}) + C_K(S_{b+1} \cap R'_{b+1})$$

$$\leq (b - a + 4)F$$

This will add an additional F to the flow time as the request might start one F early because of the extra trip.

So for the total size of requests released in $[t, t']$ the most time it takes is $(t - t') + 4F$ for any $t \leq t'$. Therefore for an odd i, R' has the earliest request belonging to $(i-1)F$ which ranges from $[(i-2)F, (i-1)F]$, and thus the oldest request in $(i-2)F$ will have to wait $4F$ before it is released at time $(i+2)F$ and an extra F in worst case for $q(t)$. So the total flow time would be $9F$.

4.4 Summary of Results

This chapter establishes sharp separations between online strategies under capacity constraints. While myopic algorithms achieve constant competitive ratios for unit capacity, they fail for higher capacities. Far-sighted batching is necessary to obtain bounded perfor-

mance as capacity increases.

Chapter 5

Conclusion

This thesis studied the online food delivery problem under capacity constraints with the objective of minimizing the maximum flow time. We focused on structured metric spaces, specifically path graphs which allow precise competitive analysis while capturing essential features of real-world delivery systems. By systematically analyzing both myopic and far-sighted algorithms, we characterized how server capacity fundamentally alters the behavior and performance of online dispatch strategies.

5.1 Summary of Contributions

The main contributions of this thesis can be summarized as follows. First, we analyzed myopic algorithms on path graphs under unit capacity and showed that simple strategies achieve constant competitive ratios. In particular, we established tight bounds for FIFO-based and anticipation-based algorithms, demonstrating that limited anticipation can significantly improve flow-time performance.

Second, we showed that increasing capacity introduces qualitative changes in algorithmic behavior. For capacity two, we proved that no myopic FIFO-based algorithm admits a bounded competitive ratio on path graphs. This impossibility result highlights the limitations of immediate dispatch strategies once batching becomes possible.

Third, to overcome these limitations, we introduced a far-sighted batching algorithm for general capacity K. By partitioning time into intervals and carefully bundling requests across adjacent intervals, the proposed algorithm achieves a constant competitive ratio on path graphs. This result demonstrates that controlled waiting and batching are essential for maintaining bounded flow time as capacity increases.

5.2 Implications

The results of this thesis provide insight into the design of real-time delivery and dispatch systems. In low-capacity settings, simple myopic strategies may suffice and are attractive due to their simplicity and responsiveness. However, as capacity increases, such strategies can lead to arbitrarily poor performance. Our analysis shows that incorporating anticipation and batching is not merely an optimization, but a necessity for achieving fairness and bounded delay.

These findings help clarify the tradeoffs faced by practical delivery platforms, where decisions must balance immediacy against efficiency under capacity constraints.

5.3 Future Work

Several directions remain open for future research. One natural extension is to study more general metric spaces beyond paths graphs and to understand how geometric properties affect the competitiveness of online algorithms. Another direction is to consider multiple servers and analyze how coordination and load balancing interact with capacity constraints.We can also look into myopic strategies and how to get bounded competitive ratios for different algorithms.

It would also be interesting to explore stochastic arrival models or partially predictive settings, where limited information about future requests is available. Finally, extending the analysis to other objectives, such as weighted flow time or combined flow-time and

distance metrics, could further bridge the gap between theoretical models and practical delivery systems.

Reference

[1] Mohammad Akbarpour, Shengwu Li, and Shayan Oveis Gharan. “Dynamic matching market design”. In: *Operations Research* 68.2 (2020), pp. 371–392.

[2] Matthew Andrews et al. “Universal stability results for greedy contention resolution protocols”. In: *Journal of the ACM* 48.1 (2001), pp. 39–69.

[3] Norbert Ascheuer, Matteo Fischetti, and Martin Grötschel. “A branch-and-cut algorithm for the capacitated vehicle routing problem”. In: *Mathematical Programming* 86.2 (2000), pp. 307–338.

[4] Giorgio Ausiello et al. “On-line algorithms for the traveling salesman problem”. In: *Theoretical Computer Science* 268.1 (2001), pp. 3–29.

[5] Yossi Azar, Andrei Z. Broder, and Anna R. Karlin. “On-line load balancing and batching”. In: *Algorithmica* 23.4 (1999), pp. 313–329.

[6] Yossi Azar et al. “On-line load balancing”. In: *Theoretical Computer Science* 130.1 (1999), pp. 73–84.

[7] Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. “Scheduling with deadlines and flow time”. In: *SIAM Journal on Computing* 34.4 (2004), pp. 904–934.

[8] Nikhil Bansal and Kirk Pruhs. “Speed scaling to manage energy and response time”. In: *SIAM Journal on Computing* 39.4 (2010), pp. 1292–1315.

[9] Yair Bartal, Amos Fiat, and Manor Mendel. “On the competitive ratio of the work function algorithm”. In: *Journal of the ACM* 48.3 (2001), pp. 546–560.

[10] Dimitris Bertsimas and Garrett van Ryzin. “A stochastic and dynamic vehicle routing problem in the Euclidean plane”. In: *Operations Research* 39.4 (1991), pp. 601–615.

[11] Marcin Bienkowski and Marek Korzeniowski. “Online routing in asymmetric metric spaces”. In: *Theoretical Computer Science* 585 (2015), pp. 1–14.

[12] Avrim Blum, Howard Karloff, and Yuval Rabani. “The competitive analysis of online algorithms”. In: *Information and Computation* 121.1 (1994), pp. 1–15.

[13] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[14] Ho-Leung Chan, Simon Fung, and Lap-Kei Lau. “Online routing with resource augmentation”. In: *Algorithmica* 62.1–2 (2012), pp. 233–252.

[15] Chandra Chekuri, Rajeev Motwani, and Balaji Natarajan. “Scheduling to minimize weighted flow time”. In: *SIAM Journal on Computing* 30.5 (2001), pp. 1468–1492.

[16] Edward G. Coffman, Michael R. Garey, and David S. Johnson. “Approximation algorithms for bin packing: A survey”. In: *Approximation Algorithms for NP-hard Problems* (1996), pp. 46–93.

[17] Matthias Englert and Heiko Röglin. “Online routing on trees”. In: *Theoretical Computer Science* 678 (2017), pp. 70–82.

[18] Ronald Fagin and Prabhakar Raghavan. “Competitive analysis of parallel algorithms”. In: *Journal of Computer and System Sciences* 52.1 (1996), pp. 13–27.

[19] Esteban Feuerstein and Leen Stougie. “On-line vehicle routing with multiple servers”. In: *Theoretical Computer Science* 268.1 (2001), pp. 355–369.

[20] Xiangyu Guo et al. “Online Food Delivery to Minimize Maximum Flow Time”. In: *arXiv preprint arXiv:2110.15772* (2021).

[21] Xiangyu Guo et al. “The online food delivery problem on stars”. In: *Theoretical Computer Science* 928 (2022), pp. 13–26. DOI: 10.1016/j.tcs.2022.06.007.

[22] Leslie A. Hall, David B. Shmoys, and Joel Wein. “Minimizing average and maximum flow time on parallel machines”. In: *Mathematics of Operations Research* 22.3 (1997), pp. 487–503.

[23] Mikael Hammar and Bengt J. Nilsson. “Competitive algorithms for the k-server problem on trees”. In: *Algorithmica* 33.3 (2002), pp. 353–371.

[24] Soumia Ichoua, Michel Gendreau, and Jean-Yves Potvin. “Dynamic vehicle dispatching with real-time information”. In: *Transportation Science* 40.4 (2006), pp. 470–486.

[25] Bala Kalyanasundaram and Kirk Pruhs. “Speed is as powerful as clairvoyance”. In: *Journal of the ACM* 47.4 (2000), pp. 617–643.

[26] Elias Koutsoupias and Christos Papadimitriou. “On the k-server conjecture”. In: *Journal of the ACM* 42.5 (1995), pp. 971–983.

[27] Gilbert Laporte. “Fifty years of vehicle routing”. In: *Transportation Science* 43.4 (2009), pp. 408–416.

[28] Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. “Competitive algorithms for on-line problems”. In: *Journal of Algorithms* 11.2 (1990), pp. 208–230.

[29] Nicole Megow, Julián Mestre, and José Verschae. “Delay management in online scheduling and routing”. In: *Operations Research Letters* 39.3 (2011), pp. 192–197.

[30] Nicole Megow, José Verschae, and Andreas Wiese. “Online scheduling with commitments”. In: *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*. 2012, pp. 953–962.

[31] Snezana Mitrovic-Minic and Gilbert Laporte. “Waiting strategies for the dynamic pickup and delivery problem”. In: *Transportation Research Part B* 38.7 (2004), pp. 635–655.

[32] Cynthia A. Phillips, Clifford Stein, and Eric Torng. “On-line scheduling of jobs with deadlines on parallel machines”. In: *SIAM Journal on Computing* 26.3 (1997), pp. 673–687.

[33] Harilaos N. Psaraftis. “Dynamic vehicle routing problems”. In: *Transportation Science* 22.3 (1988), pp. 215–228.

[34] Rene Sitters. “The online traveling salesman problem on the line”. In: *Information Processing Letters* 90.5 (2004), pp. 251–255.

[35] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. SIAM, 2002.