Title
From the Real Vehicle to the Virtual Vehicle

Author
Huang, Jiangchuan

Publication Date
2013

Peer reviewed|Thesis/dissertation

From the Real Vehicle to the Virtual Vehicle

by

Jiangchuan Huang

A dissertation submitted in partial satisfaction of the
requirements for the degree of

Doctor of Philosophy

in

Engineering – Civil and Environmental Engineering

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Raja Sengupta, Chair
Professor Carlos Daganzo
Professor Satish Rao

Fall 2013

From the Real Vehicle to the Virtual Vehicle

by

Jiangchuan Huang

Abstract

From the Real Vehicle to the Virtual Vehicle

by

Jiangchuan Huang

Doctor of Philosophy in Engineering – Civil and Environmental Engineering

University of California, Berkeley

Professor Raja Sengupta, Chair

We focus on systems with task arrivals in time and space. A good model for the single-customer case is the Dynamic Traveling Repairman Problem (DTRP) [1]. The DTRP literature has focused on optimizing the expected value of system time, defined as the elapsed time between the arrival and the completion of each task. We focus on the stability and distribution of system time, including its variance. This dissertation establishes a partially policy independent necessary and sufficient condition for stability in the DTRP. The policy class includes some of the policies proven to be optimal for system time expectation under light and heavy loads in the literature. We propose a new policy named PART-n-TSP and compute a good approximation for its system time distribution. PART-n-TSP has lower system time variance than PART-TSP [2] and Nearest Neighbor [1] when the load is neither too small or too large. We prove that PART-n-TSP is also optimal for system time expectation under light and heavy loads.

In the multi-customer case, the scheduling policies of the DTRP and other vehicle routing problems do not create performance isolation [3] between customers. We explore performance isolation between customers by borrowing the virtual machine abstraction from cloud computing. Since our servers are moving vehicles, we propose a new equivalent of the virtual machine called the virtual vehicle enabling what we call cloud computing in space. The customer operates virtual vehicles that are in reality hosted by fewer, shared provider-operated real vehicles. We show that cloud computing in space can do better than conventional cloud computing in the sense of realizing high performance isolation (e.g. 98%) while requiring significantly fewer real vehicles (e.g. approximately 1-for-5).

Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to express my deep gratitude to Professor Raja Sengupta for advising me and supporting me. He always encouraged me to work on fundamental problems. His strong intuition, deep insight into good research problems, and dedication to enlighten students always motivated me to challenge myself and do better. Through him, I learned to understand the meaning of science and the high standards of rigor held by true researchers.

During my study in Civil Systems at UC Berkeley, I also worked with Professor Christoph Kirsch in the Department of Computer Sciences at the University of Salzburg. From him, I learned how to abstract and formulate good problems. During our weekly group meetings, he gave me numerous useful suggestions from the perspective of computer science, which broadened the scope of my research.

I would like to offer special thanks to my committee members: Professor Carlos F. Daganzo in the Transportation Engineering Department and Professor Satish Rao in the Computer Science Department. They provided me with numerous useful suggestions to improve my dissertation.

During my early stay at Berkeley, I worked with Professor Adib Kanafani in the Transportation Engineering Department. He was my adviser when I was pursuing my M.S. in Transportation Engineering. He taught me how to approach difficult problems and gave me several useful suggestions during the course of my research.

I would like to thank all my colleagues at the Center for Collaborative Control of Unmanned Vehicles (C3UV) and the Cyber-Physical Cloud Computing (CPCC) Lab for creating an excellent research environment over the last four years. Eloi Pereira was my peer as a Ph.D. student. Eloi's diligence and his attention to detail constantly inspired me. He taught me a lot of software skills and was always willing to answer my questions. Dr. Ching-Ling Huang also spent a lot of time with me sharing his research experience during my early Ph.D. years. Dr. Joshua Love provided me with very valuable ideas and vast technical support on the UAV projects. I would also like to thank the other members of the C3UV group and the CPCC lab, namely Dr. Yaser Fallah, Dr. Brandon Basso, Dr. Jared Wood, Hao Chen, Jared Garvey, Mark Godwin, Ben Kehoe, Shih-Yuan Liu, Clemens Krainer, and Michael Lippautz.

I would like to give special thanks to my dearest friend Yang Zhao, Dr. Yarong Yang and Lily Hu, who supported and accompanied me most in those challenging but rewarding days. I would also like to thank all my other friends who have made my stay in Berkeley a fantastic experience. In no particular order they are: Bo, Zhen, Weihua, Yiguang, Jing, Yi, Zhuolun, Joe, and Joseph, together with my dear dancing partners Merlene, Tiffany, Caitlyn, Fei, and Cindy.

Finally and most importantly, I would like to thank my family, especially my parents,

Jianguo and Qiuyu. They have always believed in me and supported me during the most difficult of times. I would also like to thank my little brother Long'er and my little sister Fei'er for their support and love.

Chapter 1

Introduction

Consider the following systems with tasks arriving in time and space: (i) Google Street View, Google is running numerous data-collection vehicles with mounted cameras, lasers, a GPS and several computers to collect street views while minimizing the distance travelled [5]. (ii) Real-time traffic reporting, a radio station uses a helicopter to overfly accident scenes and other areas of high traffic volume for real-time traffic information. (iii) Unmanned aerial vehicle (UAV)-based sensing, a fleet of UAVs equipped with greenhouse gas sensors collect airborne measurements of greenhouse gases at several sites in California and Nevada, executing every task at its location before its deadline [6]. (iv) Enabling mobility in wireless sensor networks (WSN), mobile elements (vehicles) capable of short-range communications collect data from nearby sensor nodes as they approach on a schedule [7].

In these applications, a task is a triple (T^a, X, T^S) where T^a is the arrival time of the task, X its location in space, and T^S its execution time (task size). Servers, henceforth referred to as real vehicles, are networked vehicles each having all or some of the sensing, computation, communication, and locomotion capabilities. A real vehicle is considered to serve a task by visiting the location X and staying there for execution time T^S after the task has arrived, i.e., after time T^a. The performance metrics include system time (latency), throughput, delivery probability, and total distance traveled by the vehicles.

1.1 Real Vehicle Routing

Traditionally, these applications were modeled by (i) the vehicle routing problem (VRP) [8], its variations such as (ii) the Dynamic Travelling Repairman Problem (DTRP) [1], (iii) the stochastic and dynamic vehicle routing problem with time windows (SDVRPTW) [9], and (iv) the mobile element scheduling (MES) problem [7].

Tasks are allocated and sequenced based on their arrival times T^a, e.g., first come first served (FCFS) [1], locations X, e.g., nearest neighbor (NN) [1], and sizes T^S, e.g., shortest job first [10], to minimize the distance traveled [8], the average time each task spent in the system [1], or the deadline violation ratio [9]. A richer set of scheduling policies include direct tree search, dynamic programming, and integer linear programming for VRP [11]; FCFS, stochastic queue median (SQM), travelling salesman policy (TSP), NN [1], divide and conquer [12] for DTRP and SDVRPTW [9]; earliest deadline first, earliest deadline first with k-lookahead, the minimum weight sum first [7] and partition based scheduling [13] for MES.

The DTRP [1] is a good way to model the stochastic and dynamic vehicle routing problem for a single-customer system by assuming a Poisson task arrival process with rate λ, a general task location distribution, and a general size distribution with mean b. An earlier formulation similar to the DTRP can be found in [14]. The DTRP resembles an M/G/1 queue in the time dimension but looks like a vehicle routing problem in the space dimension. As we know in queueing theory, $\rho = \lambda b < 1$ is a necessary and sufficient condition for all work conserving M/G/1 queues [15, sec. II.4.2]. However, there is no such policy-independent stability condition for the DTRP, which seems to be a "spatial version" of the M/G/1 queue. The known stability conditions for the DTRP are policy-dependent [1, 2, 12].

Chapter 2 of this dissertation makes progress towards finding stability conditions for the DTRP that are less policy dependent than those in the literature. We establish $\rho + \lambda b_d < 1$ as a necessary and sufficient stability condition for the class of Polling-Sequencing (P-S) policies satisfying *unlimited-polling* and *economy of scale* in Theorem 2.5. This stability condition is identical to the necessary condition for stability given in [16]. We show that important policies for the DTRP in the literature fall in the P-S class and satisfy the two properties. The extra term b_d is the limit of mean travel time as the number of tasks in a polling station goes to infinity. We prove that the existence of b_d is a consequence of economy of scale. b_d is policy-dependent, but it only depends on the sequencing phase of the P-S policy. Since the value of b_d can be derived in the static setting, we do not need to analyze or simulate the dynamic queueing system of DTRP to get b_d. We only need to analyze or simulate to obtain the statistics of sequencing N tasks, where N is a random variable, and the task locations are distributed in some fashion. These results are published in our paper [17].

Our second contribution to the DTRP is the distribution of system time T, defined as the elapsed time between the arrival and the completion of each task. Knowing the distribution of the system time T, together with its expectation $E[T]$ and variance $Var[T]$ or standard deviation $\sigma[T]$, enables the expectation-variance analysis of the system under uncertainties [18, 19]. On entering a McDonalds, one may ask not just "What is my expected service time?" but also "How certain is this value?" We show in Tables 3.1 and 3.2 in Chapter 3 that two policies at the same load level can be incomparable in the sense that one has low expectation of system time but high

variance while the other has high expectation of system time but low variance. In practice, highly variable system time can be even more frustrating than large mean system times [20, 21]. The literature discusses the distribution of system time T, together with its expectation and variance, for the FCFS policy and its variations such as the SQM and partitioning-FCFS [1]. This is in sharp contrast to queueing theory where the distribution of the system time or its moments are known for a wide variety of policies. See for example [10, 22]. To illustrate our point, the expected system time of the FCFS, SQM and partitioning-FCFS policy is not as good as NN [1] and PART-TSP [2] or DC [12] at most load levels.

In Chapter 3, we propose a policy in the P-S class called the PART-n-TSP policy. We give a good approximation for the distribution of the system time that is easy to compute. We do this by utilizing approximation results for the distribution of system time T, together with $E[T]$ and $Var[T]$ known for polling systems [23, 24]. Figure 3.5 shows that the cumulative distribution function (cdf) of the system time as computed by our method is very close to the cdf of the system time as obtained by Monte-Carlo simulation. We show that FCFS, partitioning-FCFS and n-TSP [1] are special cases of PART-n-TSP, meaning PART-n-TSP can be optimized to have better performance than the three. We also compare PART-n-TSP with PART-TSP [2] and Nearest Neighbor [1] on $E[T]$ and $\sigma[T]$ in Tables 3.1 and 3.2, since the latter two are considered near optimal in the literature. The $E[T]$ and $\sigma[T]$ under PART-n-TSP are obtained by our approximation. The $E[T]$ and $\sigma[T]$ under PART-TSP and NN are obtained by simulation. The results show that NN achieves lower $E[T]$ than both PART-n-TSP and PART-TSP for all loads $\rho \in \{0.1 \ldots 0.9\}$ simulated by us. PART-n-TSP achieves lower $E[T]$ than PART-TSP when ρ is not too small or too large, e.g. when $\rho \in \{0.3, \ldots, 0.7\}$. Also, PART-$n$-TSP achieves lower $\sigma[T]$ than PART-TSP and NN when ρ is not too small or too large, e.g. when $\rho \in \{0.3, \ldots, 0.7\}$. In real systems it may be desirable for ρ to be neither too small nor too large, since small ρ results in low server utilization, and large ρ in large system times. If so, PART-n-TSP would be good in practice as it achieves lower $\sigma[T]$ than PART-TSP and NN, and lower $E[T]$ than PART-TSP. We also prove that PART-n-TSP is $E[T]$ optimal under light load ($\rho \to 0^+$) and asymptotically optimal under heavy load ($\rho \to 1^-$). See Theorem 3.1.

1.2 Virtual Vehicle Performance Isolation

In this part of the thesis we formulate the vehicle routing problem (VRP) and its variations work for multi-customer systems. In the Dynamic Travelling Repairman Problem (DTRP) tasks from different customers are indistinguishable to the scheduling policies, meaning a customer submitting more tasks would use more resources, possibly to the detriment of other customers.

To resolve this problem, we formulate the multi-customer VRP by borrowing the concept of a virtual machine from cloud computing. In cloud computing, the virtual machine abstraction creates performance isolation so that resources consumed by one virtual machine do not necessarily harm the performance of other virtual machines [3]. To extend the idea of the virtual machine [3] used in cloud computing, we define an idea called the virtual vehicle. The virtual vehicle is an analog of the virtual machine for location specific tasks. Just as the cloud computing customer has a service-level agreement (SLA) for a virtual machine, our customer would have an SLA for a virtual vehicle. The virtual vehicle enables what we call cloud computing in space, or the spatial cloud. To a customer, a virtual vehicle would be exactly like a real vehicle that travels at the virtual speed specified in the SLA. Consider again real-time traffic reporting. If the radio station (the customer) uses a helicopter to overfly accident scenes and other areas of high traffic volume (the task) at 100 mile per hour (mph), then the helicopter should arrive at an accident scene 10 miles away in 6 minutes. We now virtualize this helicopter. Instead of buying or renting, and operating the helicopter, the radio station would reserve a virtual helicopter in our spatial cloud traveling at 100 mph, and expect the accident scene to be recorded in 6 minutes. The radio station could also have more than one tasks for its virtual helicopter. In this case, the radio station would queue the multiple tasks for the virtual helicopter, just as it would do for a real helicopter. The radio station can estimate the completion time of each task by dividing the consecutive distances between tasks by the virtual speed just as they would do for their real helicopter. The task completion time expected by the radio station (customer) is given by (4.4) in Chapter 4. The radio station would enjoy the advantage of reserving the virtual helicopter only when the traffic reporting program is on, and not paying for other times. If the real helicopters executing the virtual helicopters could serve other customers at these other times, the enhanced vehicle utilization would reduce costs for all customers much like cloud computing. The analysis in Chapter 4 quantifies these gains.

The provider's responsibility is to use the real vehicles to travel to and execute each task such that the real completion time is no later than the expected completion time. The system achieves high performance isolation if a statistically dominant subset [25], e.g., 98%, of the virtual vehicle's tasks are completed no later than their expected completion times. The fraction must be small enough for the customer to consider herself adequately compensated for this loss by the reduced costs delivered by the spatial cloud. We design the spatial cloud to treat the expected completion time as a "deadline" and call it the virtual deadline. The virtual deadlines make the spatial cloud a soft real-time system [26, 27], meaning performance metrics such as tardiness and delivery probability defined in (4.5) and (4.6) can be used to measure performance isolation. These performance metrics are the performance isolation measures [28] and Jain's fairness indices [29] defined separately in (4.12) and (4.13), and in (4.14) and (4.15) in Section 4.1. The slack defined in (4.7) measures the earliness of a task completion. How the provider realizes the virtual deadlines of the tasks is of no concern to the customer. For example, the provider can use a real vehicle to travel to and execute a task as the virtual vehicle does, or migrate the information of the virtual

vehicle through a network to another real vehicle closer to the task. In the latter case, the real distance traveled is smaller than the virtual distance resulting in the phenomenon we have called migration gain. The idea works when the communication costs of migrating a virtual vehicle over the network are small. The migration cost of a virtual vehicle is defined in (4.11).

Craciunas et al. [30] proposed cyber-physical cloud computing as a concept that makes sensing a service, and described their work on the software engineering required to build a virtual vehicle. Kirsch et al. [31] described the protocols required for migrating virtual vehicles. The results of Chapter 4 show that the provider can support a given number of virtual vehicles with significantly fewer real vehicles that travel at the virtual speed while guaranteeing high performance isolation. We quantify the gain by the ratio of the number of virtual vehicles over the number of real vehicles. The gain arises from two phenomena. (i) A customer may not fully utilize her virtual vehicle, enabling the spatial cloud to multiplex several virtual vehicles onto one real vehicle. This type of gain is called multiplexing gain, and has been observed in communication networks [32] and cloud computing [33]. (ii) The real vehicles save travel distance by migrating the virtual vehicle hosting a task to another real vehicle closer to the task, creating a new type of gain we call migration gain. Chapter 4 focuses on migration gain since it is unique to this spatial cloud, virtual vehicle, and virtual speed. We use a very simple model of the task size T^S itself since we focus on the migration gain arising from the spatial distribution of the tasks. Some of the cost savings from the provider's high gain can be passed back to the customer to compensate for the small loss of missed virtual deadlines.

We analyze the system in Section 4.2 using results in queueing theory [15, 34], stochastic and dynamic vehicle routing [1, 12, 17], and soft real-time systems [26, 27]. Figure 4.1 depicts the $GI/GI/1$ queue [15] of each virtual vehicle (Theorem 4.1), and the $\Sigma GI/GI/1$ queue [34] of each real vehicle (Theorem 4.2) under the Voronoi tessellation [35] in Definition 4.1. We propose scheduling policies adapted from conventional cloud computing [36], named Earliest Virtual Deadline First (EVDF) when task size is known a priori (Definition 4.5), its variation Earliest Dynamic Virtual Deadline First (EDVDF) when task size is not known a priori (Definition 4.6), and the credit scheduling policy (Definition 4.7) in Section 4.3. Theorem 4.3 asserts that EVDF minimizes tardiness. This is based on the optimality of earliest deadline first scheduling for real-time queues [37]. Theorem 4.4 identifies a worst-case arrival process maximizing tardiness called the $\eta = 1$ arrival process in Definition 4.8. Theorem 4.5 asserts task size dependent economy of scale [38] in the sense that the largest achievable gain without compromising performance isolation increases as the square root of the number of real vehicles when the task size is zero, and increases but is upper bounded by a constant when the task size is greater than zero. Theorem 4.6 bounds the migration cost, together with slack under the $\eta = 1$ arrival process.

We simulate the system with homogeneous real vehicles (Definition 4.3) and homogeneous virtual vehicles (Definition 4.4) under the EVDF, EDVDF and credit scheduling

policies in Section 4.4. Each virtual vehicle generates an $\eta = 1$ arrival process that maximizes tardiness, and excludes multiplexing gain by having each virtual vehicle fully utilized. Thus, this is a worst-case simulation and the gain is migration gain only. Figure 4.4 shows the performance isolation and fairness index based on tardiness and delivery probability, together with slack and migration costs under EVDF for different numbers of real vehicles and gains and thus different number of virtual vehicles. We conclude that (i) the provider can support a given number of virtual vehicles with significantly fewer real vehicles that travel at the virtual speed while guaranteeing high performance isolation, e.g., the gain equals 7.5 while guaranteeing tardiness $\leq 1\%$ when the number of real vehicles is 100 as shown in Figure 4.5. (ii) The migration gain increases with the number of real vehicles while guaranteeing the same performance isolation as asserted in Theorem 4.5, and shown in Figure 4.5. (iii) The migration cost is bounded as asserted in Theorem 4.6, and shown in Figure 4.4. (iv) The virtual vehicle concept works best when the task sizes are small and the vehicle spends more time traveling to tasks than it does loitering or standing still. As the virtual vehicles spend more time standing still, cloud computing in space converges to cloud computing in time as shown in Figure 4.6. (v) The provider can easily determine the appropriate number of real vehicles for a given number of virtual vehicles and a guaranteed performance isolation because the transition between low and high performance isolation is sharp as shown in Figure 4.7. (vi) EVDF has better performance than EDVDF, and EDVDF has better performance than the credit scheduling policy as shown in Figure 4.8.

Chapter 2

Stability of the Dynamic Traveling Repairman Problem

2.1 The Dynamic Traveling Repairman Problem

We establish a necessary and sufficient stability condition for the dynamic traveling repairman problem (DTRP) introduced by Bertsimas et al. [1] under the class of polling-sequencing (P-S) policies satisfying unlimited-polling and economy of scale. For the convenience of the reader, we restate the definition of the DTRP [1]: A convex region $\mathbf{A}$ of area A contains a vehicle (server) that travels at constant speed v. Tasks arrive according to a Poisson process with rate λ. Each task i is located at $X_i \in \mathbf{A}$, and has size B_i. X_i is independent and identically distributed (i.i.d.) with probability density function (pdf) $f_X(x)$, $x \in \mathbf{A}$. B_i is i.i.d. with pdf $f_B(s)$, $s \in [0, \infty)$. $E[B_i] = b$, which is assumed to be finite. Define load $\rho = \lambda b$. The system time of task i, denoted T_i, is defined as the elapsed time between the arrival of task i and the time task i is completed. When the system is stable, T_i converges to some T in distribution. T is called the steady state system time. This chapter is focused on stability conditions for this problem and the existence of stationary distributions for T and the queue length N.

2.1.1 Literature and Results

In queueing theory, $\rho < 1$ is a necessary and sufficient condition for all work conserving M/G/1 queues [15] (sec. II.4.2), where work conserving means the server will not be idle when there are tasks waiting in the queue. However, there is no such policy-independent stability condition for the DTRP, which seems to be a "spatial version" of the M/G/1 queue. Bertsimas et al. [1] showed that $\rho < 1$ is not a sufficient stability

condition for the work conserving policies of DTRP. For example, the DTRP can be unstable when $\rho < 1$ under the first come first served (FCFS) policy.

The known stability conditions for the DTRP are policy-dependent. Bertsimas et al. [16] argued $\rho + \lambda \frac{\bar{d}}{v} \leq 1$ as a necessary condition for stability of the DTRP under a policy, where $\bar{d} = \lim_{i\to\infty} E[D_i]$, and D_i denotes the distance traveled from task i to the next task served after i under the policy. The sufficiency of this condition for stability is established for some policies for single vehicle-DTRP (1-DTRP), by deriving an expression for the mean response time, $E[T]$, or an upper bound $\overline{E[T]}$, when $\rho \to 1^-$ [1, 2, 12]. The case of many capacitated vehicles with arbitrary demand distribution was investigated in [16, 39]. Table 2.1 summarizes known results and references. Since the sufficiency of this condition is policy dependent, stability is not proven for some reasonable policies such as nearest neighbor (NN) [1] or shortest job first (SJF, the task with the smallest task size is served first [10]) discussed in the literature. Assumption (45) in [1] has not been established formally. This chapter makes progress towards finding stability conditions for the DTRP that are less policy dependent than those in this literature as described next.

Table 2.1. Stability conditions for different policies for the 1-DTRP

Policies	$E[T]$	$\overline{E[T]}$	Stability conditions	Ref.
First Come First Served	Known	Known	$\rho + 0.52\lambda\sqrt{A} < 1$	[1]
Stochastic Queue Median	Known	Known	$\rho + 0.766\lambda\sqrt{A} < 1$	[1]
Partition-FCFS	Known	Known	$\rho + \frac{0.52}{r}\lambda\sqrt{A} < 1$	[1]
n-TSP	Unknown	Known	$\rho + \lambda\beta_{TSP}\frac{\sqrt{A}}{n} < 1$	[1]
Space filling curve	Unknown	Known	$\rho < 1$	[1]
Nearest neighbor	Unknown	Unknown	Unknown	[1]
Unbiased-TSP	Unknown	Known	$\rho < 1$	[16]
Biased-TSP	Unknown	Known	$\rho < 1$	[16]
PART-TSP	Unknown	Known	$\rho < 1$	[2]
Divide & conquer	Unknown	Known	$\rho < 1$	[12]
Receding horizon	Unknown	Known	$\rho < 1$	[12]

We establish $\rho + \lambda b_d < 1$ as a necessary and sufficient condition for the class of polling-sequencing (P-S) policies satisfying *unlimited-polling* (ULP, Definition 2.4) and *economy of scale* (EoS, Definition 2.2) in Theorem 2.5 in Section 2.2.3. This stability condition is identical to the necessary condition for stability given in [16]. The unlimited polling property is adapted from the one in the literature to this spatial queueing problem. The economy of scale property is particular to the spatial aspect of the DTRP. This is the main result. Theorem 2.1 shows the queue length is Markovian. Theorem 2.2 shows it to be aperiodic and irreducible. These are separated from Theorem 2.5 because the ULP assumption is sufficient for Theorems 2.1 and 2.2, while the ergodicity proof (Theorem 2.5) also requires policies to satisfy economy of scale.

A P-S policy has two phases: the polling phase allocates the arriving tasks to each polling station. The vehicle visits each polling station in cyclic order. The sequencing phase is a vehicle routing problem inside each polling station. ULP means in each polling station the number of tasks served goes to infinity as the number of waiting tasks goes to infinity. EoS is a property of the sequencing phase policy. EoS means the mean travel time to serve a task decreases as the number of tasks increases. The extra term b_d is defined as the limit of mean travel time as the number of tasks in a polling station goes to infinity. We prove the existence of b_d is a consequence of EoS (Theorem 2.4). b_d is policy-dependent, but it only depends on the sequencing phase of the P-S policy. Since the value of b_d can be derived in the static setting, we do not need to analyze or simulate the dynamic queueing system of DTRP to get b_d. One only needs to analyze or simulate to obtain the statistics of sequencing N tasks, with N a random variable, and the task locations distributed in some fashion.

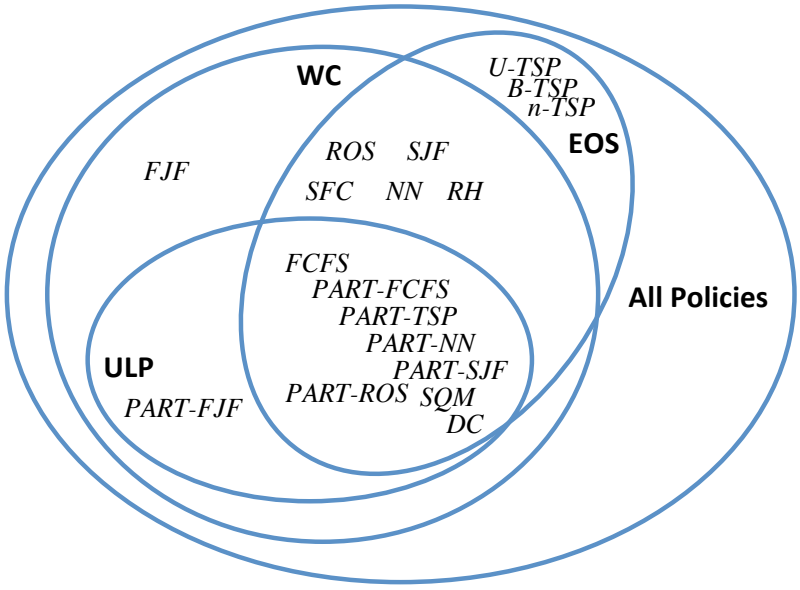


Figure 2.1. Classification of policies, WC = Work Conserving, EoS = Economy of Scale, ULP = Unlimited-Polling.

One can check that policies considered in [1, 2, 12, 40] such as FCFS, stochastic queue median (SQM), partition-FCFS (PART-FCFS), partition-traveling salesman policy (PART-TSP) and divide & conquer (DC) satisfy unlimited-polling and EoS. Other policies considered in [1, 2, 12, 40] such as space filling curve (SFC), NN, receding horizon (RH), and policies important in queueing theory but not considered for the DTRP such as SJF, and random order of service (ROS, tasks are served in a random order) satisfy EoS but not unlimited-polling. We define a polling version for these policies in Section 2.2, and name them PART-NN and PART-SJF. These satisfy both unlimited-polling and EoS. This paragraph is summarized by Figure 2.1. Theorem 2.3 proves the inclusions in the figure.

2.2 Polling-Sequencing Policies

The class of Polling-Sequencing (P-S) policies include a polling phase and a sequencing phase. For the notation of P-S policies, we use "PART-" to denote the polling (partitioning) phase, followed with the sequencing policies for the sequencing phase. For example, PART-TSP means first partition the region $\mathbf{A}$ into polling stations, and use TSP to sequence the tasks inside each polling station. Similarly, we can define PART-NN, PART-SJF, etc.

2.2.1 Spatial-Polling: Markov Chain

Polling policies are well-established in the queueing theory literature. Overviews and surveys of polling systems can be found in [41, 42, 43]. Stability and ergodicity criteria for polling systems are well established and can be found in [44, 45, 46, 47].

The polling phase of the P-S class is a *spatial-polling* policy, we divide the region $\mathbf{A}$ into an r-partition $\{\mathbf{A}^k\}_{k=1}^r$, each of area A^k. We label the r partitions as $1, 2, \ldots, r$. We regard each partition as a station in classic polling systems. In this way we generalize the polling system [44, 45] in classic queueing theory to the spatial case.

The vehicle visits the partitions in cyclic order, $1, 2, \ldots, r, 1, 2, \ldots$, and serves the tasks in each partition. Without loss of generality, we assume that the vehicle is initially at partition 1. Thus, the l-th visit of the vehicle is partition $I(l) = (l-1) \pmod r + 1$, where $l \pmod r$ means the remainder of the division of l by r. The set of tasks waiting in partition $I(l)$ on the arrival of the l-th visit of the vehicle is called the l-th queue (the queue observed at l-th visit).

We denote by:

$G^k(N)$ the number of tasks that are served in partition k when the queue observed is of length N.

$T_S^k(n)$ the total service time of n tasks in partition k. The formal definition of function $T_S^k(.)$ will be given in section 2.2.2.

$S^k(N)$ the duration of the service in partition k when the queue observed is of length N.

$$S^k(N) = T_S^k\left(G^k(N)\right) \tag{2.1}$$

Function $G^k(.)$ characterizes the polling policy and $T_S^k(.)$ characterizes the sequencing policy.

The *switch time* the vehicle takes from a random point in partition k to a random

point in partition $k+1$ is denoted by $\Delta^k, k = 1, \ldots, r-1$. The value from partition r to partition 1 is denoted by Δ^r. $\Delta^k, k = 1, \ldots, r$, are bounded above by the diameter of the region $\mathbf{A}$ divided by the speed of the vehicle v. The first moment of Δ^k is denoted by $\delta^k, k = 1, \ldots, r$. Let $\Delta = \sum_{k=1}^{r} \Delta^k$ be the total switch time in a cycle and denote by δ the first moment of Δ.

The tasks arrive at partition k with a Poisson process of parameter $\lambda^k = \int_{A^k} f_X(x)\, dx\lambda$. The task sizes are i.i.d. with common distribution B and mean b. Define $\rho^k = \lambda^k b$, and $\rho = \sum_{k=1}^{r} \rho^k$, $1 \leq k \leq r$. Let $N^k\,(t_1, t_2]$ denote the number of Poisson arrivals to partition k, $1 \leq k \leq r$, during a (random) time interval $(t_1, t_2]$. $N^k(t) \equiv N^k\,(0, t]$ is the number of Poisson arrivals in a time interval of length t.

The l-th value of the polling system is described by the random variables N_l^k, $1 \leq k \leq r$, $l \geq 1$, where N_l^k represents the number of tasks in partition k at the l-th visit of the vehicle. Let $N_l = (N_l^1, \ldots, N_l^r)$, taking values in $\{\mathbb{N}\}^r$, where $\mathbb{N}$ is the set of nonnegative integers.

Denote by S_l, the *station time*, the time interval between the arrival times of the l-th visit and the $(l+1)$st visit of the vehicle.

$$S_l = S^{I(l)}(N_l^{I(l)}) + \Delta^{I(l)} \tag{2.2}$$

Denote by C_l, the *cycle time*, the time interval between two successive arrivals of the vehicle to the same partition. $C_l = S_l + \ldots + S_{l+r-1}$.

The arrival times, the service times, the switch times are mutually independent, and are independent of the past and present system states. We adopt the rigorous independence definitions from Fricker [45] with some changes for the spatial case.

Consider a queue service starting at stopping time τ at partition k while N tasks are waiting and N^- tasks have already been served for the whole system. Let $\mathcal{F}_\tau$ be any σ-field containing the history of the service process up to random time τ. $\mathcal{F}_\tau$ is independent of the process $N(\tau, \tau + .]$ of arrivals after τ and of the task sizes $\{B_{N^-+i}\}_{i>0}$ of the tasks that have not been served up to time τ. The following four assumptions hold for all $k = 1, \ldots, r$.

A1: (G^k, S^k) is conditionally independent of $\mathcal{F}_\tau$ given N, and has the distribution of $\left(G^k(N), S^k(N)\right)$ where the expressions of the random functions $(G^k(.), T_S^k(.))$ are taken independent of N. i.e. The A-S policies do not depend on the past history of the service process such as the number of tasks being already served and the time spent serving them.

A2: (G^k, S^k) is independent of
$\left((B_{N^-+G^k+i})_{i>0}, N\left(\tau + S^k, \tau + S^k + .\right]\right)$, i.e. The selection of a task for service is independent of the required execution time and of possible future arrivals.

A3: $G^k(0) = 0$, $S^k(0) = 0$ and there exists $N > 0$ such that $G^k(N) > 0$. i.e. The vehicle leaves immediately a queue which is or becomes empty, but provides service with a positive probability once there are "enough" task(s) in the queue.

A4: $\left(G^k(N), S^k(N)\right)$ is *monotonic* and *contractive* in N. A function $g(.)$ is contractive if for every $x \geq y$, $g(x) - g(y) \leq x - y$.

N_l evolve according to the following evolution equations:

$$N_{l+1}^k = \begin{cases} N_l^k + N^k\left(S_l\right), & \text{if } I(l) \neq k \\ N_l^k - G^k\left(N_l^k\right) + N^k\left(S_l\right), & \text{if } I(l) = k \end{cases} \tag{2.3}$$

where $I(l) = (l-1) \pmod{r} + 1$.

The spatial polling system has a Markovian structure as specified by the following two theorems, which is almost identical to the theorems given in [45].

Theorem 2.1. *The sequence $\{N_l\}_{l=0}^{\infty}$ is a Markov chain.*

Proof. At the l-th polling instant τ, the server starts serving queue l (if not empty, otherwise he starts switching to queue $l+1$) according to policy $G^{I(l)}$ while the state of all queues is given by N_l. The arrival processes after l are Poisson and are independent of $\mathcal{F}_\tau$; the service times and the switch times involved after τ are also independent of $\mathcal{F}_\tau$. Because these quantities are mutually independent, it follows that given N_l, the evolution of the system after τ is independent of $\mathcal{F}_\tau$, which ensures the Markov property of the sequence. □

Remark 2.1.1. *This Markov chain is in general not homogeneous because its transitions depend on l through $G^{I(l)}$ and $\Delta^{I(l)}$, and $I(l)$ is different for each l. One can check that theorem 2.1 also holds when the task arrival process is renewal. This guarantees the arrival processes after l are independent of $\mathcal{F}_\tau$.*

Theorem 2.2. *$\{N_{lr+k}\}_{l=0}^{\infty}$ is a homogeneous, irreducible and aperiodic Markov chain with state space $\{\mathbb{N}\}^r$, $k = 1, \ldots, r$, where r is the number of polling stations.*

Proof. $\{N_{lr+k}\}_{l=0}^{\infty}$ is a subsequence of the Markov chain $\{N_l\}_{l=0}^{\infty}$ and is thus also a Markov chain which is homogeneous because $I(lr+k) = k$ and $G^{I(lr+k)} = G^k$ for $l = 0, 1, 2, \ldots$.

It is irreducible because all states communicate. Indeed, $(N^1, \ldots, N^r)$ can be reached in one step from the state $(0, \ldots, 0)$: this is realized when first no arrivals occur to all queues during the whole cycle but the last switch time Δ^{r-1}, and then the last switch time is positive and $(N^1, \ldots, N^r)$ arrivals occur during it, all this having a positive probability because the arrival processes are Poisson. On the other hand, $(0, \ldots, 0)$ is reached in (possibly) many steps from any state $(N^1, \ldots, N^r)$ with positive probability too: this is realized when there are no arrivals until it happens. By the same arguments, the state $(0, \ldots, 0)$ is aperiodic and so is the (irreducible) Markov chain. □

2.2.2 Sequencing: Economy of Scale

Under a spatial-polling policy, the number and locations of tasks are determined in each polling station in each polling cycle, which is a static vehicle routing problem. The sequencing policies sequence the set of tasks in each polling station.

Definition 2.1. *A policy for the 1-DTRP is called a Polling-Sequencing (P-S) policy if it runs a spatial-polling policy in region* $\mathbf{A}$*, and sequences the set of tasks in each polling station using some sequencing policy.*

For a set of n tasks $\{B_i, X_i\}_{i=1}^n$, each with size B_i and location X_i, denote by $T_D^P\left(\{X_i\}_{i=1}^n\right)$ and $T_S^P\left(\{B_i, X_i\}_{i=1}^n\right)$ the travel time and service time for the n tasks $\{B_i, X_i\}_{i=1}^n$ under sequencing policy P.

$$T_D^P\left(\{X_i\}_{i=1}^n\right) \equiv E_X\left[\frac{1}{v} D^P\left(X, \{X_i\}_{i=1}^n\right)\right] \tag{2.4}$$

where v is the vehicle speed, and $D^P\left(X, \{X_i\}_{i=1}^n\right)$ is the distance travelled by the vehicle to serve the tasks $\{B_i, X_i\}_{i=1}^n$ starting from a random point X in region $\mathbf{A}$ under sequencing policy P.

$$T_S^P\left(\{B_i, X_i\}_{i=1}^n\right) = T_D^P\left(\{X_i\}_{i=1}^n\right) + \sum_{i=1}^n B_i \tag{2.5}$$

Define $T_D^P(n) \equiv E_{\{X_i\}}\left[T_D^P\left(\{X_i\}_{i=1}^n\right)\right]$,

and $T_S^P(n) \equiv E_{\{X_i\}}\left[E_{\{B_i\}}\left[T_S^P\left(\{B_i, X_i\}_{i=1}^n\right)\right]\right]$, then

$$T_S^P(n) = T_D^P(n) + nb \tag{2.6}$$

$T_S^k(n) \equiv T_S^P(n)$ and $T_D^k(n) \equiv T_D^P(n)$ when sequencing policy P is used in partition k.

Definition 2.2. *A sequencing policy* P *is said to have economy of scale (EoS) if* $\frac{T_D^P(n)}{n}$ *is nonincreasing in* n*.*

Definition 2.3. *A scheduling policy is called non-location based if the distance between two consecutively executed tasks is i.i.d.. A scheduling policy is called location based if the distance between two consecutively executed tasks is dependent.*

Non-location based policies include FCFS, SJF, ROS and longest job first (LJF). Theorem 2.3(i) shows that non-location based policies satisfy economy of scale.

Location based policies include NN, furthest job first (FJF), TSP and the approximation algorithms for TSP such as Daganzo's algorithm (DA) [4]. For location based policies, there are two categories. One category try to find a shorter path connecting

the locations of the tasks, which we call *smart*. Examples include TSP, NN and the approximation algorithms of TSP such as Daganzo's algorithm (DA) [4]. The other category tries to find a longer path connecting the locations of the tasks, which we call *foolish*. Examples include furthest job first (FJF). This category does not has EoS, and is not practical. Theorem 2.3(ii) below proves that the common policies in the smart category such as TSP, NN and DA have EoS. Other policies in this category can be checked by the similar analysis or through simulation. Theorem 2.3(ii) also proves that FJF does not satisfy EoS. NN and TSP are well known. In DA, one cuts a swath of approximate width, w, covering the region $\mathbf{A}$. One possible pattern is shown in the left of Figure 2.2 with a swath of width $\frac{\sqrt{A}}{6}$. The vehicle visits the task locations by moving along the swath without backtracking.

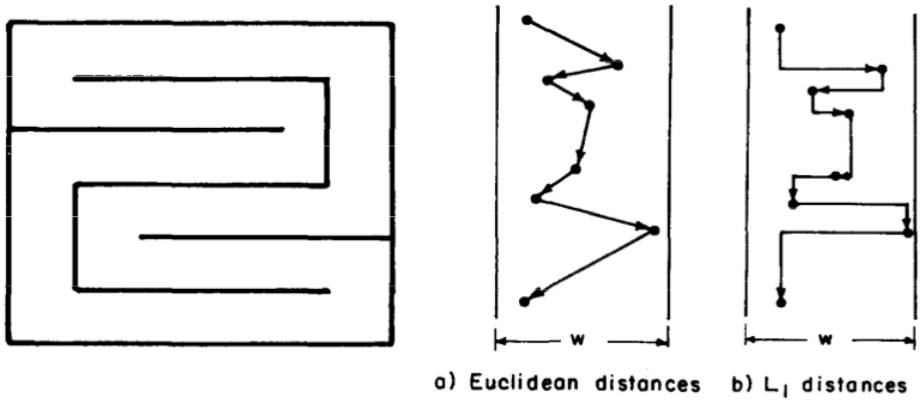


Figure 2.2. Daganzo's Algorithm, cited from [4].

Theorem 2.3. *(i) Non-location based policies satisfy economy of scale. (ii) Nearest neighbor, traveling salesman policy and Daganzo's algorithm in the location based policy class satisfy economy of scale, furthest job first in the location based policy class does not satisfy economy of scale.*

Proof. (i) This is because that X_i is independent of the arrival process, and X_i is independent of X_{i-1}. Non-location based policies do not sequence based on the locations of tasks, so $T_D^P(n) = E\left[\frac{\sum_{i=1}^n D_i}{v}\right] = \frac{\sum_{i=1}^n E[D_i]}{v}$, where $D_i = \| X_i - X_{i-1} \|$ when $i > 1$, $D_1 = \| X_1 - X_v \|$, where X_i is the location of the i-th task and X_v is the initial position of the vehicle. X_i and X_v are i.i.d. with pdf $f_X(x)$. Thus $E[D_i]$ is a constant, say d. Then $\frac{T_D^P(n)}{n} = \frac{nd}{n} = d$, which is nonincreasing in n. So non-location based policies satisfy EoS.

(ii) Under TSP, when there are n tasks, there are $n!$ *Hamiltonian paths* starting from the initial position of the vehicle. A Hamiltonian path is a path that visits each X_i exactly once. The length of each Hamiltonian path is the sum of n i.i.d. D_i's, $HP = \sum_{i=1}^n D_i$. The TSP tour is the Hamiltonian path with minimum lengths among the $n!$ Hamiltonian paths. Let L_n be the tour length, then $\frac{T_D^P(n)}{n} = \frac{E[L_n]}{nv}$. When there are $n+1$ tasks, there are $(n+1)!$ Hamiltonian paths, and L_{n+1} is the

shortest of them. $P\left(\frac{L_{n+1}}{n+1} > y\right) \leq P\left(\frac{L_n}{n} > y\right)$ because L_{n+1} is the minimum of $(n+1)!$ Hamiltonian paths and L_n is the minimum of $n!$ Hamiltonian paths. $\frac{E[L_{n+1}]}{n+1} = \int_0^\infty P\left(\frac{L_{n+1}}{n+1} > y\right) dy \leq \int_0^\infty P\left(\frac{L_n}{n} > y\right) dy = \frac{E[L_n]}{n}$. Then $\frac{T_D^P(n+1)}{n+1} \leq \frac{T_D^P(n)}{n}$. TSP satisfies EoS.

Under NN, when there are n tasks, Let L_n^{NN} be the length of the tour connecting the initial vehicle position and the locations of the n tasks, then $\frac{T_D^P(n)}{n} = \frac{E\left[L_n^{NN}\right]}{nv}$. L_n^{NN} is composed of n segments, $L_n^{NN} = \sum_{i=1}^n D_i^{NN}$, label i backwards such that D_i^{NN} is the distance from the $(n-i)$-th point to the $(n-i+1)$-th point when $i = 1, \ldots, n-1$, and D_n is the distance from the initial position of the vehicle to the 1st point. So D_i^{NN} is the minimum of i D_j's, where each D_j is the distance between two random points in the region $\mathbf{A}$. Thus $P\left(D_{i+1}^{NN} > y\right) \leq P\left(D_i^{NN} > y\right)$, this implies $E\left[D_{i+1}^{NN}\right] \leq E\left[D_i^{NN}\right]$, thus $\frac{E\left[L_{n+1}^{NN}\right]}{n+1} = \frac{\sum_{i=1}^{n+1} E\left[D_i^{NN}\right]}{n+1} \leq \frac{\sum_{i=1}^{n} E\left[D_i^{NN}\right] + \sum_{i=1}^{n} \frac{1}{n} E\left[D_i^{NN}\right]}{n+1} = \frac{\sum_{i=1}^{n} \frac{n+1}{n} E\left[D_i^{NN}\right]}{n+1} = \frac{E\left[L_n^{NN}\right]}{n}$. So $\frac{T_D^P(n+1)}{n+1} \leq \frac{T_D^P(n)}{n}$. NN satisfies EoS.

In [4], the swath was approximated to be a infinitely long strip of width w neglecting corner effect as shown in the right two of Figure 2.2. The mean travel time per task when serving n tasks, $\frac{T_D^P(n)}{n} = \frac{nd_w}{n} = d_w$, where d_w is the expected distance between two consecutive locations. Let X denote the random distance between two consecutive points along the width of the strip, and Y the distance along the side of the strip, then $E[X] = \frac{w}{3}$, $E[Y] = \frac{A}{nw}$ according to [4]. $d_w = E_{X,Y}\left(\sqrt{X^2+Y^2}\right)$ for the Euclidean metric. $d_w \approx \frac{w}{3} + \frac{A}{nw}\psi\left(\frac{nw^2}{A}\right)$, where $\psi(x) = \frac{2}{x^2}\left((1+x)log(1+x) - x\right)$. $w^* = \sqrt{\frac{2.95A}{n}}$ minimizes d_w. Substituting w^*, we see d_w is decreasing with n. Thus $\frac{T_D^P(n)}{n}$ is nonincreasing in n.

We show that FJF does not satisfy EoS by a counterexample. Consider a square of size 1×1 with uniformly distributed task locations. $\frac{T_D^P(1)}{1} = E[D_1] = 0.52$, where $D_1 = \parallel X_1 - X_v \parallel$, where X_i is the location of the i-th task and X_v is the initial position of the vehicle. X_i and X_v are i.i.d. with pdf $f_X(x) = 1$. When there are two tasks, the vehicle will choose the task further away, thus $\frac{T_D^P(2)}{2} > 0.52 = \frac{T_D^P(1)}{1}$. Thus FJF does not satisfy EoS. □

Remark 2.3.1. *Non-location based policies have trivial EoS in the sense that* $\frac{T_D^P(n)}{n}$ *is a constant.*

Theorem 2.3(ii) is supported by the simulation results in Figure 2.3. The simulations are done in a square $\mathbf{A}$ of size 1×1. The task locations and the initial vehicle position are generated independently from a uniform distribution with pdf $f_X(x) = 1$. The length of the path connecting the vehicle and the tasks is calculated under TSP, NN and DA for different number of points n.

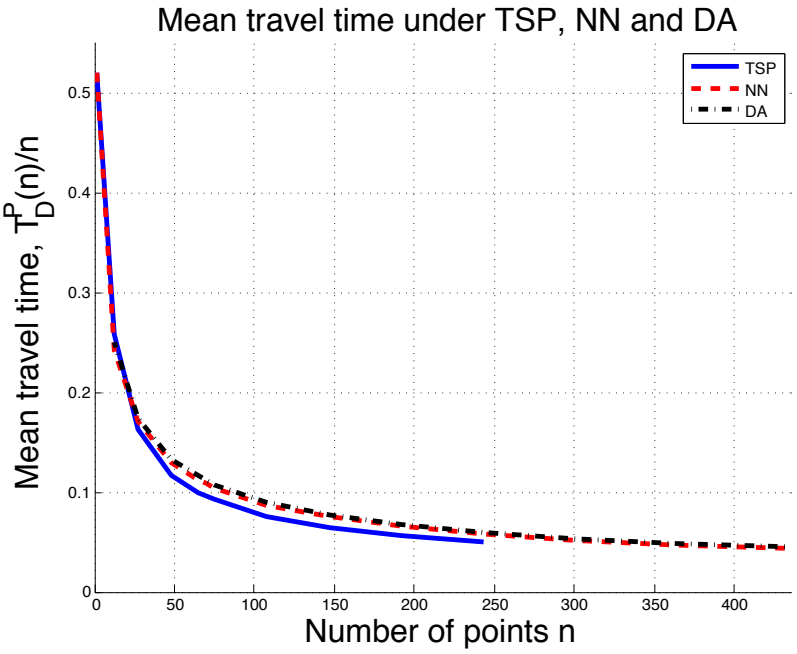


Figure 2.3. Mean travel time under TSP, NN and DA.

Theorem 2.4. *Under a sequencing policy P with economy of scale,* $\lim_{n\to\infty} \frac{T_D^P(n)}{n} = b_d \geq 0$, *and* $\exists M > 0$, *s.t.* $M \geq \frac{T_D^P(n)}{n} \geq b_d$ *for all* n.

Proof. $\frac{T_D^P(n)}{n} \geq 0$ and $\frac{T_D^P(n)}{n}$ is nonincreasing in n imply that $\lim_{n\to\infty} \frac{T_D^P(n)}{n}$ exists, say b_d.
Thus we have $\lim_{n\to\infty} \frac{T_D^P(n)}{n} = b_d$ and $M = \frac{T_D^P(1)}{1} \geq \frac{T_D^P(n)}{n} \geq b_d \geq 0$. □

Remark 2.4.1. *b_d is a measure of how well the sequencing policy can take advantage of the task locations. Let L_n denote the length of the tour connecting n points in a square of area A under TSP. From [48] we know that* $\lim_{n\to\infty} \frac{L_n}{\sqrt{n}} = \beta_{TSP}\sqrt{A}$, *where* $\beta_{TSP} \approx 0.72$, *thus* $b_d = \lim_{n\to\infty} \frac{T_D^P(n)}{n} = \lim_{n\to\infty} \frac{E[L_n]}{vn} = \lim_{n\to\infty} \beta_{TSP} \frac{\sqrt{A}}{v\sqrt{n}} = 0$. *This implies that TSP does best in taking advantage of the task locations.*

2.2.3 Stability Condition

Stability of DTRP is more complicated than in queueing theory because the stability of DTRP is policy dependent, whereas in queueing theory we have the policy-independent stability condition $\rho < 1$ for work conserving M/G/1 queues. Theorem 2.1 and 2.2 showed that $\{N_l\}_{l=0}^{\infty}$ is a Markov chain, and $\{N_{lr+k}\}_{l=0}^{\infty}$ is a homogeneous, irreducible and aperiodic Markov chain. We check the ergodicity of $\{N_{lr+k}\}_{l=0}^{\infty}$ and the stability of the DTRP under the P-S policies in this section.

Definition 2.4. *A polling policy characterized by $G^k(.)$ is called an unlimited-polling policy if $G^k(N) \to \infty$, when $N \to \infty$, $k = 1, \ldots, r$.*

One can check that the common polling policies such as the exhaustive and gated policies in [41] are unlimited-polling policies.

Lemma 2.1. *(LEMMA 3.1 in [44]) If for all $1 \leq k \leq r$ the Markov chains $\{N_{lr+k}\}_{l=0}^{\infty}$ are ergodic, then for all $1 \leq k \leq r$ $\{N_{lr+k}\}_{l=0}^{\infty}$ together with the sequence of station times $\{S_{lr+k}\}_{l=0}^{\infty}$ and the cycle times $\{C_{lr+k}\}_{l=0}^{\infty}$ converge weakly to finite random variables.*

Definition 2.5. *The DTRP under a P-S policy is said to be stable if all the r Markov chains $\{N_{lr+k}\}_{l=0}^{\infty}$ are ergodic.*

Lemma 2.2. *(Foster's Criterion [49, p.19]): Suppose a Markov chain is irreducible and let E_0 be a finite subset of the state space E. Then the chain is positive recurrent if for some $h : E \to \mathbb{R}$ and some $\epsilon > 0$ we have $\inf_x h(x) > -\infty$ and*

i) $\sum_{k \in E} p_{jk} h(k) < \infty, j \in E_0$,

ii) $\sum_{k \in E} p_{jk} h(k) \leq h(j) - \epsilon, j \notin E_0$.

where p_{jk} is the transition probability of the chain.

Theorem 2.5. *(Stability theorem): For any P-S policy with polling policy satisfying Definition 2.4 (unlimited-polling) and sequencing policy P satisfying $\lim_{n\to\infty} \frac{T_D^k(n)}{n} = b_d^k$ in each partition $\mathbf{A}^k$, assuming the partitions $\{\mathbf{A}^k\}_{k=1}^r$ are divided such that $b_d^k = \lim_{n\to\infty} \frac{T_D^P(n)}{n} = b_d$, $\forall 1 \leq k \leq r$, then when $\rho + \lambda b_d < 1$, the Markov chains $\{N_{lr+k}\}_{l=0}^{\infty}$ are ergodic, $\forall 1 \leq k \leq r$. Moreover, if the sequencing policy P satisfies Definition 2.2 (EoS), then $\rho + \lambda b_d < 1$ is necessary for the ergodicity of $\{N_{lr+k}\}_{l=0}^{\infty}$.*

Proof. Sufficiency: taking a conditional expectation in (2.3), summing over k, and substituting (2.2) and (4.3) we obtain:

$$E\left[\sum_{k=1}^r bN_{l+1}^k \,|N_l\right] = \sum_{k=1}^r bN_l^k - bG^{I(l)}\left(N_l^{I(l)}\right) + E\left[\sum_{k=1}^r bN^k\left(\Delta^{I(l)}\right)|N_l\right]$$
$$+ E\left[\sum_{k=1}^r bN^k\left(T_S^{I(l)}\left(G^{I(l)}\left(N_l^{I(l)}\right)\right)\right)|N_l\right]$$

$$= \sum_{k=1}^r bN_l^k - bG^{I(l)}\left(N_l^{I(l)}\right) + E\left[\sum_{k=1}^r bN^k\left(\Delta^{I(l)}\right)\right]$$
$$+ \sum_{k=1}^r bE[N^k(\sum_{i=1}^{G^{I(l)}(N_l^{I(l)})} B_i + T_D^{I(l)}(G^{I(l)}(N_l^{I(l)})))\,|N_l]$$

$$= \sum_{k=1}^r bN_l^k - bG^{I(l)}\left(N_l^{I(l)}\right) + \sum_{k=1}^r b\lambda_k E\left[\Delta^{I(l)}\right]$$
$$+ \sum_{k=1}^r b\lambda_k\left(G^{I(l)}\left(N_l^{I(l)}\right)b + T_D^{I(l)}\left(G^{I(l)}\left(N_l^{I(l)}\right)\right)\right)$$

$= \sum_{k=1}^{r} bN_l^k - bG^{I(l)}\left(N_l^{I(l)}\right) + \sum_{k=1}^{r} b\lambda_k \delta^{I(l)} + \rho\left(G^{I(l)}\left(N_l^{I(l)}\right) b + T_D^P\left(G^{I(l)}\left(N_l^{I(l)}\right)\right)\right)$

$= \sum_{k=1}^{r} bN_l^k + \rho\delta^{I(l)} + \left(\rho - 1 + \lambda \frac{T_D^P\left(G^{I(l)}\left(N_l^{I(l)}\right)\right)}{G^{I(l)}\left(N_l^{I(l)}\right)}\right) bG^{I(l)}\left(N_l^{I(l)}\right).$

Define $\gamma^k = \rho - 1 + \lambda \frac{T_D^P\left(G^{I(l+k)}\left(N_{l+k}^{I(l+k)}\right)\right)}{G^{I(l+k)}\left(N_{l+k}^{I(l+k)}\right)}, k = 0, \ldots, r-1,$

then $E\left[\sum_{k=1}^{r} bN_{l+1}^k \,|N_l\right] = \sum_{k=1}^{r} bN_l^k + \rho\delta^{I(l)} + \gamma^0 bG^{I(l)}\left(N_l^{I(l)}\right).$

Similarly, $E\left[\sum_{k=1}^{r} bN_{l+2}^k \,|N_l\right] = E\left[E\left[\sum_{k=1}^{r} bN_{l+2}^k \,|N_{l+1}, N_l\right]|N_l\right]$

$= E\left[E\left[\sum_{k=1}^{r} bN_{l+2}^k \,|N_{l+1}\right]|N_l\right]$

$= E\left[\sum_{k=1}^{r} bN_{l+1}^k \,|N_l\right] + \rho\delta^{I(l+1)} + E\left[\gamma^1 bG^{I(l+1)}\left(N_{l+1}^{I(l+1)}\right)|N_l\right].$

Since $N_{l+1}^{I(l+1)} = N_l^{I(l+1)} + N^{I(l+1)}(S_l) \geq N_l^{I(l+1)}$,
and $G_k(.)$ is nondecreasing, then
$E\left[G^{I(l+1)}\left(N_{l+1}^{I(l+1)}\right)|N_l\right] \geq E\left[G^{I(l+1)}\left(N_l^{I(l+1)}\right)|N_l\right] = G^{I(l+1)}\left(N_l^{I(l+1)}\right).$

$\rho + \lambda b_d < 1$ implies $\epsilon_1 = \frac{1-\rho-\lambda b_d}{\lambda} > 0.$

Since $\lim_{n\to\infty} \frac{T_D^P(n)}{n} = b_d \geq 0,$
then $\exists M_1 > 0$, s.t. $n > M_1$ implies $\frac{T_D^P(n)}{n} - b_d < \epsilon_1$, i.e. $\rho - 1 + \lambda \frac{T_D^P(n)}{n} < 0.$

Thus when $G^{I(l+1)}\left(N_l^{I(l+1)}\right) > M_1$, $G^{I(l+1)}\left(N_{l+1}^{I(l+1)}\right) > M_1$, $\gamma^1 < 0.$
This implies $E\left[\gamma^1 bG^{I(l+1)}\left(N_{l+1}^{I(l+1)}\right)|N_l\right] \leq \gamma^1 bG^{I(l+1)}\left(N_l^{I(l+1)}\right).$

So when $G^{I(l+1)}\left(N_l^{I(l+1)}\right) > M_1,$
$E\left[\sum_{k=1}^{r} bN_{l+2}^k \,|N_l\right] \leq E\left[\sum_{k=1}^{r} bN_{l+1}^k \,|N_l\right] + \rho\delta^{I(l+1)} + \gamma^1 bG^{I(l+1)}\left(N_l^{I(l+1)}\right)$
$= \sum_{k=1}^{r} bN_l^k + \rho\left(\delta^{I(l)} + \delta^{I(l+1)}\right) + \gamma^0 bG^{I(l+1)}\left(N_l^{I(l)}\right) + \gamma^1 bG^{I(l+1)}\left(N_l^{I(l+1)}\right).$

Repeating the above calculation, we obtain

$E\left[\sum_{k=1}^{r} bN_{l+r}^k \,|N_l\right] \leq \sum_{k=1}^{r} bN_l^k + \rho\delta + \sum_{k=0}^{r-1} \gamma^k bG^{I(l+k)}\left(N_l^{I(l+k)}\right),$
when $G^{I(l+k)}\left(N_l^{I(l+k)}\right) > M_1$, $k = 1, \ldots, r-1.$

Since $\gamma^k < 0$, when $G^{I(l+k)}\left(N_l^{I(l+k)}\right) > M_1$, $k = 0, \ldots, r-1,$

then $\exists M > M_1$, s.t.
$G^{I(l+k)}\left(N_l^{I(l+k)}\right) > M$ implies $-\epsilon = \rho\delta + \sum_{k=0}^{r-1}\gamma^k b G^{I(l+k)}\left(N_l^{I(l+k)}\right) < 0$.

Define $E_0 = \{N_l \in \mathbb{N}^r \left| G^{I(l+k)}\left(N_l^{I(l+k)}\right) \leq M, k = 1, \ldots, r\right.\}$,
then E_0 is a finite subset of the state space $\mathbb{N}^r$.

Define $h(N) = \sum_{k=1}^{r} bN^k$, since $b \geq 0$ and $N \in \mathbb{N}^r$, then $\inf_N h(N) > -\infty$.

It then follows that
$E\left[h\left(N_{l+r}\right)|N_l\right] \leq h\left(N_l\right) - \epsilon$, when $N_l \notin E_0$,
$E\left[h\left(N_{l+r}\right)|N_l\right] \leq \sum_{k=1}^{r} bN_l^k + \rho\delta + \sum_{k=0}^{r-1}\gamma^k b G^{I(l+k)}\left(N_l^{I(l+k)}\right)$, when $N_l \in E_0$.

Then $\{N_{lr+k}\}_{l=0}^{\infty}$ is positive recurrent by *Lemma 2.2* (Foster's Criterion), thus it is ergodic (irreducible, aperiodic and positive recurrent).

Necessity when economy of scale applies: Bertsimas et al. gave the necessary condition for stability in [16] $\rho + \lambda\frac{\bar{d}}{v} \leq 1$, where $\bar{d} = \lim_{i\to\infty} E\left[D_i\right]$, where D_i denotes the distance traveled from task i to the next task served after i, i.e. $\bar{d}$ is the steady state expected value of D_i. Let N^k be the number of tasks served in partition k in steady state. $P\left(N^k = n, X_i \in \mathbf{A}^k\right)$ denotes the probability that there are n tasks served in partition k in steady state and task i is one of them. Then
$\frac{\bar{d}}{v} = \sum_{k=1}^{r}\sum_{n=1}^{\infty}\frac{T_D^P(n)+\Delta}{n}P\left(N^k = n, X_i \in \mathbf{A}^k\right)$
$> \sum_{k=1}^{r}\sum_{n=1}^{\infty}\lim_{n\to\infty}\frac{T_D^P(n)}{n}P\left(N^k = n, X_i \in \mathbf{A}^k\right)$
$= b_d\sum_{k=1}^{r}\sum_{n=1}^{\infty}P\left(N^k = n, X_i \in \mathbf{A}^k\right) = b_d$.
So $\rho + \lambda b_d \leq \rho + \lambda\frac{\bar{d}}{v} < 1$. □

Remark 2.5.1. *The stability condition $\rho + \lambda b_d < 1$ has an additional term λb_d compared to $\rho < 1$ in queueing theory, where b_d is the mean travel time per task when $n \to \infty$.*

Remark 2.5.2. *By Lemma 2.1, ergodicity implies that the sequence of station times $\{S_{lr+k}\}_{l=0}^{\infty}$ and the cycle times $\{C_{lr+k}\}_{l=0}^{\infty}$ converge weakly to finite random variables. The i-th task arriving in partition k to be served in station time S_{lr+k} first spends time W_{Oi} to wait outside the previous cycle, $C_{(l-1)r+k}$, and spends time W_{Ii} inside the current station time S_{lr+k}. W_{Oi} and W_{Ii} are well defined based on $C_{(l-1)r+k}$ and S_{lr+k} under the P-S policy, and $W_{Oi} \leq C_{(l-1)r+k}$ and $W_{Ii} \leq S_{lr+k}$. So W_{Oi} and W_{Ii} converge weakly to finite random variables. Thus the system time $T_i = W_{Oi} + W_{Ii}$ converges weakly to finite random variable.*

2.3 Summary

We prove a necessary and sufficient condition for stability in Theorem 2.5 in the Dynamic Traveling Repairman Problem (DTRP) [1] under the class of Polling-Sequencing (P-S) policies (Definition 2.1) satisfying unlimited-polling (Definition 2.4) and economy of scale (Definition 2.2). The number of tasks inside each polling partition is shown to be a Markov chain in Theorem 2.1. Non-location based policies and some common location based policies such as TSP, NN and DA are shown to have economy of scale in Theorem 2.3. The P-S class includes some of the policies proven to be optimal for the expectation of system time under light and heavy loads in the DTRP literature.

Chapter 3

System Time Distribution in the Dynamic Traveling Repairman Problem

3.1 System Time Distribution

In the last chapter, we give a necessary and sufficient condition for the stability of the Dynamic Traveling Repairman Problem (DTRP) for the class of Polling-Sequencing (P-S) policies satisfying unlimited-polling and economy of scale. When the DTRP is stable, the distribution of the steady state system time T exists.

3.1.1 Literature and Results

To the best of our knowledge, the distribution of T for DTRP is only known under the FCFS policy by substituting the task size B_i with $\frac{D_i}{v} + B_i$ in the cdf of T for an M/G/1 queue under FCFS [15], where $D_i = \| X_i - X_{i-1} \|$.

For the expectation of system time $E[T]$, the lower bounds of $E[T]$ under light load ($\rho \to 0^+$) and heavy load ($\rho \to 1^-$) was given in [1]. The SQM policy was shown to be $E[T]$ optimal under light load in [1]. PART-TSP in [2] and DC in [12] were shown to be $E[T]$ optimal under light load and asymptotically optimal under heavy load when the number of partitions approaches infinity. The SQM, PART-TSP and DC policies fall in our P-S class. Other than these, there is no $E[T]$ optimal result for a general $\rho \in (0, 1)$. The expressions for $E[T]$ are only known for FCFS, SQM and PART-FCFS for a general $\rho \in (0, 1)$ [1] and not for other policies. Table 2.1 summarizes known results and references.

For the variance of system time $Var[T]$, Xu [2] analyzed the asymptotic behavior of $Var[T]$ under the PART-TSP policy under heavy load. The expression for $Var[T]$ is only known for FCFS for a general $\rho \in (0, 1)$. This lack of knowledge in $E[T]$ and $Var[T]$ in DTRP is in sharp contrast to queueing theory where both $E[T]$ and $Var[T]$ are known for a wide variety of policies as given in [10, 22].

Recent years have witnessed progress in the knowledge of $E[T]$ and $Var[T]$ in polling systems: Boxma et al. [23] gave the Laplace-Stieltjes transform (LST) of T, together with $E[T]$ and $E[T^2]$ for polling system with gated or globally gated policies when the sequencing policies are FCFS, LCFS, ROS and SJF. Dorsman et al. [24] provided closed form approximations for the distribution of the stead state waiting time of a task for polling systems under renewal arrival process with gated or exhaustive policies when the sequencing policy is FCFS. Since the polling phase of the P-S class for DTRP is a polling system, we can utilize these results for DTRP under the P-S policies.

Typical polling policies characterized by $G^k(.)$ defined in Section 2.2.1 were classified by Vishnevskii [43] into mainly two categories: *deterministic*, where $G^k(.)$ is a deterministic function, and *random*, where $G^k(.)$ is a random function. We focus on the deterministic policies. Polling policies can also be divided into unlimited-polling policies and limited-polling policies. We focus on the unlimited-polling policies because limited-polling policies are not efficient in the sense that they are not stable for some $\rho < 1$ even in classic queues [45]. Common deterministic unlimited-polling policies include *exhaustive* and *gated* policies:

- Exhaustive, where the server treats customers until the queue is emptied.
- Gated, where the server treats only those customers sojourned in the queue at the polling instant.

Most of the unlimited polling policies in the literature are either exhaustive or gated. We focus on the exhaustive policies because the exhaustive policies have smaller $E[T]$ than the gated policies [50].

In this chapter, we propose a policy in the P-S class called the PART-n-TSP policy. We give a good approximation for the distribution of the system time that is easy to compute by utilizing the approximation results of the distribution of system time T, together with $E[T]$ and $Var[T]$ in the polling systems [23, 24]. Figure 3.5 shows that the cumulative distribution function (cdf) of the system time by our approximation method and the cdf of the system time through simulation are very close. We show that FCFS, partitioning-FCFS and n-TSP [1] are special cases of PART-n-TSP, thus PART-n-TSP has better performance than the three by optimizing its parameters. We also compare PART-n-TSP with PART-TSP [2] and Nearest Neighbor [1] on $E[T]$ and $\sigma[T]$ in Tables 3.1 and 3.2, since the latter two are considered near optimal in

the literature. The results show that NN achieves lower $E[T]$ than PART-n-TSP and PART-TSP. PART-n-TSP achieves lower $E[T]$ than PART-TSP when ρ is not too small or too large, e.g. when $\rho \in \{0.3, \ldots, 0.7\}$. Also, PART-$n$-TSP achieves lower $\sigma[T]$ than PART-TSP and NN when ρ is not too small or too large, e.g. when $\rho \in \{0.3, \ldots, 0.7\}$. Since small ρ results in low utilization of real vehicles, and large ρ results in large system time of tasks, in practice ρ is neither too small or too large. Thus, PART-n-TSP is good in practice to achieve lower $\sigma[T]$ than PART-TSP and NN, and lower $E[T]$ than PART-TSP. We also prove that PART-n-TSP is $E[T]$ optimal under light load and asymptotically optimal under heavy load in Theorem 3.1.

3.2 PART-n-Traveling Salesman Policy

Bertsimas et al. [1] introduced the traveling salesman policy (TSP). It is based on collecting tasks into sets of size n that are then served in a TSP path. To be precise, we call this the n-TSP. This policy is a one-partition policy in the P-S class. We generalize it to multiple partitions as follows and call it the PART-n-TSP.

Definition 3.1. *A Polling-Sequencing policy in Definition 2.1 is call the PART-n-TSP policy if the sequencing phase is an n-TSP policy that collects tasks into sets of cardinality n, and then serve them using an optimal traveling salesman path.*

The polling phase involving generating an r-partition $\left\{\mathbf{A}^k\right\}_{k=1}^r$ of $\mathbf{A}$ that is simultaneously equitable with respect to $f(x)$. In particular, when the region $\mathbf{A}$ is a square region $\mathbf{A}$ with size $a \times a$, and the tasks are uniformly distributed in $\mathbf{A}$ with pdf $f(x) = \frac{1}{A}$, $\mathbf{A}$ is divided into $r = m^2$ square partitions, each has size $\frac{a}{m} \times \frac{a}{m}$, where $m > 1$ is a given integer that parameterizes the policy. The vehicle visits the partitions in a cyclic order. Partitions are numbered so that for any $k = 1, \ldots, r-1$, partition $k+1$ is adjacent to partition k, and partition r is adjacent to partition 1 when m is even, or to the diagonal of partition 1 when m is odd, as illustrated in Figure 3.1 for the case $m = 4$ and $m = 5$. The vehicle cycles through the partitions in the order $1, \ldots, r, 1, \ldots, r, \ldots$. After the vehicle finishes the tasks polled in the current partition under a sequencing policy, the vehicle moves to an adjacent partition and serves it under the same sequencing policy.

To move from one partition (polling station) to the next, the vehicle uses the projection rule shown in Figure 3.2 as introduced in [1]. Its last location in a given partition is simply "projected" onto the next partition to determine the server's new starting location. The vehicle then travels in a straight line between these two locations. This makes the distance traveled between partitions a constant, each starting location uniformly distributed, and independent of the locations of tasks in the new partition. In practice, one might use a more intelligent rule such as moving directly to the first

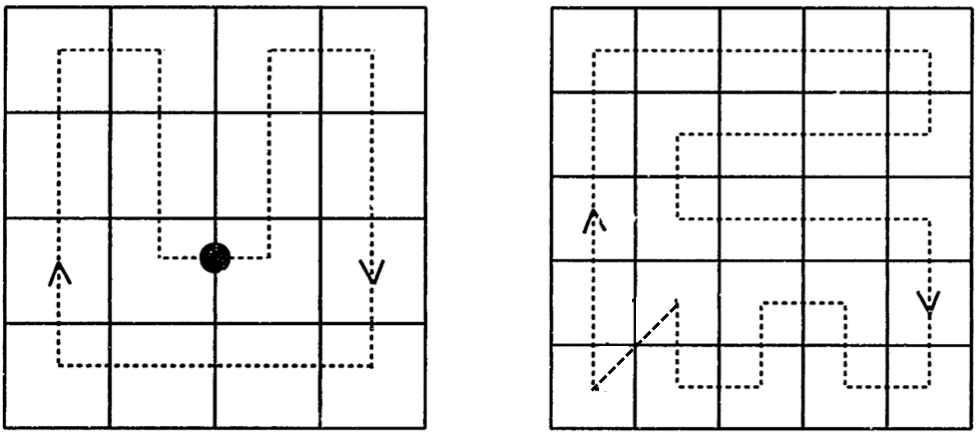


Figure 3.1. Order for serving partitions under the polling policy, cited and revised from [2].

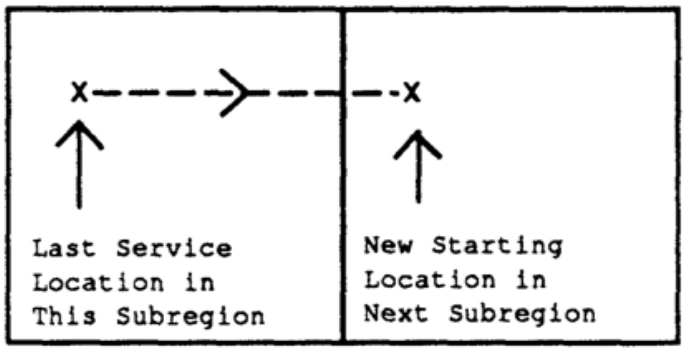


Figure 3.2. Vehicle moving to adjacent partition, cited from [1].

task in the next partition. When m is even, the vehicle always travels to an adjacent partition. Thus the switch time between two consecutive partitions is always $\Delta^k = \frac{a}{m}$, $k = 1, \ldots, r$. Thus $\Delta = \sum_{k=1}^{r} \Delta^k = ma$. When m is odd, the vehicle always travels to an adjacent partition except the last one. Thus $\Delta^k = \frac{a}{m}$ when $k = 1, \ldots, r-1$, and $\Delta^r = \frac{\sqrt{2}a}{m}$ as shown in Figure 3.1. Thus $\Delta = \sum_{k=1}^{r} \Delta^k = \frac{\left(m^2-1+\sqrt{2}\right)a}{m}$. Then we have

$$\Delta = \begin{cases} ma, & \text{if } m \text{ is even.} \\ \frac{\left(m^2-1+\sqrt{2}\right)a}{m}, & \text{if } m \text{ is odd.} \end{cases} \tag{3.1}$$

In the sequencing phase, we use the exhaustive n-TSP policy to sequence the tasks in each partition. This policy is adopted from the TSP policy in [1]. We repeat it for the convenience of the reader. Let $\mathcal{N}_l^k$ denote the l-th set of n tasks to arrive in partition k. Each set $\mathcal{N}_l^k$ has cardinality n. For example, $\mathcal{N}_1^k$ is the set of tasks $1, \ldots, n$ in partition k, and $\mathcal{N}_2^k$ is the set of tasks $n+1, \ldots 2n$ in partition k, and so on. To serve a set, we form a TSP path of the n tasks in the set starting at the initial position of the vehicle and ending at the location of the last task in the TSP path. A TSP path is the Hamiltonian path with the minimum length among all the Hamiltonian paths. A Hamiltonian path is a path that visits each task location exactly once starting at the initial position of the vehicle.

The vehicle starts at some location in partition 1. If all tasks in set $\mathcal{N}_1^1$ have arrived, we form a TSP path over these tasks. Tasks are then served by following the TSP path. If all $\mathcal{N}_2^1$ tasks have arrived when the TSP path of $\mathcal{N}_1^1$ is completed, they are also served using a TSP path. Otherwise, the vehicle moves to partition 2, and so on. The sets N_l^k are served in an FCFS order in each partition k.

We know $\rho + \lambda b_d < 1$ is the stability condition by Theorem 2.5, and the sequencing phase TSP has $b_d = 0$ by Remark 2.4.1. Thus PART-n-TSP is stable if and only if $\rho < 1$.

3.2.1 Calculation of System Time Distribution

The system time T of a task has three components:

- W_O, the time a task waits for its set to form (wait for the last task in the set to arrive).
- W_P, waiting time of the set in the polling system.
- W_I, the time it takes to complete service of the task once the task's set enters service.

Thus,

$$T = W_O + W_P + W_I \tag{3.2}$$

where W_O, W_P and W_I are independent. The distribution of T can be obtained from the distributions of W_O, W_P and W_I through convolution.

Distribution of W_O

We first obtain the distribution of W_O, together with its expectation and variance. Pick a random task. Let W_{Ol} be the waiting time of a task outside a set if it is the $(n-l)$-th task arrived in the set, $l = 0, \ldots, n-1$. Since we have equitable partitions, then the task arrival process inside each partition is Poisson with arrival rate $\frac{\lambda}{r}$. Thus $W_{O0} = 0$ and W_{Ol} is Erlang distributed with parameters $\left(l, \frac{\lambda}{r}\right)$, $l = 1, \ldots, n-1$. Thus the cdf of W_{Ol}, $F_{W_{Ol}}\left(t; l, \frac{\lambda}{r}\right) = 1 - \sum_{j=0}^{l-1} \frac{1}{j!} e^{-\frac{\lambda}{r}t} \left(\frac{\lambda}{r}t\right)^j$, $E\left[W_{Ol}\right] = \frac{lr}{\lambda}$, and $E\left[W_{Ol}^2\right] = \frac{\left(l^2+l\right)r^2}{\lambda^2}$.

Since it is equally probable that a task is the $(n-l)$-th arrived task in the set, then

$P\left(W_O \leq t\right) = \sum_{l=0}^{n-1} P\left(W_{Ol} \leq t\right) \frac{1}{n}$
$= \frac{1}{n}\left(1 + \sum_{l=1}^{n-1}\left(1 - \sum_{j=0}^{l-1} \frac{1}{j!} e^{-\frac{\lambda}{r}t} \left(\frac{\lambda}{r}t\right)^j\right)\right)$.

$E\left[W_O\right] = \sum_{l=0}^{n-1} E\left[W_{Ol}\right] \frac{1}{n} = \frac{1}{n} \sum_{l=0}^{n-1} \frac{lr}{\lambda} = \frac{(n-1)r}{2\lambda}$.

$E\left[W_O^2\right] = \sum_{l=0}^{n-1} E\left[W_{Ol}^2\right] \frac{1}{n} = \frac{1}{n} \sum_{l=0}^{n-1} \frac{\left(l^2+l\right)r^2}{\lambda^2} = \frac{\left(n^2-1\right)r^2}{3\lambda^2}$.

$Var\left[W_O\right] = E\left[W_O^2\right] - E\left[W_O\right]^2 = \frac{\left(n^2+6n-7\right)r^2}{12\lambda^2}$.

To sum up,

$$P\left(W_O \leq t\right) = \frac{1}{n}\left(1 + \sum_{l=1}^{n-1}\left(1 - \sum_{j=0}^{l-1} \frac{1}{j!} e^{-\frac{\lambda}{r}t} \left(\frac{\lambda}{r}t\right)^j\right)\right) \tag{3.3}$$

$$E\left[W_O\right] = \frac{(n-1)r}{2\lambda} \tag{3.4}$$

$$Var\left[W_O\right] = \frac{\left(n^2+6n-7\right)r^2}{12\lambda^2} \tag{3.5}$$

Distribution of W_I

Before discussing W_P, we compute the distribution of W_I, together with $E[W_I]$ and $Var[W_I]$ as follows. Let W_{Inj} be the waiting time of a task inside a set if it is the j-th task to be served in the set of n tasks, $j = 1, \ldots, n$. Then

$$W_{Inj} = \frac{D_{nj}}{v} + \sum_{i=1}^{j} B_i \tag{3.6}$$

where D_{nj} is the travel distance from the initial vehicle position to the location of the j-th task through a TSP path in a partition. B_i is i.i.d.. D_{nj} is independent of B_i. Thus $\sum_{j=1}^{k} B_i$ is the convolution of j B_i's, and W_{Inj} is the convolution of $\frac{D_{nj}}{v}$ and $\sum_{i=1}^{k} B_i$.

Let L_{nj} be the D_{nj} value when there is only one partition, i.e., $r = 1$. When $r > 1$, we assume that

$$D_{nj} =_d \frac{cL_{nj}}{\sqrt{r}} \tag{3.7}$$

where $=_d$ means identically distributed with, and c is a positive constant. In particular, when region $\mathbf{A}$ and all the partitions $\mathbf{A}^k$ are squares, and $r = m^2$ with m an integer, $c = 1$.

We obtain the empirical distribution of L_{nj}, together with $E[L_{nj}]$ and $Var[L_{nj}]$ for different n and $j = 1, \ldots, n$ through simulation on the TSP path. The ant colony optimization algorithm [51] is used to heuristically search for the TSP path. Both the number of ants and the number of iterations are set to 1000. The distribution of D_{nj} is calculated from L_{nj} by scaling in (3.7). We write L_n for L_{nn} and D_n for D_{nn}. Figure 3.3 shows the values of $E[L_n]$ and $Var[L_n]$ for different n when the region is a square of size 1×1.

Figure 3.4 shows the pdf of L_{nj} for a set of $n = 5$ tasks. The tasks are uniformly distributed on a square of 1×1.

Since it is equally probable that the task is the k-th served task in the TSP path, $k = 1, \ldots, n$, then

$$P(W_I \leq t) = \frac{1}{n} \sum_{j=1}^{n} P(W_{Inj} \leq t) \tag{3.8}$$

Then the pdf of W_I, $f_{W_I}(t) = \frac{1}{n} \sum_{j=1}^{n} f_{W_{Inj}}(t)$.

$E[W_I] = \frac{1}{n} \sum_{j=1}^{n} E[W_{Inj}]$.

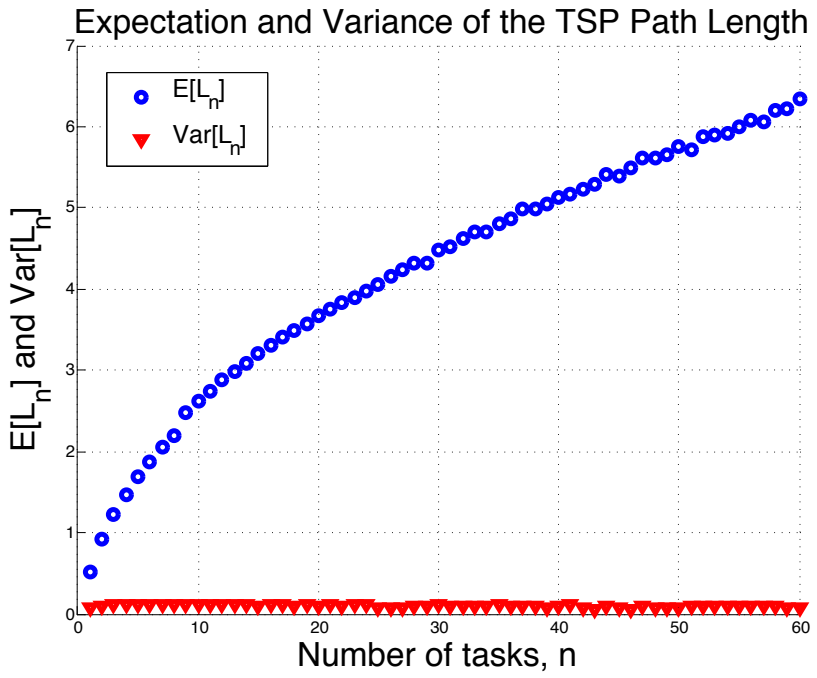


Figure 3.3. Expectation and variance of the length of the TSP path for n tasks.

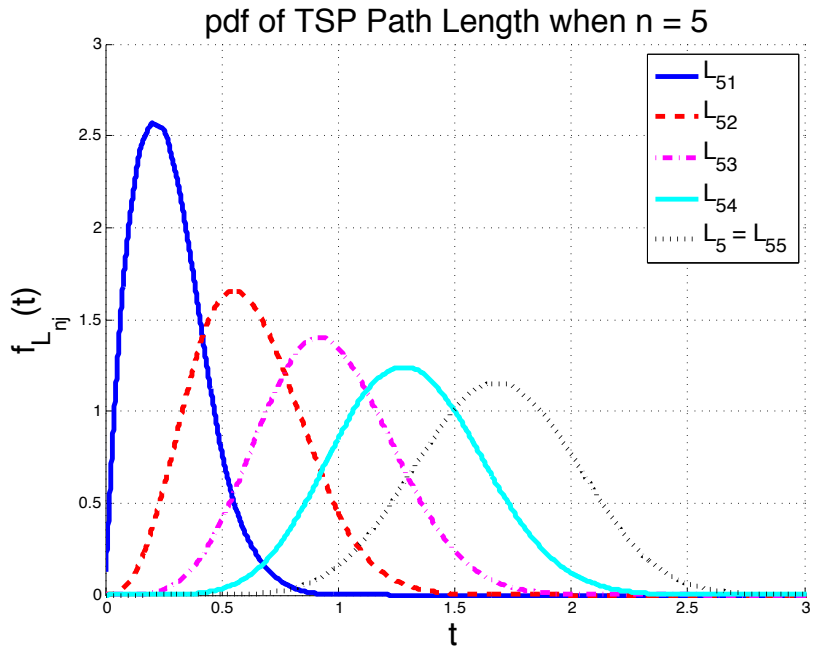


Figure 3.4. The pdf of W_{Inj} and W_I.

$E[W_I^2] = \frac{1}{n}\sum_{j=1}^{n} E\left[W_{Inj}^2\right]$.

$Var[W_I] = E[W_I^2] - E[W_I]^2$.

The distribution of W_I can be calculated from (3.6) and (3.8). W_I does not have a closed form, but can be arbitrarily accurate through simulation. Observe that in order to obtain the distribution of D_{nj} and W_I for partitions with different sizes parameterized by r, we do not need to rerun the simulation for each partition with different size. We only need to run it once for L_{nj} and store the data. D_{nj} and W_I are obtained by scaling and convolution.

Distribution of W_P

The analysis of W_P uses the results from [24] by establishing the PART-n-TSP to be equivalent to a classic polling system over jobs that are the sets $\mathcal{N}_l^k$.

Since the task arrival process is Poisson with arrival rate λ, the distribution of the interarrival time of sets, A, is Erlang of order n and arrival rate λ, i.e., $A \sim Erlang(n, \lambda)$. Let A_k be the interarrival time of sets that fall in partition $\mathbf{A}^k$. Then $A_k \sim Erlang(n, \frac{\lambda}{r})$. Thus

$$E[A_k] = \frac{nr}{\lambda}, Var[A_k] = \frac{nr^2}{\lambda^2} \tag{3.9}$$

The arrival rate of a set is

$$\lambda^s = \frac{\lambda}{n} \tag{3.10}$$

The arrival rate of a set in partition $\mathbf{A}^k$, $k = 1, \ldots, r$, is

$$\lambda_k^s = \frac{\lambda^s}{r} = \frac{\lambda}{nr} \tag{3.11}$$

The size of a set, or the time needed to travel to and execute all the tasks in the set, is W_{Inn} as given in (3.6). We write W_n for W_{Inn}. The size of each set W_n is i.i.d.. Thus, if we treat each set as a job with size W_n, and each partition as a polling station, then the system is a classic polling system on r polling stations with renewal (Erlang) arrival of rate λ^s, job size W_n, and switch time Δ^k. The load is

$$\rho^s = \lambda^s E[W_n] \tag{3.12}$$

The load in partition $\mathbf{A}^k$, $k = 1, \ldots, r$, is

$$\rho_k^s = \frac{\rho^s}{r} \tag{3.13}$$

W_P is the waiting time of each set (job) in this classic polling system. Exhaustive or gated PART-n-TSP correspond to exhaustive or gated FCFS on sets, respectively.

Dorsman et al. [24] provide closed form approximations for the distribution of the steady state waiting time of a job, W_P, for polling systems under a renewal arrival process with gated or exhaustive policies when the sequencing policy is FCFS. They claim that for exhaustive-FCFS policies,

$$P\left(W_P \leq t\right) \approx P\left(UI \leq (1-\rho^s)t\right) \tag{3.14}$$

where U is uniformly distributed on $[0,1]$, and I is Gamma distributed with parameters

$$\alpha = \frac{2E[\Delta]\delta}{\sigma^2} + 1, \beta = \frac{2E[\Delta]\delta + \sigma^2}{2\sigma^2(1-\rho^s)E\left[W_{Boon}\right]}. \tag{3.15}$$

where $\Delta = \sum_{k=1}^{r} \Delta^k$ is the total switch time in a cycle. When the region $\mathbf{A}$ and partitions $\mathbf{A}^k$ are squares as shown in Figure 3.1, Δ is given in (3.1). ρ^s is given in (3.12).

To explain δ, σ^2 and $E\left[W_{Boon}\right]$, we denote by $\hat{y}$ the value of each variable y that is a function of ρ^s evaluated at $\rho^s = 1$. $\delta = \sum_{j=1}^{r}\sum_{k=j+1}^{r} \hat{\rho^s}_j \hat{\rho^s}_k$, where $\hat{\rho^s}_k$ is given in (3.13) evaluated at $\rho^s = 1$. Since we have equitable partitions, then

$$\hat{\rho^s}_k = \frac{1}{r} \tag{3.16}$$

for all $k = 1, \ldots, r$. Thus

$$\delta = \frac{r(r-1)}{2r^2} = \frac{r-1}{2r}. \tag{3.17}$$

Again by [24]
$\sigma^2 = \sum_{k=1}^{r} \hat{\lambda^s}_k \left(Var\left[W_n\right] + \hat{\rho^s}_k^2 Var\left[\hat{A}_k\right]\right)$.

Since $\hat{\lambda} = n\hat{\lambda^s}$ by (3.11), then $Var\left[\hat{A}_k\right] = \frac{nr^2}{\hat{\lambda}^2} = \frac{r^2}{n\hat{\lambda^s}^2}$ by (3.9). Also, $\hat{\lambda^s}_k = \frac{\hat{\lambda^s}}{r}$ by (3.11), then substituting (3.16) we have $\sigma^2 = \hat{\lambda^s}\left(Var\left[W_n\right] + \frac{1}{r^2}\frac{r^2}{n\hat{\lambda^s}^2}\right)$. Thus,

$$\sigma^2 = \hat{\lambda^s}\left(Var\left[W_n\right] + \frac{1}{n\hat{\lambda^s}^2}\right), \tag{3.18}$$

where by (3.12)

$$\hat{\lambda^s} = \frac{1}{E\left[W_n\right]}. \tag{3.19}$$

From (3.6) we know

$$E\left[W_n\right] = \frac{E\left[D_n\right]}{v} + nb \tag{3.20}$$

$$Var\left[W_n\right] = \frac{Var\left[D_n\right]}{v^2} + n\sigma_B^2 \tag{3.21}$$

where $E\left[D_n\right]$ and $Var\left[D_n\right]$ are obtained from $E\left[L_n\right]$ and $Var\left[L_n\right]$ by (3.7), and $E\left[L_n\right]$ and $Var\left[L_n\right]$ are obtained from simulation as shown in Figure 3.3. Thus σ^2 is known substituting (3.19), (3.20) and (3.21).

Finally by Boon et al. [50], for equitable partitions

$$E\left[W_{Boon}\right] = \frac{K_0 + K_1\rho^s + K_2\left(\rho^s\right)^2}{1-\rho^s} \tag{3.22}$$

where $K_0 = E\left[\Delta^+\right]$. Δ^+ is called the residual of the random variable Δ with $E\left[\Delta^+\right] = \frac{E\left[\Delta^2\right]}{2E[\Delta]}$. In our case, Δ is deterministic. Thus,

$$K_0 = E\left[\Delta^+\right] = \frac{\Delta}{2} \tag{3.23}$$

with Δ given in (3.1). $K_1 = \hat{\rho^s}_k \left(\left(c^2_{\hat{A}_k}\right)^4 \mathbb{1}\{c^2_{\hat{A}_k} \leq 1\} + 2\frac{c^2_{\hat{A}_k}}{c^2_{\hat{A}_k}+1}\mathbb{1}\{c^2_{\hat{A}_k} > 1\} - 1 \right) E\left[W_n^+\right]$ $+ E\left[W_n^+\right] + \hat{\rho^s}_k \left(E\left[\Delta^+\right] - E[\Delta]\right)$, where

$$c^2_{\hat{A}_k} = \frac{Var\left[\hat{A}_k\right]}{E\left[\hat{A}_k\right]^2}, \tag{3.24}$$

and $\mathbb{1}\{\Omega\}$ is the indicator function defined as $\mathbb{1}\{\Omega\} = 1$ if Ω is true, and $= 0$ otherwise. By (3.9) we have

$$c^2_{\hat{A}_k} = \frac{1}{n}. \tag{3.25}$$

Substituting (3.1), (3.16), (3.23) and (3.25) we have

$$K_1 = E\left[W_n^+\right]\left(\frac{1}{rn^4} - \frac{1}{r} + 1\right) - \frac{\Delta}{2r}, \tag{3.26}$$

where, by definition of a residual,

$$E\left[W_n^+\right] = \frac{E\left[W_n^2\right]}{2E\left[W_n\right]} = \frac{Var\left[W_n\right] + E\left[W_n\right]^2}{2E\left[W_n\right]} \tag{3.27}$$

with $E\left[W_n\right]$ and $Var\left[W_n\right]$ given in (3.20) and (3.21). Finally, by [50]

$$K_2 = \frac{1-\hat{\rho^s}_k}{2}\left(\frac{\sigma^2}{2\delta} + E[\Delta]\right) - K_0 - K_1 \tag{3.28}$$

is known by substituting (3.1), (3.16), (3.17), (3.18), (3.23) and (3.26). Thus, we can calculate the closed form approximation for the cdf of W_P by (3.14).

After obtaining the distributions of W_O, W_P and W_I, we are able to calculate the distribution of $T = W_O + W_P + W_I$ through convolution. In the three components of T, W_O is accurate and known in closed form in (3.3). W_P has a closed form approximation in (3.14). W_I does not have a closed form, but can be arbitrarily accurate through simulation of TSP paths. It is also easy to obtain W_I because we only need to run the simulation once for the empirical distribution of L_{nj} and store the data, then calculate the distribution of D_{nj} and W_I by scaling in (3.7) and convolution in (3.8) for partitions with different sizes parameterized by r.

Figure 3.5 shows the pdf of T obtained from the convolution of its three components, and the empirical pdf of T obtained through simulation under the exhaustive PART-n-TSP when the region $\mathbf{A}$ is a square of size 1×1 with $m = 2$ and $n = 5$, and $m = 2$ and $n = 10$, separately. The tasks are uniformly distributed in the square with size $B_i \sim Unif[0, 1]$. The tasks arrive according to a Poisson process with rate $\lambda = 1$. Thus load $\rho = \lambda b = 0.5$. The simulated empirical pdf of T is regarded as the "true" value. We can see that the approximated values are very close to the true (simulated) values.

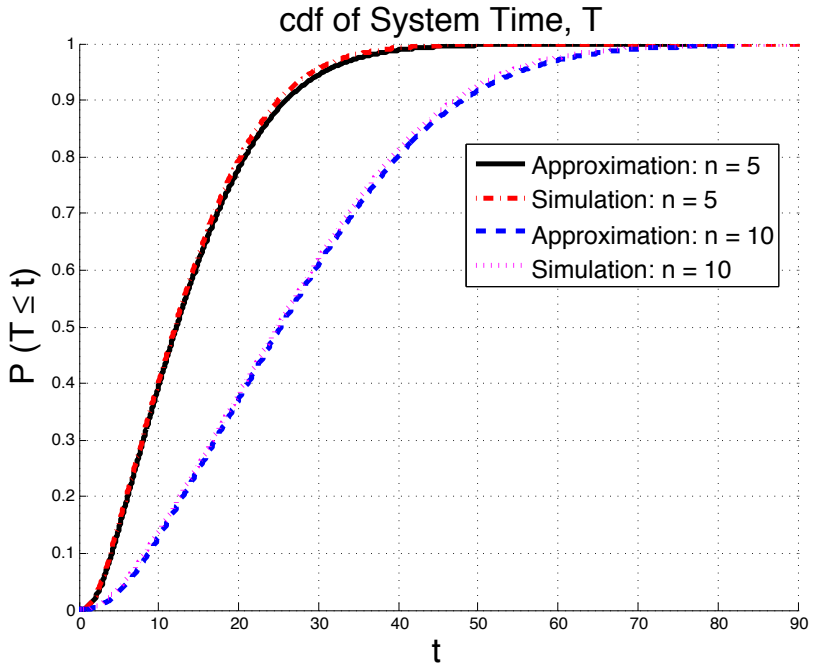


Figure 3.5. Approximated and simulated values of the cdf of the system time.

3.2.2 Comparison of PART-n-TSP, PART-TSP and Nearest Neighbor

Bertsimas et al. [1] compared the $E[T]$ of SQM, FCFS, PART-FCFS, SFC, NN and n-TSP through simulation, and concluded that NN achieves lower $E[T]$ than other

policies simulated. PART-TSP [2] or DC [12] were proven to be $E[T]$ optimal under the light and heavy loads. Here the focus is on $Var[T]$ or $\sigma[T]$.

Since the approximation for the cdf of T for PART-n-TSP is both good and easy to compute, we can optimize the two parameters n and r to minimize $E[T]$ or other performance metrics when the region $\mathbf{A}$ and partitions $\mathbf{A}^k$ are squares. Table 3.1 gives the r^* and n^* in the range $r \in \{1^2, 2^2, \ldots, 10^2\}$ and $n = \{1, \ldots, 60\}$ that minimize $E[T]$ under exhaustive PART-n-TSP and the corresponding $E[T]$ and $\sigma[T]$ values for different ρ values. The region is a square of size 1×1 and $B_i \sim Unif[0, 0.5]$. Noting that FCFS is PART-n-TSP when $r = 1$ and $n = 1$, PART-FCFS is PART-n-TSP when $n = 1$, and n-TSP is PART-n-TSP when $r = 1$. Thus by optimizing on r and n, PART-n-TSP has better performance than FCFS, PART-FCFS and n-TSP.

We compare PART-n-TSP with PART-TSP [2] and Nearest Neighbor [1] since they are considered near optimal in the literature. We simulate PART-TSP and Nearest Neighbor under the same setting. The number of partitions for PART-TSP is set to be the optimal number of partitions for PART-n-TSP. The average number of tasks served inside each gate, denoted by $E[n]$, is also shown in the table. We generate $N = 100,000$ tasks are for each load ρ value. Only the 25,000th to the 75,000th tasks are used to calculate $E[T]$ and $\sigma[T]$ to make sure that the steady state data are used. We have checked this by randomly sampling time segments in this range. Figure 3.6 shows the truncation of 1000 data points for PART-TSP when $\rho = 0.9$, $B_i \sim [0, 0.5]$, and $r = 25$. Each time segments of about 50 data is the number of tasks inside each gate, indeed this is true as shown in Table 3.1 $E[n] = 49.7$ for this case. The system time is decreasing in general in each segment because tasks arrive earlier in a gate wait more than those arrive later. Table 3.2 gives the case when $B_i \sim Unif[0, 1]$.

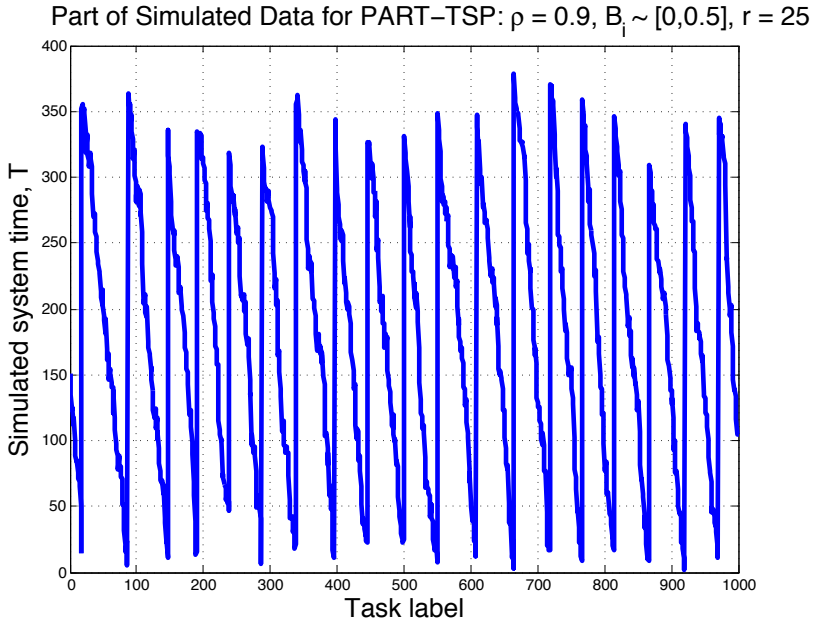


Figure 3.6. Part of simulated data for PART-TSP: $\rho = 0.9$, $B_i \sim [0, 0.5]$, and $r = 25$.

Table 3.1. Comparison of PART-n-TSP, PART-TSP and Nearest Neighbor on $E[T]$ and $\sigma[T]$: $B_i \sim Unif[0, 0.5]$

ρ		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r^* for PART-n-TSP		1	1	1	1	1	1	1	4	25
n^* for PART-n-TSP		1	2	2	4	12	24	57	59	57
$E[n]$ for PART-TSP		1.05	1.25	1.75	3.46	9.08	23.3	61.2	63.9	49.7
$E[T]$	PART-n-TSP	1.04	1.71	1.70	2.80	5.90	9.99	20.3	81.4	321
	PART-TSP	0.95 (91%)	1.26 (74%)	1.87 (110%)	3.30 (118%)	5.97 (101%)	10.9 (109%)	24.4 (120%)	51.7 (64%)	181 (56%)
	NN	**0.94** (90%)	**1.21** (71%)	**1.66** (98%)	**2.46** (88%)	**3.81** (65%)	**6.37** (64%)	**12.7** (63%)	**32.6** (40%)	**154** (48%)
$\sigma[T]$	PART-n-TSP	0.69	1.19	**0.94**	**1.39**	**2.74**	**4.36**	**8.56**	31.7	140
	PART-TSP	0.48 (70%)	0.77 (65%)	1.22 (130%)	2.11 (152%)	3.27 (119%)	5.35 (123%)	13.3 (155%)	**27.1** (85%)	**101** (72%)
	NN	**0.47** (68%)	**0.76** (64%)	1.26 (134%)	2.14 (154%)	3.57 (130%)	6.18 (142%)	12.5 (146%)	31.1 (98%)	147 (105%)

Table 3.2. Comparison of PART-n-TSP, PART-TSP and Nearest Neighbor on $E[T]$ and $\sigma[T]$: $B_i \sim Unif[0,1]$

ρ		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r^* for PART-n-TSP		1	1	1	1	1	1	1	1	9
n^* for PART-n-TSP		1	1	1	2	3	6	15	42	41
$E[n]$ for PART-TSP		1.02	1.1	1.25	1.59	2.37	4.57	13.4	40	31.1
$E[T]$	PART-n-TSP	1.20	1.51	2.15	2.23	2.94	4.96	10.8	26.3	192
	PART-TSP	**1.16** (97%)	1.37 (91%)	1.71 (80%)	2.35 (105%)	3.63 (123%)	6.24 (126%)	12.9 (119%)	27.9 (106%)	93.8 (49%)
	NN	**1.16** (97%)	**1.36** (90%)	**1.66** (77%)	**2.16** (97%)	**2.93** (100%)	**4.50** (91%)	**8.10** (75%)	**18.0** (68%)	**78.7** (41%)
$\sigma[T]$	PART-n-TSP	0.69	0.91	1.59	**1.30**	**1.59**	**2.47**	**4.85**	**11.2**	80.7
	PART-TSP	**0.54** (78%)	**0.75** (82%)	**1.07** (67%)	1.64 (126%)	2.58 (162%)	4.22 (171%)	7.63 (157%)	14.6 (130%)	**56.2** (70%)
	NN	**0.54** (78%)	0.76 (84%)	1.10 (69%)	1.71 (132%)	2.64 (166%)	4.42 (179%)	8.24 (170%)	18.3 (163%)	75.9 (94%)

The $E[T]$ and $\sigma[T]$ of PART-TSP and NN are compared to those of PART-n-TSP, The percentage following the $E[T]$ and $\sigma[T]$ of PART-TSP and NN are the ratio of these values over those of PART-n-TSP. The minimum $E[T]$ and $\sigma[T]$ at each load level of the three policies are bolded. From Tables 3.1 and 3.2, we can see that NN achieves lower $E[T]$ than PART-n-TSP and PART-TSP for all $\rho \in \{0.1 \dots 0.9\}$ in both $B_i \sim [0, 0.5]$ and $B_i \sim [0, 1]$. PART-n-TSP achieves lower $E[T]$ than PART-TSP when ρ is not too small or too large, e.g. when $\rho \in \{0.3, \dots, 0.7\}$ for $B_i \sim [0, 0.5]$, and when $\rho \in \{0.4, \dots, 0.8\}$ for $B_i \sim [0, 1]$. PART-n-TSP has higher $E[T]$ than PART-TSP when ρ is low because it is better to have the average number of tasks in a set to be between 1 and 2 as done by PART-TSP, but PART-n-TSP can only set it to be either 1 ro 2, resulting in higher $E[T]$. PART-n-TSP has higher $E[T]$ than PART-TSP when ρ is high because $r^* > 1$ when ρ is high. Then there is a switching time between partitions. By setting n to be a fixed number under PART-n-TSP, the vehicle might arrive at a partition, and find the number of tasks to be less than n. Then the vehicle would switch to the next partition without serving any task, resulting in a switching cost but no tasks served.

PART-n-TSP behaves like a "standardized" version of PART-TSP. While the fixed n reduces flexibility, it increases certainty. Thus $Var[T]$ or $\sigma[T]$ should be lower. Indeed, as shown in Tables 3.1 and 3.2, PART-n-TSP achieves lower $\sigma[T]$ than PART-TSP and NN when ρ is not too small or too large, e.g. when $\rho \in \{0.3, \dots, 0.7\}$ for $B_i \sim [0, 0.5]$, and when $\rho \in \{0.4, \dots, 0.8\}$ for $B_i \sim [0, 1]$. The performance of PART-n-TSP on $\sigma[T]$ when ρ is too small or too large is not as good for the same reasons affecting $E[T]$ as explained in the previous paragraph.

3.2.3 Optimality of PART-n-TSP under light and heavy loads

The PART-n-TSP can be modified to yield asymptotically optimal $E[T]$ under light load ($\rho \to 0^+$). First the PART-n-TSP becomes FCFS policy when setting $r = 1$ and $n = 1$. Then under FCFS policy, let the vehicle return to the median of region **A** when it becomes idle. Under light load this is the stochastic queue median (SQM) policy [1], where the vehicle travels directly to the task location from the median, executes the task, and then returns to the median after completion. SQM is proven to be $E[T]$ optimal under light load [1], proving the optimality of PART-n-TSP under light load.

Under heavy load ($\rho \to 1^-$), the following lower bound holds [16].

$$E[T] \geq \frac{\beta_{TSP,2}^2 \lambda \left(\int_A f_X^{\frac{1}{2}}(x)\, dx \right)^2}{2v^2(1-\rho)^2} \tag{3.29}$$

The following theorem shows that PART-n-TSP achieves the heavy-load lower bound (3.29) when $r \to \infty$. Thus PART-n-TSP is asymptotically optimal in $E[T]$ under heavy load.

Theorem 3.1. *Under PART-n-TSP as per Definition 3.1, when $\rho \to 1^-$ and $n \to \infty$, the system time for the 1-DTRP satisfies*

$$E[T] \leq \left(1 + \frac{1}{r}\right) \frac{\beta_{TSP,2}^2 \lambda \left(\int_A f_X^{\frac{1}{2}}(x)\, dx\right)^2}{2v^2(1-\rho)^2} \tag{3.30}$$

where r is the number of partitions.

Proof. $E[T] = E[W_O] + E[W_P] + E[W_I]$ by (3.2).

And by (3.4)

$$E[W_O] = \frac{(n-1)r}{2\lambda} < \frac{nr}{2\lambda} \tag{3.31}$$

By (3.6) and (3.8), and conditioning on the position that a given task takes within its set, and noting that the travel time around the TSP path is no more than the length of the path itself, the expected wait for completion once a task's set enters service

$$E[W_I] \leq \frac{1}{v} E[D_n] + \frac{1}{n} \sum_{j=1}^{n} jb = \frac{1}{v} E[D_n] + \frac{n+1}{2} b \tag{3.32}$$

Given that a demand falls in partition $\mathbf{A^k}$, the conditional density for its location (whose support is $\mathbf{A^k}$) is $\frac{f_X(x)}{\int_{A^k} f_X(x)\, dx}$. From [48] we know that, almost surely, $\lim_{n\to\infty} \frac{D_n}{\sqrt{n}} = \beta_{TSP,2} \int_{A^k} \sqrt{\frac{f_X(x)}{\int_{A^k} f_X(x)\, dx}}\, dx$, where $\beta_{TSP,2}$ is a constant. Let $C = \frac{1}{v} \beta_{TSP,2} \int_{A^k} \sqrt{\frac{f_X(x)}{\int_{A^k} f_X(x)\, dx}}\, dx$, thus C is a constant. So $\lim_{n\to\infty} \frac{1}{v} E[D_n] = C\sqrt{n}$.

The load of a set $\rho^s = \lambda^s E[W_n] = \frac{\lambda}{n}\left(\frac{E[D_n]}{v} + nb\right) = \lambda b + \frac{\lambda}{v}\frac{E[D_n]}{n} = \rho + \lambda \frac{E[D_n]}{nv}$ by (3.11), (3.12) and (3.20). Thus $\lim_{n\to\infty} \rho^s = \rho + \lambda \lim_{n\to\infty} \frac{E[D_n]}{nv} = \rho + \lambda \lim_{n\to\infty} \frac{C\sqrt{n}}{n} = \rho$. So $\rho \to 1^-$ implies $\rho^s \to 1^-$ when $n \to \infty$.

As for $E[W_P]$, from [52] we know that the mean waiting time in a polling system with renewal arrivals as $\rho^s \to 1^-$ is

$$E[W_P] = \frac{\omega}{1-\rho^s} + o\left((1-\rho^s)^{-1}\right), \tag{3.33}$$

where $\omega = \frac{1-\hat{\rho^s}_k}{2}\left(\frac{\sigma^2}{\sum_{k=1}^r \hat{\rho^s}_k\left(1-\hat{\rho^s}_k\right)} + E[\Delta]\right)$ under the exhaustive policy, and $\omega = \frac{1+\hat{\rho^s}_k}{2}\left(\frac{\sigma^2}{\sum_{k=1}^r \hat{\rho^s}_k\left(1+\hat{\rho^s}_k\right)} + E[\Delta]\right)$ under the gated policy. Substituting (3.16) we have

$\omega = \frac{\sigma^2}{2} + \frac{r-1}{2r}E[\Delta]$ under the exhaustive policy, and $\omega = \frac{\sigma^2}{2} + \frac{r+1}{2r}E[\Delta]$ under the gated policy. $\Delta = \sum_{k=1}^{r} \Delta^k$ is the total switch time of a polling cycle. Δ does not depend on ρ, ρ^s or n, and is given in (3.1) when the region $\mathbf{A}$ and partitions $\mathbf{A}^k$ are squares.

Thus $E[W_P] = \frac{1}{2(1-\rho^s)}\left(\sigma^2 + \frac{r\mp1}{r}E[\Delta]\right) + o\left((1-\rho^s)^{-1}\right)$ when $\rho^s \to 1^-$. Let $C' = \frac{r\mp1}{r}E[\Delta]$. So $E[W_P] = \frac{1}{2(1-\rho^s)}\left(\sigma^2 + C'\right)$ when $\rho^s \to 1^-$. Since r is a finite natural number, and Δ_k is upper bounded by the diameter of region $\mathbf{A}$, then $E[\Delta]$ is a positive finite number. Thus, C' is a positive finite number.

By (3.18) $\sigma^2 = \hat{\lambda^s}\left(Var[W_n] + \frac{1}{n\hat{\lambda^s}^2}\right)$, where $\hat{\lambda^s} = \frac{\hat{\lambda}}{n}$ by (3.11). Also, $\rho^s = \hat{\lambda^s}E[W_n]$ as $\rho^s \to 1^-$ by (3.12).

So $E[W_P] = \frac{\hat{\lambda^s}\left(Var[W_n] + \frac{1}{n\hat{\lambda^s}^2}\right) + C'}{2\left(1-\hat{\lambda^s}E[W_n]\right)}$
$= \frac{\frac{\hat{\lambda}}{n}\left(\frac{1}{v^2}Var[D_n] + n\sigma_B^2 + \frac{n}{\hat{\lambda}^2}\right) + C'}{2\left(1-\frac{\hat{\lambda}}{n}\left(\frac{1}{v}E[D_n] + nb\right)\right)}$
$= \frac{\hat{\lambda}\left(\frac{1}{\hat{\lambda}^2} + \frac{Var[D_n]}{nv^2} + \sigma_B^2\right) + C'}{2\left(1-\hat{\lambda}b - \hat{\lambda}\frac{1}{v}\frac{E[D_n]}{n}\right)}$, as $\rho^s \to 1^-$, where we substituted $\hat{\lambda^s} = \frac{\hat{\lambda}}{n}$ and (3.20) and (3.21) in the first equality.

Since $\hat{\lambda}$ is the value of λ when $\rho^s = 1$, and $\rho \to 1^-$ implies $\rho^s \to 1^-$, then we can write

$$E[W_P] = \frac{\lambda\left(\frac{1}{\lambda^2} + \frac{Var[D_n]}{nv^2} + \sigma_B^2\right) + C'}{2\left(1 - \rho - \lambda\frac{1}{v}\frac{E[D_n]}{n}\right)} \tag{3.34}$$

when $\rho \to 1^-$, where we substituted $\rho = \lambda b$.

From [53, p.189] we know $\lim_{n\to\infty} Var[D_n] = O(1)$, and therefore, $\lim_{n\to\infty} \frac{Var[D_n]}{n} = 0$.

Thus when $\rho \to 1^-$ and $n \to \infty$,
$E[T] = E[W_O] + E[W_P] + E[W_I]$
$\leq \frac{(n-1)r}{2\lambda} + \frac{1}{v}E[D_n] + \frac{n+1}{2}b + \frac{\lambda\left(\frac{1}{\lambda^2} + \frac{Var[D_n]}{nv^2} + \sigma_B^2\right) + C'}{2\left(1-\rho-\lambda\frac{1}{v}\frac{E[D_n]}{n}\right)}$
$\leq \frac{nr}{2\lambda} + C\sqrt{n} + \frac{n}{2}b + \frac{\lambda\left(\frac{1}{\lambda^2} + \sigma_B^2\right) + C'}{2\left(1-\rho-\lambda\frac{C}{\sqrt{n}}\right)}$.

Substituting $b = \frac{\rho}{\lambda}$ we have

$$E[T] \leq \frac{\lambda\left(\frac{1}{\lambda^2} + \sigma_B^2\right) + C'}{2\left(1 - \rho - \lambda\frac{C}{\sqrt{n}}\right)} + \frac{n(r+\rho)}{2\lambda} + C\sqrt{n} \tag{3.35}$$

We want to minimize (3.35) with respect to n to get the least upper bound. Noting that (3.35) is convex with respect to n, so there is indeed a minimum. First, however,

consider a change of variable $y = \frac{\lambda C}{(1-\rho)\sqrt{n}}$. With this change,

$$E[T] \leq \frac{\lambda\left(\frac{1}{\lambda^2} + \sigma_B^2\right) + C'}{2(1-\rho)(1-y)} + \frac{\lambda C^2(r+\rho)}{2(1-\rho)^2 y^2} + \frac{\lambda C^2}{(1-\rho)y} \tag{3.36}$$

For $\rho \to 1^-$, one can verify that the optimum y approaches 1. Linearizing the last two terms above about $y = 1$ we have $\frac{\lambda C^2(r+\rho)}{2(1-\rho)^2 y^2} = \frac{\lambda C^2(r+\rho)}{2(1-\rho)^2}(3-2y)$, and $\frac{\lambda C^2}{(1-\rho)y} = \frac{\lambda C^2}{1-\rho}(2-y)$.

Thus, $g(y) \equiv \frac{\lambda\left(\frac{1}{\lambda^2}+\sigma_B^2\right)+C'}{2(1-\rho)(1-y)} + \frac{\lambda C^2(r+\rho)}{2(1-\rho)^2 y^2} + \frac{\lambda C^2}{(1-\rho)y}$
$\approx \frac{C_1}{1-y} + C_2(3-2y) + C_3(2-y)$
$= \frac{C_1}{1-y} + (2C_2 + C_3)(1-y) + C_2 + C_3$, where $C_1 = \frac{\lambda\left(\frac{1}{\lambda^2}+\sigma_B^2\right)+C'}{2(1-\rho)}$, $C_2 = \frac{\lambda C^2(r+\rho)}{2(1-\rho)^2}$, and $C_3 = \frac{\lambda C^2}{1-\rho}$.

The approximation for $g(y)$ is minimized when $\frac{C_1}{1-y} = (2C_2 + C_3)(1-y)$. Substituting C_1, C_2 and C_3 we have an approximate optimum value

$$y^* = 1 - \frac{1}{C}\sqrt{\frac{\left(\frac{1}{\lambda^2} + \sigma_B^2 + \frac{C'}{\lambda}\right)(1-\rho)}{2(1+r)}} \tag{3.37}$$

Substituting (3.37) into (3.36) and noting that for $\rho \to 1^-$ the approximate y^* approaches 1 we have

$$E[T] \leq \frac{\lambda C^2(r+1)}{2(1-\rho)^2} + \frac{\lambda C\sqrt{2(r+1)\left(\frac{1}{\lambda^2} + \sigma_B^2 + \frac{C'}{\lambda}\right)}}{2(1-\rho)^{\frac{3}{2}}} + \frac{\lambda C^2}{1-\rho} \tag{3.38}$$

when $\rho \to 1^-$.

Thus $E[T] \leq \frac{\lambda C^2(r+1)}{2(1-\rho)^2} + o\left((1-\rho)^{-2}\right)$ when $\rho \to 1^-$. We have

$$E[T] \leq \frac{\lambda C^2(r+1)}{2(1-\rho)^2} \tag{3.39}$$

when $\rho \to 1^-$.

$C = \frac{1}{v}\beta_{TSP,2}\int_{A^k}\sqrt{\frac{f_X(x)}{\int_{A^k} f_X(x)\,dx}}\,dx$
$= \frac{1}{v}\beta_{TSP,2}\sqrt{r}\int_{A^k}\sqrt{f_X(x)}\,dx$
$= \beta_{TSP,2}\frac{1}{v\sqrt{r}}\int_A \sqrt{f_X(x)}\,dx.$

Substituting C in (3.39) we have

$$E[T] \leq \left(1 + \frac{1}{r}\right)\frac{\beta_{TSP,2}^2\lambda\left(\int_A f_X^{\frac{1}{2}}(x)\,dx\right)^2}{2v^2(1-\rho)^2}.$$

□

The PART-n-TSP is optimal under light load. Moreover, when $r \to \infty$, the PART-n-TSP policy achieves the heavy-load lower bound (3.29). Therefore the PART-n-TSP is both optimal under light load and arbitrarily close to optimality under heavy load, and stabilizes the system for every load $\rho \in [0, 1)$. Notice that with $r = 10$ the PART-n-TSP is already guaranteed to be within 10% of the optimal performance under heavy load.

3.3 Summary

We give a good approximation for the distribution of the system time that is easy to compute under the PART-n-TSP policy by utilizing the approximation results of the distribution of system time T, together with $E[T]$ and $Var[T]$ in the polling systems [23, 24]. We compare PART-n-TSP with PART-TSP [2] and Nearest Neighbor [1] on $E[T]$ and $\sigma[T]$ in Tables 3.1 and 3.2, since the latter two are considered near optimal in the literature. The results show that in practice PART-n-TSP achieves lower $\sigma[T]$ than PART-TSP and NN and lower $E[T]$ than PART-TSP when the load ρ is not too small or too large. We also prove that PART-n-TSP is $E[T]$ optimal under light load ($\rho \to 0^+$) and asymptotically optimal under heavy load ($\rho \to 1^-$) in Theorem 3.1.

Chapter 4

Virtual Vehicle and Cloud Computing in Space

While the current scheduling policies for the Vehicle Routing Problem (VRP) and its variations such as the Dynamic Traveling Repairman Problem (DTRP) work for single customer systems, they do not create performance isolation [3] in multi-customer systems. In this chapter we extend the idea of the virtual machine [3] used in cloud computing to an idea called the virtual vehicle to create performance isolation in multi-customer systems with location specific tasks. This enables what we call cloud computing in space, or the spatial cloud.

4.1 Model

The spatial cloud is defined as follows: A service provider controls $M \in \mathbb{N}$ real vehicles (RVs), $\{RV_m\}_{m=1}^{M}$ in a convex region $\mathbf{A}$ of area A to host $K \in \mathbb{N}$ virtual vehicles (VVs), $\{VV_k\}_{k=1}^{K}$. Each RV travels at constant speed v^R. Each VV has virtual speed v^V in the reservation contract.

To a customer, a virtual vehicle with speed v^V is a replica of a real vehicle with speed v^V. It can be reserved to host a sequence of tasks $\langle Task_{ki} \rangle_{i=1}^{\infty}$, where k denotes the k-th VV and i the i-th task hosted by it. Each $Task_{ki}$ has arrival times T_{ki}^a, location X_{ki}, and size T_{ki}^S. $\langle Task_{ki} \rangle$ is ordered by the arrival time T_{ki}^a. The sequences of tasks hosted by different VVs are independent of each other, i.e., $Task_{ki}$ are independent in k.

Each task arrival process $\{T_{ki}^a\}_{i=1}^{\infty}$ is assumed to be a renewal process. Thus the interarrival time, $I_{ki}^a \equiv T_{ki}^a - T_{k(i-1)}^a$ is independent and identically distributed (i.i.d.) in i, where $T_{k0}^a \equiv 0$. The generic interarrival time I_k^a is assumed to be integrable.

Thus the task arrival rate of each VV is

$$\lambda_k^V = \frac{1}{E\left[I_k^a\right]} \tag{4.1}$$

X_{ki} is i.i.d. in k and i, and uniformly distributed in $\mathbf{A}$ with probability density function (pdf) $f_X(x) = \frac{1}{A}$, $x \in \mathbf{A}$. The task size T_{ki}^S is i.i.d. in k and i with pdf $f_{T^S}(t)$, $t \in [0, \infty)$. We denote by T^S the generic term of T_{ki}^S. Then $E\left[T_{ki}^S\right] = E\left[T^S\right]$, which is assumed to be finite.

Let L_{ki} denote the distance between $Task_{k(i-1)}$ and $Task_{ki}$ hosted by VV_k. Then

$$L_{ki} = \parallel X_{ki} - X_{k(i-1)} \parallel \tag{4.2}$$

where $\parallel . \parallel$ is the Euclidean norm defined on region $\mathbf{A}$, $i = 1, 2, \ldots$. X_{k0} is the initial position of the first RV hosting VV_k given by the provider, which is assumed to be uniformly distributed in $\mathbf{A}$, and independent of X_{ki}. We denote by L the generic term of L_{ki}.

VV_k is assumed to serve the sequence of tasks $\langle Task_{ki} \rangle_{i=1}^{\infty}$ under the first come first served (FCFS) policy. It travels to the location X_{ki} of each task and executes it taking time T_{ki}^S. Then the virtual service time of $Task_{ki}$, denoted by T_{ki}^{Vserv}, includes the flying time and execution time as follows

$$T_{ki}^{Vserv} = \frac{L_{ki}}{v^V} + T_{ki}^S \tag{4.3}$$

Thus T_{ki}^{Vserv} is i.i.d. in k and i. We denote by T^{Vserv} the generic term of T_{ki}^{Vserv}.

Each VV_k is a queue. The virtual departure time of $Task_{ki}$ from this queue is

$$T_{ki}^{dead} = \max\left\{T_{ki}^a, T_{k(i-1)}^{dead}\right\} + T_{ki}^{Vserv} \tag{4.4}$$

where $T_{k0}^{dead} \equiv 0$. We use the superscript dead for deadline because this virtual departure time is used by our scheduling policies as a task deadline.

Each task, upon arrival, is passed to one of the M real vehicles. Thus we have M real vehicle queues. Each $Task_{ki}$ is served by some real vehicle RV_m as per an *allocation policy* given in Definition 4.1 in Section 4.2.2. The order of service is determined by a scheduling policy. The scheduling policies, discussed in Section 4.3, are the main subject of this chapter.

Each $Task_{ki}$ will be completed by an RV at some time T_{ki}^{comp}. We assume customer k will be satisfied if $T_{ki}^{comp} \leq T_{ki}^{dead}$. Then the aim of the provider is to achieve $T_{ki}^{comp} \leq T_{ki}^{dead}$ for as many tasks as possible. Thus T_{ki}^{dead} is like a "deadline" for $Task_{ki}$. We call the T_{ki}^{dead} the *virtual deadline* for $Task_{ki}$. This makes the spatial cloud a soft real-time system [26, 27].

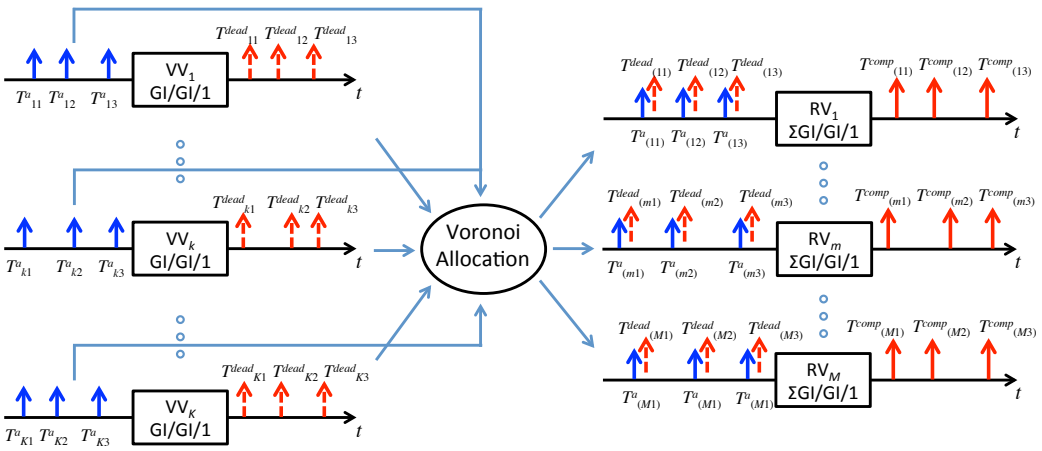


Figure 4.1. The spatial cloud with virtual vehicle queues and real vehicle queues.

We define the relative expected *tardiness* of VV_k as

$$TD_k = \frac{1}{E\left[T^{Vserv}\right]} \lim_{n\to\infty} \frac{\sum_{i=1}^{n} \max\left\{T_{ki}^{comp} - T_{ki}^{dead}, 0\right\}}{n} \tag{4.5}$$

The *delivery probability* of VV_k is

$$DP_k = \lim_{n\to\infty} \frac{\sum_{i=1}^{n} \mathbb{1}\left\{T_{ki}^{comp} \le T_{ki}^{dead}\right\}}{n} \tag{4.6}$$

where $\mathbb{1}\{\Omega\}$ is the indicator function defined as $\mathbb{1}\{\Omega\} = 1$ if Ω is true, and $= 0$ otherwise.

The relative expected *slack* of VV_k is

$$SL_k = \frac{1}{E\left[T^{Vserv}\right]} \lim_{n\to\infty} \frac{\sum_{i=1}^{n} \max\left\{T_{ki}^{dead} - T_{ki}^{comp}, 0\right\}}{n} \tag{4.7}$$

We define the virtual system time of $Task_{ki}$ as $T_{ki}^{Vsys} = T_{ki}^{dead} - T_{ki}^{a}$, and the real system time of $Task_{ki}$ as $T_{ki}^{Rsys} = T_{ki}^{comp} - T_{ki}^{a}$. Thus $T_{ki}^{comp} - T_{ki}^{dead} = T_{ki}^{Rsys} - T_{ki}^{Vsys}$, and $T_{ki}^{comp} \le T_{ki}^{dead} \Leftrightarrow T_{ki}^{Rsys} \le T_{ki}^{Vsys}$. When the queue at VV_k is stable, $T_{ki}^{Vsys} \to T_k^{Vsys}$ in distribution. When the M real vehicle queues are stable, $T_{ki}^{Vsys} \to T_k^{Vsys}$ in distribution.

Thus (4.5) is equivalent to

$$TD_k = \frac{E\left[\max\left\{T_k^{Rsys} - T_k^{Vsys}, 0\right\}\right]}{E\left[T^{Vserv}\right]} \tag{4.8}$$

(4.6) is equivalent to

$$DP_k = P\left(T_k^{Rsys} \le T_k^{Vsys}\right) \tag{4.9}$$

(4.7) is equivalent to

$$SL_k = \frac{E\left[\max\left\{T_k^{Vsys} - T_k^{Rsys}, 0\right\}\right]}{E\left[T^{Vserv}\right]} \tag{4.10}$$

These three measures are determined by the virtual vehicle queues and the real vehicle queues as shown in Section 4.2.

Two consecutive tasks of an VV might be executed by two different RVs, which involves migrating the VV from one virtual vehicle to another. Let Z_{ki} be an indicator set to 1 if there is a migration between $Task_{k(i-1)}$ and $Task_{ki}$, and 0 otherwise. When the M real vehicle queues are stable, $Z_{ki} \to Z_k$ in distribution. B_{Vk} is the number of bits to migrate VV_k at the migration time. We assume $B_{Vk} = 1$ or equivalently a constant. L is the generic distance between two consecutive tasks. We define the inter-virtual deadline time as $I_{ki}^{dead} = T_{ki}^{dead} - T_{k(i-1)}^{dead}$. Since T_{ki}^{Vserv} is i.i.d. in k and i, then $I_{ki}^{dead} \to I_k^{dead}$ in distribution. The migration cost of VV_k is

$$MC_k = B_{Vk}\frac{E\left[Z_k L\right]}{E\left[I_k^{dead}\right]} \tag{4.11}$$

The migration cost has the same unit (bit-meters/second) as in [54].

4.1.1 Performance Isolation

We measure *performance isolation* by the average of the tardiness and delivery probability.

$$TD = \frac{1}{K}\sum_{k=1}^{K} TD_k \tag{4.12}$$

$TD = 0$ implies $TD_k = 0$ for all virtual vehicle k, meaning the system achieves perfect performance isolation. Conversely $TD \to \infty$ means the relative expected tardiness of some VVs is unbounded, the system has very poor performance isolation.

$$DP = \frac{1}{K}\sum_{k=1}^{K} DP_k \tag{4.13}$$

$DP = 1$ implies $DP_k = 1$ for all virtual vehicle k, meaning the system achieves perfect performance isolation. Conversely $DP = 0$ means the delivery probability of each virtual vehicle is zero, meaning the system has no performance isolation.

Fairness is the equality of work divided among different concurrent environments [55]. We use Jain's fairness index [29] to quantify the fairness between virtual vehicles. The

fairness index based on tardiness TD_k is

$$FI\left(TD_k\right) = \frac{\left(\sum_{k=1}^{K} e^{-TD_k}\right)^2}{K\sum_{k=1}^{K}\left(e^{-TD_k}\right)^2} \tag{4.14}$$

where we use e^{-TD_k} to map TD_k from $[0,\infty)$ to $(0,1]$, the lower TD_k, the higher e^{-TD_k}, indicating higher performance.

The fairness index based on delivery probability DP_k is

$$FI\left(DP_k\right) = \frac{\left(\sum_{k=1}^{K} DP_k\right)^2}{K\sum_{k=1}^{K} DP_k^2} \tag{4.15}$$

Jain's fairness index is the ratio of the square of the first moment over the second moment of the set of performance metrics of all the VVs. If $TD_k = TD_l > 0$ (resp. $DP_k = DP_l > 0$), $\forall k, l$, then $FI\left(TD_k\right) = 1$ (resp. $FI\left(DP_k\right) = 1$), indicating completely fair. If $TD_k > 0$ and $TD_l = 0$ (resp. $DP_k > 0$ and $DP_l = 0$), $\forall l \neq k$, then $FI\left(TD_k\right) = \frac{1}{K}$ (resp. $FI\left(DP_k\right) = \frac{1}{K}$), indicating completely unfair. Thus $FI\left(TD_k\right)$ (resp. $FI\left(DP_k\right)$) ranges between $\frac{1}{K}$ and 1. The greater the fairness index, the more fair the system. Jain's fairness index has been used to evaluate virtualization systems [56], [57], [58]. Other performance metrics in this literature include throughput, latency and response time.

4.1.2 Gain

When a provider supports K virtual vehicles with M real vehicles we define the gain κ to be

$$\kappa = \frac{K}{M} \tag{4.16}$$

The provider gains if $\kappa > 1$. There are two ways a provider can gain:

- *Multiplexing gain*: a customer may not utilize her virtual vehicle fully, enabling the provider to multiplex several virtual vehicles onto one real vehicle.
- *Migration gain*: the provider gains by migrating the VV hosting the task to another RV closer to the task location.

The multiplexing gain is observed in communication networks [32] and cloud computing [33]. Migration gain is unique to cloud computing in space. When every virtual vehicles are fully utilized, there is no multiplexing gain, but there is still migration gain.

4.2 Systems

The system has queues at the virtual vehicles and queues at the real vehicles as depicted in Figure 4.1.

4.2.1 Virtual Vehicle Queues

The task arrival rate for VV_k is $\lambda_k^V = \frac{1}{E[I_k^a]}$ by (4.1), where I_k^a is the generic interarrival time of VV_k. The service time T_{ki}^{Vserv} is i.i.d. in k and i with generic term T_k^{Vserv}. We define the generic virtual vehicle service rate as

$$\mu^V = \frac{1}{E\left[T^{Vserv}\right]} = \frac{1}{\frac{E[L]}{v^V} + E\left[T^S\right]} \tag{4.17}$$

as usual, with T_{ki}^{Vserv} defined in (4.3).

The random process at the output of VV_k is the virtual deadline process $\left\{T_{ki}^{dead}\right\}_{i=1}^{\infty}$. The virtual deadline rate is defined as

$$\lambda_k^{Vdead} = \lim_{i\to\infty} \frac{1}{E\left[T_{ki}^{dead} - T_{k(i-1)}^{dead}\right]} \tag{4.18}$$

Since VV_k is a work-conserving server, it is busy if it has a queue and idle if not. Thus VV_k repeats cycles of busy and idle periods. We define Θ_{kl}^V as the l-th busy period and I_{kl}^V as the l-th idle period. We define virtual vehicle utilization as

$$u_k^V = \lim_{l\to\infty} \frac{E\left[\Theta_{kl}^V\right]}{E\left[\Theta_{kl}^V\right] + E\left[I_{kl}^V\right]} \tag{4.19}$$

A single server queuing system is $GI/GI/1$ if the interarrival times at the input and the service times are positive i.i.d. random variables, separately [15].

The tasks created by a customer are passed to the cloud at the rate chosen by the customer. The following theorem asserts that when the arrival rate is less than the service rate, the virtual deadline rate is equal to the arrival rate. However, if the customer exceeds the service rate determined by the contracted virtual speed, the virtual deadline rate is equal to the service rate, i.e., the contract throttles the customer's virtual deadline rate. A higher virtual deadline rate requires more tasks to be completed in a unit time. A customer cannot require more than her share of resources by simply generating tasks faster and faster because the virtual deadline rate is throttled by the VV service rate.

Theorem 4.1. *Each virtual vehicle VV_k is a $GI/GI/1$ queue. Moreover,*
If $\lambda_k^V < \mu^V$, then $u_k^V < 1$ and $\lambda_k^{Vdead} = \lambda_k^V$.
If $\lambda_k^V \geq \mu^V$, then $u_k^V = 1$ and $\lambda_k^{Vdead} = \mu^V$.

Proof. By assumption the task arrival process of VV_k is renewal. Thus the interarrival times are positive and i.i.d.. Also the service times T_{ki}^{Vserv} are positive and i.i.d.. Thus each virtual vehicle VV_k is a $GI/GI/1$ queue.

When $\lambda_k^V < \mu^V$, the $GI/GI/1$ queue is stable by Theorem 1.1 in [15, p. 168]. The number of tasks waiting in the queue is finite almost surely, the mean interdeparture time $E\left[T_{ki}^{dead} - T_{k(i-1)}^{dead}\right]$ of the VV is $\frac{1}{\lambda_k^V}$ because no tasks are lost and no extra task is created. Thus $\lambda_k^{Vdead} = \lambda_k^V$. Stability also ensures that $\lim_{l\to\infty} E\left[\Theta_{kl}^V\right] = E\left[\Theta_k^V\right]$ and $\lim_{l\to\infty} E\left[I_{kl}^V\right] = E\left[I_k^V\right]$, where Θ_k^V and I_k^V are the generic busy period and idle period. Thus $u_k^V = \frac{E[\Theta_k^V]}{E[\Theta_k^V]+E[I_k^V]}$. According to [22, p. 21], $u_k^V = \frac{\lambda_k^V}{\mu^V}$ since no tasks are lost or created in the system. Thus $u_k^V = \frac{\lambda_k^V}{\mu^V} < 1$.

When $\lambda_k^V > \mu^V$, the $GI/GI/1$ queue is unstable by Theorem 1.1 in [15, p. 168], the number of tasks waiting in the queue goes to infinity as time goes to infinity. The busy period tends to infinity and the idle period tends to 0, Thus $u_k^V = \lim_{l\to\infty} \frac{E\left[\Theta_{kl}^V\right]}{E\left[\Theta_{kl}^V\right]+E\left[I_{kl}^V\right]} = 1$, and the interdeparture time equals the VV service time. Then $\lambda_k^{Vdead} = \mu^V$.

When $\lambda_k^V = \mu^V$, the number of tasks waiting in the queue can either be finite, or goes to infinity as time goes to infinity depending on the arrival process $\{T_{ki}^a\}$. So this case either goes to case (i) or (ii). In either case, we have $u_k^V = 1$ and $\lambda_k^{Vdead} = \mu^V$. □

When $\lambda_k^V > \mu^V$, the $GI/GI/1$ queue at VV_k is unstable, thus the virtual system time $T_{ki}^{Vsys} = T_{ki}^{dead} - T_{ki}^a \to \infty$ almost surely [15, p. 168]. Since the customer regards a VV a replica of an RV, we assume the customer will never run the VV under the unstable condition. Thus we assume $\lambda_k^V \leq \mu^V$ from now on. Also, when $\lambda_k^V = \mu^V$, we assume the customer only provide task arrival process that results in finite virtual system time, i.e., $T_{ki}^{Vsys} \to T_k^{Vsys}$ in distribution.

4.2.2 Real Vehicle Queues

Each task, upon arrival, is passed to one of the M real vehicles. Inside each RV subregion, the RV runs a scheduling policy to decide which task of which VV to execute when the RV becomes available. The scheduling policies are discussed in Section 4.3. We allocate the tasks hosted by each VV as follows.

Definition 4.1. *The VV allocation has the following steps:*

(i) Divide the region $\mathbf{A}$ *into* M *subregions by computing an* M*-median of* $\mathbf{A}$ *that induces a Voronoi tessellation that is equitable with respect to* $f_X(x)$ *following [35]. An* M*-partition* $\{\mathbf{A}_m\}_{m=1}^M$ *is equitable with respect to* $f_X(x)$ *if* $\int_{\mathbf{A}_m} f_X(x)\,dx = \frac{1}{M}$ *for all* $m \in \{1, \ldots, M\}$.

(ii) The real vehicles assign themselves to the subregions in a one-to-one manner.

(iii) Each RV serves the tasks that fall within its own subregion. The VV hosting the task is migrated to the RV prior to task execution if the previous task was served by another RV. This migration incurs the cost quantified in (4.11).

We sequence the tasks contributed by all the virtual vehicles to subregion $\mathbf{A}_m$ by their arrival times. The sequence is denoted $\langle Task_{(mj)}\rangle_{j=1}^{\infty}$. Thus $Task_{(mj)}$ is the j-th task arrived at real vehicle m. Note each $Task_{(mj)}$ corresponds to some $Task_{ki}$ hosted by a virtual vehicle k at time T^a_{ki}. $Task_{(mj)}$ has arrival time $T^a_{(mj)} = T^a_{ki}$, location $X_{(mj)} = X_{ki}$ and size $T^S_{(mj)} = T^S_{ki}$. Note $T^a_{(m(j-1))} \leq T^a_{(mj)}$ and $X_{(mj)} \in \mathbf{A}_m$ by construction. We write $Task_{(mj)}$ if the task is labeled according to the RV, and write $Task_{ki}$ if the task is labeled according to the VV. For the different labels of the same task, $Task_{ki}$ and $Task_{(mj)}$, since $T^a_{k(i-1)} \leq T^a_{ki}$ and $T^a_{(m(j-1))} \leq T^a_{(mj)}$, then

$$i \to \infty \Leftrightarrow T^a_{ki} \to \infty \Leftrightarrow T^a_{(mj)} \to \infty \Leftrightarrow j \to \infty \tag{4.20}$$

Since the task locations X_{ki} are i.i.d. in k and i, and uniformly distributed in region $\mathbf{A}$, then $X_{(mj)}$ are i.i.d. in j and uniformly distributed in each subregion $\mathbf{A}_m$. We denote by $D_{(m)}$ the distances between two random task locations in subregion $\mathbf{A}_m$. Thus,

$$D_{(m)} = \| X_{(mj)} - X_{(ml)} \| \tag{4.21}$$

where $X_{(mj)}$ and $X_{(ml)}$ are two random task locations in subregion $\mathbf{A}_m$.

Let $D_{(mj)}$ denote the distance between $Task_{(mj)}$ and the task executed before it under a scheduling policy ϕ in subregion $\mathbf{A}_m$. In general, $D_{(mj)}$ is policy dependent. We define a class of policies that produce i.i.d. $D_{(mj)}$ as follows.

Definition 4.2. *A scheduling policy* ϕ *in a real vehicle subregion* $\mathbf{A}_m$ *is called non-location based if the distance between two consecutively executed tasks is i.i.d..*

The common policies in queueing theory such as FCFS, last come first served (LCFS), random order of service (ROS), and shortest job first (SJF) [10] are non-location based policies in the sense of Definition 4.2. The scheduling policies we propose in Section 4.3 such as Earliest Virtual Deadline First (EVDF), Earliest Dynamic Virtual Deadline First (EDVDF) and credit scheduling policy are shown to satisfy Definition 4.2 in Theorem 4.3.

The scheduling policies that utilize the location of the tasks such as nearest neighbor (NN) and traveling salesman policy (TSP) on a given set of tasks are not non-location based because the distances between two consecutively executed tasks are not independent.

Definition 4.3. *The set of M real vehicle subregions $\{\mathbf{A}_m\}_{m=1}^{M}$ are said to be homogeneous if they all have the same scheduling policy, and $D_{(m)}$ is i.i.d. for all $m = 1, \ldots, M$.*

In this section and next, we only consider the homogeneous RV subregions under non-location based scheduling policies. We denote by D the generic term of $D_{(m)}$. Then $D_{(mj)}$ is i.i.d. in m and j, and has the same distribution as D.

The service time of $Task_{(mj)}$ by RV_m, denoted by $T_{(mj)}^{Rserv}$, is

$$T_{(mj)}^{Rserv} = \frac{D_{(mj)}}{v^R} + T_{(mj)}^{S} \tag{4.22}$$

Since $D_{(mj)}$ and $T_{(mj)}^{S}$ are i.i.d. in m and j, separately, then $T_{(mj)}^{Rserv}$ is i.i.d. in m and j. We denote by T^{Rserv} the generic term of $T_{(mj)}^{Rserv}$, and define the generic real vehicle service rate μ^R as

$$\mu^R = \frac{1}{E\left[T^{Rserv}\right]} = \frac{1}{\frac{E[D]}{v^R} + E\left[T^S\right]} \tag{4.23}$$

We define

$$\kappa^c = \frac{\mu^R}{\mu^V} = \frac{\frac{E[L]}{v^V} + E\left[T^S\right]}{\frac{E[D]}{v^R} + E\left[T^S\right]} \tag{4.24}$$

Before analyzing the queues at the real vehicles, we list some of the basic results of the thinning and superposition of stochastic processes for convenience below.

(i) Example 4.3(a) in [59, pp. 75-76], thinning of renewal processes: Given a renewal process $\{S_n\}$ with rate λ, let each point S_n for $n = 1, 2, \ldots$ be omitted from the sequence with probability $1 - p$ and retained with probability p for some constant p in $0 < p < 1$, each such point S_n being treated independently. The sequence of retained points, denoted by $\{S_n^p\}$, is called the thinned process with retaining probability p. Then $\{S_n^p\}$ is also renewal with rate $p\lambda$.

(ii) From [60, Section 14], we know the following proposition for the superposition of independent, stationary processes. Let $N_k(t)$ be a stationary process with rate λ_k, then the superposition of K such independent processes $N(t) = \sum_{k=1}^{K} N_k(t)$ is also stationary with rate $\lambda = \sum_{k=1}^{K} \lambda_k$.

(iii) Two stationary stochastic processes are said to be probabilistic replicas of each other if their generic interarrival times are identically distributed [22, p. 21].

The following theorem asserts the queueing systems at each real vehicle is $\Sigma GI/GI/1$ [34] under non-location based policies. This means the arrival process at each real vehicle is the superposition of independent renewal processes and the service time process has positive i.i.d. interarrival times. It also establishes the critical role of κ^c in stability of these queues under the assumption that every customer keeps their VV stable, i.e., $\lambda_k^V \leq \mu^V$.

Theorem 4.2. *Under non-location based scheduling policies, each real vehicle RV_m is a $\Sigma GI/GI/1$ queue with task arrival rate $\lambda^R = \frac{\sum_{k=1}^K \lambda_k^V}{M}$. Moreover, assume homogeneous real vehicle subregions as in Definition 4.3. Then*

(i) all the real vehicle $\Sigma GI/GI/1$ queues are probabilistic replicas of each other, i.e., the interarrival time and service time of each queue are identically distributed, separately.

(ii) Let $\lambda_k^V \leq \mu^V$. When $\kappa < \kappa^c$, the $\Sigma GI/GI/1$ queue at each real vehicle is stable and TD_k exists. When $\kappa > \kappa^c$, the $\Sigma GI/GI/1$ queue at each real vehicle is unstable when $\lambda_k^V = \mu^V$, with κ and κ^c defined in (4.16) and (4.24).

Proof. (i) By Definition 4.1, our M-Voronoi tessellation creates equitable subregions and $\mathbf{A}_m$ is a Voronoi subregion. Then each task in the sequence $\langle Task_{ki}\rangle_{i=1}^{\infty}$ falls in subregion $\mathbf{A}_m$ with probability $\frac{1}{M}$. Hence the arrival time of tasks in the sequence $\langle Task_{ki}\rangle_{i=1}^{\infty}$ that fall in $\mathbf{A}_m$ is a thinned process of $\{T_{ki}^a\}_{i=1}^{\infty}$ with retaining probability $p = \frac{1}{M}$. We denote the thinned arrival process as $\{T_{ki}^{ap}\}_{i=1}^{\infty}$. Since $\{T_{ki}^a\}_{i=1}^{\infty}$ is renewal with rate λ_k^V, then $\{T_{ki}^{ap}\}_{i=1}^{\infty}$ is renewal with rate $\frac{\lambda_k^V}{M}$ by Example 4.3(a) in [59, pp. 75-76]. Thus the arrival process in subregion $\mathbf{A}_m$, $\left\{T_{(mj)}^a\right\}_{j=1}^{\infty}$, is the superposition of $\{T_{ki}^{ap}\}_{i=1}^{\infty}$, $k = 1, \ldots, K$, or K independent renewal processes. This proves the ΣGI. A renewal process is also stationary, so $\left\{T_{(mj)}^a\right\}_{j=1}^{\infty}$ is stationary, and the arrival rate at RV_m is $\lambda^R = \frac{\sum_{k=1}^K \lambda_k^V}{M}$ by [60, Section 14].

Moreover, the M thinned processes $\{T_{ki}^{ap}\}_{i=1}^{\infty}$ generated from the same VV_k arrival process are probabilistic replicas of each other because the retaining probabilities are all $\frac{1}{M}$. Since the arrival process of each RV subregion, $\left\{T_{(mj)}^a\right\}_{j=1}^{\infty}$, is the superposition of thinned replicas from each VV, then $\left\{T_{(mj)}^a\right\}_{j=1}^{\infty}$ are probabilistic replicas in m. We know that the service times at RV_m, $T_{(mj)}^{Rserv}$, are i.i.d. in m and j. This proves the second GI, and all the real vehicle $\Sigma GI/GI/1$ queues are probabilistic replicas of each other. (i) follows.

(ii) When $\lambda_k^V \leq \mu^V$, when $\kappa < \kappa^c$, $\lambda^R = \frac{\sum_{k=1}^K \lambda_k^V}{M} \leq \frac{\sum_{k=1}^K \mu^V}{M} = \kappa\mu^V < \kappa^c\mu^V = \mu^R$, thus each $\Sigma GI/GI/1$ queue at RV_m is stable by Loynes' stability condition [61]. $Task_{ki}$ corresponds to $Task_{(mj)}$, we denote by $Task_{ki(mj)}$ for the same task. By (4.20) we

know

$$T^{Rsys}_{ki(mj)} \to_p T^{Rsys}_{k(m)} \tag{4.25}$$

Since the all the RV $\Sigma GI/GI/1$ queues are probabilistic replicas of each other, then $T^{Rsys}_{k(m)}$ is i.i.d. in m, we denote by T^{Rsys}_k the generic term of $T^{Rsys}_{k(m)}$. Since a task from VV_k can fall in any of the RV subregions with probability $\frac{1}{M}$, thus (4.25) becomes

$$T^{Rsys}_{ki(mj)} \to_p T^{Rsys}_k \tag{4.26}$$

Also, when $\lambda^V_k < \mu^V$, the $GI/GI/1$ queue at each VV is stable, $T^{Vsys}_{ki} \to T^{Vsys}_k$ in distribution. When $\lambda^V_k = \mu^V$, by our assumption that follows Theorem 4.1, the customer will provide arrival process that guarantees $T^{Vsys}_{ki} \to T^{Vsys}_k$ in distribution. Thus both T^{Rsys}_{ki} and T^{Vsys}_{ki} converges to T^{Rsys}_k and T^{Vsys}_k in distribution. The tardiness TD_k defined in (4.8) exists.

When $\kappa > \kappa^c$ and $\lambda^V_k = \mu^V$, $\lambda^R = \frac{\sum_{k=1}^K \lambda^V_k}{M} = \frac{\sum_{k=1}^K \mu^V}{M} = \kappa\mu^V > \kappa^c\mu^V = \mu^R$. Thus each $\Sigma GI/GI/1$ queue at RV_m is unstable by Loynes' stability condition [61]. □

4.3 Scheduling Policies

In this section, we design the scheduling policies inside each RV subregion. We assumed that the task arrival process $\{T^a_{ki}\}$ of VV_k is renewal with generic interarrival time I^a_k in Section 4.1. In this section we further assume that I^a_k is i.i.d. in k, i.e., the interarrival times $I^a_{ki} = T^a_{ki} - T^a_{k(i-1)}$ are i.i.d. in k and i. We thus denote by I^a the generic term of I^a_k. Then the task arrival rate are the same for each VV, $\lambda^V_k = \lambda^V = \frac{1}{E[I^a]}$.

Definition 4.4. *The set of K virtual vehicles are said to host homogeneous tasks if the task interarrival times I^a_{ki}, locations X_{ki} and sizes T^S_{ki} are all i.i.d. in k and i, separately, $k = 1, \ldots, K$ and $i = 1, 2, \ldots$.*

In this section we assume that all the VVs host homogeneous tasks. Then the steady state real system time for VV_k, T^{Rsys}_k, is identically distributed in k. We denote by T^{Rsys} the generic term of T^{Rsys}_k. Then (4.25) and (4.26) become

$$T^{Rsys}_{ki(mj)} \to_p T^{Rsys} \tag{4.27}$$

Also the steady state virtual system time of VV_k, T^{Vsys}_k, is i.i.d. in k. We denote by T^{Vsys} the generic value of T^{Vsys}_k. Thus

$$T^{Vsys}_{ki(mj)} \to_p T^{Vsys} \tag{4.28}$$

Then the tardiness TD_k are the same for all the VVs, thus TD is not only the average value but also the generic term of TD_k. We have

$$TD_k = TD = \frac{E\left[\max\left\{T^{Rsys} - T^{Vsys}, 0\right\}\right]}{E\left[T^{Vserv}\right]} \tag{4.29}$$

From (4.27), (4.28) and (4.29) we know that when the virtual vehicles are homogeneous and the real vehicle subregions are homogeneous, it suffices to analyze only one RV subregion, and the results represent the generic results of the system.

Let Φ denote a class of non-location based scheduling policies that are non-preemptive and deadline smooth. A scheduling policy is said to be non-preemptive if under this policy the real vehicle always complete an initiated task even when a priority task enters the system in the meanwhile. A scheduling policy is said to be deadline smooth if under this policy the RV serves all the tasks including those whose deadline has passed [37]. The common policies in queueing theory such as FCFS, LCFS, ROS and SJF [10] are non-location based, non-preemptive and deadline smooth policies. Our scheduling policies introduced in this section such as EVDF, EDVDF and credit scheduling policy are shown to be non-location based, non-preemptive and deadline smooth policies in Theorem 4.3.

Let TD^{ϕ} denote the tardiness as defined by (4.29) under scheduling policy $\phi \in \Phi$. We consider the following optimization problem:

Find $\psi \in \Phi$ s.t.

$$TD^{\psi} = \min_{\phi \in \Phi} TD^{\phi} \tag{4.30}$$

for every $k = 1, \ldots, K$.

We propose the scheduling policies Earliest Virtual Deadline First (EVDF) when the task size is known a priori in Definition 4.5, its variation Earliest Dynamic Virtual Deadline First (EDVDF) when the task size is not known a priori in Definition 4.6, and the credit scheduling policy in Definition 4.7. These scheduling policies are motivated by the simple earliest deadline first and credit scheduler in the cloud computing literature [36]. The scheduling quantum for us is the task size. The size cannot be preempted. The Xen schedulers [36] have a constant scheduling quantum and can preempt tasks. In our kind of cloud computing preemption would waste the time spent traveling to the location. In this chapter we analyze the value of cloud computing with moving servers without preemption.

Definition 4.5. *Under the Earliest Virtual Deadline First (EVDF) scheduling policy, when a real vehicle becomes available, the real vehicle always hosts the virtual vehicle whose current task has the earliest virtual deadline as defined in (4.4) from the pool of virtual vehicles whose current task falls in the real vehicle subregion.*

Definition 4.6. *Under the Earliest Dynamic Virtual Deadline First (EDVDF) scheduling policy, when a real vehicle becomes available, the real vehicle always hosts*

the virtual vehicle whose current task has the earliest dynamic virtual deadline as defined in (4.36) from the pool of virtual vehicles whose current task falls in the real vehicle subregion.

Definition 4.7. *Under the credit scheduling policy, when a real vehicle becomes available, the real vehicle always hosts the virtual vehicle with the maximum current credit as described in Section 4.3.3 from the pool of virtual vehicles whose current task falls in the real vehicle subregion.*

4.3.1 Earliest Virtual Deadline First

The optimality of our EVDF scheduling policy among all the non-location based non-preemtive and deadline smooth scheduling policies follows from Theorem 1 of [37]. We restate this result for convenience below.

Theorem 1 of [37]: For any convex function $g : \mathbb{R} \to \mathbb{R}$, $E\left[g\left(R^{\phi}\right)\right] \leq E\left[g\left(R^{\psi}\right)\right]$ whenever $\phi \ll \psi$.

Theorem 1 of [37] assumes a $G/GI/1$ queue served by non-preemtive and deadline smooth scheduling policies. The G in a $G/GI/1$ queue means the task arrival process is stationary and ergodic. ϕ and ψ denote two admissible non-preemptive and deadline smooth scheduling policies. $\phi \ll \psi$ when ϕ always chooses a customer having a deadline earlier than that of the customer chosen by ψ. In particular the earliest deadline first (EDF) scheduling policy always gives priority to the customer having the earliest deadline, and the latest deadline first (LDF) one gives priority to the customer having the latest deadline. Then, by definition $EDF \ll \phi \ll LDF$ for any admissible scheduling policy ϕ. R^{ϕ} is the steady-state value of $R_n = D_n - T_n - W_n$ under policy ϕ, where D_n, T_n and W_n are the deadline, arrival time and waiting time of the n-th task.

The following theorem shows that our EVDF, EDVDF and credit scheduling policy are in the class of non-location based, non-preemptive, and deadline smooth scheduling policies Φ. Moreover, EVDF optimizes tardiness within this class.

Theorem 4.3. *(i) Let $\phi \in \{EVDF, EDVDF, Credit\}$, as in Definitions 4.5, 4.6 and 4.7, then $\phi \in \Phi$, i.e., ϕ is non-location based, non-preemptive, and deadline smooth.*

(ii) Assume homogeneous virtual vehicles and homogeneous real vehicle subregions, let $\lambda^R < \mu^R$, with λ^R and μ^R defined in Theorem 4.2 and (4.23). Then $TD^{EVDF} = \min_{\phi \in \Phi} TD^{\phi}$.

Proof. (i) The EVDF, EDVDF and credit scheduling policies schedule only based on virtual deadlines, dynamic virtual deadlines of each task and credit of each VV, separately. They are independent of the distance between two consecutively executed

tasks, $D_{(mj)}$. Thus $D_{(mj)}$ is the distance between two random task locations in subregion $\mathbf{A}_m$. Thus $D_{(mj)}$ is i.i.d. in j and has the same distribution as $D_{(m)}$. Thus the three policies are non-location based. The three policies always serve all tasks, even if deadlines have passed. Thus they are smooth with respect to the virtual deadlines as defined in (4.4). The three policies always completes an initiated task even when a priority task enters the system in the meanwhile. Thus the three policies are non-preemptive. This proves part (i).

(ii) Since $\phi \in \Phi$ is non-location based, the queue in an RV subregion is $\Sigma GI/GI/1$ by Theorem 4.2. Since ΣGI is a subset of G, then a $\Sigma GI/GI/1$ queue is also a $G/GI/1$ queue. Also, ϕ is non-preemptive and deadline smooth, Thus Theorem 1 of [37] holds in each RV subregion under $\phi \in \Phi$.

We define $R_{(mj)} = T^{dead}_{(mj)} - T^{a}_{(mj)} - W^{R}_{(mj)}$, where $W^{R}_{(mj)}$ is the waiting time of $Task_{(mj)}$, and is defined as the time difference between the arrival time $T^{a}_{(mj)}$ and when RV_m begins to travel to $Task_{(mj)}$. Thus $T^{Rsys}_{(mj)} = W^{R}_{(mj)} + T^{Rserv}_{(mj)}$.

Thus
$\max\left\{T^{Rsys}_{(mj)} - T^{Vsys}_{(mj)}, 0\right\} = -\min\left\{T^{Vsys}_{(mj)} - T^{Rsys}_{(mj)}, 0\right\}$
$= -\min\left\{T^{dead}_{(mj)} - T^{a}_{(mj)} - W^{R}_{(mj)} - T^{Rserv}_{(mj)}, 0\right\}$
$= -\min\left\{R_{(mj)} - T^{Rserv}_{(mj)}, 0\right\}$.

When $\lambda^R < \mu^R$, the $\Sigma GI/GI/1$ queue at RV_m is stable, we have $R_{(mj)} \to R_{(m)} = T^{dead}_{(m)} - T^{a}_{(m)} - W^{R}_{(m)}$ in distribution. Since all the RV subregions are homogeneous, we can write the generic term $R = T^{dead} - T^{a} - W^{R}$.

Thus $E\left[\max\left\{T^{Rsys} - T^{Vsys}, 0\right\}\right]$
$= E\left[-\min\left\{R - T^{Rserv}, 0\right\}\right]$
$= \int_{t=0}^{\infty} E\left[-\min\left\{R - t, 0\right\}\right] dF_{T^{Rserv}}(t)$,
where $F_{T^{Rserv}}(t)$ is the cumulative distribution function (cdf) of T^{Rserv}. Since function $g(x) = -\min\left\{x - t, 0\right\}$ is a convex function when t is a constant, and R has the same definition as R in Theorem 1 of [37], then $E\left[-\min\left\{R^{\phi} - t, 0\right\}\right] \leq E\left[-\min\left\{R^{\psi} - t, 0\right\}\right]$ when $\phi \ll \psi$ for any constant t by Theorem 1 of [37]. Thus $E\left[\max\left\{T^{Rsys} - T^{Vsys}, 0\right\}\right]^{\phi} \leq E\left[\max\left\{T^{Rsys} - T^{Vsys}, 0\right\}\right]^{\psi}$ when $\phi \ll \psi$, where the superscript $^{\phi}$ means the value is obtained when the scheduling policy is ϕ.

We know $TD = \frac{E\left[\max\left\{T^{Rsys} - T^{Vsys}, 0\right\}\right]}{E[T^{Vserv}]}$ by (4.29). Notice that $E\left[T^{Vserv}\right]$ is a constant, then $TD^{\phi} \leq TD^{\psi}$ when $\phi \ll \psi$. In particular, $TD^{EVDF} \leq TD^{\phi}$ since $EVDF \ll \phi$ for any $\phi \in \Phi$. Thus $TD^{EVDF} = \min_{\phi \in \Phi} TD^{\phi}$. □

Different task arrival processes will generate different tardiness values. We identify a

worst-case arrival process maximizing tardiness. We prove the special case $\eta = 1$ of Definition 4.8 generates the worst case. This is Theorem 4.4.

Definition 4.8. *An arrival process $\{T_{ki}^a\}$ is called an η-arrival process if*

$$T_{ki}^a = \begin{cases} 0, & i \leq \eta \\ T_{k(i-\eta)}^{dead}, & i > \eta \end{cases} \tag{4.31}$$

where $\eta \in \mathbb{N}$.

For $\eta = 1$ the definition implies the arrival of the current task is the virtual deadline of the previous task. Thus the service times are also the interarrival times and the process at the output of the VV is identical to the arrival process, i.e., it is also an $\eta = 1$ process. The following theorem establishes the special role of the $\eta = 1$ process.

Theorem 4.4. *Assume homogeneous virtual vehicles and homogeneous real vehicle subregions under the EVDF scheduling policy, let $\kappa \leq \kappa^c$, then the η-arrival process with $\eta = 1$ for all the virtual vehicles achieves the maximum TD among all the renewal processes.*

Proof. To prove the η-arrival process with $\eta = 1$ maximizes TD, we show for any given task locations $\{X_{ki}\}$ and task sizes $\left\{T_{ki}^S\right\}$, an arbitrary renewal arrival process $\{T_{ki}^a\}$ will have a TD less than that of the $\eta = 1$ arrival process. We denote by $\left\{T_{ki}^{a(\eta=1)}\right\}$ the $\eta = 1$ arrival process. Thus $T_{ki}^{a(\eta=1)} = T_{k(i-1)}^{dead(\eta=1)}$ by Definition 4.8. For an arbitrary renewal arrival process $\{T_{ki}^a\}$, we construct an arrival process $\{T_{ki}^{a1}\}$ such that $T_{ki}^{a1} = \max\left\{T_{ki}^a, T_{ki}^{a(\eta=1)}\right\}$ for all k and i. Thus $T_{ki}^{a1} = \max\left\{T_{ki}^a, T_{k(i-1)}^{dead(\eta=1)}\right\}$. We construct another arrival process $\{T_{ki}^{a2}\}$ such that $T_{ki}^{a2} = \max\left\{T_{ki}^{a1}, T_{k(i-1)}^{dead1}\right\}$. Note the three processes are constructed to have the same $\{X_{ki}\}$ and $\left\{T_{ki}^S\right\}$. With the same $\{X_{ki}\}$ and $\left\{T_{ki}^S\right\}$, we denote by $TD^{\phi}\left(\{T_{ki}^a\}\right)$ the tardiness under policy ϕ when the arrival processes of the VV_k are $\{T_{ki}^a\}$. We want to show $TD^{EVDF}\left(\{T_{ki}^a\}\right) \leq TD^{EVDF}\left(\{T_{ki}^{a2}\}\right)$ and $TD^{EVDF}\left(\{T_{ki}^{a2}\}\right) \leq TD^{EVDF}\left(\left\{T_{ki}^{a(\eta=1)}\right\}\right)$, separately.

Since $\{X_{ki}\}$ and $\left\{T_{ki}^S\right\}$ are the same for all three processes, then the service times T_{ki}^{Vserv} are the same for the processes. For an arbitrary arrival process $\{T_{ki}^a\}$, we have $T_{ki}^{dead} \geq T_{ki}^{dead(\eta=1)}$ by (4.4) and Definition 4.8.

Comparing $\{T_{ki}^a\}$ and $\{T_{ki}^{a1}\}$, we have $T_{ki}^a \leq T_{ki}^{a1}$, and $T_{ki}^{dead} = T_{ki}^{dead1}$ for all k and i. The latter can be seen by induction on (4.4). First, $T_{k1}^{dead} = T_{k1}^{dead1}$. Second, if $T_{k(i-1)}^{dead} = T_{k(i-1)}^{dead1}$, then $T_{ki}^{dead} = \max\left\{T_{ki}^a, T_{k(i-1)}^{dead}\right\} + T_{ki}^{Vserv}$, and $T_{ki}^{dead1} = \max\left\{\max\left\{T_{ki}^a, T_{k(i-1)}^{dead(\eta=1)}\right\}, T_{k(i-1)}^{dead1}\right\} + T_{ki}^{Vserv} = \max\left\{T_{ki}^a, T_{k(i-1)}^{dead1}\right\} + T_{ki}^{Vserv}$ since $T_{k(i-1)}^{dead1} \geq T_{k(i-1)}^{dead(\eta=1)}$. Thus $T_{k(i-1)}^{dead} = T_{k(i-1)}^{dead1}$ implies $T_{ki}^{dead} = T_{ki}^{dead1}$. This is true

for every i by induction. Similarly, comparing $\{T_{ki}^{a1}\}$ and $\{T_{ki}^{a2}\}$, we have $T_{ki}^{a1} \leq T_{ki}^{a2}$, and $T_{ki}^{dead1} = T_{ki}^{dead2}$ by induction. Thus $T_{ki}^{a} \leq T_{ki}^{a1} \leq T_{ki}^{a2}$ and $T_{ki}^{dead} = T_{ki}^{dead1} = T_{ki}^{dead2}$ for all k and i.

In each RV_m subregion, we construct a scheduling policy $\phi \in \Phi$, possibly non-work-conserving, such that under ϕ and $\{T_{ki}^{a}\}$, $k = 1, \ldots, K$, RV_m will copy the behavior of RV_m under $EVDF$ and $\{T_{ki}^{a2}\}$, i.e., RV_m flies to, executes, and completes each task at the same time, separately, comparing the two cases. Such ϕ exists because the set of tasks available to RV_m under $\{T_{ki}^{a2}\}$ is always a subset of the set of tasks available to RV_m under $\{T_{ki}^{a}\}$ at any time since $T_{ki}^{a} \leq T_{ki}^{a2}$ for all k and i. Thus $TD^{\phi}(\{T_{ki}^{a}\}) = TD^{EVDF}(\{T_{ki}^{a2}\})$. Also $TD^{EVDF}(\{T_{ki}^{a}\}) \leq TD^{\phi}(\{T_{ki}^{a}\})$ by Theorem 4.3. So $TD^{EVDF}(\{T_{ki}^{a}\}) \leq TD^{EVDF}(\{T_{ki}^{a2}\})$ for any given $\{X_{ki}\}$ and $\{T_{ki}^{S}\}$.

Under the arrival process $\{T_{ki}^{a2}\}$, we have $T_{ki}^{a2} = \max\left\{T_{ki}^{a1}, T_{k(i-1)}^{dead1}\right\}$ and $T_{k(i-1)}^{dead1} = T_{k(i-1)}^{dead2}$. By (4.4) $T_{ki}^{dead2} = \max\left\{\max\left\{T_{ki}^{a1}, T_{k(i-1)}^{dead1}\right\}, T_{k(i-1)}^{dead2}\right\} + T_{ki}^{Vserv} = \max\left\{T_{ki}^{a1}, T_{k(i-1)}^{dead1}\right\} + T_{ki}^{Vserv} = T_{ki}^{a2} + T_{ki}^{Vserv}$. Thus $T_{ki}^{Vsys2} = T_{ki}^{dead2} - T_{ki}^{a2} = T_{ki}^{Vserv}$. Thus the virtual system time is the same under both $\{T_{ki}^{a2}\}$ and $\left\{T_{ki}^{a(\eta=1)}\right\}$, and equals T_{ki}^{Vserv}. Also, $T_{ki}^{a2} - T_{k(i-1)}^{dead2} = \max\left\{T_{ki}^{a1}, T_{k(i-1)}^{dead1}\right\} - T_{k(i-1)}^{dead2} = \max\left\{T_{ki}^{a1}, T_{k(i-1)}^{dead2}\right\} - T_{k(i-1)}^{dead2} \geq 0$. Thus under $\{T_{ki}^{a2}\}$, VV_k idles for $T_{ki}^{a2} - T_{k(i-1)}^{dead2} \geq 0$ before generating $Task_{ki}$. But under $\left\{T_{ki}^{a(\eta=1)}\right\}$, VV_k never idles, i.e., $T_{ki}^{a(\eta=1)} - T_{k(i-1)}^{dead(\eta=1)} = 0$.

When every VV is under $\left\{T_{ki}^{a(\eta=1)}\right\}$, the interarrival time is T_{ki}^{Vserv}. The arrival process of the subregion of RV_m, $\left\{T_{(mj)}^{a(\eta=1)}\right\}$, is the superposition of K thinned renewal processes of $\left\{T_{ki}^{a(\eta=1)}\right\}$ by Theorem 4.2. When every VV is under $\{T_{ki}^{a2}\}$, the interarrival time is $T_{ki}^{Vserv} + Id_{ki}$, where $Id_{ki} \geq 0$ is the idle time between two consecutive tasks. Thus the arrival process of the subregion of RV_m, $\left\{T_{(mj)}^{a2}\right\}$, is the superposition of K thinned renewal processes of $\{T_{ki}^{a2}\}$. Thus the generic interarrival time of $\left\{T_{(mj)}^{a2}\right\}$, $I_{(m)}^{a2}$, is stochastically greater than that of $\left\{T_{(mj)}^{a(\eta=1)}\right\}$, $I_{(m)}^{a(\eta=1)}$, i.e., $I_{(m)}^{a2} \geq_{st} I_{(m)}^{a(\eta=1)}$. A random variable A is stochastically greater than B, denoted $A \geq_s tB$, if $P(A > t) \geq P(B > t)$ for all $-\infty < t < \infty$. Thus, we can construct a scheduling policy $\phi \in \Phi$ in the subregion of RV_m under $\left\{T_{(mj)}^{a2}\right\}$ such that under ϕ, RV_m executes tasks in the same order as EVDF but always idles for $I_{(m)}^{a2} - I_{(m)}^{a(\eta=1)}$ right after the completion of each task. Since the virtual system time of each task are the same for both cases. Thus $TD^{\phi}(\{T_{ki}^{a2}\}) = TD^{EVDF}\left(\left\{T_{ki}^{a(\eta=1)}\right\}\right)$. Also $TD^{EVDF}(\{T_{ki}^{a2}\}) \leq TD^{\phi}(\{T_{ki}^{a2}\})$ by Theorem 4.3. Thus $TD^{EVDF}(\{T_{ki}^{a2}\}) \leq TD^{EVDF}\left(\left\{T_{ki}^{a(\eta=1)}\right\}\right)$ for any given $\{X_{ki}\}$ and $\{T_{ki}^{S}\}$.

So we have $TD^{EVDF}\left(\{T^a_{ki}\}\right) \leq TD^{EVDF}\left(\left\{T^{a(\eta=1)}_{ki}\right\}\right)$ for an arbitrary renewal arrival process $\{T^a_{ki}\}$ for any given $\{X_{ki}\}$ and $\left\{T^S_{ki}\right\}$. Thus this is also true when we take expectation on $\{X_{ki}\}$ and $\left\{T^S_{ki}\right\}$. So the η-arrival process with $\eta = 1$ achieves the maximum TD among all the renewal processes. □

The provider wants to know the right gain κ for a given number of real vehicles M and a given task arrival process $\{T^a_{ki}\}$ for each virtual vehicle for a guaranteed level of tardiness. We assume homogeneous virtual vehicles and homogeneous real vehicle subregions, and consider the case when every customer is fully utilizing their VVs, i.e., $\lambda^V = \mu^V$. Under a scheduling policy ϕ, if one fixes the number of real vehicles M and increases the number of virtual vehicles, one increases the gain κ, but increases the tardiness TD, thus reducing the performance isolation. Conversely at any level, say $TD = \alpha$ for the tardiness there is a largest value of κ in the sense that any increase in the number of virtual vehicles without increase in M will increase the tardiness TD above the level α. We denote this largest value of κ by $\kappa^\phi_\alpha(M)$ at each α. We denote by $TD^\phi(M, \kappa)$ the tardiness when the number of RVs is M and the gain is κ under scheduling policy ϕ. Thus $\kappa^\phi_\alpha(M)$ is defined as

$$\kappa^\phi_\alpha(M) = \max_{TD^\phi(M,\kappa)\leq\alpha, \lambda^V=\mu^V} \kappa \tag{4.32}$$

Recall that L is the generic term of L_{ki} as defined in (4.2), and D is the generic term of $D_{(m)}$ as defined in (4.21). We assume each RV subregion satisfies

$$E[D] = \frac{c_1 E[L]}{\sqrt{M}}, E\left[D^2\right] = \frac{c_2 E\left[L^2\right]}{M} \tag{4.33}$$

where c_1 and c_2 are positive constants. In particular, when region $\mathbf{A}$ and subregions $\mathbf{A}_m$ are squares, $c_1 = c_2 = 1$.

The following theorem asserts the largest achievable gain without compromising performance isolation actually increases with the number of real vehicles M. In other words, larger systems are able to support more customers per real vehicle without compromising performance isolation. We call this economy of scale [38].

Theorem 4.5. *Assume homogeneous virtual vehicles and homogeneous real vehicle subregions satisfying (4.33) under the EVDF scheduling policy.* $\kappa^{EVDF}_\alpha(M)$ *increases with* M. *Moreover, when* $T^S_{ki} = 0$, $\kappa^{EVDF}_\alpha(M)$ *is* $\Theta\left(\sqrt{M}\right)$. *When* $E\left[T^S\right] > 0$, $\kappa^{EVDF}_\alpha(M) < 1 + \frac{E[L]}{v^V E[T^S]}$.

Proof. Since this theorem is only about the EVDF scheduling policy, we omit the superscript ^{EVDF} in the notations.

From (4.29) we know it suffices to analyze only one RV subregion to obtain the average tardiness of the system $TD = \frac{E\left[\max\left\{T^{Rsys} - T^{Vsys}, 0\right\}\right]}{E[T^{Vserv}]}$.

From Theorem 4.2 we know that each RV_m is a $\Sigma GI/GI/1$ queue with arrival rate $\lambda^R = \frac{\sum_{k=1}^{K} \lambda_k^V}{M} = \frac{K\lambda^V}{M} = \kappa\lambda^V = \kappa\mu^V$. Let $T^{Rsys}(M,\kappa)$ denote the real system time when the number of RVs is M and the gain is κ. Let $E[D](M)$ and $\mu^R(M)$ denote the $E[D]$ and μ^R values when the number of RVs is M. If $M_1 < M_2$, then $E[D](M_1) > E[D](M_2)$ by (4.33). Thus $\mu^R(M_1) < \mu^R(M_2)$ by (4.23). Notice that the arrival rate of both the $\Sigma GI/GI/1$ queues under M_1 and M_2 are the same $\lambda^R = \kappa\mu^V$. Then $T^{Rsys}(M_1,\kappa) >_{st} T^{Rsys}(M_2,\kappa)$ because the arrival rate does not change but service rate increases. Utilizing the fact that $A \geq_{st} B$ if and only if for all non-decreasing functions g, $E[g(A)] \geq E[g(B)]$, and noticing that the virtual system time T^{Vsys} does not change with M or κ, and $g(x) = \max(x,0)$ is non-decreasing, we have $TD(M_1,\kappa) > TD(M_2,\kappa)$. Thus $\exists\kappa' > \kappa$, s.t. $TD(M_1,\kappa) = TD(M_2,\kappa')$. Thus $\kappa_\alpha(M_1) < \kappa_\alpha(M_2)$ by (4.32). Thus $\kappa_\alpha(M)$ increases with M.

$\kappa_\alpha(M)$ is defined based on $\lambda^V = \mu^V$. By Theorem 4.2 we know that the $\Sigma GI/GI/1$ queue at each real vehicle is unstable when $\kappa > \kappa^c$, then we should keep $\kappa \leq \kappa^c$. Thus $\kappa_\alpha(M) \leq \kappa^c = \frac{\frac{E[L]}{v^V} + E\left[T^S\right]}{\frac{E[D]}{v^R} + E[T^S]}$ by (4.24). When $T_{ki}^S = 0$, substituting (4.33) we have $\kappa_\alpha(M) \leq \kappa^c = \frac{v^R\sqrt{M}}{c_1 v^V}$. Then $\kappa_\alpha(M)$ is $O\left(\sqrt{M}\right)$.

To establish the lower bound of $\kappa_\alpha(M)$, we first analyze the tardiness under FCFS. Since $K = \kappa_\alpha(M)M$, and $\kappa_\alpha(M)$ increases with M, then as $M \to \infty$, $K \to \infty$. The superposition of independent renewal processes converges to a Poisson process as the number of component processes tend to infinity [62]. Then the $\Sigma GI/GI/1$ queue of each subregion becomes an $M/GI/1$ queue with arrival rate $\lambda^R = \kappa_\alpha(M)\mu^V$ as both M and K goes to infinity. Since the task size $T_{ki}^S = 0$, the service rate of the queue is $\frac{v^R}{E[D]}$ by (4.23). By Pollaczek-Khinchine formula [63, 64] we have the expected waiting time of a task in the $M/GI/1$ queue under FCFS, $E\left[W^{R(FCFS)}\right] = \frac{\lambda^R \frac{E\left[D^2\right]}{(v^R)^2}}{2\left(1-\lambda^R\frac{E[D]}{v^R}\right)}$.

$T^{Rsys(FCFS)} = W^{R(FCFS)} + \frac{D}{v^R}$, $T^{Vsys} = W^V + \frac{L}{v^V}$, where W^V is the steady state waiting time of a task in the $GI/GI/1$ queue at each virtual vehicle. T^{Vsys} and T^{Vserv} is only determined by the $GI/GI/1$ queue at each virtual vehicle, and do not depend on the scheduling policy of each RV subregion. When $M \to \infty$, $\frac{D}{v^R} \leq \frac{L}{v^V}$ almost surely (a.s.). Thus $T^{Rsys(FCFS)} - T^{Vsys} = W^{R(FCFS)} + \frac{D}{v^R} - W^V - \frac{L}{v^V} \leq W^{R(FCFS)}$ a.s..

Thus $TD^{FCFS} = \frac{E\left[\max\left\{T^{Rsys(FCFS)} - T^{Vsys}, 0\right\}\right]}{E[T^{Vserv}]}$

$\leq \frac{E\left[\max\left\{W^{R(FCFS)}, 0\right\}\right]}{E[T^{Vserv}]} = \frac{E[W^{R(FCFS)}]}{E[T^{Vserv}]}$

$= \frac{\lambda^R \frac{E\left[D^2\right]}{(v^R)^2}}{2E[T^{Vserv}]\left(1-\lambda^R\frac{E[D]}{v^R}\right)}$ a.s. when $M \to \infty$.

Since EVDF achieves minimum TD by Theorem 4.3, $TD^{EVDF} = \alpha$ implies $\alpha \leq \frac{\lambda^R \frac{E\left[D^2\right]}{(v^R)^2}}{2E[T^{Vserv}]\left(1-\lambda^R \frac{E[D]}{v^R}\right)}$.

Substituting (4.33) and $\lambda^R = \kappa_\alpha(M)\mu^V$ we have
$\kappa_\alpha(M) \geq \frac{2\alpha E\left[T^{Vserv}\right]}{2\alpha E[T^{Vserv}]\mu^V c_1 \frac{E[L]}{v^R\sqrt{M}} + \mu^V c_2 \frac{E[L^2]}{(v^R)^2 M}}$
$\geq \frac{2\alpha E\left[T^{Vserv}\right]}{2\alpha E[T^{Vserv}]\mu^V c_1' \frac{E[L]}{v^R\sqrt{M}}} = \frac{v^R\sqrt{M}}{\mu^V c_1' E[L]}$ for some $c_1' > c_1$ when $M \to \infty$. Thus $\kappa_\alpha(M)$ is $\Omega\left(\sqrt{M}\right)$.

Thus we have $\kappa(\alpha)$ is $\Theta\left(\sqrt{M}\right)$ when $T_{ki}^S = 0$.

When $E\left[T^S\right] > 0$, $\kappa_\alpha(M) \leq \kappa^c = \frac{\frac{E[L]}{v^V} + E\left[T^S\right]}{\frac{E[D]}{v^R} + E[T^S]} < \frac{\frac{E[L]}{v^V} + E\left[T^S\right]}{E[T^S]} = 1 + \frac{E[L]}{v^V E[T^S]}$. □

We define the travel ratio of VV_k as the expected travel time over the expected service time of a task.

$$r_{tr} = \frac{\frac{E[L]}{v^V}}{E\left[T^{Vserv}\right]} = \frac{\frac{E[L]}{v^V}}{\frac{E[L]}{v^V} + E\left[T^S\right]} \tag{4.34}$$

Theorem 4.6. *(i) Under the allocation policy in Definition 4.1, $MC_k \leq r_{tr} B_{Vk} v^V$. Moreover, when VV_k is fully utilized $MC_k \geq \left(1 - \frac{1}{M}\right) r_{tr} B_{Vk} v^V$.*

(ii) Under the η-arrival process with $\eta = 1$, $SL_k \leq r_{tr} \leq 1$.

Proof. (i) By definition, $MC_k = B_{Vk} \frac{E[Z_k L]}{E\left[I_k^{dead}\right]}$. Z_k indicates migration between two consecutive tasks. Thus $E\left[Z_k L\right] \leq E[L]$. By Theorem 4.1 $\lambda_k^{Vdead} \leq \mu^V$. Since $\lambda_k^{Vdead} = \frac{1}{E\left[I_k^{dead}\right]}$ and $\mu^V = \frac{1}{E[T^{Vserv}]}$, then $E\left[I_k^{dead}\right] \geq E\left[T^{Vserv}\right]$. Thus $MC_k \leq B_{Vk} \frac{E[L]}{E[T^{Vserv}]} = B_{Vk} \frac{\frac{E[L]}{v^V}}{E[T^{Vserv}]} v^V = B_{Vk} r_{tr} v^V$ when substituting (4.34).

When VV_k is fully utilized, $\lambda_k^{Vdead} = \mu^V$ by Theorem 4.1, thus $E\left[I_k^{dead}\right] = E\left[T^{Vserv}\right]$. Greater L implies the distance between two consecutive tasks are larger, then it is more probable that the two tasks will fall in different RV subregions and cause a migration, i.e., $P\left(Z_k = 1 \,|L\right)$ increases with L. Thus L and Z_k are positively correlated. Then $Cov\left(Z_k, L\right) \geq 0$. Thus $E\left[Z_k L\right] = E\left[Z_k\right] E[L] + Cov\left(Z_k, L\right) \geq E\left[Z_k\right] E[L]$. Also, $P\left(Z_k = 0\right) = \frac{1}{M}$ and $P\left(Z_k = 1\right) = 1 - \frac{1}{M}$ since Z_k indicates whether two consecutive tasks fall in the same RV subregion. Thus $E\left[Z_k\right] = 1 - \frac{1}{M}$. Then $MC_k \geq B_{Vk} \frac{E[Z_k]E[L]}{E[T^{Vserv}]} = \left(1 - \frac{1}{M}\right) B_{Vk} \frac{E[L]}{E[T^{Vserv}]} = \left(1 - \frac{1}{M}\right) B_{Vk} r_{tr} v^V$.

(ii) By Definition 4.8, $T_k^{Vsys} = \frac{L}{v^V} + T^S$. Also $T_k^{Rsys} \geq \frac{D}{v} + T^S$. Thus $T_k^{Vsys} - T_k^{Rsys} \leq$

$\frac{L}{v^V} + T^S - \frac{D}{v} - T^S \leq \frac{L}{v^V}$. Thus $SL_k = \frac{E\left[\max\left\{T_k^{Vsys} - T_k^{Rsys}, 0\right\}\right]}{E[T^{Vserv}]} \leq \frac{E\left[\max\left\{\frac{L}{v^V}, 0\right\}\right]}{E[T^{Vserv}]} = \frac{E\left[\frac{L}{v^V}\right]}{E[T^{Vserv}]} = r_{tr} = \frac{\frac{E[L]}{v^V}}{\frac{E[L]}{v^V} + E[T^S]} \leq 1.$ □

4.3.2 Earliest Dynamic Virtual Deadline First

When the task sizes are not known a priori, the provider only knows the task size after execution, the RV hosts the VV whose current task has the earliest dynamic virtual deadline.

The task size may not be known a priori in practice as assumed by our EVDF scheduling policy. Therefore we define and evaluate another scheduling policy named Earliest Dynamic Virtual Deadline First (EDVDF) which assigns task virtual deadlines based on an estimated task size prior to task completion and updates size to the true value after completion. (4.35) and (4.36) specify the deadline computation. $T_{ki}^{S'}$ denotes the task size estimate. We define the estimated service time of VV_k on $Task_{ki}$ as

$$T_{ki}^{Vserv'} = \frac{L_{ki}}{v^V} + T_{ki}^{S'} \tag{4.35}$$

The dynamic virtual deadline $T_{ki}^{dead}(t)$ is calculated as follows.

$$T_{ki}^{dead}(t) = \begin{cases} \max\left\{T_{ki}^a, T_{k(i-1)}^{dead}(t)\right\} + T_{ki}^{Vserv'}, & t < T_{ki}^{comp} \\ \max\left\{T_{ki}^a, T_{k(i-1)}^{dead}(t)\right\} + T_{ki}^{Vserv}, & t \geq T_{ki}^{comp} \end{cases} \tag{4.36}$$

where $T_{k0}^{dead} \equiv 0$ and T_{ki}^{Vserv} is given in (4.3).

When $Task_{ki}$ is completed at $t = T_{ki}^{comp}$ by an RV, T_{ki}^S is known, $T_{ki}^{dead}(t)$ is updated, the scheduling policy updates all the tasks hosted by VV_k following $Task_{ki}$, i.e., updates $T_{k(i+1)}^{dead}(t), T_{k(i+2)}^{dead}(t), \ldots$ according to equation (4.36). The real vehicle then serves the next task with the earliest $T_{ki}^{dead}(t)$. In this way the scheduling policy utilizes the actual task sizes as they become known.

In our implementation of EDVDF in Section 4.4, the task size estimate is set to be $T_{ki}^{S'} = E\left[T_k^{Sexe}\right]$, where $\left\{T_k^{Sexe}\right\}$ denotes the sizes of the set of previously executed tasks hosted by VV_k. $E\left[T_k^{Sexe}\right] = 0$ if $\left\{T_k^{Sexe}\right\}$ is empty. Time-series methods can also be used to estimate $T_{ki}^{S'}$.

4.3.3 Credit Scheduling Policy

We adopt the credit scheduler described in [65] with some changes for the our spatial case. Under the credit scheduling policy, each VV keeps a balance of credits which

can be negative. Each credit has a value of 1 second of RV time while it emulates the VV perfectly. A token bucket algorithm [66] is implemented to manage the credits of each VV. Each VV has a bucket. Credits are added to the bucket at constant rate 1 per second, and are expended during service. The bucket can hold at most c credits. The inflow credits are discarded when the bucket is full.

As shown in Figure 4.2, VVs are divided into three states: UNDER, with a non-negative credit balance, OVER, with a negative credit balance, and INACTIVE or halted. The VVs are listed in decreasing order of credit balance. Thus, those in UNDER state are ahead of those in OVER state. The VV at the head of the queue has the most credits and is selected for execution when an RV becomes available. In work-conserving (WC) mode, when no VVs are in the UNDER state, one in the OVER state will be chosen, allowing it to receive more than its share of RV time. In non-work-conserving (NWC) mode, the RV will go idle instead.

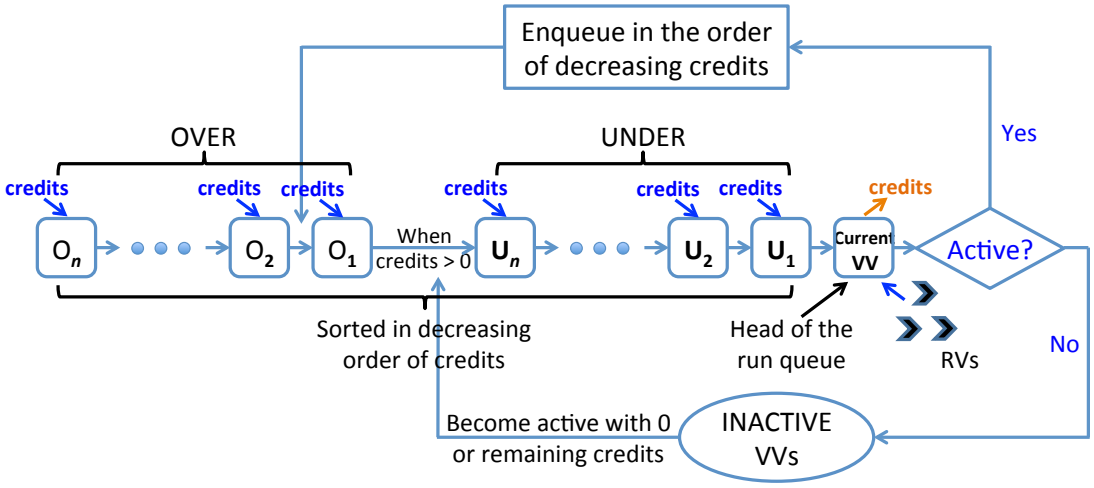


Figure 4.2. The credit scheduling policy.

The VV with the most credits is called the current VV, say VV_k. The token bucket algorithm is illustrated in Figure 4.3. When an RV becomes available, the RV will travel to and execute the current task hosted by VV_k, say $Task_{ki}$. The scheduler debits $T_{ki}^{Vserv'}$ credits from the bucket of VV_k. $T_{ki}^{Vserv'}$ is calculated according to (4.35). The scheduler finds the VV with the maximum credit balance, and the VV with the most credits will become the new current VV. When $Task_{ki}$ is completed, the scheduler will know the size T_{ki}^{S} and compute the true service time T_{ki}^{Vserv}. The consumed credits of $Task_{ki}$, T_{ki}^{Vserv}, is calculated according to (4.3). This is the appropriate credits, or RV time, the scheduler should debit for executing $Task_{ki}$. The scheduler returns $T_{ki}^{Vserv'} - T_{ki}^{Vserv}$ credits to VV_k to adjust the debited credits to be exactly T_{ki}^{Vserv}. This method of debiting $T_{ki}^{Vserv'}$ before execution and adjusting to T_{ki}^{Vserv} after execution is because the scheduler does not know the size of $Task_{ki}$, and thus does not know T_{ki}^{Vserv}, before execution. If the task size T_{ki}^{S} is known a priori, then $T_{ki}^{Vserv'} = T_{ki}^{Vserv}$. The right amount of credits will be debited at beginning and the adjusted amount equals 0.

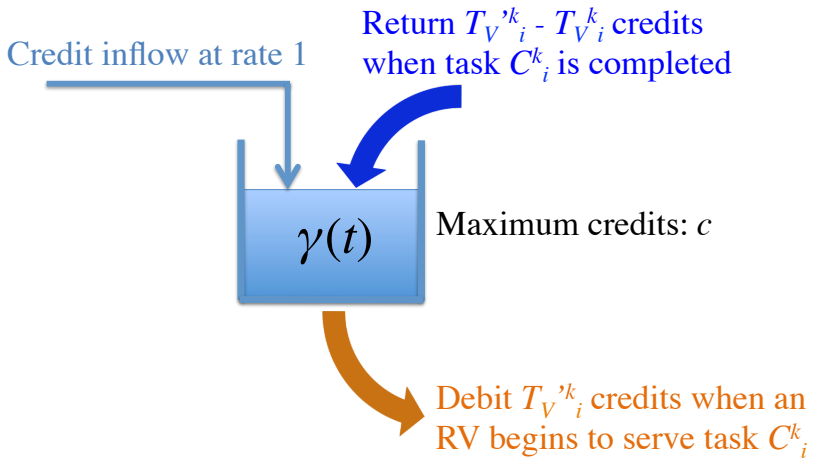


Figure 4.3. The token bucket algorithm.

When $Task_{ki}$ is completed, there are two ways VV_k goes:

(i) Enqueue according to the credit balance such that the list of VVs is in decreasing order of credits.

(ii) Go INACTIVE if the next $Task_{k(i+1)}$ has not arrived yet, or if the customer's reservation of VV_k ends.

VVs in an INACTIVE state are divided into two categories:

(i) The VV is still under the reservation of a customer, all the arrived tasks hosted by the VV have been executed and the next task has not arrived yet. In this case, the credits continue flowing into the bucket up to a maximum of c. The VV becomes active again with the remaining credit balance and enqueues in decreasing order of credits upon arrival of the next task.

(ii) The VV is not under reservation. In this case, the credits stop flowing to the bucket. The VV becomes active again with 0 credit balance and enqueues the active VVs in decreasing order of credits when a customer begins to reserve it.

4.4 Experiments

We simulate the system under homogeneous real vehicles and homogeneous virtual vehicles under Earliest Virtual Deadline First (EVDF), Earliest Dynamic Virtual Deadline First (EDVDF) and credit scheduling policies in this section.

4.4.1 Simulation Setup

All the simulations are done in a square region $\mathbf{A}$ of size $a \times a$, where $a = 10\,\text{m}$. The number of RVs is M, $\sqrt{M} \in \mathbb{N}$. The speed of the RVs equals the virtual speed $v^R = v^V = 1\,\text{m/s}$. Each virtual vehicle hosts tasks with $\eta = 1$ arrival process. Since the $\eta = 1$ process maximizes tardiness, and excludes multiplexing gain by having each virtual vehicle fully utilized. So this is a worst-case simulation, and the gain observed is migration gain only. The performance measures include the performance isolation and fairness index based on tardiness and delivery probability, together with slack and migration cost. We simulate 1300 tasks per VV but calculate the metrics using only the 100-th to 600-th tasks to ensure the metrics are computed at steady-state. When the 600-th task of each VV is under execution, all the other VVs still have tasks. Each task is uniformly distributed in region $\mathbf{A}$. The square region $\mathbf{A}$ is divided into M square subregions, each with edge length $\frac{a}{\sqrt{M}}$. Table 4.1 summarizes the simulation setup.

Table 4.1. Simulation setup

Size of region $\mathbf{A}$	10 m $\times$ 10 m
Virtual speed, v^V	1 m/s
RV speed, v^R	1 m/s
Task size, T^S_{ki}	Uniformly distributed
Task location, X_{ki}	Uniformly distributed in region $\mathbf{A}$
Task arrival process, $\{T^a_{ki}\}$	$\eta = 1$ arrival process
# Tasks per VV	1300
Tasks used in metric calculation	100th - 600th task of each VV
Policies	EVDF, EDVDF and Credit scheduling policy

4.4.2 Simulation Results

Figure 4.4 shows the performance isolations and fairness indices based on tardiness and delivery probability, together with slacks and migration costs under EVDF for different numbers of real vehicles and gains, or different numbers of virtual vehicles, when the task size is zero.

Figure 4.5 verifies Theorem 4.5. Subfigures 4.5(a) and 4.5(b) are the contours of performance isolations based on tardiness and delivery probability shown in subfigures 4.4(a) and 4.4(b) when the task size is zero. We can see that the provider supports a given number of virtual vehicles with significantly fewer real vehicles that travel at the virtual speed while guaranteeing high performance isolation, for example, 750 VVs Vs. 100 RVs while guaranteeing the relative expected tardiness to be less than 1% in 4.5(a), and 560 VVs Vs. 100 RVs while guaranteeing the average delivery probability is greater than 98%. The migration gain increases in the order of the square root

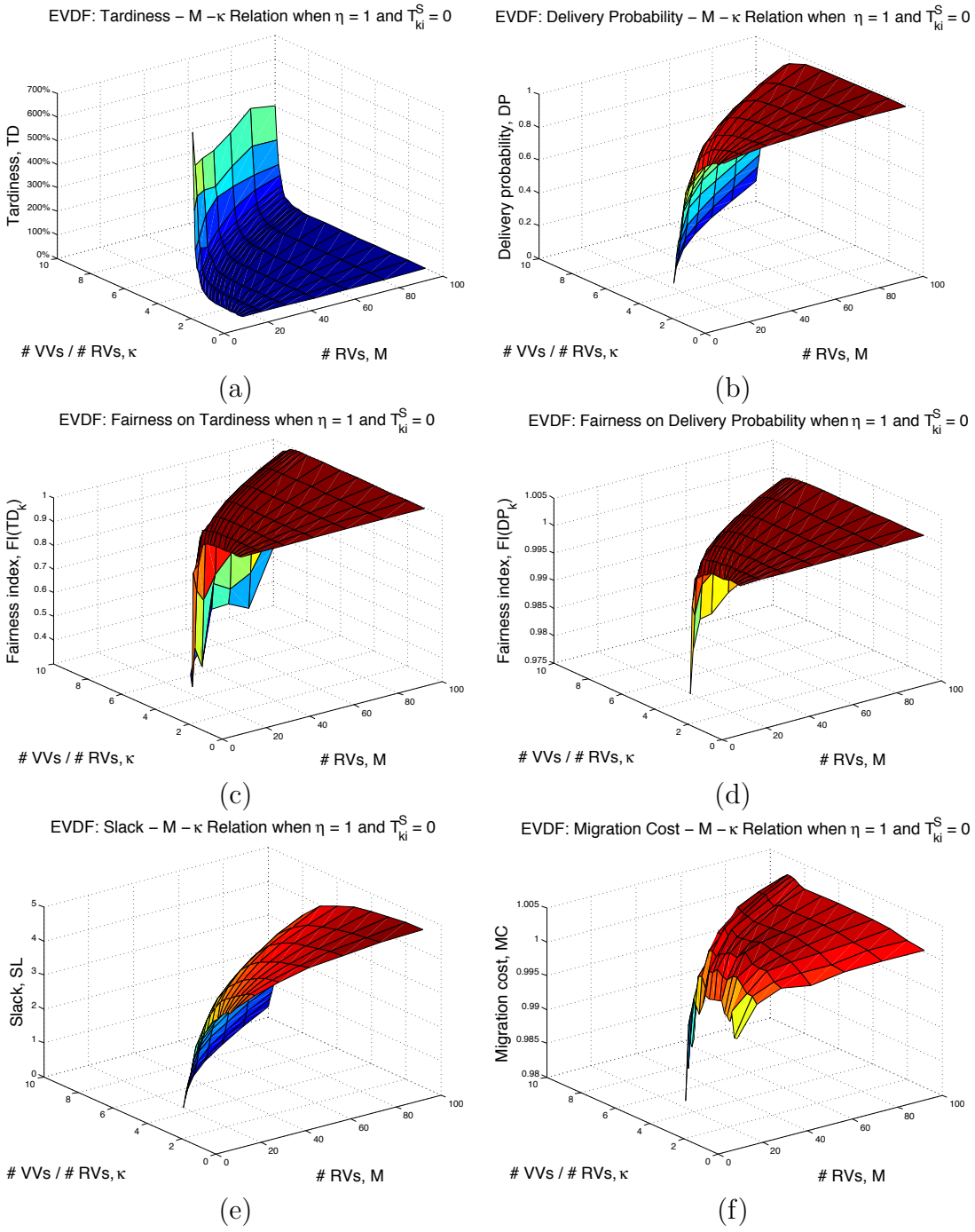


Figure 4.4. Tardinesses, delivery probabilities, fairness indices, slacks, and migration costs with different numbers of RVs and gains when the task size is zero under the $\eta = 1$ process under the EVDF scheduling policy.

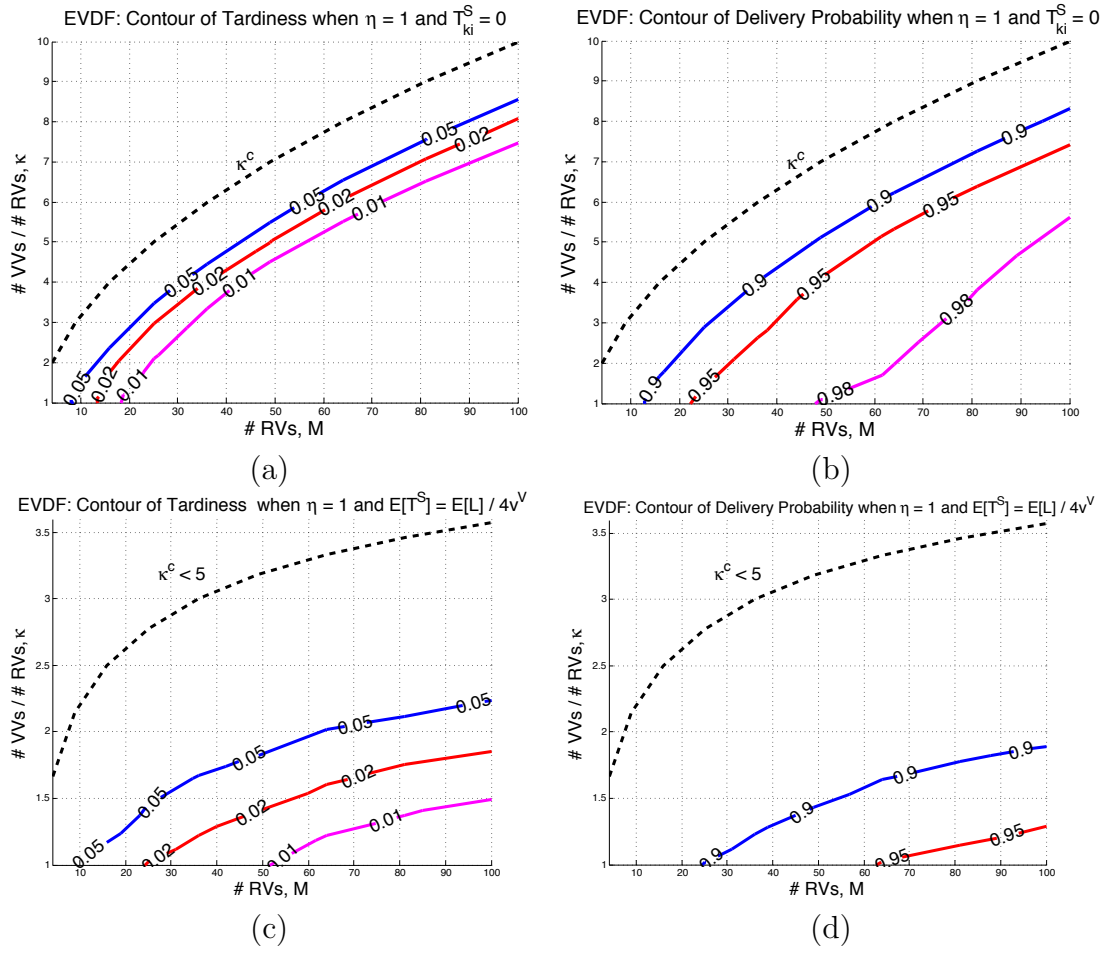


Figure 4.5. Contours of tardiness and delivery probability with different numbers of RVs and gains when the task size is zero and $E\left[T^S\right] = \frac{E[L]}{4v^V}$ under the $\eta = 1$ process under the EVDF scheduling policy.

of the number of real vehicles while guaranteeing the same performance isolation based on tardiness and delivery probability, showing economy of scale. Subfigures 4.5(c) and 4.5(d) are the contours of performance isolations based on tardiness and delivery probability when the mean task size is 25% of the mean flying time, i.e., $E\left[T^S\right] = \frac{E[L]}{4v^V}$. We can see that the gain is upper bounded by a constant as asserted in Theorem 4.5. Also, for a given performance isolation level and number of real vehicles, the gain or the number of virtual vehicles hosted decreases as the task size increases comparing Subfigures 4.5(a) and 4.5(c), 4.5(b) and 4.5(d), separately, for example, 150 VVs Vs. 100 RVs while guaranteeing the relative expected tardiness to be less than 1% in 4.5(a), and 130 VVs Vs. 100 RVs while guaranteeing the average delivery probability is greater than 98%. The gain diminishes as the task size increases. The virtual vehicle generates gain when traveling, not when unmoving executing.

Figure 4.6 also shows the diminishing gain as the relative task size $\frac{E\left[T^S\right]v^V}{E[L]}$ increases. For example, to guarantee the average relative tardiness to be less than 1% using 100 RVs, the number of VVs hosted decreases from 750 to 150 as the relative task size increases from 0 to 25%.

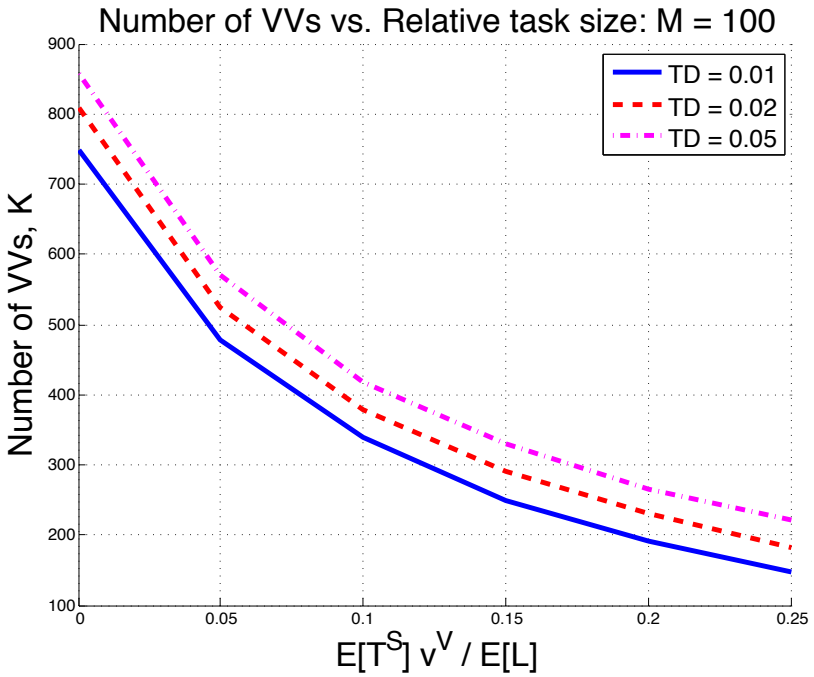
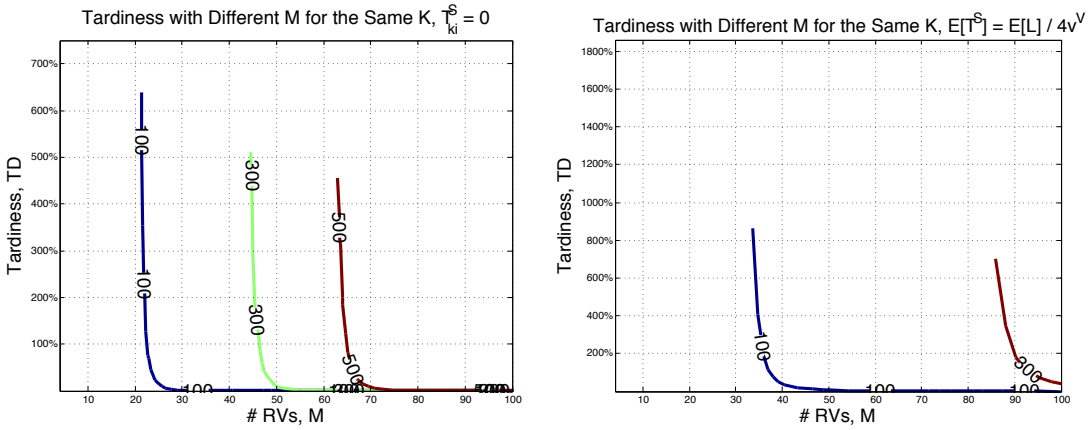


Figure 4.6. The number of virtual vehicles hosted by 100 real vehicles to guarantee a certain level of tardiness under EVDF with different task sizes.

Figure 4.7 shows the performance isolations based on tardiness with different numbers of RVs M to host the same number of VVs K under EVDF with different task sizes. We can see in Subfigure 4.7(a) that when $T_{ki}^S = 0$ and the number of VVs K is fixed, average relative expected tardiness decreases as the number of RVs M increases. This change becomes very sharp when $M \approx 23$ for the case $K = 100$. The average tardiness

is very small once M is greater than 28. This sharp change is also revealed in the case when $K = 300$ and $K = 500$, where the average tardiness becomes very small once $M \geq 52$ and $M \geq 74$, respectively. The transition between low and high performance isolations is sharp. This implies that the system is very easy to operate because the provider can easily determine the appropriate number of real vehicles to host a give number of virtual vehicle and a guaranteed performance isolation. For the case when the mean task size $E\left[T^S\right] = \frac{E[L]}{4v^V}$ as shown in Subfigure 4.7(b), similar phenomenon follows, but the number of RVs needed to guarantee the same performance isolation increases to host the same number of VVs. This also implies that the gain diminishes as the task size increases.



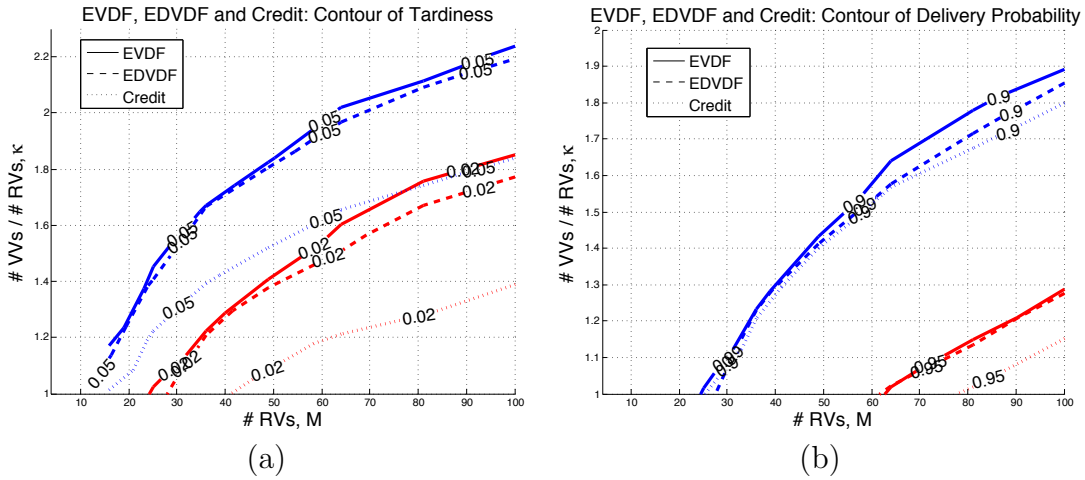
(a) (b)

Figure 4.7. Tardiness with different numbers of RVs M for the same number of VVs under EVDF for different task sizes.

Figure 4.8 compares EVDF, EDVDF and credit scheduling policies on the same settings as Subfigures 4.5(c) and 4.5(d). We can see that EVDF achieves higher gain than EDVDF, and EDVDF achieves higher gain than the credit scheduling policy for a given number of RVs and a guaranteed performance isolation based on both tardiness and delivery probability.

4.5 Summary

We proposed the concept of a virtual vehicle in multi-customer systems with location specific tasks to create performance isolation [3], which enables cloud computing in space. In Section 4.1, we illustrated the role of the virtual vehicle in cloud computing in space. In the service-level agreement (SLA), each virtual vehicle has a virtual speed v^V, and the performance of each virtual vehicle is measured by tardiness and delivery probability. To quantify performance isolation and fairness across virtual vehicles, we used the measure PI [28] and Jain's fairness indices FI [29]. In Section 4.2, we designed virtual vehicle allocation and scheduling policies. The allocation



(a) (b)

Figure 4.8. Comparison of EVDF, EDVDF and Credit scheduling policies on tardiness and delivery probability when the mean task size $E\left[T^S\right] = \frac{E[L]}{4v^V}$ under the $\eta = 1$ process.

involves dividing the service region into equitable subregions. The real vehicles travel less after this division. The scheduling policies include earliest virtual deadline first, earliest dynamic virtual deadline first and credit scheduling policies, adapted from the CPU schedulers in conventional cloud computing [36].

We analyzed the system in Section 4.2 following standard results in queueing theory [15, 34], stochastic and dynamic vehicle routing [1, 12, 17], and soft real-time systems [26, 27]. Each virtual vehicle is a $GI/GI/1$ queue [15] by Theorem 4.1, and each real vehicle is a $\Sigma GI/GI/1$ queue [34] by Theorem 4.2. We showed that EVDF minimizes tardiness in Theorem 4.3 by utilizing the optimality of earliest deadline first in real-time queues [37]. We also identified the $\eta = 1$ arrival process (Definition 4.8) as the worst-case arrival process that maximizes tardiness among all the renewal processes in Theorem 4.4. We showed task size dependent economy of scale [38] in Theorem 4.5.

Afterwards, we simulated the system under the three scheduling policies under the worst-case arrival process - the $\eta = 1$ arrival process. Simulation results show that (i) a virtual vehicle performs as well as a real vehicle with high performance isolation. (ii) The provider can support a given number of virtual vehicles with significantly fewer real vehicles that travel at the virtual speed while guaranteeing high performance isolation.

Chapter 5

Conclusion

The contribution of this dissertation to systems with task arrivals in time and space is twofold: (i) In the single-customer case, we provided a stability condition in Chapter 2, and an approximation for the system time distribution in Chapter 3 for a (real) vehicle routing problem, the Dynamic Traveling Repairman Problem (DTRP) [1]. (ii) In the multi-customer case, we proposed the concept of a virtual vehicle to create performance isolation [3] in Chapter 4, enabling cloud computing in space.

First in Chapter 2, we proved a necessary and sufficient condition for stability in Theorem 2.5 in the Dynamic Traveling Repairman Problem (DTRP) [1] under the class of Polling-Sequencing (P-S) policies (Definition 2.1) satisfying unlimited-polling (Definition 2.4) and economy of scale (Definition 2.2). The number of tasks inside each polling partition was shown to be a Markov chain in Theorem 2.1. Non-location based policies and some common location based policies such as TSP, NN and DA were shown to have economy of scale in Theorem 2.3. The P-S class includes some of the policies proven to be optimal for the expectation of system time under light and heavy loads in the DTRP literature.

Then in Chapter 3, we gave a good approximation of the distribution of the system time that is easy to compute under the PART-n-TSP policy by utilizing the approximation results of the distribution of system time T, together with $E[T]$ and $Var[T]$ known for polling systems [23, 24]. We compared PART-n-TSP with PART-TSP [2] and Nearest Neighbor [1] on $E[T]$ and $\sigma[T]$ in Tables 3.1 and 3.2, since the latter two are considered near optimal in the literature. The results show that in practice PART-n-TSP achieves lower $\sigma[T]$ than PART-TSP and NN and lower $E[T]$ than PART-TSP when the load ρ is not too small or too large. We also proved that PART-n-TSP is $E[T]$ optimal under light load ($\rho \to 0^+$) and asymptotically optimal under heavy load ($\rho \to 1^-$) in Theorem 3.1.

Next in Chapter 4, we proposed the concept of a virtual vehicle in the multi-customer

case to create performance isolation [3], which enables cloud computing in space. In Section 4.1, we illustrated the role of the virtual vehicle in cloud computing in space. In the service-level agreement (SLA), each virtual vehicle has a virtual speed v^V, and the performance of each virtual vehicle is measured by tardiness and delivery probability. To quantify performance isolation and fairness across virtual vehicles, we used the measure PI [28] and Jain's fairness indices FI [29]. In Section 4.2, we designed virtual vehicle allocation and scheduling policies. The allocation involves dividing the service region into equitable subregions. The real vehicles travel less after this division. The scheduling policies include earliest virtual deadline first, earliest dynamic virtual deadline first and credit scheduling policies, adapted from the CPU schedulers in conventional cloud computing [36].

We analyzed the system in Section 4.2 using results in queueing theory [15, 34], stochastic and dynamic vehicle routing [1, 12, 17], and soft real-time systems [26, 27]. Each virtual vehicle is a $GI/GI/1$ queue [15] by Theorem 4.1, and each real vehicle is a $\Sigma GI/GI/1$ queue [34] by Theorem 4.2. We showed that EVDF minimizes tardiness in Theorem 4.3 by utilizing the optimality of earliest deadline first in real-time queues [37]. We also identified the $\eta = 1$ arrival process (Definition 4.8) as the worst-case arrival process that maximizes tardiness among all the renewal processes in Theorem 4.4. We showed task size dependent economy of scale [38] in Theorem 4.5.

Afterwards, we simulated the system under our three scheduling policies for the worst-case arrival process - the $\eta = 1$ arrival process. Simulation results show that (i) a virtual vehicle performs as well as a real vehicle with high performance isolation. (ii) The provider can support a given number of virtual vehicles with significantly fewer real vehicles that travel at the virtual speed while guaranteeing high performance isolation.

Bibliography

[1] D. J. Bertsimas and G. Van Ryzin, "A stochastic and dynamic vehicle routing problem in the euclidean plane," *Operations Research*, vol. 39, no. 4, pp. 601–615, 1991.

[2] H. Xu, *Optimal policies for stochastic and dynamic vehicle routing problems*. Cambridge, 1994. [Online]. Available: http://books.google.com/books?id=brJzNwAACAAJ

[3] M. Rosenblum, "The reincarnation of virtual machines," *Queue*, vol. 2, no. 5, pp. 34–40, Jul. 2004. [Online]. Available: http://doi.acm.org/10.1145/1016998.1017000

[4] C. F. Daganzo, "The length of tours in zones of different shapes," *Transportation Research Part B: Methodological*, vol. 18, no. 2, pp. 135 – 145, 1984. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0191261584900274

[5] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver, "Google street view: Capturing the world at street level," *Computer*, vol. 43, 2010. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5481932&tag=1

[6] K. Jenvey, "NASA Unmanned Aircraft Measures Low Altitude Greenhouse Gases," December 2011. [Online]. Available: http://www.nasa.gov/centers/ames/news/releases/2011/11-101AR_prt.htm

[7] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava, "Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines," in *Proceedings of the 25th IEEE International Real-Time Systems Symposium*, ser. RTSS '04, Washington, DC, USA, 2004, pp. 296–305. [Online]. Available: http://dx.doi.org/10.1109/REAL.2004.31

[8] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. pp. 80–91, 1959. [Online]. Available: http://www.jstor.org/stable/2627477

[9] M. Pavone, N. Bisnik, E. Frazzoli, and V. Isler, "A stochastic and dynamic vehicle routing problem with time windows and customer impatience," *Mob.*

Netw. Appl., vol. 14, no. 3, pp. 350–364, Jun. 2009. [Online]. Available: http://dx.doi.org/10.1007/s11036-008-0101-1

[10] A. Wierman, "Scheduling for today's computer systems: bridging theory and practice," Ph.D. dissertation, Pittsburgh, PA, USA, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1329734

[11] G. Laporte and Y. Nobert, "Exact algorithms for the vehicle routing problem," in *Surveys in Combinatorial Optimization*, ser. North-Holland Mathematics Studies, M. M. Silvano Martello, Gilbert Laporte and C. Ribeiro, Eds. North-Holland, 1987, vol. 132, pp. 147 – 184. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304020808732353

[12] M. Pavone, E. Frazzoli, and F. Bullo, "Adaptive and distributed algorithms for vehicle routing in a stochastic and dynamic environment," *Automatic Control, IEEE Transactions on*, vol. 56, no. 6, pp. 1259–1274, June 2011.

[13] Y. Gu, D. Bozdag, E. Ekici, F. zgner, and C.-G. Lee, "Partitioning based mobile element scheduling in wireless sensor networks," in *In. Proc. Second Annual IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, 2005, pp. 386–395.

[14] C. F. Daganzo, "An approximate analytic model of many-to-many demand responsive transportation systems," *Transportation Research*, vol. 12, no. 5, pp. 325 – 333, 1978. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0041164778900072

[15] J. Cohen, *The single server queue*, ser. North-Holland series in applied mathematics and mechanics. North-Holland Pub. Co., 1982.

[16] D. J. Bertsimas and G. Van Ryzin, "Stochastic and dynamic vehicle routing with general demand and interarrival time distributions," *Advances in Applied Probability*, pp. 947–978, 1993.

[17] J. Huang and R. Sengupta, "Stability of dynamic traveling repairman problem under polling-sequencing policies," in *European Control Conference*, July 2013.

[18] P. De, J. B. Ghosh, and C. E. Wells, "Expectation-variance analyss of job sequences under processing time uncertainty," *International Journal of Production Economics*, vol. 28, no. 3, pp. 289 – 297, 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0925527392900172

[19] S. Sarin, B. Nagarajan, S. Jain, and L. Liao, "Analytic evaluation of the expectation and variance of different performance measures of a schedule on a single machine under processing time variability," *Journal of Combinatorial Optimization*, vol. 17, pp. 400–416, 2009. [Online]. Available: http://dx.doi.org/10.1007/s10878-007-9122-0

[20] B. G. C. Dellaert and B. E. Kahn, "How tolerable is delay? consumers' evaluations of internet web sites after waiting," *Journal of Interactive Marketing*, vol. 13, pp. 41–54, 1999.

[21] M. K. Hui and L. Zhou, "How does waiting duration information influence customers' reactions to waiting for services?1," *Journal of Applied Social Psychology*, vol. 26, no. 19, pp. 1702–1717, 1996. [Online]. Available: http://dx.doi.org/10.1111/j.1559-1816.1996.tb00093.x

[22] H. Takagi, *Queueing analysis: a foundation of performance evaluation, vol. 1 : vacation and priority systems*, ser. Queueing Analysis. North-Holland, 1991. [Online]. Available: http://books.google.com/books?id=DXZRAAAAMAAJ

[23] O. Boxma, J. Bruin, and B. Fralix, "Sojourn times in polling systems with various service disciplines," *Performance Evaluation*, vol. 66, no. 11, pp. 621 – 639, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166531609000765

[24] J. L. Dorsman, R. D. van der Mei, and E. M. M. Winands, "A new method for deriving Waiting-Time approximations in polling systems with renewal arrivals," *Stochastic Models*, vol. 27, pp. 318 – 332, 2011.

[25] G. J. Popek and R. P. Goldberg, "Formal requirements for virtualizable third generation architectures," *Commun. ACM*, vol. 17, no. 7, pp. 412–421, Jul. 1974. [Online]. Available: http://doi.acm.org/10.1145/361011.361073

[26] G. Buttazzo, *Soft real-time systems: predictability vs. efficiency*, ser. Series in computer science. Springer, 2005. [Online]. Available: http://books.google.com/books?id=egYUXKLUG8YC

[27] L. Abeni and G. Buttazzo, "Qos guarantee using probabilistic deadlines," in *Real-Time Systems, 1999. Proceedings of the 11th Euromicro Conference on*, 1999, pp. 242–249.

[28] K. Ye, X. Jiang, D. Ye, and D. Huang, "Two optimization mechanisms to improve the isolation property of server consolidation in virtualized multi-core server," in *Proceedings of the 2010 IEEE 12th International Conference on HPCC*, Washington, DC, USA, 2010, pp. 281–288. [Online]. Available: http://dx.doi.org/10.1109/HPCC.2010.95

[29] R. Jain, D.-M. Chiu, and W. R. Hawe, *A quantitative measure of fairness and discrimination for resource allocation in shared computer system.* Eastern Research Laboratory, Digital Equipment Corporation, 1984.

[30] S. Craciunas, A. Haas, C. Kirsch, H. Payer, H. Röck, A. Rottmann, A. Sokolova, R. Trummer, J. Love, and R. Sengupta, "Information-Acquisition-as-a-Service for Cyber-Physical Cloud Computing," in *Proc. Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2010.

[31] C. Kirsch, E. Pereira, R. Sengupta, H. Chen, R. Hansen, J. Huang, F. Landolt, M. Lippautz, A. Rottmann, R. Swick, R. Trummer, and D. Vizzini, "Cyber-Physical Cloud Computing: The Binding and Migration Problem," in *Design, Automation and Test in Europe*, 2012.

[32] A. Host-Madsen and A. Nosratinia, "The multiplexing gain of wireless networks," in *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*, 2005, pp. 2065–2069.

[33] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman *et al.*, "Reservoir-when one cloud is not enough," *Computer*, vol. 44, no. 3, pp. 44–51, 2011.

[34] S. L. Albin, "On poisson approximations for superposition arrival processes in queues," *Management Science*, vol. 28, no. 2, pp. 126–137, 1982. [Online]. Available: http://EconPapers.repec.org/RePEc:inm:ormnsc:v:28:y:1982:i:2:p:126-137

[35] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo, "Equitable partitioning policies for robotic networks," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, 2009, pp. 2356 –2361.

[36] L. Cherkasova, D. Gupta, and A. Vahdat, "Comparison of the three CPU schedulers in Xen," *SIGMETRICS Performormance Evaluation Review*, vol. 35, no. 2, pp. 42–51, Sep. 2007. [Online]. Available: http://dx.doi.org/10.1145/1330555.1330556

[37] P. Moyal, "Convex comparison of service disciplines in real time queues," *Operations Research Letters*, vol. 36, no. 4, pp. 496 – 499, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167637708000060

[38] A. Mas-Colell, M. Whinston, and J. Green, *Microeconomic theory*. Oxford University Press, 1995. [Online]. Available: http://books.google.com/books?id=KGtegVXqD8wC

[39] D. J. Bertsimas and G. Van Ryzin, "Stochastic and dynamic vehicle routing in the euclidean plane with multiple capacitated vehicles," *Operations Research*, vol. 41, no. 1, pp. 60–76, 1993.

[40] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. Smith, "Dynamic vehicle routing for robotic systems," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1482–1504, Sept. 2011.

[41] H. Takagi, *Analysis of polling systems*, ser. MIT Press series in computer systems. MIT Press, 1986. [Online]. Available: http://books.google.com/books?id=fWpRAAAAMAAJ

[42] ——, "Queueing analysis of polling models: progress in 1990-1994," *Frontiers in Queueing*, pp. 119–146, 1997.

[43] V. Vishnevskii and O. Semenova, "Mathematical methods to study the polling systems," *Automation and Remote Control*, vol. 67, pp. 173–220, 2006. [Online]. Available: http://dx.doi.org/10.1134/S0005117906020019

[44] E. Altman, P. Konstantopoulos, and Z. Liu, "Stability, monotonicity and invariant quantities in general polling systems," *Queueing Systems*, vol. 11, pp. 35–57, 1992. [Online]. Available: http://dx.doi.org/10.1007/BF01159286

[45] C. Fricker and M. Jaibi, "Monotonicity and stability of periodic polling models," *Queueing Systems*, vol. 15, pp. 211–238, 1994. [Online]. Available: http://dx.doi.org/10.1007/BF01189238

[46] Z. Rosberg, "A positive recurrence criterion associated with multidimensional queueing processes," *Journal of Applied Probability*, pp. 790–801, 1980.

[47] A. Borovkov and R. Schassberger, "Ergodicity of a polling network," *Stochastic Processes and their Applications*, vol. 50, no. 2, pp. 253 – 262, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0304414994901228

[48] J. M. Steele, "Probabilistic and worst case analyses of classical problems of combinatorial optimization in euclidean space," *Math. Oper. Res.*, vol. 15, pp. 749–770, October 1990. [Online]. Available: http://dl.acm.org/citation.cfm?id=89215.89225

[49] S. Asmussen, *Applied Probability and Queues*, ser. Applications of Mathematics. Springer, 2003. [Online]. Available: http://books.google.co.uk/books?id=BeYaTxesKy0C

[50] M. A. A. Boon, E. M. M. Winands, I. J. B. F. Adan, and A. C. C. van Wijk, "Closed-form waiting time approximations for polling systems," *Perform. Eval.*, vol. 68, no. 3, pp. 290–306, Mar. 2011.

[51] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *Biosystems*, vol. 43, no. 2, pp. 73 – 81, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0303264797017085

[52] R. D. Van Der Mei and E. M. M. Winands, "A note on polling models with renewal arrivals and nonzero switch-over times," *Oper. Res. Lett.*, vol. 36, no. 4, pp. 500–505, Jul. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.orl.2008.01.008

[53] E. Lawler, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. New York: Wiley, 1985.

[54] P. Gupta and P. Kumar, "The capacity of wireless networks," *Information Theory, IEEE Transactions on*, vol. 46, no. 2, pp. 388–404, 2000.

[55] D. Boutcher and A. Chandra, "Does virtualization make disk scheduling passé?" *ACM SIGOPS Operating Systems Review*, vol. 44, no. 1, pp. 20–24, 2010.

[56] A. Zhong, H. Jin, S. Wu, X. Shi, and W. Gao, "Performance implications of non-uniform vcpu-pcpu mapping in virtualization environment," *Cluster Computing*, pp. 1–12, 2012.

[57] M. Kesavan, A. Gavrilovska, and K. Schwan, "On disk i/o scheduling in virtual machines," in *Proceedings of the 2nd conference on I/O virtualization*. USENIX Association, 2010, pp. 6–6.

[58] F. Anhalt and P. V.-B. Primet, "Analysis and experimental evaluation of data plane virtualization with xen," *International conference on Networking and Services (ICNS'06)*, vol. 0, pp. 198–203, 2009.

[59] D. Daley and D. Vere-Jones, *An introduction to the theory of point processes*, ser. Springer series in statistics. Springer-Verlag, 1988. [Online]. Available: http://books.google.com/books?id=bU4_AQAAIAAJ

[60] A. Khinchin, *Mathematical methods in the theory of queueing*, ser. Griffin's statistical monographs & courses. Griffin, 1969. [Online]. Available: http://books.google.com/books?id=neVQAAAAMAAJ

[61] R. Loynes, "The stability of a queue with non-independent inter-arrival and service times," in *Proc. Cambridge Philos. Soc*, vol. 58, no. 3. Cambridge Univ Press, 1962, pp. 497–520.

[62] E. Cinlar, "Superposition of point processes," in *In Stochastic Point Processes: Statistical Analysis, Theory and Applications (P. A. W. Lewis, ed.)*. New York: Wiley, 1972, pp. 549–606.

[63] F. Pollaczek, "Über eine aufgabe der wahrscheinlichkeitstheorie. i," *Mathematische Zeitschrift*, vol. 32, no. 1, pp. 64–100, 1930.

[64] A. Khinchin and R. C. S. M. CALIF., *The Mathematical Theory of a Stationary Queue*. Defense Technical Information Center, 1967. [Online]. Available: http://books.google.com/books?id=OJhNOAAACAAJ

[65] F. Zhou, M. Goel, P. Desnoyers, and R. Sundaram, "Scheduler vulnerabilities and attacks in cloud computing," *CoRR*, vol. abs/1103.0759, 2011.

[66] S. Shenker and J. Wroclawski, "General characterization parameters for integrated service network elements," *RFC 2215*, sept. 1997.