وزارة التعليم العالي والبحث العلمي

Ministry of Higher Education and Scientific Research

BADJI MOKHTAR-ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR ANNABA

جامعة باجي مختار ـ عنابة ـ

Faculty of Sciences
Department of Mathematics

Year: 2025/2026

THESIS

Presented with a view to obtaining the doctorate degree

Analysis of Retrial Queueing System With Different Types of Vacations

Stream
Applied Mathematics

Speciality
Probability and Statistics

By
HADJADJ Houari

SUPERVISOR: **Mrs. Nawel ARRAR** **MCA** **ENSIA – Algiers**

CO-SUPERVISOR: **Mr. Lahcene YAHIAOUI** **MCA** **U.D.M.T – Saida**

In front of the jury

PRESIDENT:	**Mrs. Nacira SEDDIK AMEUR**	**Prof.**	**U.B.M.- Annaba**
EXAMINER:	**Mr. Hacene BOUTABIA**	**Prof.**	**U.B.M. - Annaba**
EXAMINER:	**Mr. Mohamed BOUALEM**	**Prof.**	**U.A.M. - Bejaia**

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

BADJI MOKHTAR-ANNABA

UNIVERSITY

UNIVERSITE BADJI MOKHTAR

ANNABA

جامعة باجي مختار

ـعنابةـ

Faculté des Sciences

Département de Mathématiques

Année : 2025/2026

THÈSE

Présentée en vue de l'obtention du diplôme de Doctorat

Analyse stochastique de systèmes de files d'attente avec rappels et politiques de vacances variées.

Filière
Mathématiques Appliquées

Spécialité
Probabilités et Statistique

Par

HADJADJ Houari

DIRECTRICE DE THÈSE: Mme. Nawel Khadidja ARRAR MCA. ENSIA – Alger

CO-DIRECTEUR DE THÈSE: M. Lahcene YAHIAOUI MCA. UDMT – Saida

Devant le jury

PRESIDENT:	**Mme. Nacira SEDDIK AMEUR**	**Prof. U.B.M - Annaba**
EXAMINATEUR :	**M. Hacene BOUTABIA**	**Prof. U.B.M - Annaba**
EXAMINATEUR :	**M. Mohamed BOUALEM**	**Prof. U.A.M – Béjaïa**

Dedication

In the name of **Allah**, the Most Gracious, the Most Merciful.
All praise be to **Allah**, by whose grace good deeds are completed.

♡ To my beloved **wife** my support and my light along the path,
and to my dear son **Lokman** the flower of my days and the ever-renewing hope of my life.

♡ To my parents: **Baghdad** and **Aicha**, who raised me with tenderness and compassion,
and to my parents: **Mohamed** and **Mokhtaria**, who gave me life the root of my being,
whose prayers are the source of my blessings.

♡ To my dear brothers and sisters companions of my journey and my constant support.

♡ And to my loyal friends sources of inspiration **Lahcene** and **Mohamed Amine**;
thank you for your honesty and devotion.

To you all, I dedicate the fruit of my effort and the labor of my days.

Acknowledgements

I would like to express my deep gratitude to my thesis supervisor **Dr. Nawel ARRAR** for her continuous support, patience and advice throughout the duration of my thesis research. I also thank, **Dr. Lahcene YAHIAOUI** my thesis co-supervisor for his support.

I would like to thank the members of my committee, for their trust for accepting to evaluate my work. **Prof. Nacira SEDDIK AMEUR, Prof.Hacene BOUTABIA, Prof. Mohamed BOUALEM** They were very kind to agree to be on my committee.

Warm thanks to all the members of **Probability and Statistics Laboratory, Badji Mokhtar University**, for their kindness and expertise.

Many thanks to all my friends and colleagues, for their encouragement, backing, and understanding,

I would like to thank everybody who was important to the successful realization of thesis, as well as expressing my apology that I could not mention personally one by one.

Finally, I wish to express my deepest appreciation to my family for their love, understanding, and belief in me, particulary my parents, for their love and support over the years. Without the words of wisdom and teaching of my parents, I would never have persisted in the pursuit of my goals, from the smallest task to the largest achievement. I dedicate this thesis to them.

الملخص

تعالج هذه الأطروحة نماذج متقدمة من أنظمة الطوابير ذات إعادة المحاولة، والتي تدمج عدداً من الخصائص التشغيلية الواقعية كما هو الحال في أنظمة الاتصالات والحوسبة السحابية والمراكز البيانية المقيدة بالطاقة. تم اقتراح وتحليل نموذجين رئيسيين في هذا العمل.

في الجزء الأول، تم دراسة نظام طابور بخادم واحد يتضمن إجازات تشغيلية، وزمن إعداد، وإيقاف تلقائي للحفاظ على الطاقة، وآلية إصلاح مثالية. العملاء الذين لا يجدون الخادم متاحاً يدخلون في مجموعة إعادة المحاولة، ويحاولون مجدداً بعد زمن عشوائي. وعند قدوم عميل جديد في حالة خمول الخادم، يتم الدخول في مرحلة الإعداد، والتي قد تتعرض للفشل وتتطلب إصلاحاً قبل استئناف الخدمة. تم تحليل ديناميكية النظام باستخدام تقنية المتغيرات المساعدة، كما تم تقديم مقاييس الأداء، وتحليل الموثوقية، والدراسة التحسينية مدعومة بأمثلة عددية باستخدام لغة Python.

في الجزء الثاني، تم توسيع النموذج السابق بإدخال العملاء السلبيين ونظام إصلاح ثانٍ مختلف. العملاء السلبيون، عند وصولهم، يتسببون مباشرة في تعطل الخادم، ما يؤدي إلى إصلاح من نوع خاص. هذا النموذج الأعمق يعكس سيناريوهات أعطال أكثر واقعية. كما تم تطبيق خوارزمية التحسين عبر السرب الجزيئي (PSO) لتحسين أداء النظام، بالإضافة إلى تحليل استراتيجيات مثلى لتحديد أفضل تكوينات تشغيلية. وقد تم التحقق من صحة النتائج النظرية عبر تجارب عددية، مما أبرز تأثير السياسات الخدمية وسلوك إعادة المحاولة وآليات الإصلاح على أداء وموثوقية النظام.

Abstract

This thesis addresses advanced models of retrial queuing systems, integrating several realistic operational features found in communication systems, cloud computing, and energy-constrained data centers. Two main models are proposed and analyzed in this work.

In the first part, a single-server queuing system with retrial is studied, incorporating working vacations, setup time, automatic power-off to conserve energy, and perfect repair mechanisms. Customers who find the server unavailable join the orbit and attempt service again after a random delay. When a new customer arrives during server idleness, the system enters a setup phase, which may fail and require repair before service can resume. The system dynamics are analyzed using the supplementary variable technique. Performance measures, reliability analysis, and an optimization study are provided, supported by numerical experiments using Python.

In the second part, the previous model is extended by introducing negative customers and a second distinct repair mechanism. Negative customers cause immediate server failure upon arrival, leading to a specific repair process. This extended model captures more realistic failure scenarios. Additionally, the Particle Swarm Optimization (PSO) algorithm is applied to enhance system performance, along with optimal strategy analysis to determine the best operational configurations. The theoretical results are validated through numerical experiments, highlighting the impact of service policies,orbiting retrial behavior, and repair mechanisms on system performance and reliability.

Résumé

Cette thèse traite des modèles avancés de systèmes de files d'attente avec rappels, en intégrant plusieurs caractéristiques opérationnelles réalistes observées dans les systèmes de communication, l'informatique en nuage et les centres de données contraints en énergie. Deux modèles principaux sont proposés et analysés dans ce travail.

Dans la première partie, un système de file d'attente avec rappel à serveur unique est étudié, intégrant des vacances actives, un temps de configuration, une mise hors tension automatique pour économiser l'énergie, et un mécanisme de réparation parfaite. Les clients qui ne trouvent pas le serveur disponible rejoignent une orbite et tentent d'obtenir un service après un certain temps aléatoire. Lorsqu'un nouveau client arrive pendant l'inactivité du serveur, le système entame une phase de mise en route, qui peut échouer et nécessiter une réparation avant de reprendre le service. Les dynamiques du système sont analysées à l'aide de la technique des variables supplémentaires. Des mesures de performance, une analyse de fiabilité et une étude d'optimisation sont fournies, appuyées par des simulations numériques en Python.

Dans la deuxième partie, le modèle précédent est étendu par l'introduction de clients négatifs et d'un deuxième mécanisme de réparation distinct. Ces clients provoquent une panne immédiate du serveur à leur arrivée, nécessitant un processus de réparation spécifique. Ce modèle approfondi reflète des scénarios de défaillance plus réalistes. En outre, l'algorithme d'optimisation par essaim de particules (PSO) est appliqué pour améliorer la performance du système, accompagné d'une analyse stratégique optimale visant à déterminer les meilleures configurations opérationnelles. Les résultats théoriques sont validés par des expérimentations numériques, mettant en évidence l'impact des politiques de service, du comportement des rappels de l'orbite et des mécanismes de réparation sur la performance et la fiabilité du système.

Contents

List of Figures

List of Tables

Introduction

Queueing theory provides a fundamental framework for modeling systems in which demands for service compete for limited resources. Originating from the pioneering work of Agner Krarup Erlang on telephone traffic congestion in the early 20th century [29, 30], the theory has evolved into a rich mathematical discipline with profound implications across diverse domains such as telecommunications, computer networks, production systems, transportation logistics, manufacturing, and healthcare [50, 73, 19, 33]. Among its many applications, queueing theory constitutes a cornerstone of applied probability, enabling the analysis and optimization of stochastic service systems under uncertainty.

Among the various extensions of classical queueing models, *retrial queueing systems* have received significant attention due to their ability to capture realistic customer behaviors that traditional waiting-line models cannot adequately represent. In such systems, customers who find all servers occupied upon arrival do not join a conventional queue nor abandon the system permanently; instead, they enter a so-called *retrial orbit* and attempt service again after random time intervals governed by predefined retrial policies [31, 53, 6]. This mechanism naturally arises in numerous practical contexts such as repeated connection attempts in mobile and wireless networks, job resubmissions in cloud computing environments, or retry requests in distributed service architectures where blocked requests persistently seek access rather than being lost or queued [1, 49, 72, 84, 95]. Figure 1 illustrates the general structure of a retrial queue, showing the interaction between the main service facility and the orbit where customers wait and retry for access.

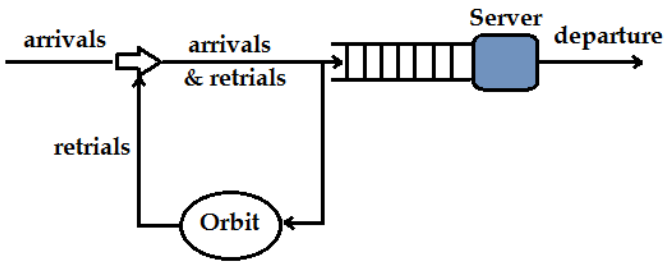

Figure 1: General structure of a retrial queue

Foundational contributions to retrial queueing theory were made by Yang and Templeton, Falin and Templeton [32], and later extended by Artalejo and Gómez-Corral [36], as well as Arrar et al. [2, 3, 4]. The study of retrial queues has expanded remarkably in recent decades, driven by their relevance to emerging technologies. Modern research has thus incorporated new operational characteristics into retrial models, such as server vacations, setup times, breakdown–repair mechanisms, and energy-saving policies [7, 26, 55].

Of particular practical relevance are queueing systems incorporating *server vacations*, which enable dynamic resource management by allowing servers to temporarily suspend or reduce service during idle periods. First formalized by Levy and Yechiali [59], such models have become indispensable for analyzing production lines, service centers, inventory control, and other stochastic environments. In classical vacation models, service halts entirely during server breaks; however, numerous extensions have been proposed to account for *vacation interruptions*, with seminal work by Keilson and Servi [44] and later contributions by Li and Tian [61]. Takagi [83] further enriched the literature by analyzing single-server queues under Bernoulli vacation schedules.

To enhance system efficiency without fully suspending service, the concept of *working vacations* was introduced, wherein servers operate at a reduced rate during vacation periods. This paradigm was pioneered by Servi and Finn [79] in the context of an M/M/1 queue with exponentially distributed working vacations. Subsequent developments were comprehensively surveyed by Tian et al. [85] and Chandrasekaran et al. [17]. The integration of retrial behavior with working vacations was first examined by Do, followed by Banik et al. [13], who proposed a general framework for such systems. Arivudainambi et al. [5] analyzed single-server retrial queues under working vacation regimes, while Zhang and Hou [97] incorporated *vacation interruption* into M/G/1 models. More recently, Gupta and Kumar [37] extended the framework to include server breakdowns and repairs within retrial–working vacation systems.

Furthermore, the incorporation of *setup times* has emerged as a critical feature, particularly in energy-efficient computing and green communications. Setup periods allow servers to deactivate during idle intervals and require a preparatory phase before resuming full-capacity service, thereby enabling significant power savings. Phung-Duc [75, 76] was among the first to integrate setup times into retrial queueing models. Gupta and Kumar [38] later analyzed a retrial system with feedback, setup times, working vacations, and perfect repair mechanisms. Additional insights can be found in the works of Manoharan and Jeeva [70, 41].

To emphasize the modern relevance of these models, this thesis draws motivation from recent technological contexts such as artificial intelligence inference serving under limited GPU resources, serverless machine learning functions, bioinformatics workflows, and AI-driven network slicing in 5G/6G systems. These examples reveal how retrial phenomena emerge naturally

in data-driven and resource-constrained environments, underscoring the need for advanced analytical models that jointly account for performance, reliability, and energy efficiency.

Building upon these motivations, the thesis investigates advanced retrial queueing systems with energy-saving mechanisms, setup and repair processes, and reliability considerations. The remainder of this thesis is organized into three main chapters as follows:

Chapter 1 introduces the fundamentals of retrial queueing systems and surveys key developments in the literature, with a focus on models involving server vacations, setup procedures, interruptions, breakdown–repair mechanisms, and optimization frameworks. The chapter also reviews analytical tools such as the supplementary variable technique and optimization algorithms like Particle Swarm Optimization (PSO).

Chapter 2 develops a single-server retrial queueing system with working vacations, setup time, automatic power-off, setup failure, and perfect repair. The system's steady-state behavior is analyzed using the supplementary variable method, and performance measures are derived to assess server availability and energy efficiency. A cost-based optimization problem is then formulated to determine optimal configurations, supported by numerical validation.

Chapter 3 extends the previous model by introducing *negative customers*—entities that can cause server breakdowns during regular service periods—and a second repair mechanism distinct from setup-failure repair. The system's steady-state distribution is obtained analytically, and reliability as well as optimization analyses are performed using PSO. This chapter further explores strategic policies balancing energy consumption, reliability, and service performance under dual repair conditions.

Finally, the thesis concludes with a synthesis of the main results and outlines directions for future research in the modeling and optimization of complex retrial queueing systems.

Chapter 1

Overview of Retrial Queueing Theory

Since the pioneering works of the 1950s, retrial queueing models have become indispensable tools for stochastic modeling in telecommunications, computer networks, and everyday service systems. In modern complex systems such as cloud computing infrastructures, edge networks, and IoT-based service platforms retrial behavior is often intertwined with diverse operational phenomena, giving rise to a rich taxonomy of model variants. Below, we systematically categorize and analyze the most prominent types, incorporating recent theoretical and applied advancements.

1.1 Retrial Queues with Setup Time

In many real-world service systems, the server does not immediately resume operation after becoming idle. Instead, it undergoes a preparatory phase, commonly referred to as **setup time**, before service can restart. Levy and Kleinrock [60] first formally incorporated this mechanism into the classical M/M/1 queue. In their model, the server is deactivated during idle periods and must complete a setup phase upon reactivation, forcing arriving customers either to wait or to retry.

Because of its practical relevance, the setup-time framework has been widely studied. Bischof [15] analyzed M/G/1 queues with setup times and vacations under several service disciplines and provided detailed performance measures. Later, Gandhi, Doroudi, Harchol-Balter, and Scheller-Wolf [34] developed an exact analysis of the M/M/k/setup model, emphasizing its applicability to energy-efficient server management.

A significant theoretical extension was proposed by Phung-Duc [76], who generalized both setup and service times to arbitrary distributions and derived the stationary queue-length distribution using matrix-analytic methods. More recently, Chang and Wang [18] extended the framework to unreliable M/M/1/1 retrial queues with setup time, allowing server breakdowns

during the setup phase—an important feature for modeling fault-tolerant systems. **Recent Extension (2023–2024)**: In cloud and edge computing, setup times are now modeled as *cold-start delays* in serverless architectures. These delays impact Quality of Service (QoS) and motivate adaptive strategies to mitigate latency .

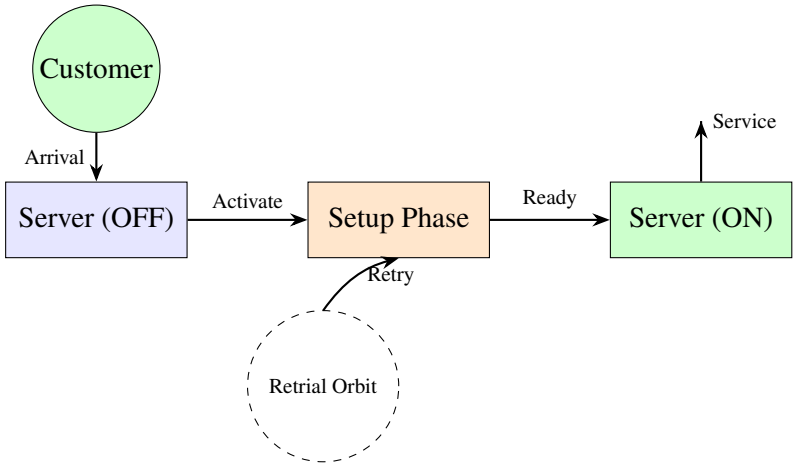

Figure 1.1: Retrial queue with setup time: the first customer triggers the setup and is served after preparation. Other arrivals may join the retrial orbit until the server is ready.

1.2 Retrial Queues with Negative Customers

Queueing systems incorporating **negative customers**, commonly referred to as G-networks, provide a powerful analytical framework for modeling cancellation mechanisms, congestion control, and disruptive events in communication and manufacturing systems. In these systems, positive customers request service in the usual manner, whereas negative customers—arriving only during active service periods—remove positive customers from the system without joining the queue.

G-networks introduced a novel class of queueing models in which special signals eliminate customers, leading to tractable product-form solutions and stimulating extensive research in both continuous- and discrete-time settings.

In the context of retrial queues, Wang and Zhang [90] investigated a discrete-time retrial system with negative arrivals and an unreliable server, highlighting the combined impact of cancellations and breakdowns on system performance. Subsequently, Krishnakumar, Vijayakumar, and Sophia [52] examined retrial models with multiple service phases and vacation mechanisms. Wu and Lian [92] incorporated priority customers and disaster effects, while Gao and Wang [35] analyzed performance and reliability characteristics of an M/G/1-G retrial queue using soft computing techniques. Further extensions were proposed by Zhang and Liu [99],

who studied non-exhaustive random vacations, and by Peng, Liu, and Wu [74], who introduced preemptive resume priority under an N-policy framework.

Modern Application: In cybersecurity systems, the concept of negative customers has been used to model malicious activities such as malware infections or denial-of-service attacks. For instance, negative arrivals can represent malicious packets that remove legitimate requests from the system, thereby disrupting normal operations. Recent studies have applied retrial queueing models with negative arrivals to capture the reliability and cost impact of malware-infected systems, showing how such attacks can degrade performance and increase downtime .

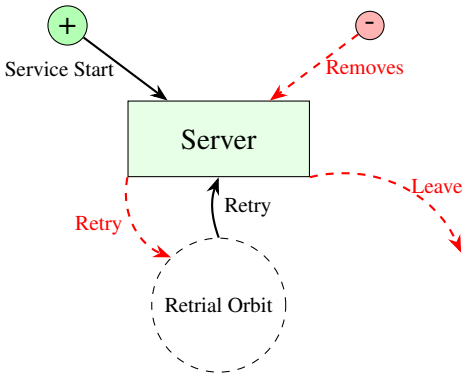

Figure 1.2: Retrial queue with negative customers: positive customers enter service, while negative customers may remove them. Removed customers either retry via the orbit or leave the system.

1.3 Retrial Queues with Server Vacations

Server vacations model periods when the server performs secondary tasks (e.g., maintenance, diagnostics, or auxiliary processing) when no customers are present. Vacation policies are classified into several types, each with distinct operational rules and performance implications.

1.3.1 Single Vacation

Under the single vacation policy, the server takes exactly one vacation after each busy period. Upon returning, if the system is empty, the server remains idle instead of initiating another vacation. This mechanism is frequently adopted in production and maintenance systems where downtime is scheduled once per operating cycle.

Lee et al [56] analyzed the $M^X/G/1$ queue with N-policy and server vacations, incorporating batch arrivals into the single vacation framework. Choudhury [21] extended classical vacation models by studying batch arrival queues under a single vacation policy. Madan and Abu Al-Rub

[67] investigated a two-phase optional $M^X/G/1$ system with a single vacation, while Madan and Al-Rawwash [68] examined feedback mechanisms combined with optional server vacations.

Ke [42] further enriched the literature by analyzing an $M^X/G/1$ system with startup, standby spare, and breakdown features, thereby integrating reliability aspects into the single vacation setting.

Recent Development: Adaptive mechanisms in cloud resource allocation have been proposed to balance energy efficiency and Quality of Service (QoS). Instead of keeping all servers active, some resources can temporarily enter an idle or sleep statesimilar in spirit to vacation models in queueing theory. Recent approaches, such as the Look-ahead Energy Efficient VM Allocation (LAA), use workload prediction techniques (e.g., Holt-Winters forecasting) to dynamically adjust the number of active servers. This adaptive strategy reduces power consumption significantly while ensuring that QoS requirements are still met .

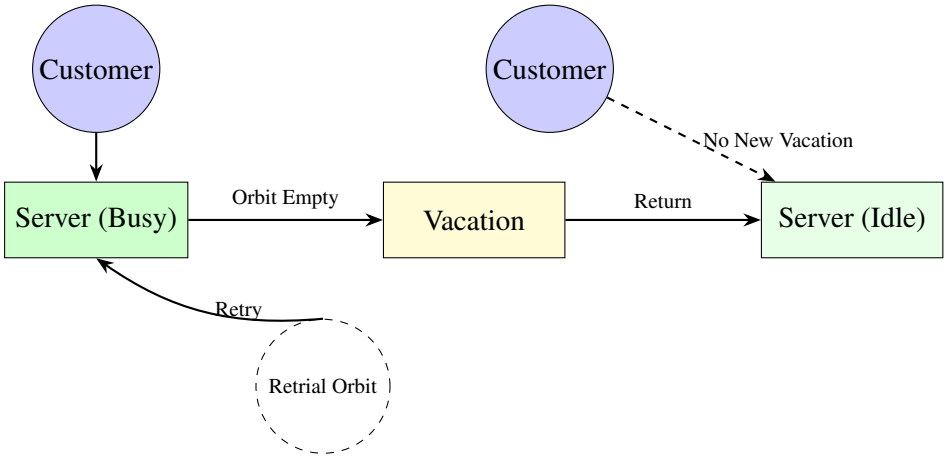

Figure 1.3: Single vacation policy: after completing a busy period, the server takes exactly one vacation. If the orbit remains empty, it returns to the idle state without further vacations

1.3.2 Bernoulli Vacation

Under the Bernoulli vacation policy, upon service completion and when the orbit is empty, the server takes a vacation with probability p or remains active with probability $q = 1 - p$. This probabilistic control mechanism has been widely adopted in communication and manufacturing systems to regulate workload and improve operational flexibility.

Keilson and Servi [45] first introduced the foundational Bernoulli vacation model, establishing a stochastic framework in which vacation decisions are made probabilistically after service completion. Madan [65] extended related queueing structures by incorporating a second optional service phase, while Madan [66] further examined systems with random breakdowns.

Atencia and Moreno [9] investigated retrial queues with general retrial times in discrete time, broadening the applicability of Bernoulli-type mechanisms. Wang and Li [89] analyzed a repairable M/G/1 retrial queue with Bernoulli vacations and two-phase service, explicitly integrating reliability features. Ke and Chang [43] proposed modified vacation policies for batch arrival systems with breakdowns and multiple vacations. Choudhury and Ke [23] provided a detailed study of unreliable batch arrival queues operating under a Bernoulli vacation schedule combined with multiple vacation behavior.

More recently, Liu, Zeng, Cui, and Song [64] explored reinforcement learning-based charging scheduling for electric vehicles, where charging units probabilistically switch to low-power modes. Although developed in the context of smart energy systems, their probabilistic control mechanism reflects the same structural principle underlying Bernoulli vacation policies.

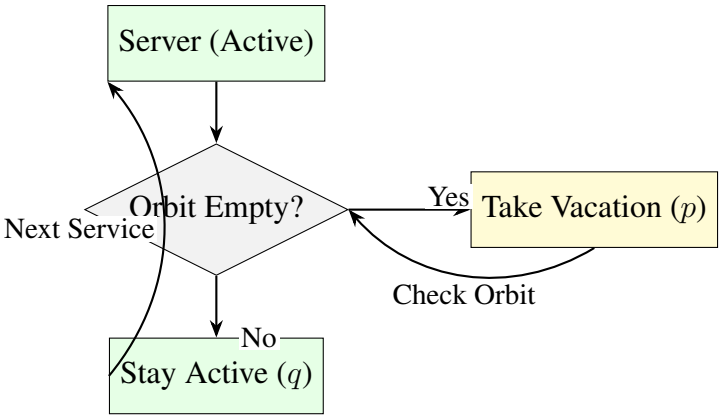

Figure 1.4: Bernoulli vacation schedule: After service, the server takes a vacation with probability p or stays active with probability $q = 1 - p$.

1.3.3 Multiple Vacations

The server repeatedly takes vacations until it finds at least one customer upon return. This mechanism is particularly relevant in systems requiring periodic diagnostics or segmented maintenance, such as telecom switches and database servers. The foundational work in this area was established by Baba [11], who analyzed the M^X/G/1 queue with vacation times, and subsequently extended by Lee and Srinivasan [57] through their investigation of optimal control policies. Krishna Reddy et al. [51] further enriched the model by incorporating N-policy with setup times, thereby addressing more complex operational scenarios.

The discrete-time generalization of these models was later developed by Tian and Zhang [86], while finite-buffer and bulk-service variants were rigorously analyzed by Sikdar and Gupta [81] and Arumuganathan and Jeyakumar [8], respectively. These theoretical advancements have

laid the groundwork for contemporary applications in edge computing environments, where the multiple vacations model effectively characterizes micro-sleep cycles for energy conservation. In this context, Kim et al. [48] demonstrated that optimized vacation sequencing can achieve energy consumption reductions of up to 30% in 5G base stations without compromising latency service level agreements.

1.3.4 Working Vacation

Unlike traditional vacations, during a **working vacation**, the server operates at a reduced rate instead of halting completely. This mechanism effectively models scenarios involving preventive maintenance during low-load periods or background processing operations in web servers. The foundational analysis of working vacation policies was established by Servi and Finn [80] for M/M/1 queueing systems. Subsequently, Liu et al. [63] developed important stochastic decomposition properties for these models, while Baba [12] extended the framework to GI/M/1 queues with comprehensive sojourn time analysis. Further generalizations were provided by Wu and Takagi [93], who analyzed M/G/1 systems with general service distributions, and Zhang and Hou [98], who derived queue-length distributions using matrix-analytic methods.

State-of-the-Art (2024): Recent advances in sustainable AI inference have highlighted dynamic precision and adaptive execution as effective techniques for reducing energy consumption. Barad et al. [14] demonstrated that early-exit strategies can significantly reduce inference costs while maintaining acceptable accuracy levels. This emerging trend naturally extends to the concept of "hybrid working vacations," wherein servers alternate between full-rate and reduced-rate operational modes to achieve an optimal balance between computational precision and energy efficiency.

1.3.5 Retrial Queues with Unreliable Server

Real-world servers are inherently susceptible to failures, which may occur either **actively** (during service delivery) or **passively** (during idle or setup periods). These failure modes significantly compromise system reliability and necessitate explicit modeling within queueing-theoretic frameworks. The seminal contribution in this domain was made by Yang and Li [94], who pioneered the analysis of M/G/1 retrial queues with *starting failures*, establishing fundamental stability conditions and stochastic decomposition laws. Wang et al. [88] subsequently integrated comprehensive reliability metrics including availability and failure rates into the analysis. The incorporation of server failures with vacation policies and two-phase service structures was later investigated by Wang and Li [89] and Ke and Chang [43]. Furthermore,

Wang and Zhou [91] introduced admission control mechanisms for batch-arrival retrial systems, while Choudhury and Deka [22] developed models incorporating delayed repair processes. The optimization of multi-server retrial systems under economic constraints using matrix-analytic techniques was addressed by Yuh et al. [96].

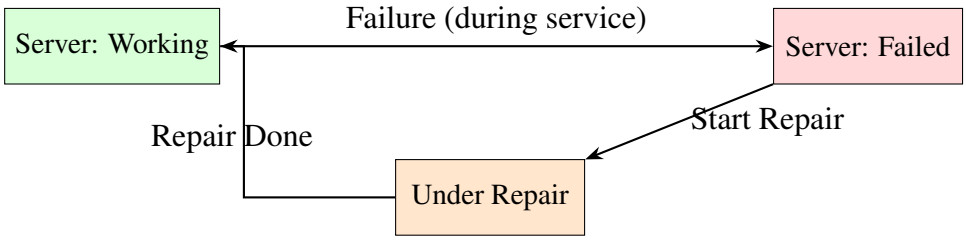

Figure 1.5: Unreliable server with repair: The server may fail during service, undergo repair, and return to the working state. Customers retry during downtime.

1.4 Supplementary Variable Method

Due to the memoryless property inherent to exponential inter-arrival and service time distributions in M/M/1 queues, the system's occupancy level can be characterized as a Markov process, denoted by $\mathcal{Q}(s)$. However, this Markovian property is lost when arrival or service intervals follow non-exponential distributions. To address this fundamental limitation, Cox [25] introduced the supplementary variable method, thereby enabling rigorous analysis of non-Markovian stochastic systems. Keilson and Kooharian [46] noted that this methodological approach was subsequently adopted by Henderson [40], Choi et al. [20], and Medhi [71]. By introducing an auxiliary state descriptor representing residual service times, differential equations analogous to those governing Markovian systems can be systematically derived.

For illustrative purposes, consider queueing systems with general service time distributions. In such cases, the process $\mathcal{Q}(s)$ becomes non-Markovian. To restore analytical tractability, we define $\tau(s)$ as the residual service duration of the customer in service at epoch s. The augmented process $(\mathcal{Q}(s), \tau(s))$ then constitutes a Markovian system amenable to standard analysis techniques. This approach offers significant computational advantages over alternative methodologies such as embedded Markov chains [58]. The supplementary variable technique was first applied to M/G/1 retrial queues by Keilson et al. [47], who analyzed the joint steady-state distribution of server status and orbit population. Li and Yang [62] subsequently extended this framework to incorporate server vacations and finite source populations.

1.5 Optimization

Optimization constitutes an essential component in enhancing queueing system performance, whether through minimizing waiting times, controlling queue lengths, or maximizing throughput. In complex model configurations incorporating retrials, vacations, and server failures, performance metrics typically emerge as nonlinear functions of multiple system parameters. Analytical optimization often proves infeasible in such contexts, necessitating the application of numerical methods to identify optimal configurations under operational constraints.

1.5.1 Quadratic Interpolation Search

This derivative-free optimization method locates extrema of unimodal functions by fitting a quadratic polynomial through three evaluation points. Given three distinct abscissas $\chi < \psi < \phi$ and their corresponding function evaluations $\mathcal{J}(\chi), \mathcal{J}(\psi), \mathcal{J}(\phi)$, the extremum θ^* of the interpolating parabola is approximated by:

$$\theta^* \approx \frac{1}{2}\left[\frac{\mathcal{J}(\chi)(\psi^2-\phi^2)+\mathcal{J}(\psi)(\phi^2-\chi^2)+\mathcal{J}(\phi)(\chi^2-\psi^2)}{\mathcal{J}(\chi)(\psi-\phi)+\mathcal{J}(\psi)(\phi-\chi)+\mathcal{J}(\phi)(\chi-\psi)}\right]$$

The iteration proceeds by replacing the worst-performing point among $\{\chi, \psi, \phi\}$ with the newly computed θ^*, continuing until convergence criteria are satisfied.

1.5.2 Particle Swarm Optimization (PSO)

PSO represents a bio-inspired metaheuristic that emulates swarm intelligence observed in natural systems. This optimization paradigm is particularly favored for its algorithmic simplicity, rapid convergence characteristics, and robust capability to handle non-convex, high-dimensional optimization landscapes. Within queueing systems research, PSO has been successfully deployed to optimize cost functions, delay metrics, and overall system efficiency.

The algorithm initializes a swarm comprising M agents (particles), each representing a candidate solution in the parameter space. Each particle α is characterized by a position vector $\boldsymbol{\xi}_\alpha^{[n]}$ and velocity vector $\boldsymbol{\eta}_\alpha^{[n]}$ at iteration n. Particles iteratively update their trajectories by incorporating both personal memory (the best position previously visited, $\boldsymbol{\rho}_\alpha$) and collective memory (the global best position discovered by the entire swarm, $\boldsymbol{\Gamma}$).

Notable applications of PSO in retrial queueing systems include the work of Upadhyaya [87], who minimized operational costs in discrete-time retrial queues with startup failures.

Algorithm 1: Particle Swarm Optimization (PSO)

Input: Cost function $\mathcal{J}(\cdot)$, swarm size M, dimension D, max iterations $N_{\max}$, parameters ζ, κ, λ
Output: Global minimizer estimate $\boldsymbol{\Gamma}$
Initialize positions $\boldsymbol{\xi}_\alpha^{[0]}$ and velocities $\boldsymbol{\eta}_\alpha^{[0]}$ for all particles $\alpha = 1, \ldots, M$;
Set personal bests $\boldsymbol{\rho}_\alpha = \boldsymbol{\xi}_\alpha^{[0]}$, compute $\mathcal{J}(\boldsymbol{\rho}_\alpha)$;
Set global best $\boldsymbol{\Gamma} = \arg\min_\alpha \mathcal{J}(\boldsymbol{\rho}_\alpha)$;
for $n = 0$ **to** $N_{\max} - 1$ **do**
  for *each particle* $\alpha = 1, \ldots, M$ **do**
    Draw $\epsilon_1, \epsilon_2 \sim \mathcal{U}(0, 1)$;
    Update velocity:
    $\boldsymbol{\eta}_\alpha^{[n+1]} = \zeta \boldsymbol{\eta}_\alpha^{[n]} + \kappa \epsilon_1 (\boldsymbol{\rho}_\alpha - \boldsymbol{\xi}_\alpha^{[n]}) + \lambda \epsilon_2 (\boldsymbol{\Gamma} - \boldsymbol{\xi}_\alpha^{[n]})$;
    Update position:
    $\boldsymbol{\xi}_\alpha^{[n+1]} = \boldsymbol{\xi}_\alpha^{[n]} + \boldsymbol{\eta}_\alpha^{[n+1]}$;
    Evaluate $\mathcal{J}(\boldsymbol{\xi}_\alpha^{[n+1]})$;
    if $\mathcal{J}(\boldsymbol{\xi}_\alpha^{[n+1]}) < \mathcal{J}(\boldsymbol{\rho}_\alpha)$ **then**
      $\boldsymbol{\rho}_\alpha = \boldsymbol{\xi}_\alpha^{[n+1]}$;
    if $\mathcal{J}(\boldsymbol{\rho}_\alpha) < \mathcal{J}(\boldsymbol{\Gamma})$ **then**
      $\boldsymbol{\Gamma} = \boldsymbol{\rho}_\alpha$;
return $\boldsymbol{\Gamma}$

Zhang et al. [100] optimized state-dependent service rates, while Malik et al. [69] demonstrated that PSO outperforms genetic algorithms in terms of convergence speed and solution accuracy. More recently, Harini and Indhira [39] applied PSO to hospital management systems, validating its robustness in dynamic operational environments.

1.6 Some Motivating Examples

1.6.1 AI Inference Serving with Limited GPU Resources

Contemporary AI deployments involving large language models and deep learning workloads are typically hosted on GPU clusters. However, the substantial cost and limited availability of GPU resources frequently necessitate oversubscription strategies. When all GPUs are occupied, incoming inference requests may be blocked or subjected to significant delays. Retrial queueing models provide an elegant alternative framework wherein blocked or rejected requests reattempt service after a specified delay period. Rather than maintaining extensive waiting queues that consume memory resources and risk timeout failures, rejected requests enter a *virtual orbit* and retry following a random or adaptively determined backoff interval. This modeling

approach has been extensively investigated in the literature, ranging from stability analysis of GI/G/c/K retrial queues [10] to multi-class systems with sophisticated retrial policies [28], and more recently in cloud computing contexts [24]. Such mechanisms effectively balance system load, enhance resource utilization efficiency, and mitigate tail latency under bursty traffic conditions.

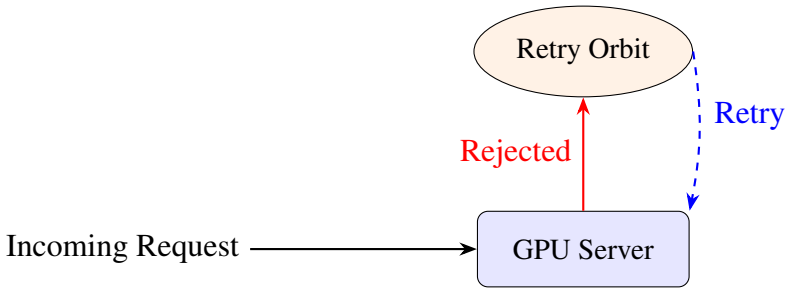

Figure 1.6: Simplified retrial queueing model for GPU inference. Rejected requests enter a virtual orbit and retry after a delay.

1.6.2 Serverless AI Functions

Serverless computing platforms such as AWS Lambda are increasingly adopted for machine learning workloads due to their inherent elasticity and cost efficiency. These systems exhibit specific performance characteristics including strict concurrency limits that may trigger request throttling under heavy load, and cold start phenomena that introduce substantial initialization delays, particularly for large ML models [16]. Although the serverless computing literature has not extensively employed retrial queueing formalisms, the analogy provides a rigorous analytical lens. By mapping concurrency limits to server capacity, cold starts to setup times, and retry policies to retrial processes, retrial queueing theory offers powerful predictive capabilities for latency, rejection probabilities, and resource utilization in serverless ML pipelines. This conceptual connection illustrates how classical queueing models can be extended to capture emerging cloud-native architectural patterns.

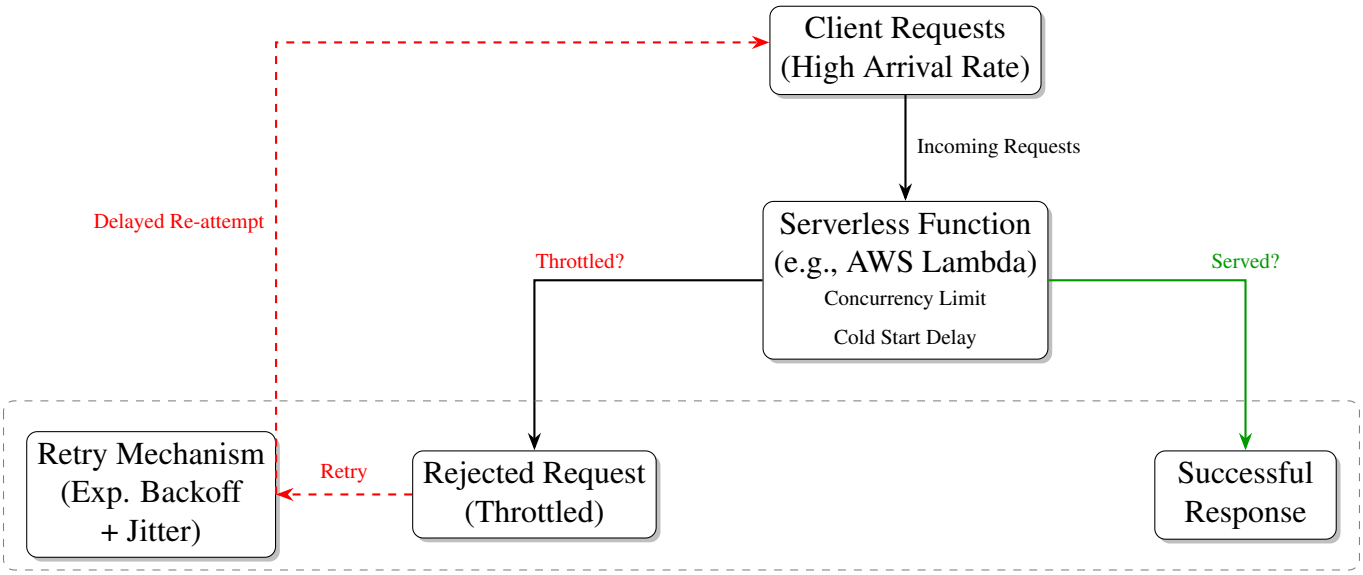

Figure 1.7: Performance dynamics of serverless AI functions with retry behavior under concurrency limits and cold starts.

1.6.3 Bioinformatics and Computational Biology Workflows

High-throughput genomic pipelines provide compelling examples where retrial queueing theory offers valuable analytical insights. In such environments, thousands of computational jobs are simultaneously submitted to high-performance computing clusters or cloud platforms, frequently exceeding available resources or encountering transient failures that necessitate job resubmission. Workflow management systems such as Nextflow are explicitly designed to address these challenges through built-in automatic retry mechanisms, ensuring resilient computation without manual intervention. This operational behavior closely parallels retrial queue dynamics, wherein jobs that do not receive immediate service re-enter an orbit and attempt execution again after a specified interval. Adopting this perspective, job resubmissions can be interpreted as customer retrials, scheduler queues as service nodes with finite capacity, and resource failures as server breakdowns. Such queueing-theoretic abstractions underscore the critical importance of retrial policies in ensuring robustness and reproducibility in large-scale genomic data processing [78, 82, 27].

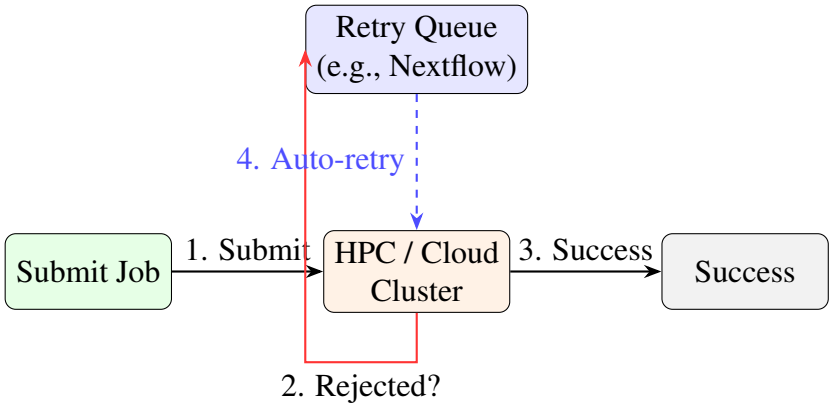

Figure 1.8: Rejected jobs are queued for automatic retry, enabling resilient execution in genomic pipelines.

1.6.4 AI-Driven Network Slicing in 5G/6G for Smart Cities

Fifth-generation (5G) mobile networks employ *network slicing* technology, enabling the creation of multiple virtual sub-networks atop shared physical infrastructure, each optimized for distinct service requirements. For instance, autonomous vehicular applications require ultra-reliable low-latency communication slices, whereas video streaming services prioritize high bandwidth allocation. These slices operate independently while coexisting on common resources, ensuring heterogeneous applications receive appropriate quality of service guarantees. When network capacity becomes temporarily saturated, slice requests are not discarded; instead, they are redirected into an *orbit* where they retry access after a randomized interval. This mechanism can be naturally modeled using retrial queueing theory, which has been extensively applied to telecommunication systems for analyzing congestion dynamics, delay characteristics, and resource allocation optimization [101].

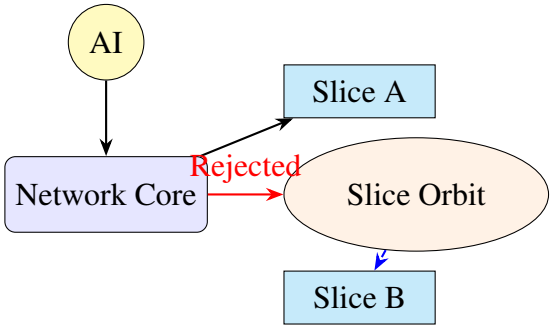

Figure 1.9: Network slicing with retrial: AI-managed system retries rejected slice requests via orbit.

Chapter 2

Unreliable M/G/1 Queue with General Retrial Time, Working Vacation and Setup Time

In this chapter, we present a comprehensive analysis of an **M/G/1 retrial queue** with a general retrial time distribution, incorporating **setup times** and **working vacations**. The server may enter a working vacation state serving customers at a reduced rate or undergo a setup phase before resuming normal operations. Crucially, the server is susceptible to random breakdowns during both service and setup periods, with repair times governed by general probabilistic distributions. We derive the steady-state distribution of the system, compute key performance and reliability metrics, and conduct sensitivity and cost optimization analyses to evaluate the operational and economic impact of system parameters.

2.1 Justifications of the Model and Practical Relevance

Consider a stylized manufacturing system designed for paper recycling, comprising three key components: a recycling machine (server), a foreman (primary server), and an assistant worker (auxiliary server). Waste paper batches, conceptualized as *customers*, arrive stochastically to be processed into finished goods. The foreman operates the machine at full capacity whenever waste paper is available. However, in the event of supply disruptions such as transportation delays causing customer unavailability the foreman may enter a *vacation state*, temporarily suspending direct involvement in production.

During the foreman's vacation, if new waste paper arrives, the assistant worker assumes

operational control of the machine. In this phase, termed a *working vacation*, production continues, but at a reduced service rate, reflecting the worker's limited capacity compared to the foreman. Upon completion of a production batch, the worker triggers a *vacation interruption* mechanism, recalling the foreman to resume operations at the standard, higher service rate.

Alternatively, if the foreman's vacation concludes naturally (i.e., without interruption), he returns to the system and assesses its state. If waste paper is present upon his return, he immediately restarts the machine and resumes full-capacity production. If no customers are waiting, the foreman may initiate a *shutdown procedure* to conserve energy effectively deactivating the server.

Subsequently, when new waste paper arrives after such a shutdown, the system must undergo a *setup period* before production can recommence. This setup phase represents the time required to reconfigure and warm up the machine. If the setup is successfully completed, normal service resumes. However, if the setup fails modeled as a probabilistic event the machine enters a *breakdown state* and is sent for repair. During the repair period, no production occurs, and arriving customers may join a retrial orbit.

This scenario encapsulates a rich interplay of queueing phenomena: **server vacations, working vacations, vacation interruptions, setup times, and server breakdowns with repair** all embedded within a practical, energy-aware manufacturing context.

2.2 Model Description

In this study, we analyze a single-server retrial queueing system characterized by server unreliability, a single working vacation policy, and non-negligible setup times. The system dynamics are governed by the following assumptions:

- **Arrival and Retrial Mechanism:**
  Customers arrive according to a Poisson process with rate $\pi_1 > 0$. Upon arrival, if the server is idle, the customer enters service immediately. Otherwise, if the server is busy, on working vacation, undergoing setup, or under repair the customer joins a retrial orbit, adhering to a First-Come-First-Served (FCFS) discipline. While in orbit, each customer independently retries after a random time interval. The inter-retrial times are governed by a general probability distribution function $A_1(\xi)$, with corresponding Laplace-Stieltjes Transform (LST) denoted by $\mathcal{A}_1^*(\theta)$.

- **Normal Service Phase:**
  When the server is idle and a new or retrial customer arrives, normal service commences

immediately. The service time during this phase follows a general distribution $A_2(\xi)$, with LST $\mathcal{A}_2^*(\theta)$.

- **Working Vacation Policy:**
  If the orbit becomes empty, the server initiates a single working vacation, the duration of which is exponentially distributed with rate ω. During this phase, arriving customers are still served, albeit at a reduced service rate. Service times during working vacation follow a general distribution $A_3(\xi)$, with LST $\mathcal{A}_3^*(\theta)$. Upon completion of the working vacation, if no customers are present in the orbit, the server is deactivated to conserve energy.

- **Setup Phase and Server Activation:**
  When the server is in the off-state and a customer arrives, the server must undergo a setup procedure before resuming service. The setup time is modeled by a general distribution $A_5(\xi)$, with LST $\mathcal{A}_5^*(\theta)$. During the setup period, newly arriving customers are directed to the orbit.

- **Server Breakdown During Setup:**
  The setup process may fail with probability $\bar{\alpha} = 1 - \alpha$. In such an event, the server is sent for repair. The repair time follows an arbitrary distribution $A_4(\xi)$, with corresponding LST $\mathcal{A}_4^*(\theta)$. No service is rendered during the repair period.

2.3 Steady-State Analysis

Let $\mathscr{X}_1(t), \mathscr{X}_2(t), \mathscr{X}_3(t), \mathscr{X}_4(t), \mathscr{X}_5(t)$ be the elapsed time in retrial, time in regular service, working vacation time,repair time and setup time sequentially at time t.
Let supose that :
$A_1(0) = 0, A_1(\infty) = 1, A_2(0) = 0, A_2(\infty) = 1, A_3(0) = 0, A_3(\infty) = 1, A_5(0) = 0, A_5(\infty) = 1, A_4(0) = 0, A_4(\infty) = 1$ are continuous at $\xi = 0$. Consequently, we specify the hazard rate functions $\pi_2(\xi), \pi_3(\xi), \pi_4(\xi), \pi_5(\xi), \pi_6(\xi)$, for retrial, normal service, lower rate service, delayed repair and repair, respectively.

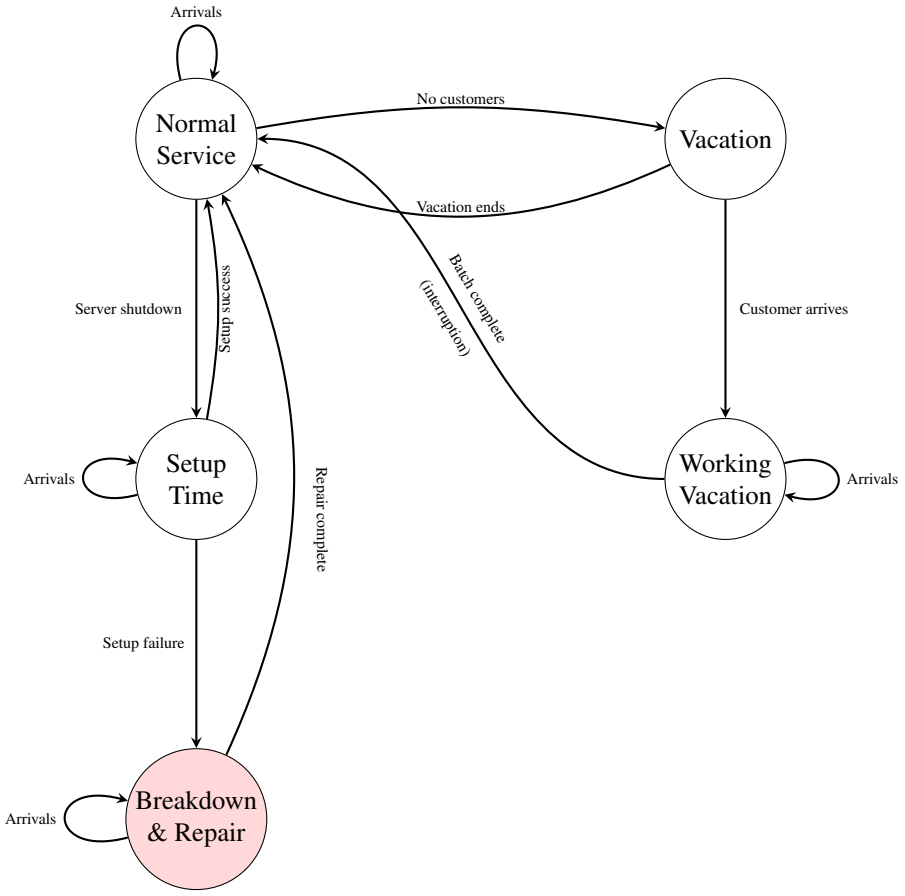

Figure 2.1: Schematic diagram of the proposed model.

$$\pi_2(\xi)d\xi = \frac{dA_1(\xi)}{1 - A_1(\xi)}$$

$$\pi_3(\xi)d\xi = \frac{dA_2(\xi)}{1 - A_2(\xi)}$$

$$\pi_4(\xi)d\xi = \frac{dA_3(\xi)}{1 - A_3(\xi)}$$

$$\pi_5(\xi)d\xi = \frac{dA_4(\xi)}{1 - A_4(\xi)}$$

$$\pi_6(\xi)d\xi = \frac{dA_5(\xi)}{1 - A_5(\xi)}$$

The state of the system at time t can be defined by the Markov process $\{N(t); t \geq 0\} = \{R(t), X(t), \mathscr{X}_1(t), \mathscr{X}_2(t), \mathscr{X}_3(t), \mathscr{X}_4(t), \mathscr{X}_5(t)t \geq 0\}$, where $X(t)$ denotes the number of cus-

tomers in the orbit at time t and

$$R(t) = \begin{cases} 0, & \text{server idle during regular service period} \\ 1, & \text{server idle during working vacation period} \\ 2, & \text{server busy providing regular service} \\ 3, & \text{server busy providing service at reduced rate} \\ 4, & \text{server undergoing repair} \\ 5, & \text{server in setup/initialization phase} \end{cases}$$

Depending on the server state $R(t)$, the elapsed time processes are defined as follows:

- If $R(t) = 0$ and $X(t) > 0$, then $\mathscr{X}_1(t)$ denotes the elapsed retrial time at time t.
- If $R(t) = 2$ and $X(t) > 0$ then $\mathscr{X}_2(t)$ denotes the elapsed service time during the regular busy period at time t.
- If $R(t) = 3$ and $X(t) \geq 0$, then $\mathscr{X}_3(t)$ denotes the elapsed working vacation time at time t.
- If $R(t) = 4$ and $X(t) \geq 1$, then $\mathscr{X}_4(t)$ denotes the elapsed repair time at time t.
- If $R(t) = 5$ and $X(t) \geq 1$, then $\mathscr{X}_5(t)$ denotes the elapsed setup time at time t.

2.3.1 Stability and ergodicity conditions

Let $\{t_n; n \in N\}$ be the sequence of epochs of either service completion times or vacation termination time. The sequence of random vectors $Z_n = \{R(t_n+), X(t_n+)\}$ form a Markov chain which is the embedded Markov chain for our queueing system. Its state space is $S = \{0, 1, 2, 3, 4 \text{ and } 5\} \times N$.

Theorem 2.3.1. *The embedded Markov chain $\{Z_n; n \in N\}$ is ergodic if and only if* $\rho = \pi_1 \left(\frac{1}{\pi_3} + \frac{1}{\pi_4} + \omega \mathcal{A}_3^{*\prime}(\omega) \right) < \mathcal{A}_1^*(\pi_1)$.

2.3.2 Governing Equations

For the Markov process $\{N(t); t \geq 0\}$, we define the probability

$$\Gamma_0(t) = \{R(t) = 0, X(t) = 0\} \Gamma_1(t) = \{R(t) = 1, X(t) = 0\}$$

and the probability densities

$$\Upsilon_{0,n}(\xi,t)d\xi = \{R(t)=0, X(t)=n, \xi \le \mathscr{X}_1(t) < \xi + d\xi\}, \xi \ge 0, n \ge 1$$

$$\Upsilon_{1,n}(\xi,t)d\xi = \{R(t)=2, X(t)=n, \xi \le \mathscr{X}_2(t) < \xi + d\xi\}, \xi \ge 0, n \ge 0$$

$$\Upsilon_{2,n}(\xi,t)d\xi = \{R(t)=3, X(t)=n, \xi \le \mathscr{X}_3(t) < \xi + d\xi\}, \xi \ge 0, n \ge 0$$

$$\Upsilon_{3,n}(\xi,t)d\xi = \{R(t)=4, X(t)=n, \xi \le \mathscr{X}_4(t) < \xi + d\xi\}, \xi \ge 0, n \ge 1$$

$$\Upsilon_{4,n}(\xi,t)d\xi = \{R(t)=5, X(t)=n, \xi \le \mathscr{X}_5(t) < \xi + d\xi\}, \xi \ge 0, n \ge 1$$

We assume that the stability condition is fulfilled in the sequel and so that we can set $\Gamma_0 = \lim_{t\to\infty} \Gamma_0(t)$ and $\Gamma_1 = \lim_{t\to\infty} \Gamma_1(t)$ limiting densities for $\xi > 0$ and $n \ge 0$

$$\Upsilon_{1,n}(\xi) = \lim_{t\to\infty} \Upsilon_{1,n}(\xi,t), \Upsilon_{0,n}(\xi) = \lim_{t\to\infty} \Upsilon_{0,n}(\xi,t) \text{ and } \Upsilon_{2,n}(\xi) = \lim_{t\to\infty} \Upsilon_{2,n}(\xi,t)$$

$$\text{and } \Upsilon_{3,n}(\xi) = \lim_{t\to\infty} \Upsilon_{3,n}(\xi,t) \text{ and } \Upsilon_{4,n}(\xi) = \lim_{t\to\infty} \Upsilon_{4,n}(\xi,t).$$

Under the stated assumptions and notations, our model is described by a system of differential-difference equations,

$$\pi_1 \Gamma_0 = \omega \Gamma_1 \tag{2.1}$$

$$(\pi_1 + \omega)\Gamma_1 = \int_0^\infty \Upsilon_{1,0}(\xi)\pi_3(\xi)d\xi + \int_0^\infty \Upsilon_{2,0}(\xi)\pi_4(\xi)d\xi \tag{2.2}$$

$$\frac{d}{d\xi}\Upsilon_{0,n}(\xi) + (\pi_1 + \pi_2(\xi))\Upsilon_{0,n}(\xi) = 0, \quad \xi > 0, \quad n \ge 1 \tag{2.3}$$

$$\frac{d}{d\xi}\Upsilon_{1,0}(\xi) + (\pi_1 + \pi_3(\xi))\,\Upsilon_{1,0}(\xi) = 0, \quad \xi > 0 \tag{2.4}$$

$$\frac{d}{d\xi}\Upsilon_{1,n}(\xi) + (\pi_1 + \pi_3(\xi))\,\Upsilon_{1,n}(\xi) = \pi_1 \Upsilon_{1,n-1}(\xi), \quad \xi > 0, \quad n \ge 1 \tag{2.5}$$

$$\frac{d}{d\xi}\Upsilon_{2,0}(\xi) + (\pi_1 + \omega + \pi_4(\xi))\,\Upsilon_{2,0}(\xi) = 0, \quad \xi > 0 \tag{2.6}$$

$$\frac{d}{d\xi}\Upsilon_{2,n}(\xi) + (\pi_1 + \omega + \pi_4(\xi))\,\Upsilon_{2,n}(\xi) = \pi_1 \Upsilon_{2,n-1}(\xi), \quad \xi > 0, \quad n \ge 1 \tag{2.7}$$

$$\frac{d}{d\xi}\Upsilon_{3,0}(\xi) + (\pi_1 + \pi_5(\xi))\Upsilon_{3,0}(\xi) = 0, \quad \xi > 0 \tag{2.8}$$

$$\frac{d}{d\xi}\Upsilon_{3,n}(\xi) + (\pi_1 + \pi_5(\xi))\Upsilon_{3,n}(\xi) = \pi_1 \Upsilon_{3,n-1}(\xi), \quad \xi > 0, \quad n \ge 1 \tag{2.9}$$

$$\frac{d}{d\xi}\Upsilon_{4,0}(\xi) + (\pi_1 + \pi_6(\xi))\Upsilon_{4,0}(\xi) = 0, \quad \xi > 0 \tag{2.10}$$

$$\frac{d}{d\xi}\Upsilon_{4,n}(\xi) + (\pi_1 + \pi_6(\xi))\Upsilon_{4,n}(\xi) = \pi_1\Upsilon_{4,n-1}(\xi), \quad \xi > 0, \quad n \geq 1 \tag{2.11}$$

The boundary conditions at $\xi = 0$ include $\Upsilon_{0,n}(0), \Upsilon_{1,0}(0), \Upsilon_{1,n}(0), \Upsilon_{2,0}(0), \Upsilon_{3,0}(0), \Upsilon_{3,n}(0), \Upsilon_{4,0}(0)$. The description of these terms at $\xi = 0$ are described as follows:

$$\Upsilon_{0,n}(0) = \int_0^\infty \Upsilon_{2,n}(\xi)\pi_4(\xi)d\xi + \int_0^\infty \Upsilon_{1,n}(\xi)\pi_3(\xi)d\xi, \quad n \geq 1 \tag{2.12}$$

$$\Upsilon_{1,0}(0) = \alpha\int_0^\infty \Upsilon_{4,0}(\xi)\pi_6(\xi)d\xi + \int_0^\infty P_1(\xi)\pi_2(\xi)d\xi + \omega\int_0^\infty \Upsilon_{2,0}(\xi)d\xi + \int_0^\infty \Upsilon_{3,0}(\xi)\pi_5(\xi)d\xi \tag{2.13}$$

$$\Upsilon_{1,n}(0) = \alpha\int_0^\infty \Upsilon_{4,n}(\xi)\pi_6(\xi)d\xi + \int_0^\infty P_{n+1}(\xi)\pi_2(\xi)d\xi + \omega\int_0^\infty \Upsilon_{2,n}(\xi)d\xi + \pi_1\int_0^\infty \Upsilon_{0,n}(\xi)d\xi + \int_0^\infty \Upsilon_{3,n}(\xi)\pi_5(\xi)d\xi, \quad n \geq 1 \tag{2.14}$$

$$\Upsilon_{2,0}(0) = \pi_1\Gamma_1 \tag{2.15}$$

$$\Upsilon_{2,n}(0) = 0, \quad n \geq 1 \tag{2.16}$$

$$\Upsilon_{4,0}(0) = \pi_1\Gamma_0 \tag{2.17}$$

$$\Upsilon_{4,n}(0) = 0, \quad n \geq 1 \tag{2.18}$$

$$\Upsilon_{3,0}(0) = \bar{\alpha}\int_0^\infty \Upsilon_{4,0}(\xi)\pi_6(\xi)d\xi \tag{2.19}$$

$$\Upsilon_{3,n}(0) = \bar{\alpha}\int_0^\infty \Upsilon_{4,n}(\xi)\pi_6(\xi)d\xi, \quad n \geq 1 \tag{2.20}$$

The normalization condition is given by

$$\Gamma_0 + \Gamma_1 + \sum_{n=1}^{\infty}\int_0^\infty \Upsilon_{0,n}(\xi)d\xi + \sum_{n=0}^{\infty}\int_0^\infty \Upsilon_{1,n}(\xi)d\xi + \sum_{n=0}^{\infty}\int_0^\infty \Upsilon_{2,n}(\xi)d\xi + \sum_{n=0}^{\infty}\int_0^\infty \Upsilon_{3,n}(\xi)d\xi + \sum_{n=0}^{\infty}\int_0^\infty \Upsilon_{4,n}(\xi)d\xi = 1$$

In order to solve the above set of equations we define the generating functions as

$$\Upsilon_0(\xi,\zeta) = \sum_{n=1}^{\infty}\zeta^n\Upsilon_{0,n}(\xi) \qquad \text{for } |\zeta| \leq 1 \text{ and } \xi > 0,$$

$$\Upsilon_0(0,\zeta) = \sum_{n=1}^{\infty} \zeta^n \Upsilon_{0,n}(0) \qquad \text{for } |\zeta| \leq 1,$$

$$\Upsilon_1(\xi,\zeta) = \sum_{n=0}^{\infty} \zeta^n \Upsilon_{1,n}(\xi) \qquad \text{for } |\zeta| \leq 1 \text{ and } \xi > 0,$$

$$\Upsilon_1(0,\zeta) = \sum_{n=0}^{\infty} \zeta^n \Upsilon_{1,n}(0),$$

$$\Upsilon_2(\xi,\zeta) = \sum_{n=0}^{\infty} \zeta^n \Upsilon_{2,n}(\xi) \qquad \text{for } |\zeta| \leq 1 \text{ and } \xi > 0,$$

$$\Upsilon_2(0,\zeta) = \sum_{n=0}^{\infty} \zeta^n \Upsilon_{2,n}(0),$$

$$\Upsilon_3(\xi,\zeta) = \sum_{n=0}^{\infty} \zeta^n \Upsilon_{3,n}(\xi) \qquad \text{for } |\zeta| \leq 1,$$

$$\Upsilon_3(0,\zeta) = \sum_{n=0}^{\infty} \zeta^n \Upsilon_{3,n}(0) \qquad \text{for } |\zeta| \leq 1,$$

$$\Upsilon_4(\xi,\zeta) = \sum_{n=0}^{\infty} \zeta^n \Upsilon_{4,n}(\xi) \qquad \text{for } |\zeta| \leq 1,$$

$$\Upsilon_4(0,\zeta) = \sum_{n=0}^{\infty} \zeta^n \Upsilon_{4,n}(0) \qquad \text{for } |\zeta| \leq 1.$$

Multiplying equations (2.2)-(2.11) by suitable powers of ζ and summing over n, we obtain the following set of partial differential equations :

$$\frac{\partial \Upsilon_0(\xi,\zeta)}{\partial \xi} + (\pi_1 + \pi_2(\xi))\Upsilon_0(\xi,\zeta) = 0, \tag{2.21}$$

$$\frac{\partial \Upsilon_1(\xi,\zeta)}{\partial \xi} + (\pi_1 - \pi_1\zeta + \pi_3(\xi))\,\Upsilon_1(\xi,\zeta) = 0, \tag{2.22}$$

$$\frac{\partial \Upsilon_2(\xi,\zeta)}{\partial \xi} + (\pi_1 - \pi_1\zeta + \omega + \pi_4(\xi))\,\Upsilon_2(\xi,\zeta) = 0, \tag{2.23}$$

$$\frac{\partial \Upsilon_3(\xi,\zeta)}{\partial \xi} + (\pi_1 - \pi_1\zeta + \pi_5(\xi))\Upsilon_3(\xi,\zeta) = 0, \tag{2.24}$$

$$\frac{\partial \Upsilon_4(\xi,\zeta)}{\partial \xi} + (\pi_1 - \pi_1\zeta + \pi_6(\xi))\Upsilon_4(\xi,\zeta) = 0. \tag{2.25}$$

Solving the above partial differential equations (2.21) to (2.25)

$$\Upsilon_0(\xi,\zeta) = \Upsilon_0(0,\zeta)[1 - A_1(\xi)]e^{-\pi_1\xi}, \tag{2.26}$$

$$\Upsilon_1(\xi,\zeta) = \Upsilon_1(0,\zeta)\left[1 - A_2(\xi)\right]e^{-I(\zeta)\xi}, \tag{2.27}$$

$$\Upsilon_2(\xi,\zeta) = \Upsilon_2(0,\zeta)\left[1 - A_3(\xi)\right]e^{-(I(\zeta)+\eta)\xi}, \tag{2.28}$$

$$\Upsilon_3(\xi,\zeta) = \Upsilon_3(0,\zeta)\left[1 - A_4(\xi)\right]e^{-I(\zeta)\cdot\xi}, \tag{2.29}$$

$$\Upsilon_4(\xi,\zeta) = \Upsilon_4(0,\zeta)\left[1 - A_5(\xi)\right]e^{-I(\zeta)\cdot\xi}. \tag{2.30}$$

where $I(\zeta) = \pi_1(1-\zeta)$
Multiplying equation (2.12) by suitable powers of ζ, summing over n from 1 to ∞ and using equations (2.1) and (2.2) after some algebraic manipulations, we get

$$\Upsilon_0(0,\zeta) = \int_0^\infty \Upsilon_2(\xi,\zeta)\pi_4(\xi)d\xi + \int_0^\infty \Upsilon_1(\xi,\zeta)\pi_3(\xi)d\xi - \left(\frac{\pi_1+\omega}{\omega}\right)\pi_1\Gamma_0 \tag{2.31}$$

Similarly, multiplying equations (2.13)-(2.20) by suitable powers of ζ, summing over n and after some algebraic manipulations, we obtain

$$\Upsilon_1(0,\zeta) = \frac{\alpha}{\zeta}\int_0^\infty \Upsilon_4(\xi,\zeta)\pi_6(\xi)d\xi + \pi_1\int_0^\infty \Upsilon_0(\xi,\zeta)d\xi + \frac{1}{\zeta}\int_0^{+\infty} \Upsilon_0(\xi,\zeta)\pi_2(\xi)d\xi$$
$$+ \omega\int_0^{+\infty} \Upsilon_2(\xi,\zeta)d\xi + \int_0^{+\infty} \Upsilon_3(\xi,\zeta)\pi_5(\xi)d\xi, \tag{2.32}$$

$$\Upsilon_2(0,\zeta) = \Upsilon_{2,0}(0), \tag{2.33}$$

$$\Upsilon_3(0,\zeta) = \bar{\alpha}\int_0^\infty \Upsilon_4(\xi,\zeta)\pi_6(\xi)d\xi, \tag{2.34}$$

$$\Upsilon_4(0,\zeta) = \Upsilon_{4,0}(0). \tag{2.35}$$

inserting equations 2.27-2.28 in 2.31 we get:

$$\Upsilon_0(0,\zeta) = \Upsilon_2(0,\zeta)\mathcal{A}_3^*(I(\zeta)+\eta) + \Upsilon_1(0,\zeta)G^*(I(\zeta)) - \left(\frac{\pi_1+\omega}{\omega}\right)\pi_1\Gamma_0 \tag{2.36}$$

in similary way inserting equations 2.26-2.30 in 2.32 we get :

$$\Upsilon_1(0,\zeta) = \frac{1}{\zeta}\Upsilon_0(0,\zeta)\mathcal{A}_1^*(\pi_1) + \alpha\mathcal{A}_5^*(I(\zeta))\Upsilon_4(0,\zeta) + V(\zeta)\Upsilon_2(0,\zeta)$$
$$+\Upsilon_0(0,\zeta)(1-\mathcal{A}_1^*(\pi_1)) + \Upsilon_3(0,\zeta)\mathcal{A}_4^*(I(\zeta)) \tag{2.37}$$

where $V(\zeta) = \frac{\omega}{I(\zeta)+\omega}(1 - \mathcal{A}_3^*(I(\zeta) + \omega))$ using equations 2.15 and 2.1 :

$$\Upsilon_2(0,\zeta) = \frac{\pi_1^2}{\omega}\Gamma_0 \tag{2.38}$$

using equation 2.13 :

$$\Upsilon_4(0,\zeta) = \pi_1\Gamma_0 \tag{2.39}$$

using equation 2.30 in 2.34 :

$$\Upsilon_3(0,\zeta) = \overline{\alpha}\mathcal{A}_5^*(I(\zeta))\Upsilon_4(0,\zeta) \tag{2.40}$$

using equation 2.39 in 2.40 :

$$\Upsilon_3(0,\zeta) = \overline{\alpha}\pi_1\Gamma_0\mathcal{A}_5^*(I(\zeta)) \tag{2.41}$$

using equations 2.38 -2.39 and 2.41 in 2.36 and 2.37 we get

$$\Upsilon_0(0,\zeta) = \frac{N_r(\zeta)}{D_r(\zeta)} \tag{2.42}$$

Where $N_r(\zeta) = \pi_1\Gamma_0\zeta\mathcal{A}_2^*(I(\zeta))\mathcal{A}_5^*(I(\zeta))[(\alpha-1)\omega\mathcal{A}_4^*(I(\zeta)) - \alpha\omega] + \omega$
$- \pi_1\mathcal{A}_2^*(I(\zeta))V(\zeta) - \pi_1(\mathcal{A}_3^*(I(\zeta)+\omega) - 1)$
$D_r(\zeta) = \omega\mathcal{A}_2^*(I(\zeta))[\zeta(1 - \mathcal{A}_1^*(\pi_1)) + \mathcal{A}_1^*(\pi_1)] - \zeta$

$$\begin{aligned}\Upsilon_1(0,\zeta) = \frac{-\pi_1\Gamma_0}{D_r(\zeta)}\Big\{&\omega\zeta\big(-\alpha\mathcal{A}_4^*(I(\zeta)) + \alpha + \mathcal{A}_4^*(I(\zeta))\big)\mathcal{A}_5^*(I(\zeta)) \\ &+ \pi_1\zeta V(\zeta) + \zeta\big(\mathcal{A}_1^*(\pi_1) - 1\big)\big(\omega - \pi_1\mathcal{A}_3^*(I(\zeta)+\omega) + \pi_1\big) \\ &- \big(\omega - \pi_1\mathcal{A}_3^*(I(\zeta)+\omega) + \pi_1\big)\mathcal{A}_1^*(\pi_1)\Big\}\end{aligned} \tag{2.43}$$

Substituting equations (2.38-2.43) in 2.26-2.30 .

2.3.3 Steady state results

Under the steady-state condition $\rho < \mathcal{A}_1^*(\pi_1)$, the probability generating functions (PGFs) for the number of customers in the orbit, conditioned on the server's operational state, are given as follows:

1. **Server idle:**

$$\Upsilon_0(\zeta) = \int_0^\infty \Upsilon_0(\xi, \zeta)\, d\xi = \frac{\Upsilon_0(0, \zeta)\,(1 - \mathcal{A}_1^*(\pi_1))}{\pi_1} \tag{2.44}$$

2. **Server busy (regular service):**

$$\Upsilon_1(\zeta) = \int_0^\infty \Upsilon_1(\xi, \zeta)\, d\xi = \frac{\Upsilon_1(0, \zeta)\,(1 - \mathcal{A}_2^*(I(\zeta)))}{I(\zeta)} \tag{2.45}$$

3. **Server busy (reduced-rate service / working vacation):**

$$\Upsilon_2(\zeta) = \int_0^\infty \Upsilon_2(\xi, \zeta)\, d\xi = \frac{\Upsilon_2(0, \zeta)\,(1 - \mathcal{A}_3^*(I(\zeta) + \omega))}{I(\zeta) + \omega} \tag{2.46}$$

4. **Server under repair:**

$$\Upsilon_3(\zeta) = \int_0^\infty \Upsilon_3(\xi, \zeta)\, d\xi = \frac{\Upsilon_3(0, \zeta)\,(1 - \mathcal{A}_4^*(I(\zeta)))}{I(\zeta)} \tag{2.47}$$

5. **Server in setup phase:**

$$\Upsilon_4(\zeta) = \int_0^\infty \Upsilon_4(\xi, \zeta)\, d\xi = \frac{\Upsilon_4(0, \zeta)\,(1 - \mathcal{A}_5^*(I(\zeta)))}{I(\zeta)} \tag{2.48}$$

Among these expressions, the only unknown quantity is Γ_0, which can be determined using the normalization condition:

$$\Gamma_0 + \Gamma_1 + \Upsilon_0(1) + \Upsilon_1(1) + \Upsilon_2(1) + \Upsilon_3(1) + \Upsilon_4(1) = 1.$$

Solving for Γ_0, we obtain:

$$\begin{aligned}\Gamma_0 = \frac{\omega^2\,(\mathcal{A}^*(\pi_1) - \pi_1\,\mathbb{E}[A_3])}{\omega^2\,(\pi_1\,(\mathbb{E}[A_4](1-\alpha) + \mathbb{E}[A_5]) + \mathcal{A}_1^*(\pi_1))} \\ + \omega\pi_1^2\,(2\,\mathcal{A}_3^{*\prime}(\omega)\,(\pi_1\mathbb{E}[A_2] - \mathcal{A}_1^*(\pi_1) + 1) - \mathbb{E}[A_2]\,\mathcal{A}_3^*(\omega)) \\ + \pi_1\mathcal{A}_1^*(\pi_1) + \pi_1^2\,(1 - \mathcal{A}_3^*(\omega))\end{aligned} \tag{2.49}$$

Corollaire 2.3.1. *Under steady-state conditions, the probability generating function (PGF) of the number of customers in the system, $K_s(\zeta)$, is given by:*

$$K_s(\zeta) = \Gamma_0 + \Gamma_1 + \Upsilon_0(\zeta) + \zeta\Upsilon_1(\zeta) + \zeta\Upsilon_2(\zeta) + \zeta\Upsilon_3(\zeta) + \zeta\Upsilon_4(\zeta). \tag{2.50}$$

The PGF of the number of customers in the orbit, $K_o(\zeta)$, is given by:

$$K_o(\zeta) = \Gamma_0 + \Gamma_1 + \Upsilon_0(\zeta) + \Upsilon_1(\zeta) + \Upsilon_2(\zeta) + \Upsilon_3(\zeta) + \Upsilon_4(\zeta). \tag{2.51}$$

2.4 Performance Measures and Reliability

Our analysis relies on the following key characteristics of the retrial queueing system.

2.4.1 System state probabilities

1. The steady-state probability that the server is idle during the retrial period is expressed as:

$$I = \Upsilon_0(1)$$

$$= \frac{-\pi_1\Gamma_0\big(\mathcal{A}_1^*(\pi_1)-1\big)}{\omega^2\big(\mathcal{A}_1^*(\pi_1)-\pi_1E(A_2)\big)}\Bigg\{\omega\Big(-E(A_2)\pi_1\big(\mathcal{A}_3^*(\omega)-1\big)+\omega\big(-\alpha E(A_4)+E(A_2)+E(A_4)\big)\Big)$$

$$+\omega E(A_4)+\pi_1\omega\mathcal{A}_3^{*'}(\omega)+\pi_1\big(\omega\mathcal{A}_3^{*'}(\omega)-\mathcal{A}_3^*(\omega)+1\big)\Bigg\}$$

2. The steady-state probability that the server is actively serving customers during a regular service period is given by:

$$U = \Upsilon_1(1)$$

$$= \frac{\pi_1\Gamma_0E(A_2)}{\omega^2\big(\mathcal{A}_1^*(\pi_1)-\pi_1E(A_2)\big)}\Bigg\{\omega^2\Big(-\alpha E(A_4)\pi_1+E(A_4)\pi_1+E(A_4)\pi_1+\mathcal{A}_1^*(\pi_1)\Big)$$

$$+\omega\Big(2\pi_1^2\mathcal{A}_3^{*'}(\omega)-\pi_1\mathcal{A}_3^*(\omega)\mathcal{A}_1^*(\pi_1)+\pi_1\mathcal{A}_1^*(\pi_1)\Big)$$

$$-\pi_1^2\mathcal{A}_3^*(\omega)+\pi_1^2\Bigg\}$$

3. The steady-state probability that the server is occupied during a working vacation is:

$$V = \Upsilon_2(1) = \frac{\pi_1^2\Gamma_0\cdot(1-\mathcal{A}_3^*(\omega))}{\omega^2}$$

4. The steady-state probability that the server is undergoing repair is:

$$S = \Upsilon_3(1) = E\left(A_4\right)\pi_1\Gamma_0\cdot(1-\alpha)$$

5. The steady-state probability that the server is in setup mode is:

$$K = \Upsilon_4(1) = E\left(A_4\right)\pi_1\Gamma_0$$

2.4.2 Mean system size and orbit size

(i) To quantify congestion in the orbit, we compute the expected number of waiting customers, denoted L_q. This is achieved by taking the first derivative of the orbit's probability generating function (Equation 2.51) with respect to z, then evaluating the limit as $z \longrightarrow 1$:

$$L_q = K_o'(1) = \lim_{z \to 1} \frac{d}{dz} K_o(z)$$

(ii) Similarly, the expected total number of customers present in the entire system including those in service and in orbit is captured by L_s. It is obtained by differentiating the system's PGF (Equation 2.50) at $z = 1$:

$$L_s = K_s'(1) = \lim_{z \to 1} \frac{d}{dz} K_s(z)$$

(iii) Using Little's Law we derive the average sojourn time in the system (W_s) and the average waiting time in the orbit (W_q), both scaled by the effective arrival rate π_1:

$$W_s = \frac{L_s}{\pi_1} \quad \text{and} \quad W_q = \frac{L_q}{\pi_1}.$$

2.4.3 Reliability Measures

In retrial queueing systems with server unreliability, reliability metrics play a pivotal role in evaluating system resilience and guiding performance enhancement strategies.

(i) The steady-state availability of the server, denoted A_v, reflects the proportion of time the server is operational either actively serving customers or remaining idle and ready for service. It is formally expressed as:

$$A_v = 1 - \Upsilon_3(1) = 1 - E\left(A_4\right) \pi_1 \Gamma_0 \cdot (1 - \alpha) \tag{2.52}$$

(ii) The steady-state probability of server failure, denoted F_f, quantifies the likelihood that the server is in a failed state during setup. This is given by:

$$F_f = \alpha \Upsilon_4(1) = \alpha E\left(A_4\right) \pi_1 \Gamma_0$$

Theorem 2.4.1. *Let $E(T_b)$ and $E(T_c)$ denote the expected durations of a busy period and a complete busy cycle, respectively, under steady-state conditions. These quantities are derived as follows:*

$$E(T_b) = \frac{1}{\omega^2\big(A^*(\pi_1) - \pi_1 E(A_2)\big)} \Bigg\{ \omega^2 \Big(- \alpha E(A_4) + E(A_2) + E(A_5) + E(A_4) \Big)$$

$$+\omega\Big(-E(A_2)\pi_1\mathcal{A}_3^*(\omega)+2\pi_1\mathcal{A}_3^{*'}(\omega)\big(E(A_2)\pi_1-\mathcal{A}_1^*(\pi_1)+1\big)\Big)$$

$$+\omega\mathcal{A}_1^*(\pi_1)-\pi_1\mathcal{A}_3^*(\omega)+\pi_1\Bigg\} \tag{2.53}$$

$$E(T_c)=\frac{1}{\omega^2\pi_1\big(\mathcal{A}_1^*(\pi_1)-\pi_1E(A_2)\big)}\Bigg\{\omega^2\Big(\pi_1\big(E(A_4)(1-\alpha)+E(A_4)\big)+\mathcal{A}_1^*(\pi_1)\Big)$$

$$+\omega\Big(\pi_1^2\Big[2\mathcal{A}_3^{*'}(\omega)\big(\pi_1E(A_2)-\mathcal{A}_1^*(\pi_1)+1\big)\Big]\Big)$$

$$-\pi_1^2\omega E(A_2)\mathcal{A}_3^*(\omega)+\pi_1\mathcal{A}_1^*(\pi_1)+\pi_1^2\big(1-\mathcal{A}_3^*(\omega)\big)\Bigg\} \tag{2.54}$$

Proof. The derivation follows from the classical framework of alternating renewal processes. Specifically, the idle probability Γ_0 satisfies:

$$\Gamma_0=\frac{E\left(T_0\right)}{E\left(T_b\right)+E\left(T_0\right)}\quad\Rightarrow\quad E\left(T_b\right)=\frac{1}{\pi_1}\left(\frac{1}{\Gamma_0}-1\right),\quad E\left(T_c\right)=E\left(T_0\right)+E\left(T_b\right) \tag{2.55}$$

where T_0 represents the duration during which the system remains empty. Given that customer inter-arrival times are exponentially distributed with rate π_1, it follows that $E(T_0)=1/\pi_1$. Substituting Equation (2.49) into (2.55) and performing direct algebraic simplifications yields the expressions in (2.53) and (2.54).

2.5 Cost and Optimization Analysis

In practice, queue managers strive to minimize the system's operating cost per unit time. In this section, we formulate a steady-state expected cost function per unit time, treating the service rate π_3 as the decision variable. Our goal is to determine the optimal value of π_3 that minimizes this cost function.

To do so, we first define the following cost components:

- $\mathcal{F}_1$: The holding cost per customer in the system per unit time.
- $\mathcal{F}_2$: The cost per unit time to keep the server running (even when idle).
- $\mathcal{F}_3$: The service cost per unit time (proportional to the service rate).

- $\mathcal{F}_4$: The setup or startup cost per unit time required to activate the server.

Let $\mathcal{T}_c$ denote the total expected cost per unit time. It is given by:

$$\mathcal{T}_c = \mathcal{F}_1 L_s + \mathcal{F}_2(1 - \Gamma_0) + \mathcal{F}_3\pi_3 + \mathcal{F}_4\Gamma_0.$$

2.5.1 Quadratic Fit Search Method

In this section, we turn our attention to the cost optimization challenge, approached through the lens of the Quadratic Fit Search Method (QFSM), under the predefined cost structure. As outlined by Rardin [77], this method thoughtfully constructs a quadratic approximation of the objective function using a three-point sampling pattern ensuring, under mild conditions, the existence of a unique minimum within the interval of interest. Our central aim is to fine-tune the service rate π_3 across varying operational scenarios, with the goal of minimizing the expected total cost function $\mathcal{T}_c$, which for simplicity and readability we denote here as H.

We assume for the sake of focus and analytical clarity that all system parameters remain fixed, leaving π_3 as the sole decision variable under our control. Thus, the optimization problem reduces elegantly to:

$$\text{Minimize: } H(\pi_3) = \mathcal{F}_1 L_s + \mathcal{F}_2(1 - \Gamma_0) + \mathcal{F}_3\pi_3 + \mathcal{F}_4\Gamma_0.$$

As Laxmi et al. [54] insightfully observed, given any triplet of points (z^l, z^m, z^u), one can construct a quadratic interpolant that exactly matches the values of $H(z)$ at these locations. This surrogate function, despite its simplicity, reliably yields a unique minimum denoted z^q which serves as an intelligent estimate of the true optimum. This estimate is then used to strategically update the search interval by replacing the least informative point, thereby accelerating convergence with grace and efficiency.

The analytical expression for this interpolated minimum z^q derived via quadratic interpolation through the three sample points is given by:

$$z^q \approx \frac{1}{2}\left[\frac{H(z^l)\left((z^m)^2 - (z^u)^2\right) + H(z^m)\left((z^u)^2 - (z^l)^2\right) + H(z^u)\left((z^l)^2 - (z^m)^2\right)}{H(z^l)(z^m - z^u) + H(z^m)(z^u - z^l) + H(z^u)(z^l - z^m)}\right].$$

To ensure consistency and reproducibility across all numerical experiments presented in this section, we fix the cost coefficients as follows: $\mathcal{F}_1 = 10$, $\mathcal{F}_2 = 350$, $\mathcal{F}_3 = 20$, $\mathcal{F}_4 = 120$.

2.5.2 Optimization analysis

To carry out the numerical analysis for parameter optimization in the considered queueing system, we adopt the following baseline parameter values: $\alpha = 0.7$, $\pi_1 = 2$, $\omega = 3$, $\pi_4 = 2$, $\pi_6 = 5$, $\pi_5 = 4$, and $\pi_2 = 8$. The Quadratic Fit Search Method (QFSM) is implemented with a convergence tolerance of $\epsilon = 10^{-6}$, ensuring high precision in locating the optimal solution.

As illustrated in **Figure** 2.2, the cost curves exhibit clear convex behavior a reassuring indicator that, for the given set of model parameters, there exists a well-defined service rate π_3 that globally minimizes the expected total cost function. Guided by this convexity, we initialize the QFSM algorithm with the three-point pattern $\left(\pi_3^l, \pi_3^m, \pi_3^u\right) = (3.05, 3.5, 3.75)$. After a finite number of carefully orchestrated iterations, the algorithm converges gracefully to the optimal solution: a minimal expected operating cost of $H = 372.29$, achieved at the optimal service rate $\pi_3^* = 3.282$. This result not only validates the effectiveness of QFSM in this context but also offers a tangible, actionable insight for system designers seeking cost-efficient performance tuning.

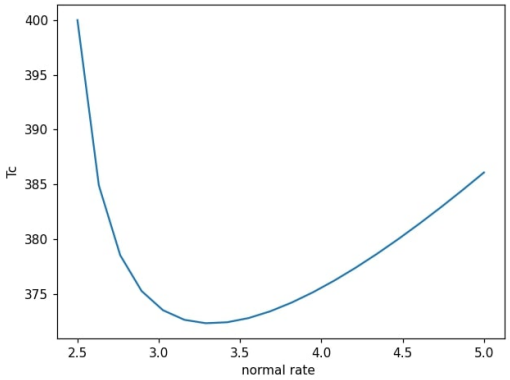

Figure 2.2: The optimum service rate π_3^*

Moreover, we conduct a systematic sensitivity analysis to explore how variations in the cost parameters influence the behavior of the expected cost function H. To ensure comparability across scenarios, all system parameters are held constant at the following baseline values: $\alpha = 0.7$, $\pi_1 = 2$, $\pi_4 = 2$, $\omega = 3$, $\pi_6 = 5$, $\pi_5 = 4$, and $\pi_2 = 8$.

The impact of pairwise cost parameter adjustments is visualized in **Tables 1–3**, which respectively examine the effects of varying: (i) $(\mathcal{F}_1, \mathcal{F}_2)$ representing holding and failure-related costs, (ii) $(\mathcal{F}_2, \mathcal{F}_4)$ linking failure penalties with reward incentives, and (iii) $(\mathcal{F}_4, \mathcal{F}_3)$ balancing rewards against service effort costs.

Across all configurations, a consistent and intuitive pattern emerges: the expected cost function H exhibits a near-linear upward trend as the underlying cost parameters increase. This monotonic behavior not only reinforces the economic rationality of the model but also provides decision-makers with predictable, interpretable guidance higher operational or penalty costs naturally lead to higher total expected expenditures, underscoring the importance of strategic parameter calibration.

Table 2.1: Sensitivity of $\mathcal{T}_c$ to $(\mathcal{F}_1, \mathcal{F}_2)$, with $\mathcal{F}_3 = 20$, $\mathcal{F}_4 = 120$.

$(\mathcal{F}_1, \mathcal{F}_2)$	(100,250)	(100,150)	(100,100)	(150,250)	(200,250)
$\mathcal{T}_c$	222.1301	163.4991	134.1836	225.0850	228.0805

Table 2.2: Sensitivity of $\mathcal{T}_c$ to $(\mathcal{F}_2, \mathcal{F}_4)$, with $\mathcal{F}_1 = 10$, $\mathcal{F}_3 = 20$.

$(\mathcal{F}_2, \mathcal{F}_4)$	(350,150)	(350,200)	(250,200)	(150,120)	(100,120)
$\mathcal{T}_c$	293.1718	313.8563	255.2253	163.4991	134.1836

Table 2.3: Sensitivity of $\mathcal{T}_c$ to $(\mathcal{F}_4, \mathcal{F}_3)$, with $\mathcal{F}_1 = 10$, $\mathcal{F}_2 = 350$.

$(\mathcal{F}_4, \mathcal{F}_3)$	(100,10)	(200,10)	(200,15)	(120,5)	(120,20)
$\mathcal{T}_c$	272.4873	313.8563	323.8563	270.7611	300.7611

2.6 Numerical Examples and Sensitivity Analysis

In this section, we present a series of numerical experiments implemented in Python to explore how variations in key system parameters influence critical performance measures. To ensure analytical tractability and realism, we assume that all stochastic processes including retrial times, service times, lower-speed service durations, vacation periods, setup times, and repair times follow exponential distributions, characterized by the probability density function $k(x) = ve^{-vx}$ for $x > 0$.

Parameter values are carefully selected to satisfy the necessary stability conditions of the system, ensuring meaningful and convergent results. The subsequent tables summarize the computed steady-state performance metrics, including:

- Γ_0: the probability that the server is idle;

- L_q: the mean number of customers in orbit (mean orbit size);
- π_2 The retrial rate ;
- π_3 The normal rate service ;
- π_4 The lower rate service ;
- π_5 The repair rate ;
- π_6 The setup rate.

These metrics not only quantify system efficiency and resource utilization but also provide intuitive, decision-ready insights for practitioners seeking to balance responsiveness, cost, and reliability in real-world queueing environments.

As depicted in **Figure** 2.3, we observe a clear and intuitive trend: the probability that the server remains idle, Γ_0, increases monotonically with higher values of both the lower-speed service rate π_3 and the regular service rate π_4. This behavior reflects the system's improved responsiveness as service capacity increases (even at reduced speed), customers are served more promptly, reducing congestion and allowing the server to return to idle states more frequently. These findings offer valuable insight for capacity planning, highlighting how even modest enhancements in service rates can meaningfully elevate system efficiency and resource availability.

As shown in **Figure** 2.4, the mean orbit size L_q decreases monotonically with increasing values of both the lower-speed service rate π_4 and the retrial rate π_2. This trend reveals an important operational insight: enhancing either the system's ability to serve customers even at a reduced speed or encouraging customers to retry more frequently contributes to alleviating congestion in the orbit. In practical terms, this means decision-makers can reduce waiting populations not only by investing in service capacity but also by designing retry-friendly policies that keep the system fluid and responsive.

As illustrated in **Figure** 2.5, the mean orbit size L_q exhibits a clear decreasing trend as both the lower service rate π_4 and the regular service rate π_3 increase. This behavior underscores a fundamental principle of queueing dynamics: whether through baseline service capacity (π_3) or auxiliary, perhaps energy-efficient, lower-rate processing (π_4), enhancing service throughput in any form contributes to reducing customer congestion in the orbit. These results empower system designers to strategically allocate resources: even modest improvements in secondary service modes can meaningfully alleviate pressure on the primary queue.

As shown in **Figure** 2.6, the server's idle probability Γ_0 increases with a higher setup rate, assuming a fixed repair rate. This trend reflects an intuitive operational dynamic: when the server transitions more rapidly from setup to active service, it spends less time in non-productive

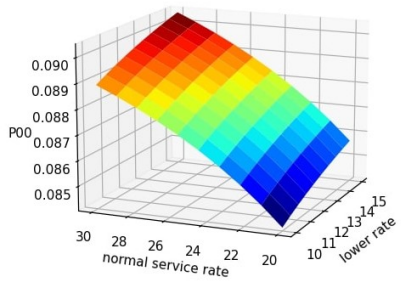

Figure 2.3: Variation in Γ_0 with π_3 and π_4

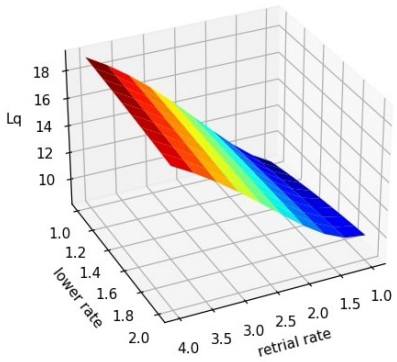

Figure 2.4: Impact of ξ and π_4 on (L_q)

preparatory states thereby increasing the likelihood of returning to an idle, available state after completing tasks. In practical terms, accelerating setup processes not only enhances system responsiveness but also improves resource availability, offering system designers a valuable lever for optimizing efficiency without altering repair protocols.

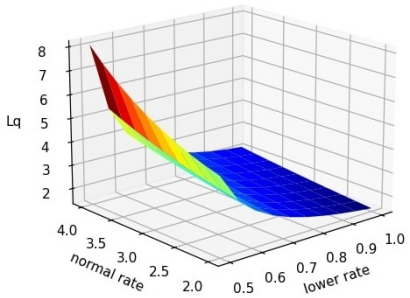

Figure 2.5: Impact of π_3 and π_4 on (L_q)

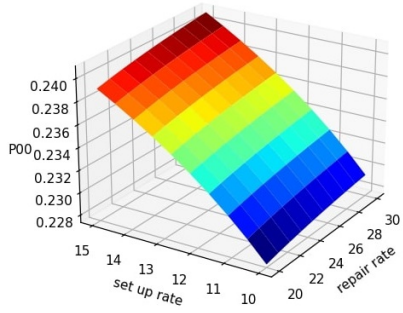

Figure 2.6: Effect of the setup rate and repair rate on Γ_0

As depicted in **Figure** 2.7, the probability of the server being in the repair state decreases as the repair rate increases. This intuitively expected behavior highlights a core principle of system reliability: faster repair processes reduce the server's downtime, allowing it to return more swiftly to active or idle operational states. From a managerial perspective, this underscores the value of investing in maintenance efficiency not only to minimize disruption, but also to enhance overall system availability and responsiveness without altering other structural parameters.

As shown in **Figure** 2.8, an increase in the service rate π_3 is associated with a higher probability of the server being in the setup state. This counterintuitive yet meaningful trend may arise from the server's accelerated cycling between active service and idle periods leading to more frequent transitions into setup as the system resets or reconfigures between bursts of activity. In

essence, while higher service rates improve throughput, they may also increase the frequency of preparatory phases, highlighting a subtle trade-off between speed and operational overhead.

As observed in **Figure** 2.9, the probability of the server being in the vacation state decreases as the working vacation rate increases. This behavior is both intuitive and analytically sound: a higher vacation rate corresponds to shorter average vacation durations, causing the server to spend less time in this passive state and transition more rapidly back into active service or idle readiness. From an operational standpoint, this highlights how tuning the vacation rate can serve as an effective mechanism for balancing energy conservation (or resource idling) with service responsiveness enabling systems to remain agile without sacrificing efficiency.

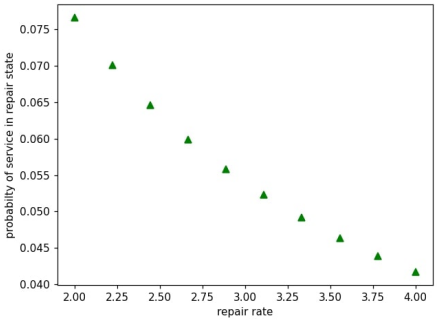

Figure 2.7: Effect of repair rate on repair state probability of server.

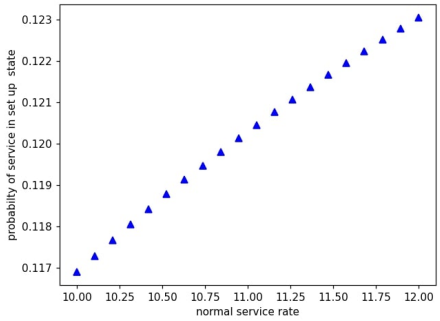

Figure 2.8: Probability of server in setup versus π_3.

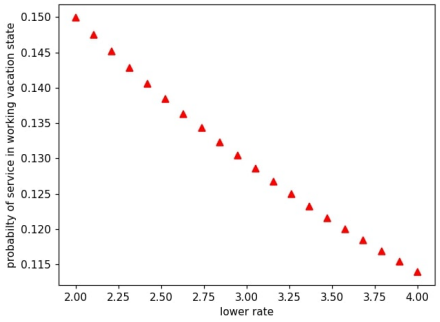

Figure 2.9: Probability of server in vacation versus vacation rate π_4.

Chapter 3

M/G/1 Retrial queue with Negative Customers Under Working Vacation and Setup-Time

Queues with negative customers, also known as G-queues, have gained considerable interest due to their wide-ranging applications in computer systems, communication networks, and manufacturing processes. Negative customers arrive into the system only during the service time of positive customers; they do not join the queue nor receive any service. Instead, a negative customer removes one positive customer currently in service. The removed positive customer may either rejoin the queue (orbit) for another service attempt or leave the system, depending on certain probabilities.

In this chapter, we extend the retrial queue model by incorporating two types of server breakdowns that occur under working vacation and setup time conditions. The first type of breakdown occurs as in the previous model, while the second type is triggered specifically by the arrival of negative customers during normal service. The server undergoes repairs depending on the cause of breakdown, and these repair times follow different probabilistic distributions. We conduct a steady-state analysis of this advanced model, derive system performance and cost measures, and perform sensitivity analysis to understand the impact of various parameters on system behavior and optimization and optimal strategy analysis.

3.1 Justifications of the Model and Practical Relevance

In a private medical clinic, the patients arrive randomly in time one after another for MRI scans. A doctor primarily handles patients with the assistance of a medical assistant. The doctor works faster and more efficiently than the assistant. But if the doctor goes away for example, to attend a conference or on vacation the assistant handles the job, although less efficiently.

If the physician is recalled and there are no patients waiting, the MRI device is shut off to save energy. When a patient arrives while the machine is shut off, it is restarted, however The device may fail to start, triggering a repair phase before it becomes operational again.

In addition, even if the MRI equipment is functioning, unexpected disruptions otherwise known as "negative customers" while in service can cause abrupt failures. These interruptions cause repair processes that interrupt service until finished.

3.2 Model Desription

In this work, we analyze an M/G/1 unreliable G-retrial queue with single working vacation , setup time. The following premises define the structure of the model:

1. The arrival of positive Customers follows a Poisson process with a rate $\lambda > 0$.

2. Upon arrival, if the server is idle, the positive customer enters service immediately. on the other hand, if the server is busy, undergoing setup, on a working vacation, or out of order due to a breakdown, the positive Customer leave the service area and join a queueing pool called the orbit with probability q or balk with complementary probability $1-q$. In the orbit, customers are managed according to a FCFS discipline. While in orbit, the positive Customer waits a random amount of time before retrying. The inter-retrial times follow an arbitrary probability distribution function $\Psi_1(x)$ with corresponding LST function ${\Psi_1}^*(\theta)$.

3. The server starts servicing both new and retry positive Customer as soon as they arrive if it is free. The normal service time distributed with general distribution function $\Psi_2(x)$ having LST ${\Psi_2}^*(\theta)$

4. When the orbit is empty, the server automatically enters a single working vacation exponentially distributed with rate ω. If a positive Customer arrives during the working vacation, they are served at a reduced rate. Once the working vacation is over, the server resumes normal service. The service time distributed with general distribution function $\Psi_3(x)$ having LST $\Psi_3^*(\theta)$

5. When the working vacation ends and no positive Customers are waiting, the server instantly powers off to save energy.

6. In the server's off-state, any arriving positive Customer , they will wait at the front of the server until it is activated (setup time) . the setup time is assumed to follow general

distribution with probability distribution function $\Psi_4(x)$ with corresponding LST function $\Psi_4^*(\theta)$.

7. During the setup phase, the server may fail with a certain probability $\bar{\alpha} = (1 - \alpha)$. In the case of a setup failure the server undergoes a repair process, where the repair time is governed by a general distribution function $\Psi_5(x)$ with corresponding LST function $\Psi_5^*(\theta)$.

8. Negative Customers arrive from outside the system following a Poisson process with rate β, but only during the regular service of positive customers. Unlike positive Customers, negative Customers neither queue nor receive service; they instantly remove the positive Customer being served, causing a server breakdown that halts service (active breakdown). When a breakdown occurs, the server is immediately sent for repair. The repair time, follows a general probability distribution function $\Psi_6(x)$ with corresponding LST function $\Psi_6^*(\theta)$.

3.3 Steady-state analysis

The state of the system at time t can be defined by the Markov process $\{L(t); t \geq 0\} = \{A(t), B(t), \zeta_1(t), \zeta_2(t), \zeta_3(t), \zeta_4(t), \zeta_5(t), \zeta_6(t), t \geq 0\}$, where $X(t)$ denotes the number of customers in the orbit at time t and

$$A(t) = \begin{cases} 0, & \text{if the server is idle in a normal period} \\ 1, & \text{if the server is idle in a working vacation period} \\ 2, & \text{if the server is busy on normal service} \\ 3, & \text{if the server is busy on working vacation period} \\ 4, & \text{if the server is on repair} \\ 5, & \text{if the server is on setup state} \\ 6, & \text{if the server is on repair tim for an active breakdown} \end{cases}$$

If $A(t) = 0$ and $B(t) > 0$, then $\zeta_1(t)$ represents the elapsed retrial time, if $A(t) = 2$, then $\zeta_2(t)$ represents the elapsed service time during normal busy period at time t, if $A(t) = 3$ and $B(t) \geq 0$ then $\zeta_3(t)$ represents the elapsed working vacation time at time t and if $A(t) = 4$ and $B(t) \geq 1$ then $\zeta_4(t)$ represents the elapsed repair time at time t and if $A(t) = 5$ and $B(t) \geq 1$ then $\zeta_5(t)$ represents the elapsed setup time at time t and $B(t) \geq 0$ then $\zeta_6(t)$ represents the

elapsed repair time for an active breakdown at time t. The functions $f_i(x), \forall i = 1, \cdots, 6$ are the conditional completion rates for retrial time, service, working vacation, setup time, repair time, repair time for an active breakdown respectively, at time t i.e.

$$f_i(x)dx = \frac{d\Psi_i(x)}{1 - \Psi_i(x)}, \forall i = 1, \cdots, 6.$$

3.3.1 Stability and ergodicity condition

The sequence of random vectors $Z_n = \{A(t_n+), B(t_n+)\}$ forms a Markov chain which is embedded in the retrial queueing system.

Theorem 3.3.1. *The embedded Markov chain $\{Z_n; n \in N\}$ is ergodic if and only if $\varrho < \Psi_1^*(\lambda)$ for our system to be stable, where $\varrho = (\lambda/\beta)(1 - \Psi_2^*(\beta))\left(1 + \beta c^{(1)}\right)$. where $c^{(1)}$ the first moment of repair time for an active breakdown .*

Proof. *Demonstrating the sufficient condition for ergodicity often benefits from applying Foster's criterion, as discussed in Pakes [29] , which states that the chain $\{Z_n; n \in N\}$ is an irreducible and aperiodic Markov chain is ergodic if there exists a non-negative function $h(j), j \in N$ and $\varepsilon > 0$, such that mean drift $\varphi_j = E[h(z_{n+1}) - h(z_n)/z_n = j]$ is finite for all $j \in N$ and $\varphi_j \leqslant -\varepsilon$ for all $j \in N$, except perhaps for a finite number j 's. In our case, we consider the function $h(j) = j$. then we have*

$$\varphi_j = \begin{cases} \varrho - 1, & \text{if } j = 0, \\ \varrho - \Psi_1^*(\lambda), & \text{if } j = 1, 2, \ldots \end{cases}$$

Clearly the inequality $\varrho < \Psi_1^*(\lambda)$ is sufficient condition for Ergodicity.

To prove the necessary condition, as noted in Sennott et al. [30], if the Markov chain $\{Z_n; n \geqslant 1\}$ satisfies Kaplan's condition, namely, $\varphi_j < \infty$ for all $j \geqslant 0$ and there exits $j_0 \in N$ such that $\varphi_j \geqslant 0$ for $j \geqslant j_0$. Notice that, in our case, Kaplan's condition holds because there is a k such that $m_{ij} = 0$ for $j < i - k$ and $i > 0$, where $M = (m_{ij})$ is the one step transition matrix of $\{Z_n; n \in N\}$. Then $\varrho \geqslant \Psi^*(\lambda)$ implies the non-Ergodicity of the Markov chain.

3.3.2 Governing Equations

For the Markov process $\{L(t); t \geq 0\}$, we define the probability

$$\Pi_0(t) = \{A(t) = 0, B(t) = 0\} \Pi_1(t) = \{A(t) = 1, B(t) = 0\}$$

and the probability densities

$$\Pi_{2,n}(x,t)dx = \{A(t)=0, B(t)=n, x \le \chi_1(t) < x+dx\}, x \ge 0, n \ge 1$$
$$\Pi_{3,n}(x,t)dx = \{A(t)=2, B(t)=n, x \le \chi_2(t) < x+dx\}, x \ge 0, n \ge 0$$
$$\Pi_{4,n}(x,t)dx = \{A(t)=3, B(t)=n, x \le \chi_3(t) < x+dx\}, x \ge 0, n \ge 0$$
$$\Pi_{5,n}(x,t)dx = \{A(t)=4, B(t)=n, x \le \chi_4(t) < x+dx\}, x \ge 0, n \ge 1$$
$$\Pi_{6,n}(x,t)dx = \{A(t)=5, B(t)=n, x \le \chi_5(t) < x+dx\}, x \ge 0, n \ge 1$$
$$\Pi_{7,n}(x,t)dx = \{A(t)=6, B(t)=n, x \le \chi_5(t) < x+dx\}, x \ge 0, n \ge 1$$

Based on the above assumptions and notations, our model is governed by the following set of differential difference equations,

$$\lambda \Pi_0 = \omega \Pi_1, \tag{3.1}$$

$$(\lambda+\omega)\Pi_1 = \int_0^\infty \Pi_{3,0}(x) f_2(x) dx + \int_0^\infty \Pi_{4,0}(x) f_3(x) dx + \int_0^\infty \Pi_{7,0}(x) f_6(x) dx, \tag{3.2}$$

$$\frac{d}{dx}\Pi_{2,n}(x) + (\lambda + f_1(x))\Pi_{2,n}(x) = 0, \qquad x>0, n \ge 1, \tag{3.3}$$

$$\frac{d}{dx}\Pi_{3,0}(x) + (\lambda q + \beta + f_2(x))\,\Pi_{3,0}(x) = 0, \qquad x>0 \tag{3.4}$$

$$\frac{d}{dx}\Pi_{3,n}(x) + (\lambda q + \beta + f_2(x))\,\Pi_{3,n}(x) = \lambda q \Pi_{3,n-1}(x), \qquad x>0, n \ge 1 \tag{3.5}$$

$$\frac{d}{dx}\Pi_{4,0}(x) + (\lambda q + \omega + f_3(x))\,\Pi_{4,0}(x) = 0, \qquad x>0 \tag{3.6}$$

$$\frac{d}{dx}\Pi_{4,n}(x) + (\lambda q + \omega + f_3(x))\,\Pi_{4,n}(x) = \lambda q \Pi_{4,n-1}(x), \qquad x>0, n \ge 1 \tag{3.7}$$

$$\frac{d}{dx}\Pi_{5,0}(x) + (\lambda q + f_4(x))\Pi_{5,0}(x) = 0, \qquad x>0 \tag{3.8}$$

$$\frac{d}{dx}\Pi_{5,n}(x) + (\lambda q + f_4(x))\Pi_{5,n}(x) = \lambda q \Pi_{5,n-1}(x), \qquad x>0, n \ge 1 \tag{3.9}$$

$$\frac{d}{dx}\Pi_{6,0}(x) + (\lambda q + f_5(x))\Pi_{6,0}(x) = 0, \qquad x > 0 \tag{3.10}$$

$$\frac{d}{dx}\Pi_{6,n}(x) + (\lambda q + f_5(x))\Pi_{6,n}(x) = \lambda q \Pi_{6,n-1}(x), \qquad x > 0, n \geq 1 \tag{3.11}$$

$$\frac{d}{dx}\Pi_{7,0}(x) + (\lambda q + f_6(x))\Pi_{7,0}(x) = 0, \qquad x > 0 \tag{3.12}$$

$$\frac{d}{dx}\Pi_{7,n}(x) + (\lambda q + f_6(x))\Pi_{7,n}(x) = \lambda q \Pi_{7,n-1}(x), \qquad x > 0, n \geq 1. \tag{3.13}$$

The boundary conditions at $x = 0$ include $\Pi_{2,n}(0), \Pi_{3,0}(0), \Pi_{3,n}(0), \Pi_{4,0}(0), \Pi_{5,0}(0), \Pi_{5,n}(0), \Pi_{6,0}(0), \Pi_{7,n}$. The description of these terms at $x = 0$ are described as follows:

$$\Pi_{2,n}(0) = \int_0^\infty \Pi_{4,n}(x) f_3(x) dx + \int_0^\infty \Pi_{3,n}(x) f_2(x) dx + \int_0^\infty \Pi_{7,n}(x) f_6(x) dx, \qquad n \geq 1 \tag{3.14}$$

$$\Pi_{3,0}(0) = \alpha \int_0^\infty \Pi_{6,n}(x) f_5(x) dx + \int_0^\infty \Pi_{2,1}(x) f_1(x) dx + \omega \int_0^\infty \Pi_{4,0}(x) dx + \int_0^\infty \Pi_{5,0}(x) f_4(x) dx \tag{3.15}$$

$$\Pi_{3,n}(0) = \alpha \int_0^\infty \Pi_{6,n}(x) f_5(x) dx + \int_0^\infty \Pi_{2,n+1}(x) f_1(x) dx + \omega \int_0^\infty \Pi_{4,n}(x) dx + \lambda \int_0^\infty \Pi_{2,n}(x) dx + \int_0^\infty \Pi_{5,n}(x) f_4(x) dx, \qquad n \geq 1 \tag{3.16}$$

$$\Pi_{4,0}(0) = \lambda \Pi_1, \qquad \Pi_{4,n}(0) = 0, \qquad n \geq 1 \tag{3.17}$$

$$\Pi_{5,0}(0) = \bar{\alpha} \int_0^\infty \Pi_{6,0}(x) f_5(x) dx, \qquad \Pi_{5,n}(0) = \bar{\alpha} \int_0^\infty \Pi_{6,n}(x) f_5(x) dx, \qquad n \geq 1 \tag{3.18}$$

$$\Pi_{6,0}(0) = \lambda \Pi_0, \qquad \Pi_{6,n}(0) = 0, \qquad n \geq 1 \tag{3.19}$$

$$\Pi_{7,0}(0) = \beta \int_0^\infty \Pi_{3,0}(x) f_2(x) dx, \qquad \Pi_{7,n}(0) = \beta \int_0^\infty \Pi_{3,n}(x) f_2(x) dx, \qquad n \geq 1. \tag{3.20}$$

The normalization condition is given by

$$\Pi_0 + \Pi_1 + \sum_{n=1}^{\infty}\int_0^{\infty}\Pi_{2,n}(x)dx + \sum_{n=0}^{\infty}\int_0^{\infty}\Pi_{3,n}(x)dx + \sum_{n=0}^{\infty}\int_0^{\infty}\Pi_{4,n}(x)dx + \sum_{n=0}^{\infty}\int_0^{\infty}\Pi_{5,n}(x)dx + \sum_{n=0}^{\infty}\int_0^{\infty}\Pi_{6,n}(x)dx + \sum_{n=0}^{\infty}\int_0^{\infty}\Pi_{7,n}(x)dx = 1$$

In order to solve the above set of equations we define the generating functions as:

$$\Pi_2(x,z) = \sum_{n=1}^{\infty} z^n \Pi_{2,n}(x), \qquad \text{for } |z| \leq 1 \text{ and } x > 0,$$

$$\Pi_2(0,z) = \sum_{n=1}^{\infty} z^n \Pi_{2,n}(0), \qquad \text{for } |z| \leq 1,$$

$$\Pi_3(x,z) = \sum_{n=0}^{\infty} z^n \Pi_{3,n}(x), \qquad \text{for } |z| \leq 1 \text{ and } x > 0,$$

$$\Pi_3(0,z) = \sum_{n=0}^{\infty} z^n \Pi_{3,n}(0),$$

$$\Pi_4(x,z) = \sum_{n=0}^{\infty} z^n \Pi_{4,n}(x), \qquad \text{for } |z| \leq 1 \text{ and } x > 0,$$

$$\Pi_4(0,z) = \sum_{n=0}^{\infty} z^n \Pi_{4,n}(0),$$

$$\Pi_5(x,z) = \sum_{n=0}^{\infty} z^n \Pi_{5,n}(x), \qquad \text{for } |z| \leq 1,$$

$$\Pi_5(0,z) = \sum_{n=0}^{\infty} z^n \Pi_{5,n}(0),$$

$$\Pi_6(x,z) = \sum_{n=0}^{\infty} z^n \Pi_{6,n}(x), \qquad \text{for } |z| \leq 1,$$

$$\Pi_6(0,z) = \sum_{n=0}^{\infty} z^n \Pi_{6,n}(0),$$

$$\Pi_7(x,z) = \sum_{n=0}^{\infty} z^n \Pi_{7,n}(x), \qquad \text{for } |z| \leq 1,$$

$$\Pi_7(0,z) = \sum_{n=0}^{\infty} z^n \Pi_{7,n}(0). \qquad \text{for } |z| \leq 1$$

Multiplying equations (3.2)-(3.13) by suitable powers of z and summing over n, we obtain the following set of partial differential equations

$$\frac{\partial \Pi_2(x,z)}{\partial x} + (\lambda + f_1(x))\Pi_2(x,z) = 0 \tag{3.21}$$

$$\frac{\partial \Pi_3(x,z)}{\partial x} + (\lambda q - \lambda q z + \beta + f_2(x))\,\Pi_3(x,z) = 0 \tag{3.22}$$

$$\frac{\partial \Pi_4(x,z)}{\partial x} + (\lambda q - \lambda q z + \omega + f_3(x))\,\Pi_4(x,z) = 0 \tag{3.23}$$

$$\frac{\partial \Pi_5(x,z)}{\partial x} + (\lambda q - \lambda q z + f_4(x))\Pi_5(x,z) = 0 \tag{3.24}$$

$$\frac{\partial \Pi_6(x,z)}{\partial x} + (\lambda q - \lambda q z + f_5(x))\Pi_6(x,z) = 0 \tag{3.25}$$

$$\frac{\partial \Pi_7(x,z)}{\partial x} + (\lambda q - \lambda q z + f_6(x))\Pi_7(x,z) = 0 \tag{3.26}$$

Solving the above partial differential equations (3.21) to (3.26)

$$\Pi_2(x,z) = \Pi_2(0,z)[1 - \Psi_1(x)]e^{-\lambda x} \tag{3.27}$$

$$\Pi_3(x,z) = \Pi_3(0,z)\,[1 - \Psi_2(x)]\,e^{-A_b(z)x} \tag{3.28}$$

$$\Pi_4(x,z) = \Pi_4(0,z)\,[1 - \Psi_3(x)]\,e^{-A_v(z)x} \tag{3.29}$$

$$\Pi_5(x,z) = \Pi_5(0,z)\,[1 - \Psi_4(x)]\,e^{-I(z)\cdot x} \tag{3.30}$$

$$\Pi_6(x,z) = \Pi_6(0,z)\,[1 - \Psi_5(x)]\,e^{-I(z)\cdot x} \tag{3.31}$$

$$\Pi_7(x,z) = \Pi_7(0,z)\,[1 - \Psi_6(x)]\,e^{-I(z)\cdot x} \tag{3.32}$$

where $I(z) = \lambda q(1-z)$,$A_b(z) = \lambda q(1-z) + \beta$, $A_v(z) = \lambda q(1-z) + \omega$
Multiplying equation (3.14) by suitable powers of z, summing over n from 1 to ∞ and using equations (3.1) and (3.2) after some algebraic manipulations, we get

$$\Pi_2(0,z) = \int_0^\infty \Pi_4(x,z) f_3(x)\,dx + \int_0^\infty \Pi_3(x,z) f_2(x)\,dx + \int_0^\infty \Pi_7(x,z) f_6(x)\,dx - \left(\frac{\lambda+\omega}{\omega}\right)\lambda\Pi_0 \tag{3.33}$$

Similarly, multiplying equations (3.15)-(3.20) by suitable powers of z, summing over n and after some algebraic manipulations, we obtain

$$\Pi_3(0,z) = \frac{\alpha}{z}\int_0^\infty \Pi_6(x,z)f_5(x)\,dx + \lambda\int_0^\infty \Pi_2(x,z)\,dx + \frac{1}{z}\int_0^\infty \Pi_2(x,z)f_1(x)\,dx + \omega\int_0^\infty \Pi_4(x,z)\,dx + \int_0^\infty \Pi_5(x,z)f_4(x)\,dx \quad (3.34)$$

$$\Pi_4(0,z) = \Pi_{4,0}(0) \quad (3.35)$$

$$\Pi_5(0,z) = \bar{\alpha}\int_0^\infty \Pi_6(x,z)f_5(x)\,dx \quad (3.36)$$

$$\Pi_6(0,z) = \Pi_{6,0}(0) \quad (3.37)$$

$$\Pi_7(0,z) = \beta\int_0^\infty \Pi_3(x,z)\,dx \quad (3.38)$$

Inserting equations 3.28- 3.29 and 3.32 in 3.33 we get :

$$\Pi_2(0,z) = \Pi_4(0,z)\,\Psi_3^*(A_v(z)) + \Pi_3(0,z)\,\Psi_2^*(A_b(z)) + \Pi_7(0,z)\,\Psi_6^*(I(z)) - \left(\frac{\lambda+\omega}{\omega}\right)\lambda\Pi_0 \quad (3.39)$$

In similary way inserting equations 3.27- 3.32 in 3.34-3.38 we get :

$$\Pi_3(0,z) = \frac{1}{z}\,\Pi_2(0,z)\,\Psi_1^*(\lambda) + \alpha\,\Psi_4^*(I(z))\,\Pi_6(0,z) + V(z)\,\Pi_4(0,z) + \Pi_2(0,z)\,(1-\Psi_1^*(\lambda)) + \Pi_5(0,z)\,\Psi_5^*(I(z)) \quad (3.40)$$

where $V(z) = \frac{\omega}{A_v(z)}(1-\Psi_3^*(A_v(z)))$ using equations 3.32 and 3.42 :

$$\Pi_7(0,z) = D(z)\Pi_3(0,z) \quad (3.41)$$

where $D(z) = \frac{\beta}{A_b(z)}(1-\Psi_2^*(A_b(z)))$

The following are the symbolic solutions for the generating functions $\Pi_2(0,z)$, $\Pi_3(0,z)$, $\Pi_4(0,z)$, $\Pi_5(0,z)$, $\Pi_6(0,z)$, and $\Pi_7(0,z)$, expressed in terms of Π_0:

$$\Pi_4(0,z) = \frac{\lambda^2}{\omega}\Pi_0, \quad (3.42)$$

$$\Pi_6(0,z) = \lambda\Pi_0, \quad (3.43)$$

$$\Pi_5(0,z) = \overline{\alpha}\lambda\Psi_4^*(I(z))\Pi_0, \quad (3.44)$$

for $\Pi_2(0,z)$, $\Pi_3(0,z)$, and $\Pi_7(0,z)$, we define the following helper terms:

- Denominator:

$$\text{Den}(z) = 1 - [\Psi_2^*(A_b(z)) + D(z)\Psi_6^*(I(z))] \left(1 - \Psi_1^*(\lambda) + \frac{\Psi_1^*(\lambda)}{z}\right). \tag{3.45}$$

- Numerator for $\Pi_2(0,z)$, denoted $\text{Num}_P(z)$:

$$\begin{aligned}\text{Num}_P(z) = &\left(\frac{\lambda^2}{\omega}\Psi_3^*(A_v(z)) - \left(\frac{\lambda+\omega}{\omega}\right)\lambda\right) \\ &+ [\Psi_2^*(A_b(z)) + D(z)\Psi_6^*(I(z))] \left(\alpha\lambda\Psi_4^*(I(z)) + \frac{\lambda^2}{\omega}V(z) + \overline{\alpha}\lambda\Psi_4^*(I(z))\Psi_5^*(I(z))\right).\end{aligned} \tag{3.46}$$

- Numerator for $\Pi_3(0,z)$, denoted $\text{Num}_M(z)$:

$$\begin{aligned}\text{Num}_M(z) = &\left(1 - \Psi_1^*(\lambda) + \frac{\Psi_1^*(\lambda)}{z}\right)\left(\frac{\lambda^2}{\omega}\Psi_3^*(A_v(z)) - \left(\frac{\lambda+\omega}{\omega}\right)\lambda\right) \\ &+ \left(\alpha\lambda\Psi_4^*(I(z)) + \frac{\lambda^2}{\omega}V(z) + \overline{\alpha}\lambda\Psi_4^*(I(z))\Psi_5^*(I(z))\right).\end{aligned} \tag{3.47}$$

Using these helper terms, the remaining solutions are:

$$\Pi_2(0,z) = \Pi_0\left(\frac{\text{Num}_P(z)}{\text{Den}(z)}\right), \tag{3.48}$$

$$\Pi_3(0,z) = \Pi_0\left(\frac{\text{Num}_M(z)}{\text{Den}(z)}\right), \tag{3.49}$$

$$\Pi_7(0,z) = \Pi_0\left(D(z)\frac{\text{Num}_M(z)}{\text{Den}(z)}\right). \tag{3.50}$$

3.3.3 Steady state results

If the system is in steady state condition $\rho < \Psi_1^*(\lambda)$, the PGFs are as follows:
(I) the number of customers in the orbit when the server is idle;

$$\Pi_2(z) == \int_0^\infty \Pi_2(x,z)dx = \frac{\Pi_2(0,z)(1-\Psi_1^*(\lambda))}{\lambda} \tag{3.51}$$

(II) the number of customers in the orbit when the server is regularly busy;

$$\Pi_3(z) = \int_0^\infty \Pi_3(x,z)dx = \frac{\Pi_3(0,z)(1-\Psi_2^*(A_b(z))}{A_b(z)} \tag{3.52}$$

(III) the number of customers in the orbit when the server is at a lower speed service;

$$\Pi_4(z) = \int_0^\infty \Pi_4(x,z)dx = \frac{\Pi_4(0,z)(1-\Psi_3^*(A_v(z))}{A_v(z)} \tag{3.53}$$

(IV) the number of customers in the orbit when the server is at repair time;

$$\Pi_5(z) = \int_0^\infty \Pi_5(x,z)dx = \frac{\Pi_5(0,z)(1-\Psi_5^*(I(z))}{I(z)} \tag{3.54}$$

(V) the number of customers in the orbit when the server is at setup time;

$$\Pi_6(z) = \int_0^\infty \Pi_6(x,z)dx = \frac{\Pi_6(0,z)(1-\Psi_4^*(I(z))}{I(z)} \tag{3.55}$$

(VI) the number of customers in the orbit when the server is at repair time for an active breakdown;

$$\Pi_7(z) = \int_0^\infty \Pi_7(x,z)dx = \frac{\Pi_7(0,z)(1-\Psi_6^*(I(z))}{I(z)} \tag{3.56}$$

From the above equations, the only unknown is Π_0 which can be obtained by using the normalization condition $\Pi_0 + \Pi_1 + \Pi_2(1) + \Pi_3(1) + \Pi_4(1) + \Pi_5(1) + \Pi_6(1) + \Pi_7(1) = 1$ as

$$\Pi_0 = \frac{\beta\omega^2\left(\rho - \Psi_1(\lambda)\right)}{F(\alpha,\beta,\lambda,\omega,\rho,\Psi_1,\cdots,\Psi_6)}, \tag{3.57}$$

where

$$\begin{aligned}
F(\alpha,\beta,\lambda,\omega,\rho,\Psi_1,\ldots,\Psi_6) = {} & \beta\big(\rho-\Psi_1(\lambda)\big)\Big[-\lambda^2\big(\Psi_3(\omega)-1\big)+\lambda\omega+\omega^2\big(\lambda\bar{\alpha}\mathrm{E}(\Psi_5)+\lambda\mathrm{E}(\Psi_4)+1\big)\Big] \\
& -\lambda q\big(\Psi_1(\lambda)-1\big)\Big[\beta\lambda\big(\Psi_3(\omega)-1\big)-\beta\omega^2\big(\bar{\alpha}\mathrm{E}(\Psi_5)+\mathrm{E}(\Psi_4)\big) \\
& +\omega\big(-\beta\mathrm{E}(\Psi_6)-1\big)\big(\lambda\big(\Psi_3(\omega)-1\big)-\omega\big)\big(\Psi_2(\beta)-1\big)\Big] \\
& -\big(\Psi_2(\beta)-1\big)\Big[-\lambda^2 q\big(\Psi_3(\omega)-1\big)+\lambda\omega^2 q\big(\bar{\alpha}\mathrm{E}(\Psi_5)+\mathrm{E}(\Psi_4)\big) \\
& +\omega\big(-\lambda\Psi_3(\omega)+\lambda+\omega\big)\Psi_1(\lambda)\Big]\Big[\mathrm{E}(\Psi_6)\big(\Psi_2(\beta)-1\big)+1\Big].
\end{aligned}$$

Corollaire 3.3.1. *If the system satisfies the steady state condition, The PGF of the number of customers in the system* $(K_s(z))$ *is obtained using*

$$K_s(z) = \Pi_0 + \Pi_1 + \Pi_2(z) + z\Pi_3(z) + z\Pi_4(z) + \Pi_5(z) + \Pi_6(z) + \Pi_7(z). \tag{3.58}$$

The PGF of the number of customers in the orbit $(K_0(z))$ *is obtained using*

$$K_o(z) = \Pi_0 + \Pi_1 + \Pi_2(z) + \Pi_3(z) + \Pi_4(z) + \Pi_5(z) + \Pi_6(z) + \Pi_7(z).. \tag{3.59}$$

3.4 Performance Measures

Our analysis is based on the following system characteristics of the retrial queueing system.

3.4.1 System state probabilities

1. The steady state probability that the server is idle during the retrial time is given by

$$\mathbf{I} = \Pi_2(1) = \frac{\Pi_0\lambda q\,(1-\Psi_1(\lambda))}{\beta\omega^2\,(\rho-\Psi_1(\lambda))}$$
$$\times\left\{-\beta\lambda\,(\Psi_3(\omega)-1) + \beta\omega^2\,(\mathrm{E}(\Psi_4) + \bar{\alpha}\mathrm{E}(\Psi_5)) - \omega\,(-\beta\mathrm{E}(\Psi_6)-1)\,(\lambda\,(\Psi_3(\omega)-1)-\omega)\,(\Psi_2(\beta)-1)\right\}.$$

2. The steady state probability that the server is busy on normal service period is given by

$$\mathbf{U} = \Pi_3(1) = \frac{\Pi_0\lambda\big(1-\Psi_2(\beta)\big)}{\beta\omega^2\big(\rho-\Psi_1(\lambda)\big)}\Bigg\{-\lambda^2 q\big(\Psi_3(\omega)-1\big)$$
$$+\lambda\omega^2 q\Big(\mathrm{E}(\Psi_4)+\bar{\alpha}\mathrm{E}(\Psi_5)\Big) - \omega\big(\lambda\Psi_3(\omega)-\lambda-\omega\big)\Psi_1(\lambda)\Bigg\}.$$

3. The steady state probability that the server is busy on working vacation period is given by

$$\mathbf{V} = \Pi_4(1) = \frac{\Pi_0\lambda^2\cdot(1-\Psi_3(\omega))}{\omega^2}$$

4. The steady state probability that the server is under repair time is given by

$$\mathbf{S} = \Pi_5(1) = \Pi_0\lambda\,(1-\alpha)\,\mathrm{E}(\Psi_5)$$

5. The steady state probability that the server is under setup time is given by

$$\mathbf{K} = \mathit{\Pi}_6(1) = \Pi_0 \lambda \mathrm{E}(\Psi_4)$$

6. The steady state probability that the server is under repair time for an active breakdown is given by

$$\begin{aligned}\mathbf{R} = \mathit{\Pi}_7(1) = & \frac{\Pi_0 \lambda \left(1 - \Psi_2(\beta)\right)^2 \mathrm{E}(\Psi_6)}{\beta\omega^2 \left(\rho - \Psi_1(\lambda)\right)} \\ & - \lambda^2 q \left(\Psi_3(\omega) - 1\right) + \lambda\omega^2 q \left(\mathrm{E}(\Psi_4) + \bar{\alpha}\mathrm{E}(\Psi_5)\right) \\ & \times \omega \left(\lambda\Psi_3(\omega) - \lambda - \omega\right) \Psi_1(\lambda).\end{aligned}$$

3.4.2 Mean system size and orbit size

(i) The expected number of customers in the orbit (L_q) is obtained by differentiating equation 52 with respect to z and evaluating at $z = 1$

$$L_q = K_o'(1) = \lim_{z \to 1} \frac{d}{dz} K_o(z)$$

(ii) The expected number of customers in the system (L_s) is obtained by differentiating equation 51 with respect to z and evaluating at z $= 1$

$$L_s = K_s'(1) = \lim_{z \to 1} \frac{d}{dz} K_s(z)$$

(iii) The average time a customer spends in the system (W_s) and the average time a customer spends in the queue (W_q) are found using Little's formula

$$W_s = \frac{L_s}{\lambda_1} \quad \text{and} \quad W_q = \frac{L_q}{\lambda_1}.$$

where $\lambda_1 = \lambda q \left(\Pi_3(1) + \Pi_4(1) + \Pi_5(1) + \Pi_6(1) + \Pi_7(1)\right)$

3.5 Cost and Optimization Analysis

To optimize costs, the retrial queueing system's configurations must be carefully analyzed. It's not just about one thing you've got to consider stuff like the service rate and other important

numbers that make the system work well. In this section, we desire to draw out the Most efficient values service rate μ_b and working vacation rate μ_w by By applying cost optimization techniques. this involves balancing operational efficiency and cost to ensure smooth system performance. we begin by formulating a steady-state expected cost function per unit time, where the service rate μ_b and μ_w is the decision variables. Our objective is to find the optimal values of μ_b and μ_w to minimize the expected cost function. To reach this, We need to define cost elements as follows:
$\mathcal{T}_1$: is the cost of each consomer in the system per unit of time.
$\mathcal{T}_2$: represents the cost per unit time to leave the server functioning
$\mathcal{T}_3$: Cost per service per unit time.
$\mathcal{T}_4$: represents the cost per unit time needed to prepar starting up the server.
$\mathcal{T}_5$: represents the setup cost per busy cycle.
Let $\mathcal{T}_c$ be the total expected cost per unit time of the system:

$$\mathcal{T}_c = \mathcal{T}_1 L_s + \mathcal{T}_2(1 - \Pi_0) + \mathcal{T}_3\mu_b + \mathcal{T}_4\Pi_0 + \mathcal{T}_5\lambda.$$

3.5.1 Cost Model Formulation

The recent surge in communications and computer networks has created an urgent demand for the analysis of queueing models, so as to offer suitable service and allay congestion. This can happen if the maximum number of persons reach to a cost-effective system. Hence, systems that pose some cost optimization are very helpful and recommendable. Jain et al. (2020) and Jain and Meena (2020) have recently attempted to cost analysis of a queueing model with vacation time and unreliable server. The present authors have tried to explore financial constraints in network systems through particle swarm optimization (PSO). This is the method that is able to find the best-suited solution from an extremely large solution space. Another advantage of PSO is that Different from other optimization techniques, the objective function need not be differentiable. This method has a variety of applications, some of which include signal processing, telecommunications, workflow design, and management, among others. The procedure entails initializing a certain number of candidates or particles, and subjecting them to movement within the search space, governed by constraints concerning the objective function and candidates' positions and velocities. Each particle is assigned a fitness value that is compared to obtain personal best (pbest) and global best (gbest) values. Any particle having a pbest that is superior to gbest becomes the new global best. This process keeps on repeating up to the required iteration limit.

3.5.2 Application of Particle Swarm Optimization (PSO)

To conduct the numerical analysis for parameter optimization in the queueing system under consideration, we use the following default values for the parameters: $\alpha = 0.7, \lambda = 2, \delta = 8, \mu_w = 2, s = 5, r = 4$ and $\xi = 8$.

From **Figure** 3.1, The curves clearly show convexity, which means , the existence of a specific μ_b and μ_w that minimizes the total expected cost function for the given set of model parameters. we observe that the minimum expected operating cost per unit time converges to the solution $H = 42.88825$ at $(\mu_b^*, \mu_w^*) =$ (9.51 , 5),

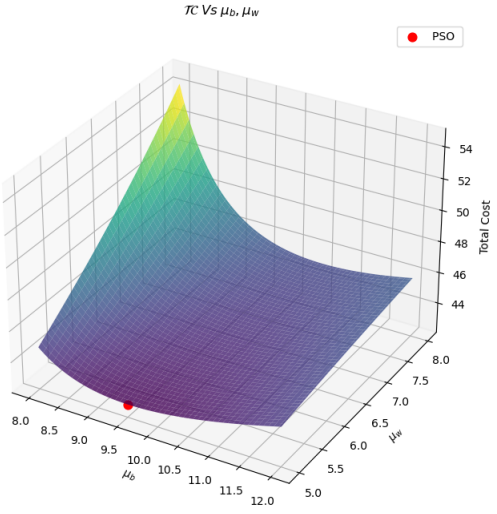

Figure 3.1: Optimality of the cost function using PSO

3.5.3 Convergence Analysis of PSO

In queue models for cost optimization, convergence means that an optimization algorithm will iteratively change the system parameters in order to minimize a given cost function.The convergence plots for cost function presented in **Figure** 3.2 show rapid decrease in best cost during the first few iterations, implying that the PSO algorithm is strong in quickly obtaining near-optimal solutions. Continuing the pre-defined number of iterations, it stabilizes, implying convergence to a global or near-global minimum.

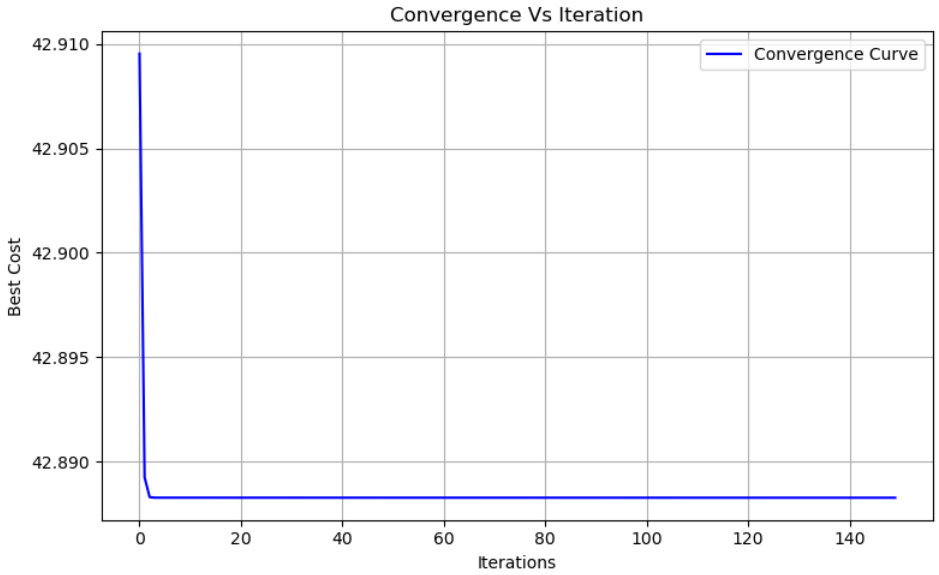

Figure 3.2: Convergence of the cost function using PSO

3.6 Sensitivity Analysis and Numerical Examples

This part offers Python-based numerical examples showing how various factors influence system performance measures. We assume that retrial times, service times, lower-speed service times, vacation times, setup times, repair times and repair times for an active breakdown all follow exponential distributions with parametres $\xi, \mu_b, \mu_w, \omega, s, r, \eta$ respectively . The parameter values are chosen to satisfy the system's stability conditions. The following tables present computed values for various model characteristics, such as the probability that the server is idle (Π_0), the mean orbit size (L_q), probability that server in working vacation , probability that server in setup time, and probability that server in repair time . The exponential distribution is $k(x) = ve^{-vx}, x > 0$.

Figure 3.3 , Illustrate that Π_0 decreases as λ increases. in addition, for a fixed λ , Π_0 decreases with μ_b . This is because when the service is fast, the server is more likely to become idle.

Figure 3.4 shows that for a fixed μ_b the probability of serer become idle decreases this happens because if the arrival rate of negative customers increases(β) rises, They disturb the server. This causes the server fail and go for repair, thus accumulating positive customers and resulting in the probability of the server becoming empty decreasing. **Figure** 3.5 shows

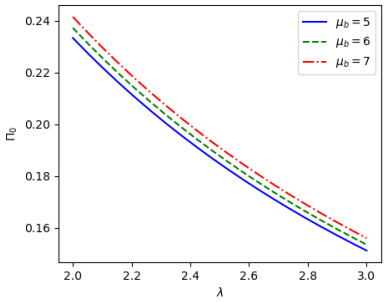

Figure 3.3: Variation in Π_0 versus λ for different values of μ_b

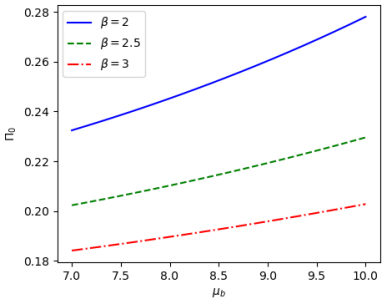

Figure 3.4: Variation in Π_0 versus μ_b for different values of β

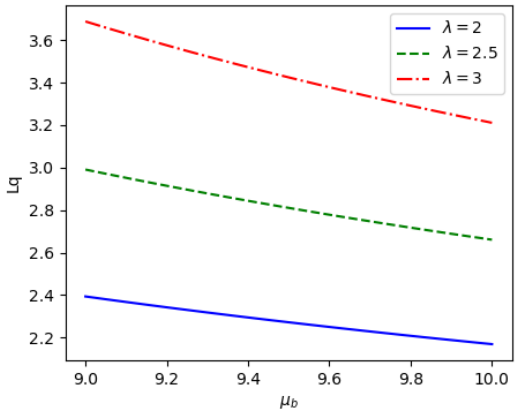

Figure 3.5: Variation in L_q versus μ_b for different values of λ

the mean orbit size L_q increases with μ_b and decreases with λ . Because if the server's work frequency increases, many customers are served, and this makes their number decrease .Unlike if the frequency of entry of positive customers increases.

3.7 Optimal strategy analysis

In this section, we introduce the reward-cost structure and focus on balanced customer enrollment strategies (at equilibrium) and the issue of social benefit maximization. After the end of the service, each customer receives a reward of $\mathcal{R}$, each customer also pays the cost of staying in the system, as the cost of waiting in the system per unit time is $\mathcal{C}$. All clients are risk neutral and act rationally with the aim of maximizing their benefits. According to the reward-cost structure shown above, the expected benefit to the individual (referred client) who finds the server

unavailable and decides to enter the orbit is:

$$\mathcal{U}(q) = \mathcal{R} - \mathcal{C}W_q$$

When customers arrive at the system, they judge whether to join based on what they gain or lose. From the individual utility function, we can find the customers' equilibrium joining probability q_e as follows.

(i) If $\mathcal{U}(1) > 0$, then the equilibrium strategy is $q_e = 1$, this means that when customers choose to join the system, their expected individual utility is positive.

(ii) If $\mathcal{U}(0) < 0$, then the equilibrium strategy is $q_e = 0$, which means that when customers decide to join the system, their expected individual utility is negative.

(iii) In addition, the necessary and sufficient condition for $q_e \in (0, 1)$ to be an equilibrium joining probability is that $\mathcal{U}(q_e) = 0$, this means that when customers choose to join the system, their expected individual utility is zero.

We continue with the problem of maximizing the social welfare per time unit. The social welfare per time unit is

$$\mathcal{S}(q) = \lambda^* \mathcal{R} - \mathcal{C}L_q$$

where λ^* is the effective arrival rate of customers, and

$$\lambda^* = \lambda \Pi_2(1) + \lambda q \left(\Pi_3(1) + \Pi_4(1) + \Pi_5(1) + \Pi_6(1) + \Pi_7(1)\right)$$

From the above equation $\mathcal{S}(q)$, a socially optimal strategy q_s is determined, which maximizes the social benefit per time unit.

Because the expressions obtained for individual and social benefits are very complex, it is hard to get specific results through traditional calculations. In the following analysis, we can use the particle swarm algorithm (PSO algorithm) to find the numerical solve this problem. When it comes to PSO algorithm, the most significant advantage is that it does not require too much analytic property of the objective function. It is an optimization algorithm based on swarm intelligence theory. During each iterative search, the particles in the swarm can dynamically adjust their position and velocity by tracking the two extremes of the swarm: the optimal solution P-best found by the particle itself and the optimal solution G-best found by the swarm. Through many iterations, the global optimal solution can be obtained.

In **Figure** 3.6 , As λ increases the probability q_e and q_s decreases . Whereas the numbers of cunsomers increasing in the system leads the orbital load will increase . This makes customers less willing to join, due to the lower chance of getting the service quickly.

Figure 3.7 illustrates that the probabilities q_e and q_s rise with η . The repair rate rise , in this

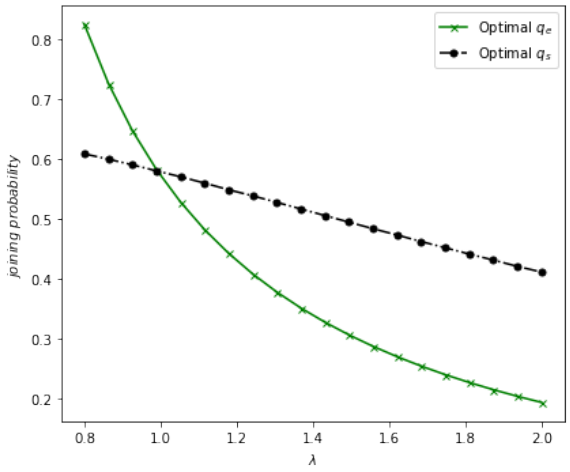

Figure 3.6: Effect of λ on q_e and q_s

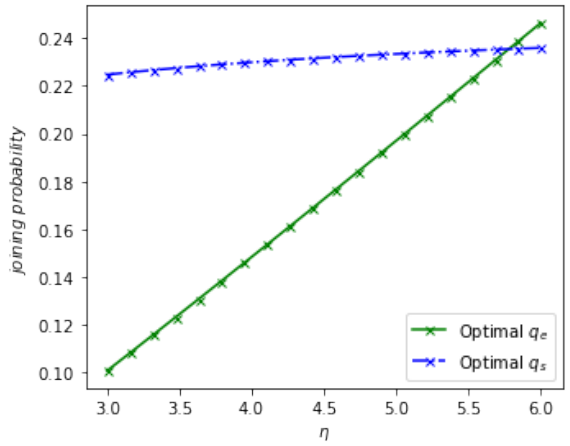

Figure 3.7: Effect of η on q_e and q_s

case, the server is quick to serve customers.

Figure 3.8 that the duo q_e and q_s increase with respect μ_b . This is because faster service means that customers are served faster. which encourages new customers to enter the orbit in anticipation of later service.

We observe from **Figure** 3.9 that both q_e and q_s drop as the negative arrival rate increases. This is because negative customers cause server disruptions, leading to longer waiting times, so customers decide not to join the system. **Figure** 3.10 presents that the welfare maximum increases as the value of the reward $\mathcal{R}$ increases. Conversely, welfare decreases as the cost of waiting $\mathcal{C}$ increases since high costs discourage customers from participating and weaken the overall benefit of the system.

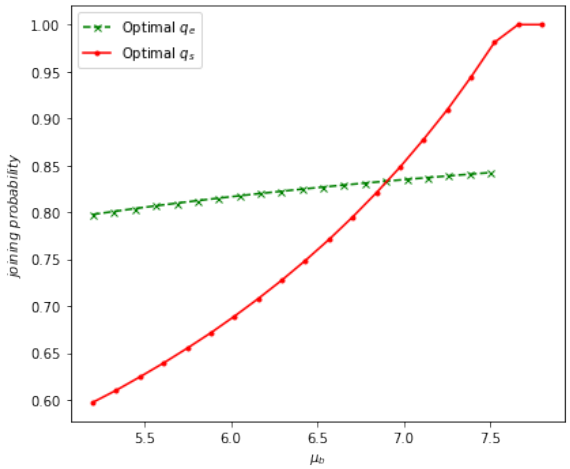

Figure 3.8: q_e and q_s as functions of μ_b

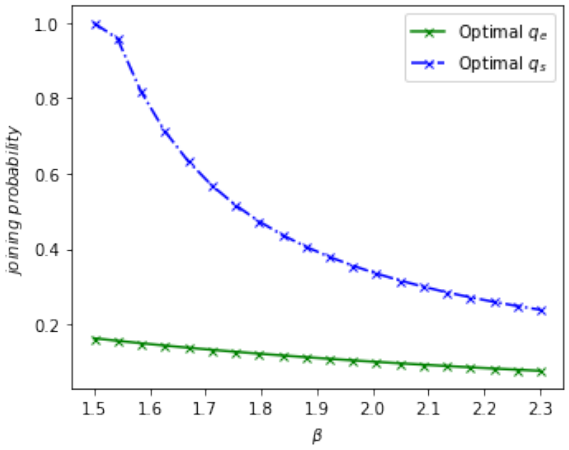

Figure 3.9: Effect of β on q_e and q_s

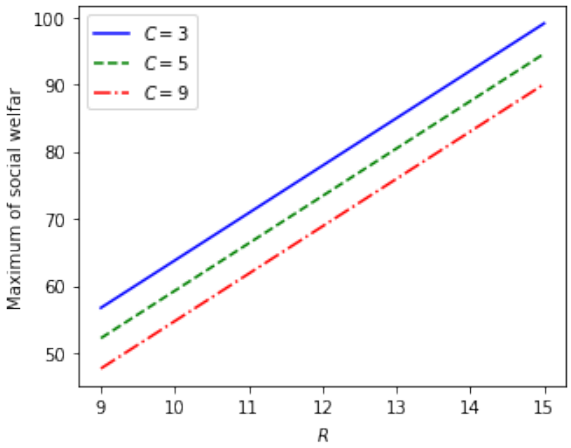

Figure 3.10: Maximum of social welfare versus $\mathcal{R}$ for different values of $\mathcal{C}$

Figure 3.11 displays As the value α increases, the server is more likely to be up and running successfully when a customer arrives, reducing failures and improving system performance. On the other hand, a higher repair rate r indicates that when the server fails, it is fixed more quickly , reducing downtime. For this reason, social welfare functions increase as both α and r, because the system becomes more up and provides better service.

Figure 3.12 shows the relationship between the social welfare and the positive arrival rate λ for different service working vacation rate μ_w . It presents that the social welfare increases with respect to the λ and μ_w. The shorter the λ and lower service time, the more customers are willing to enter the system, promoting the growth of social benefit.

Figure 3.13 displays that the growth of both η and ξ can increase the maximum social

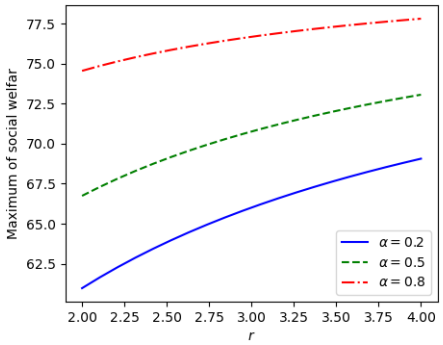

Figure 3.11: Maximum of social welfare versus r for different values of α

welfare. In this case, this reduces the waiting, which encourages customers to join the system. Therefore, the social benefit function increases.

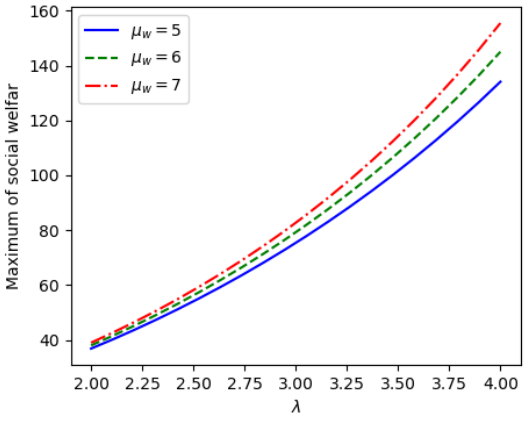

Figure 3.12: Maximum of social welfare versus λ for different values of μ_w.

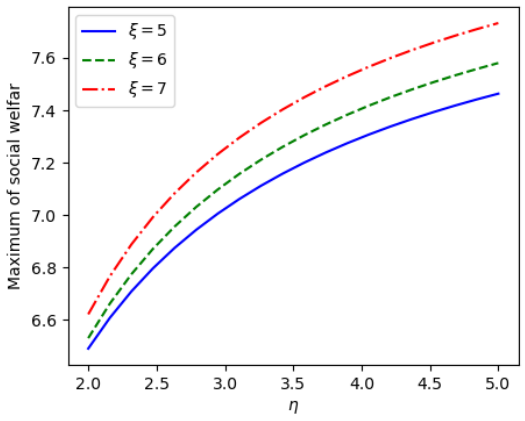

Figure 3.13: Maximum of social welfare versus η for different values of ξ.

General Conclusion and Future Work

This thesis has presented a comprehensive study of advanced retrial queueing systems incorporating features such as setup times, working vacations, unreliable servers, and the presence of negative customers. Through rigorous steady-state analysis and the use of supplementary variable techniques, the models were characterized, and explicit expressions for key performance measures were derived. The inclusion of reliability considerations and cost optimization frameworks further enriched the practical applicability of the results. Numerical examples demonstrated the significant impact of system parameters on performance metrics, revealing critical insights into system design and management. Moreover, the use of optimization methods, including quadratic fit search and particle swarm optimization, allowed for the identification of optimal operating conditions that balance system performance and operational costs. The models developed in this thesis can be extended to more complex systems involving multiple servers, different retrial policies, or more general service time distributions. Future research may also explore the integration of game-theoretic approaches to better understand strategic behavior of customers in retrial queueing environments. Overall, this work contributes both theoretically and practically to the field of retrial queueing theory, providing a solid foundation for ongoing investigations and real-world applications.

Bibliography

[1] Allen, A.O. (1978). *Probability, Statistics, and Queueing Theory with Computer Science Applications*. Academic Press. 1

[2] Arrar, N., Djellab, N., and Baillon, J.-B. (2012). On the asymptotic behavior of M/G/1 retrial queue with batch arrivals and impatience phenomenon. *Mathematical and Computer Modeling*, 55, 654–665. 2

[3] Arrar, N., Djellab, N., and Baillon, J.-B. (2017). On the stochastic decomposition of single server retrial queueing systems. *Turkish Journal of Mathematics*, 41(4), 918–932. 2

[4] Arrar, N., Derouiche, L., and Djellab, N. (2018). On the asymptotic behaviour of M/G/1 retrial queue with priority customers, Bernoulli schedule and general retrial times. *International Journal of Applied Mathematics*, 48(2), 206–213. 2

[5] Arivudainambi, D., Godhandaraman, P., and Rajadurai, P. (2014). Performance analysis of a single server retrial queue with working vacation. *OPSEARCH*, 51, 434–462. 2

[6] Artalejo, J.R. (1999). Accessible bibliography on retrial queues. *Mathematical and Computer Modelling*, 30(1–2), 1–6. 1

[7] Artalejo, J.R. and Gómez-Corral, A. (2008). *Retrial Queueing Systems: A Computational Approach*. Springer. 2

[8] Arumuganathan, R. and Jeyakumar, S. (2005). Steady state analysis of a bulk queue with multiple vacations, setup times with N-policy and closedown times. *Applied Mathematical Modelling*, 29(10), 972–986. 8

[9] Atencia, I. and Moreno, P. (2005). A discrete-time Geo/G/1 retrial queue with general retrial times. *Queueing Systems*, 48(1–2), 5–21. 8

[10] Avrachenkov, K. and Morozov, E. (2010). Stability Analysis of GI/G/c/K Retrial Queue with Constant Retrial Rate. arXiv preprint arXiv:1007.1548. 13

[11] Baba, Y. (1986). On the M^X/G/1 queue with vacation time. *Operations Research Letters*, 5(2), 93–98. 8

[12] Baba, Y. (2005). Analysis of a GI/M/1 queue with multiple working vacations. *Operations Research Letters*, 33(2), 201–209. 9

[13] Banik, A.D., Gupta, U.C., and Pathak, S.S. (2007). On the GI/M/1/N queue with multiple working vacations-analytic analysis and computation. *Applied Mathematical Modelling*, 31, 1701–1710. 2

[14] Barad, H., Achterberg, J., Chou, T.P., and Yu, J. (2024). Accelerated AI Inference via Dynamic Execution Methods. arXiv preprint. Available at: `https://arxiv.org/abs/2411.00853`. 9

[15] Bischof, W. (2001). Analysis of M/G/1 queues with setup times and vacations under six different service disciplines. *Queueing Systems*, 39(4), 265–301. 4

[16] Boualem, M. (2020). Stochastic analysis of a single server unreliable queue with balking and general retrial time. *RUDN Journal of Mathematics, Information Sciences and Physics*, 28(4), 373–384. 13

[17] Chandrasekaran, V.M., Indhira, K., Saravanarajan, M.C., and Rajadurai, P. (2016). A survey on working vacation queueing models. *International Journal of Pure and Applied Mathematics*, 106, 33–41. 2

[18] Chang, J. and Wang, J. (2018). Unreliable M/M/1/1 retrial queues with set-up time. *Quality Technology & Quantitative Management*, 15(5), 589–601. 4

[19] Choi, B.D., Kulkarni, V.G., and Gautam, N. (1995). M/M/1 Retrial Queue with Variable Service Rates. *Queueing Systems*, 21(1–2), 153–164. 1

[20] Choi, B.D., Kim, Y.C., and Lee, Y.W. (1998). The M/G/1 retrial queue with Bernoulli schedule. *Queueing Systems*, 21(3–4), 401–419. 10

[21] Choudhury, G. (2002). A batch arrival queue with a vacation time under single vacation policy. *Computers & Operations Research*, 29(14), 1941–1955. 6

[22] Choudhury, G. and Deka, M. (2012). An M/G/1 unreliable retrial queue with orbital search of customers at service completion. *Applied Mathematics and Computation*, 218(23), 11562–11572. 10

[23] Choudhury, G. and Ke, J.C. (2012). A batch arrival unreliable queue with Bernoulli vacation schedule under multiple vacation policy. *Applied Mathematics and Computation*, 218(23), 11562–11572. 8

[24] Chuong, D.T., Cuong, H.L., Long, H.D., and Hung, D.D. (2023). An Efficient Method to Compute the Rate Matrix for Multi-Server Retrial Queues with Cloud Computing Systems. International Journal of Computer Networks & Communications (IJCNC), 15(1), 53–70. 13

[25] Cox, D.R. (1955). The use of supplementary variables in the analysis of stochastic processes. *Journal of the Royal Statistical Society: Series B*, 17(1), 91–105. 10

[26] Cox, D.R. (1965). Some Statistical Methods Connected with Series of Events. *Journal of the Royal Statistical Society: Series B (Methodological)*, 27(2), 129–157. 2

[27] Di Tommaso, P., Ewels, P., et al. (2025). Empowering bioinformatics communities with Nextflow and nf-core. *Genome Biology*, 26(1), 34. 14

[28] Dimitriou, I. (2018). A two-class queueing system with constant retrial policy and general class dependent service times. arXiv preprint arXiv:1802.07451. 13

[29] Erlang, A.K. (1909). Solution of Some Problems in the Theory of Probabilities of Significance in Automatic Telephone Exchanges. Copenhagen. 1

[30] Erlang, A.K. (1917). The Calculation of Time Delays in an Automatic Telephone Exchange. Copenhagen. 1

[31] Falin, G.I. (1990). On sufficient conditions for ergodicity of multichannel queueing systems with repeated calls. *Advances in Applied Probability*, 18(1), 193–203. 1

[32] Falin, G.I. and Templeton, J.G.C. (1997). *Retrial Queues*. Chapman and Hall, London, UK. 2

[33] Falin, G.I. and Templeton, J.G.C. (1993). *Stochastic Models of Computer and Communication Systems*. Oxford University Press. 1

[34] Gandhi, A., Doroudi, S., Harchol-Balter, M., and Scheller-Wolf, A. (2014). Exact analysis of the M/M/k/setup class of Markov chains via recursive renewal reward. *Queueing Systems*, 77(2), 177–209. 4

[35] Gao, S. and Wang, J. (2014). Performance and reliability analysis of an M/G/1-G retrial queue with orbital search: A soft computing approach. *Neural Computing and Applications*, 24(7), 1621–1632. 5

[36] Gomez-Corral, A. (1999). Stochastic analysis of a single server retrial queue with general retrial time. *Naval Research Logistics*, 46(5), 561–581. 2

[37] Gupta, P. and Kumar, N. (2021). Performance Analysis of Retrial Queueing Model with Working Vacation, Interruption, Waiting Server, Breakdown, and Repair. *Journal of Scientific Research*, 13(3), 833–844. doi: `http://dx.doi.org/10.3329/jsr.v13i3.52546`. 2

[38] Gupta, P. and Kumar, N. (2021). Study of feedback retrial queueing system with working vacation, setup time and perfect repair. *Ratio Mathematica*, 41, 291–307. 2

[39] Harini, R. and Indhira, K. (2024). Dynamical behavior and metaheuristic optimization of a hospital management software system. *Heliyon*, 10(16), e35937. 12

[40] Henderson, W. (1972). The supplementary variable technique and its applications to probability theory. *Journal of Applied Probability*, 9(3), 561–571. 10

[41] Manoharan, P. and Jeeva, T. (2020). Impatient customers in a Markovian queue with Bernoulli schedule working vacation interruption and setup time. *Applications and Applied Mathematics*, 15(2), 725–739. 2

[42] Ke, J.C. (2007). Operating characteristic analysis on the M^X/G/1 system with standby spare, startup and breakdown. *Applied Mathematical Modelling*, 31(3), 482–496. 7

[43] Ke, J.C. and Chang, F.M. (2009). Modified T vacation policy for a batch arrival queue with server breakdowns and multiple vacations. *Applied Mathematical Modelling*, 33(8), 3543–3557. 8, 9

[44] Keilson, J. and Servi, L.D. (1986). Oscillating random walk models for GI/G/1 vacation systems with Bernoulli schedule. *Journal of Applied Probability*, 23(3), 790–802. 2

[45] Keilson, J. and Servi, L.D. (1986). Oscillating Random Walk Models for G1/G/1 Vacation Systems with Bernoulli Schedules. *Journal of Applied Probability*, 23(1), 790–802. 7

[46] Keilson, J. and Kooharian, A. (1960). On time dependent queuing processes. *Annals of Mathematical Statistics*, 31(1), 104–112. 10

[47] Keilson, J., Nunn, W.R., and Sumita, U. (1968). A generalization of the M/G/1 queue. *Operations Research*, 16(3), 612–619. 10

[48] Kim, T., Lee, S., Choi, H., Park, H.S., and Choi, J. (2023). An Energy-Efficient Multi-Level Sleep Strategy for Periodic Uplink Transmission in Industrial Private 5G Networks. Sensors, 23(22), 9070. 9

[49] Kleinrock, L. (1975). *Queueing Systems, Volume I: Theory*. Wiley. 1

[50] Kleinrock, L. (1976). *Queueing Systems, Volume II: Computer Applications*. Wiley. 1

[51] Krishna Reddy, G.V., Nadarajan, R., and Arumuganathan, R. (1998). Analysis of a bulk queue with N-policy multiple vacations and setup times. *Computers & Operations Research*, 25(11), 957–967. 8

[52] Krishnakumar, B., Vijayakumar, A., and Sophia, S. (2013). An M/G/1 retrial queueing system with two phases of service and vacation. *Journal of Industrial and Management Optimization*, 9(1), 171–181. 5

[53] Kulkarni, V.G. and Liang, H. (1997). Retrial queues revisited. *Communications in Statistics: Stochastic Models*, 13(1), 167–189. 1

[54] Laxmi, P.V., Goswami, V., and Jyothsna, K. (2013). Optimization of balking and reneging queue with vacation interruption under N-policy. *Journal of Optimization*, 683708, 9. 30

[55] Lee, H.W., Lee, S.S., Park, J.O., and Chae, K.C. (1994). Analysis of the M^X/G/1 Queue with N-Policy and Multiple Vacations. *Journal of Applied Probability*, 31, 465–480. 2

[56] Lee, H.W., Lee, S.S., Park, J.O., and Chae, K.C. (1995). Analysis of the M^X/G/1 queue with N-policy and server vacations. *Journal of Applied Probability*, 32(3), 671–683. 6

[57] Lee, H.S. and Srinivasan, M.M. (1989). Control policies for the M^X/G/1 queueing system. *Management Science*, 35(6), 708–720. 8

[58] Leonard, T.C. (1975). An application of Markov chain methods to denumerable state space queueing problems. *Operations Research*, 23(5), 934–945. 10

[59] Levy, Y. and Yechiali, U. (1975). Utilization of idle time in an M/G/1 queueing system. *Management Science*, 22, 202–211. doi:10.1287/mnsc.22.2.202. 2

[60] Levy, H. and Kleinrock, L. (1986). A queue with starter and a queue with vacations: delay analysis by decomposition. *Operations Research*, 34(3), 426–436. 4

[61] Li, J. and Tian, N. (2007). The M/M/1 queue with working vacations and vacation interruptions. *Journal of Systems Science and Systems Engineering*, 16, 121–127. 2

[62] Li, H. and Yang, T. (1995). An M/G/1 retrial queue with priority calls. *Performance Evaluation*, 24(4), 277–285. 10

[63] Liu, W., Xu, X., and Tian, N. (2007). Stochastic decompositions in the M/M/1 queue with working vacations. *Operations Research Letters*, 35(5), 596–601. 9

[64] Liu, D., Zeng, P., Cui, S., and Song, C. (2023). Deep Reinforcement Learning for Charging Scheduling of Electric Vehicles Considering Distribution Network Voltage Stability. Sensors, 23(3), 1618. `https://doi.org/10.3390/s23031618`. 8

[65] Madan, K.C. (2000). An M/G/1 queue with second optional service. *Queueing Systems*, 34(1), 37–46. 7

[66] Madan, K.C. (2001). On a batch arrival Poisson queue with a random breakdown. *International Journal of Information and Management Sciences*, 12(3), 35–46. 7

[67] Madan, K.C. and Abu Al-Rub, K.A. (2004). A two-phase optional M^X/G/1 queue with a single vacation policy. *Information and Management Sciences*, 15(1), 1–17. 7

[68] Madan, K.C. and Al-Rawwash, M. (2005). On the M^X/G/1 queue with feedback and optional server vacations. *Applied Mathematics and Computation*, 161(3), 925–939. 7

[69] Malik, G., Upadhyaya, S., and Sharma, R. (2021). Cost inspection of an M/G/1 retrial model using particle swarm optimization and genetic algorithm. *Ain Shams Engineering Journal*, 12(4), 2241–2254. 12

[70] Manoharan, P. and Jeeva, T. (2019). Impatient customers in an M/M/1 working vacation queue with a waiting server and setup time. *Journal of Computer and Mathematical Sciences*, 10(5), 1189–1196. 2

[71] Medhi, J. (2003). *Stochastic Models in Queueing Theory*. Academic Press. 10

[72] Neuts, M.F. (1984). *Matrix-Geometric Solutions in Stochastic Models*. Johns Hopkins University Press. 1

[73] Parthasarathy, P.R. and Sharafali, M. (1987). A constructive approach to the theory of stochastic processes. *Operations Research Letters*, 6(4), 197–203. 1

[74] Peng, Y., Liu, Z., and Wu, J. (2014). An M/G/1 retrial G-queue with preemptive resume priority under N-policy. *Journal of Computational Information Systems*, 10(12), 5197–5205. 6

[75] Phung-Duc, T. (2015). M/M/1/1 retrial queues with setup time. *Advances in Intelligent Systems and Computing*, 383, 93–104. 2

[76] Phung-Duc, T. (2017). Single server retrial queues with setup time. *Journal of Industrial and Management Optimization*, 13(3), 1329–1345. doi:10.3934/jimo.2016075. 2, 4

[77] Rardin, R.L. (1997). *Optimization in Operations Research*. Prentice-Hall, Upper Saddle River. 30

[78] Sáez-Rodríguez, J., et al. (2021). Design considerations for workflow management systems use in production genomics research and the clinic. *Nature Communications*, 12(1), 5975. 14

[79] Servi, L.D. and Finn, S.G. (2002). M/M/1 queues with working vacations (M/M/1/WV). *Performance Evaluation*, 50, 41–52. doi:10.1016/S0166-5316(02)00057-3. 2

[80] Servi, L.D. and Finn, R. (2002). M/M/1 Queues with Working Vacations (M/M/1/WV). *Performance Evaluation*, 59(1), 1–15. 9

[81] Sikdar, K. and Gupta, U.C. (2005). Queue length distributions in a batch service queue with multiple vacations. *Quality Technology & Quantitative Management*, 2(1), 75–89. 8

[82] Taylor, I.J. and Deelman, E. (2020). Managing Failures in Task-Based Parallel Workflows in Distributed Computing Environments. In *Workflows in Support of Large-Scale Science*, Springer, pp. 411–426. 14

[83] Takagi, H. (1991). *Queueing Analysis: Vacation and Priority Systems, Volume I*. North-Holland, Amsterdam. 2

[84] Takagi, H. (1988). *Queuing Analysis: A Foundation of Performance Evaluation*. North-Holland. 1

[85] Tian, N.S., Li, J.H., and Zhang, Z.G. (2009). Matrix analytic method and working vacation queues-a survey. *International Journal of Information Management Science*, 20, 603–633. 2

[86] Tian, N. and Zhang, Z.G. (2002). The discrete-time GI/Geo/1 queue with multiple vacations. *Queueing Systems*, 40(3), 247–270. 8

[87] Upadhyaya, S. (2020). Cost optimization of a discrete-time retrial queue with Bernoulli feedback and starting failure. *International Journal of Industrial and Systems Engineering*, 36(2), 165–196. 11

[88] Wang, J., Cao, J., and Li, W. (2001). Reliability analysis of M/G/1 queueing systems with server breakdowns and repairs. *Journal of Systems Science and Mathematical Sciences*, 21(2), 184–190. 9

[89] Wang, J. and Li, J. (2008). A repairable M/G/1 retrial queue with Bernoulli vacation and two-phase service. *Quality Technology & Quantitative Management*, 5(2), 179–197. 8, 9

[90] Wang, J. and Zhang, P. (2009). A discrete-time retrial queue with negative customers and unreliable server. *Applied Mathematical Modelling*, 33(8), 3616–3629. 5

[91] Wang, J. and Zhou, Q. (2010). An M^X/G/1 retrial queue with active breakdowns and general retrial times. *Mathematical and Computer Modelling*, 51(5–6), 421–431. 10

[92] Wu, J. and Lian, Z. (2013). A single-server retrial queue with priority customers and disasters. *Acta Applicandae Mathematicae*, 124(1), 1–20. 5

[93] Wu, D. and Takagi, H. (2006). M/G/1 queue with multiple working vacations. *Performance Evaluation*, 63(7), 654–681. 9

[94] Yang, T. and Li, H. (1994). The M/G/1 retrial queue with the server subject to starting failures. *Queueing Systems*, 16(1–2), 83–96. 9

[95] Yechiali, U. (1993). *Queues with Server Vacations and Related Topics*. Technion-Israel Institute of Technology. 1

[96] Yuh, Z., Yang, D.Y., and Wu, C.H. (2015). A multi-server retrial queueing system with Bernoulli feedback and starting failure. *American Journal of Operations Research*, 5(3), 191–208. 10

[97] Zhang, M. and Hou, Z. (2010). Performance analysis of M/G/1 queue with working vacations and vacation interruption. *Journal of Computational and Applied Mathematics*, 234, 2977–2985. 2

[98] Zhang, M. and Hou, Z. (2012). Steady-state analysis of an M/G/1 queue with single working vacation. *International Journal of Computer Mathematics*, 89(13–14), 1894–1905. 9

[99] Zhang, Y. and Liu, B. (2015). An M/G/1 retrial G-queue with non-exhaustive random vacations. *Journal of Systems Science and Complexity*, 28(3), 559–579. 5

[100] Zhang, X.L., Wang, J.T., and Ma, Q. (2017). Optimal design for a retrial queueing system with state dependent service rate. *Journal of Systems Science and Complexity*, 30(2), 883–900. 12

[101] Zhang, H., Liu, N., Chu, X., Long, K., Aghvami, A.H., and Leung, V.C. (2020). Network slicing based 5G and future mobile networks: mobility, resource management, and challenges. *IEEE Communications Magazine*, 58(8), 12–18. 15