ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

INGEGNERIA BIOMEDICA, ELETTRICA E DEI SISTEMI

Ciclo 38

Settore Concorsuale: 01/A6 - RICERCA OPERATIVA

Settore Scientifico Disciplinare: MAT/09 - RICERCA OPERATIVA

A DISCRETE EVENT SIMULATION AND DEEP REINFORCEMENT LEARNING FRAMEWORK FOR QUEUE MANAGEMENT AND PATIENT PRIORITIZATION IN AN EMERGENCY DEPARTMENT

Presentata da: Luca Zattoni

Coordinatore Dottorato

Michele Monaci

Supervisore

Enrico Malaguti

Co-supervisore

Paolo Tubertini

Esame finale anno 2026

Abstract

Department of Electrical, Electronic, and Information Engineering
"Guglielmo Marconi" - DEI
PhD Programme in Biomedical, Electrical and System Engineering - IBES
Automatic Control and Operational Research Curriculum

A Discrete Event Simulation and Deep Reinforcement Learning Framework for Queue Management and Patient Prioritization in an Emergency Department

By *Luca Zattoni*

The Emergency Department (ED) represents one of the main access points to hospital care. Its complexity derives from the intrinsic high operational variability, related to the non-elective nature of the services provided. One of the most challenging and widely documented problems in this setting is overcrowding, which occurs when the demand for emergency services exceeds the capacity to provide them in a timely manner. Overcrowding, directly correlated with prolonged patient length of stay (LOS), is also strongly associated with reduced quality of care, discomfort and distress for both patients and staff.

This thesis proposes a data-driven operational decision support tool for mitigating overcrowding through adaptive patient flow management. The approach combines Discrete Event Simulation (DES), Deep Learning (DL) and Deep Reinforcement Learning (DRL) within a unified framework, and it is applied to the case study of the General ED of IRCCS Azienda Ospedaliero-Universitaria di Bologna, a large hospital in Northern Italy.

First, a DES is developed and validated as a Digital Twin of the real ED, fed with historical data that allow a realistic and dynamic representation of the system. The patient prioritization problem is then formulated as a Markov Decision Process (MDP), with the objective of reducing LOS while ensuring timely access to care. The MDP formulation is further enriched through a DL model that estimates the likelihood of each patient approaching the end of their clinical pathway.

Two Actor-Critic models are trained and evaluated within the simulated environment, and their performance is compared against the prioritization policy currently adopted in the real ED. The results indicate a reduction in both LOS and crowding levels, while taking into account clinical constraints.

These findings demonstrate the feasibility of adopting similar approaches to im-

prove ED patient flow, opening a promising research direction towards data-driven and adaptive control of emergency operations.

Acknowledgements

First, I would like to express my gratitude to my supervisor, Prof. Enrico Malaguti, for his scientific guidance and continuous support throughout the development of this thesis. Thanks to Dr. Paolo Tubertini, for giving me the opportunity to start this project, to Dr. Cristiano Fabbri, who provided the initial sparkle for this work, and to IRCCS AOU Bologna Medical Management staff. My gratitude also goes to Prof. Michele Monaci, coordinator of the doctoral programme, for making this journey possible.

Thanks also to Dr. Andrea Borghesi for his availability and the ideas provided, which helped me through the initial stage of development of the project.

Thanks to Dr. Alberto Santini and Dr. Dimitri Thomopulos for the time they spent reviewing this thesis.

A special mention goes to my companion in misfortune, Andrea Eusebi, whose presence has made the challenging moments far less lonely.

I am deeply grateful to my long-time friend Dr. Giorgio Franceschelli: although not directly involved in this project, he has always been a reference I could rely on, and more than one of the developments presented in this thesis owes much to his suggestions.

I wish to thank Prof. Luca Grieco and Prof. Daniele Catanzaro for welcoming me into their research groups –CORU and CORE respectively– during my two visiting periods abroad, which enriched these three years through meaningful exchanges with other researchers and students.

I would also like to thank all the friends and colleagues who stood by my side during these three years: your encouragement has been fundamental to bring this work to completion.

Finally, I would like to thank my girlfriend, Giulia, for her patience, understanding, and support over the past few months, and my parents for everything they've done for me, and without whom I wouldn't have come this far.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Meaning
A2C	(Synchronous) Advantage Actor-Critic
A3C	Asynchronous Advantage Actor-Critic
AI	Artificial Intelligence
AUC	Area Under the Curve
CV	Coefficient of Variation
DES	Discrete Event Simulation
DL	Deep Learning
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
DTDT	Door-to-Doctor Time
ED	Emergency Department
GCS	Glasgow Coma Scale
LOS	Length of Stay
ML	Machine Learning
MDP	Markov Decision Process
OR	Operations Research
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
ROC	Receiver Operating Characteristic

1 Introduction

Health systems governance is a topic of growing global interest, encompassing the management of all activities and resources aimed at ensuring health services are accessible, equitable, efficient, affordable, and of high quality for the entire population (World Health Organization, 2025b).

The steady increase in life expectancy and the consequent ageing of the population have led to a higher incidence of chronic diseases (Brennan et al., 2017; World Health Organization, 2025a). This demographic transition is likely to further intensify the demand for healthcare services, particularly among older people.

Modern healthcare systems show intrinsic complexity, due to the interdependencies between their components and the frequent unpredictability of their dynamics (Kannampallil et al., 2011; Plsek & Greenhalgh, 2001). Managing such complexity requires the adoption of approaches, techniques, and tools that can effectively support decision-making processes, both clinical and organizational. The wide availability of data in contemporary healthcare enabled the development of advanced analytical methods (Subrahmanya et al., 2022) in multiple settings of the health domain. These methods can be classified in four categories: *descriptive*, *diagnostic*, *predictive* and *prescriptive* analytics (El Morr & Ali-Hassan, 2019). While descriptive and diagnostic analytics are useful to describe what happened and why it happened, predictive analytics make use of available data to project predictions of what will happen. Finally, prescriptive analytics goes a step further, involving those techniques aimed at suggesting actions to address the problems identified through diagnostic analytics and produce a desired outcome.

Within this last category, Artificial Intelligence (AI) and Operations Research (OR) have emerged as key disciplines capable of transforming data into actual insight to improve decision-making and overall system performance (Al Kuwaiti et al., 2023; Rais & Viana, 2011).

1.1 Context and motivation: emergency services and Emergency Department

Health services can be divided in two categories: *elective care* and *non-elective care*. The former comprises treatments that can be postponed for days or weeks without significant medical implications, while the latter concerns an unexpected demand that needs to be planned on short notice (Gupta & Denton, 2008; Mans et al., 2015). The process of delivery of health emergency services involves several actors, with the Emergency Department (ED) –also known as Emergency Room– assuming a central role and representing the main topic of this thesis.

The ED constitutes one of the main access points to care services in hospitals worldwide. Nowadays, its mission involves not only the management of emergencies, but it also increasingly encompasses the treatment of non-urgent cases that would traditionally fall under the responsibility of general practitioners (Calnan, 1984; Steinbrook, 1996; Zaboli et al., 2024). This dual function, combined with the unpredictability of demand, makes ED operations highly complex and resource-intensive.

In Italy, the trend of ED visits remained relatively stable over the past decade (Ministero della Salute – Ufficio di Statistica, 2025). However, the progressive reduction in the number of public facilities dedicated to emergency services (Ministero della Salute – Ufficio di Statistica, 2016) has likely increased the pressure on the remaining ones. The two recent COVID-19 pandemic waves led to a temporary decline in the number of ED accesses (Golinelli et al., 2021), likely due to containment measures and fear of contagion, but volumes seemed to rapidly restore –even if not completely– right after (Mattiuzzi et al., 2025), confirming that the demand for emergency services is still a major concern.

The continuous inflow of patients, many of whom require urgent or emergency interventions, can put considerable pressure on both operational and clinical decision-making in EDs. Tackling this challenge means seeking methodologies and analytical frameworks aimed not only at the optimal allocation and dimensioning of resources –such as medical staff and equipment– but also at enhancing the efficiency of existing resources by optimizing patient pathways and prioritization mechanisms.

1.2 Case study: IRCCS AOU Bologna Emergency Department

The case study presented in this thesis focuses on the General ED of the IRCCS Azienda Ospedaliero–Universitaria di Bologna (IRCCS AOU Bologna), the largest hospital in Emilia–Romagna region, Italy (AOU Bologna, 2025). This ED, an exemplary setting due to the high volume of annual patient visits, shares common characteristics in both organization and process with most EDs worldwide (Di Leva & Sulis, 2017; Fabbri et al., 2024; Rojas et al., 2019). Its layout includes a waiting area, a triage area, and two treatment areas –major and minor intensity– dedicated to patients with different acuity levels. In addition, a temporary observation area is used for short-term monitoring. Finally, an adjacent short-stay observation unit (*Osservazione Breve Intensiva*, OBI) accommodates patients requiring up to 48 hours of observation and treatment before either home discharge or hospital admission.

1.2.1 The ED process

Throughout their stay in the ED, patients interact with multiple resources, including physicians, nurses and other clinical staff, technicians, diagnostic devices, beds and stretchers, and dedicated spaces. The ED process can be divided into three interconnected phases: (i) arrival and triage; (ii) medical visits; (iii) diagnostics and treatment. Figure 1.1 summarizes the whole process following the BPMN notation introduced by Object Management Group (OMG), 2022.

Patient arrival and Triage Patients may arrive to the ED either autonomously or by ambulance. Upon arrival, they undergo a procedure called *triage*, typically conducted by a trained nurse, in order to be assigned an appropriate level of severity, so as to regulate their access to limited resources according to a priority-based policy (Leo et al., 2016). The primary goal of triage is therefore to assess the urgency of the patient treatment, ensuring that critical cases receive immediate attention while maintaining an efficient and fair resource allocation (FitzGerald et al., 2010; Yancey & O'Rourke, 2025).

The triage procedure usually includes a brief interview about the presented symptoms, vital signs measurement, and the classification of the case into a main problem category. This allows a contextualization of the patient's clinical condition, as well as an anticipation of its potential evolution (Sapra et al., 2023; Sterling et al., 2025). Depending on the assessed problem, some patients may be redirected to other spe-

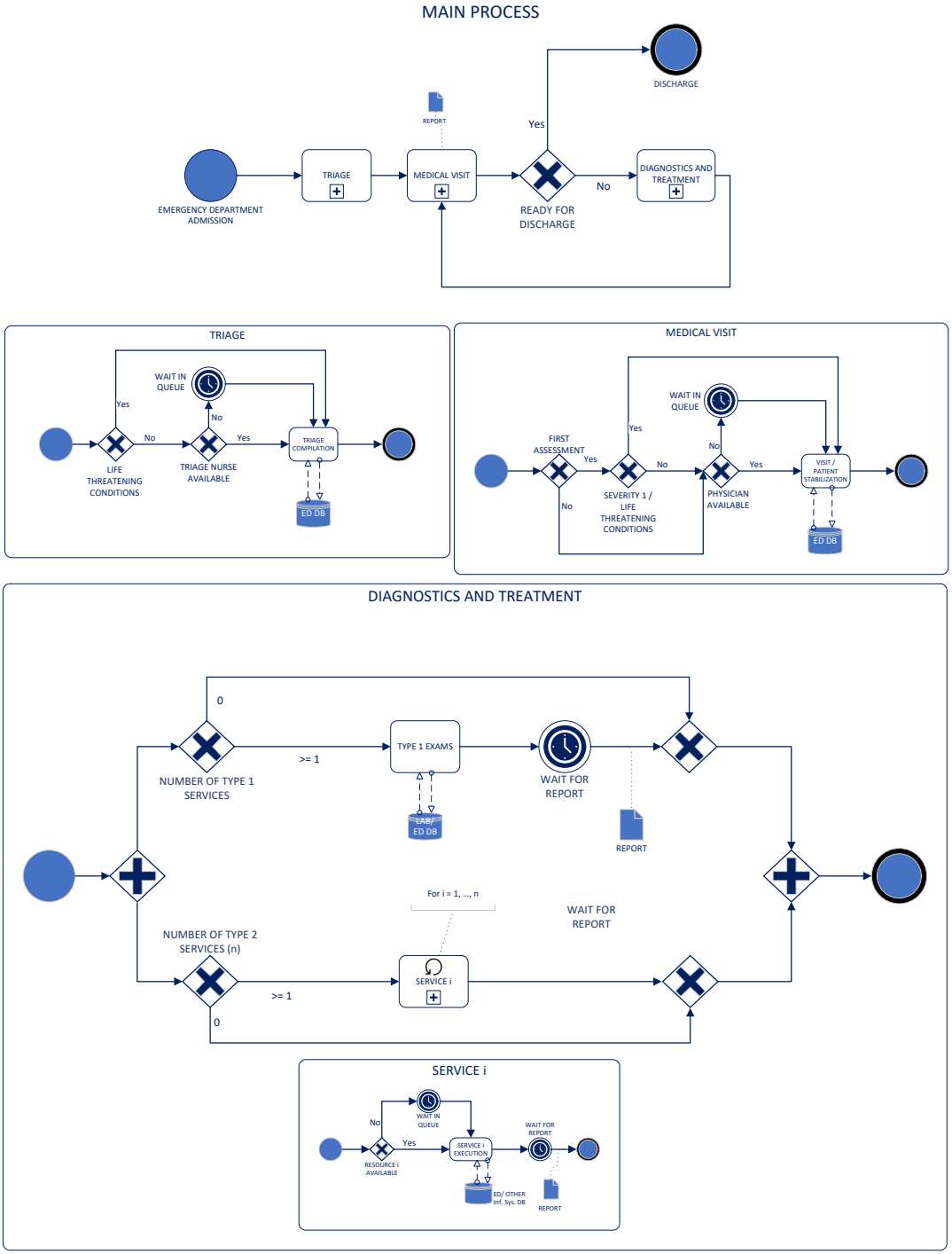

Figure 1.1: BPMN representation of the ED process

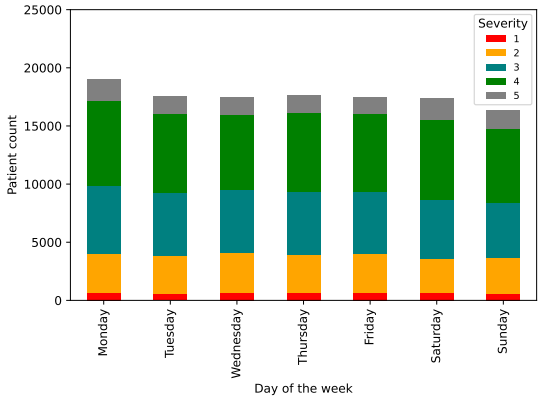

Figure 1.2: Weekly trend of patient arrivals

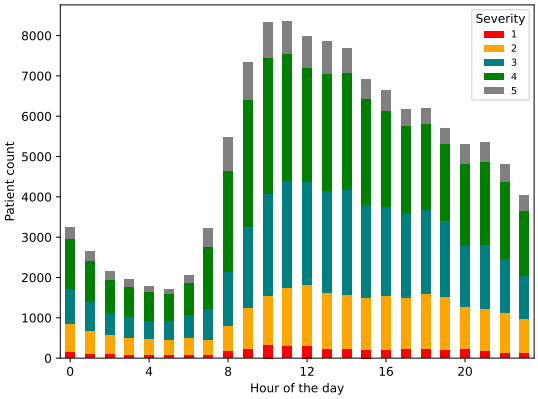

Figure 1.3: Daily trend of patient arrivals

cialized EDs within the hospital (e.g., ophthalmic or pediatric EDs) or to dedicated fast-track pathways (e.g., some orthopedic conditions).

Table 1.1 reports the distribution of the main problem categories assigned to patients between January 1, 2022, and December 31, 2023 (122,902 patients). A more detailed representation, listing all the 201 main problems included in the ED information system, is provided in Appendix Table B.1.

The triage outcome is the assignment of a specific code to the patient, according to a scale which can differ on a national or regional level (Christ et al., 2010). From mid-2021, the Emilia-Romagna Region adopted a five-level triage system, replacing the previous four-level triage (Regione Emilia-Romagna, 2021). Each severity level, identified by a color code, corresponds to a maximum waiting time before the patient receives the first physician visit (medical assessment), as summarized in Table 1.2. This time interval is often referred to as *door-to-doctor time* (DTDT). The classification in severity codes reflects the conventional distinction between emergent, urgent, and non-urgent cases (Schneider et al., 1998), based on the seriousness of the problem and on the required level of care. This system allows fair resource allocation and contributes to the containment of patients' length of stay (LOS).

Table 1.3 shows the distribution of severity codes assigned over the considered period, while Figures 1.2 and 1.3 display the weekly and daily trends of patient arrivals, stratified by severity level.

Medical visit After triage classification, patients are examined by an emergency physician, who follows them throughout their diagnostic and therapeutic pathway. The physician role consists in performing clinical assessments, prescribing diagnostic tests and treatments, and determining the final disposition according to exam

Table 1.1: Frequency of the main problem categories identified at triage. A dash (−) indicates that no patient has been assigned the problem category in the target period.

Problem category	**Frequency**
Cardiovascular	19.7%
General disorders and minor problems	15.2%
Gastrointestinal	14.2%
Neurological	12.6%
Respiratory	8.7%
Genitourinary	8.4%
Trauma	5.6%
Psychiatry	3.4%
Orthopedic	2.7%
ENT - lips, oral cavity, neck	2.5%
Dermatological	2.5%
Substance abuse	1.5%
ENT - nose	1.2%
ENT - ear	1.1%
Ophtalmology	0.7%
Environmental	0.1%
Obstetric-Gynecological	<0.1%
Social	−
Other	−

Table 1.2: Current triage system adopted in Emilia-Romagna region

Severity code	**Color**	**Correspondent urgency**	**Maximum waiting time for first medical assessment**
1	Red	Emergency	0 minutes
2	Orange	Very urgent	15 minutes
3	Light blue	Urgent	60 minutes
4	Green	Less urgent	120 minutes
5	White	Non-urgent	240 minutes

Table 1.3: Distribution of ED patients by severity level

Severity	**Frequency**
1	3.6%
2	18.5%
3	30.3%
4	38.4%
5	9.2%

reports (Brown & Cadogan, 2020; Schneider et al., 1998). With few exceptions, continuity of care is ensured by having the same physician follow the patient throughout their stay. Table 1.4 reports the distribution of the outcomes of ED accesses during the analyzed period.

For simplicity, from now on the term ***discharge*** will be used to indicate the end of the patient clinical pathway in the ED, independently of the specific outcome.

Table 1.4: Frequency of patient outcomes

Outcome	**Frequency**
To General Practitioner (ordinary discharge)	60.2%
Hospitalization	14.4%
Transfer to short-stay observation unit	11.0%
Left before medical assessment	9.4%
Left after medical assessment	1.9%
Transfer to another hospital	1.8%
Refusal of hospital admission	0.4%
Discharge to residential facility	0.3%
Death	0.2%
Discharge to outpatient facility	0.2%
Deceased before arrival	<0.1%
Transfer to another ED	<0.1%
Transfer to another hospital outside region	<0.1%

Diagnostics and treatment Depending on the clinical assessment, patients may undergo one or more diagnostic or therapeutic services, such as diagnostic imaging, specialist consultations, or laboratory exams. These can be categorized as: (i) *Type-1 services*, which do not require the patient's physical presence during execution (e.g., laboratory analyses performed on collected samples), and (ii) *Type-2 services*, which require the patient to be physically present (e.g., imaging procedures

or therapies).

This means that type-1 services can be parallelized to other activities, while type-2 services must be executed sequentially (except for their report generation, which can be parallelized as well). This conceptual separation has been formalized in similar process analyses, such as in L. Jiang and Giachetti, 2008. Once all reports are available, patients undergo a follow-up visit for re-evaluation and possibly further diagnostics or discharge decision.

Appendix figures A.1–A.5 illustrate, through heatmaps, the correlation between patient problem categories (see Table 1.1), severity levels, and prescribed services, revealing distinctive patterns. Each value represents the probability, given a severity level and a main problem category, of a patient requiring a certain service during their stay in the ED.

1.2.2 Overcrowding

Throughout the described process, patients concur for the same limited set of resources. As illustrated in Figure 1.1, life-threatening cases (severity level 1 at the time of arrival) bypass queues and receive immediate assessment, while access for all the other patients is regulated according to the severity they are assigned and their waiting time. When the demand for emergency services exceeds the ability of the ED to provide them in a timely manner, the structure may experience overcrowding (Asplin et al., 2003).

Due to its well-documented negative effects on patient conditions and staff performance (Derlet & Richards, 2000; Di Somma et al., 2015; Morley et al., 2018; Pearce et al., 2023), ED overcrowding has been recognized as a public health concern and health equity issue by the International Federation for Emergency Medicine (Javidan et al., 2021). Interest in measuring and monitoring such phenomena generated multiple indicators (Badr et al., 2022), such as NEDOCS (Weiss et al., 2004). By integrating multiple operational factors such as resource availability and patients in ED, this method returns a score on a scale from "not busy" to "dangerously overcrowded". However, its adequacy across different settings remains debated (Wang et al., 2014).

Conceptual frameworks, such as the input-throughput-output model proposed by Asplin et al., 2003, decompose crowding contributing factors into three interacting components: (i) input - demand and patient arrival, (ii) throughput - internal process efficiency, and (iii) output - discharge or hospitalization constraints.

Recent works like Savioli et al., 2022 and Sartini et al., 2022 use this kind of framework to try and propose possible countermeasures. They emphasize the central

role of throughput efficiency –particularly the optimization of diagnostic pathways– in mitigating overcrowding. Consequently, reducing the average LOS, due to its strict relation with crowding, becomes a strategic goal to improve ED performance and patient experience. In line with this, since 2019 the Emilia-Romagna region has introduced a LOS target of six hours for patient, plus one additional hour for complex cases (Regione Emilia-Romagna, 2019). Although the current triage and prioritization policy has proven reasonably effective in maintaining service accessibility, recurrent high-congestion episodes highlight the potential benefit of exploring advanced and data-driven approaches to operational management. Due to the stochastic and dynamic nature of ED process, effective solutions should rely on adaptive tools capable of continuously monitoring the system state and translating this information into real-time decisions to optimize patient flow. According to the classification proposed by Hulshof et al., 2012, such methods fall within the domain of *online operational planning*, which represents the conceptual foundation for the decision-support techniques explored in this thesis.

1.3 Research objectives and methodology

This thesis aims to address the problem of ED overcrowding through the development of an integrated Decision Support framework to improve patient flow. The main focus concerns assessing the impact of adaptive queue management and patient prioritization strategies on system performance, compared to the standard policy currently adopted, which is based on severity code and waiting time as patient selection criteria. The proposed approach considers exclusively decisions on patient priority management, without altering staff shifts or the number of available resources. In this way, it seeks to demonstrate that significant performance improvement can be achieved by optimizing the use of existing resources, rather than by increasing capacity. The primary objective is to reduce patient LOS, while ensuring that patients, after triage completion, receive the first medical assessment in a timely manner (monitoring and containment of DTDT).

Recent works like Sartini et al., 2022 and Maninchedda et al., 2023 mention how emerging AI tools can support the design and implementation of effective strategies to mitigate overcrowding. Building on these insights, the approach developed in this thesis relies on techniques derived from the field of Reinforcement Learning (RL). RL is a branch of AI and optimal control which studies how intelligent and trainable agents, by interacting with controllable systems, adapt their behavior in order to maximize an objective function according to the observation of the condition of

the system (Sutton & Barto, 2018). Given the inherent complexity and stochastic nature of ED, most classical and basic RL methods are inadequate to capture its dynamic. Therefore, this work employs Deep Reinforcement Learning (DRL), a class of RL techniques that model agents through the use of Deep Neural Networks (DNNs) (Plaat, 2022). More specifically, the study adopts Actor-Critic algorithms, which belong to the family of policy gradient methods.

The RL algorithms described in this thesis operate within a Discrete Event Simulation (DES) that emulates the real system, allowing for a reliable representation of the main dynamics of the process. The suitability of DES as an engine for RL frameworks has been highlighted in studies such as those from Capocchi and Santucci, 2022 and Belsare et al., 2022.

In addition, a DNN-based predictive model is embedded in the framework to estimate, in real time, each patient's likelihood of approaching the end of their clinical pathway (and so being discharged from the system). This prediction at the patient-level is integrated for a more detailed representation of the state of the system.

All models that will be presented have been trained and evaluated on realistic instances, obtained through historical data from the ED chosen as case study (section 1.2). The motivation is therefore not merely theoretical, but rather coherent with the development of a tool with potential practical relevance.

All the computational experiments were conducted using Python 3.9.7 language on a server with an AMD Ryzen 7 2700X 8-core processor, 64GB RAM, under the Ubuntu Linux operating system.

The following section provides an overview of the available literature on the use of OR and AI methods for ED process management, highlighting the current gaps and positioning the proposed work within this research landscape.

1.4 Literature review and main contributions

The use of OR techniques, and in particular DES, has grown significantly in the analysis and improvement of ED processes, as well as in broader applications in the healthcare domain. This diffusion is due to the ability of these methods to effectively capture the correlation between system inputs (e.g., resource allocation and process design) and performance outputs (Jacobson et al., 2013).

Several contributions can be found on the use of DES to investigate ED operational dynamics, allowing risk-free analyses and exploring how different resource management strategies impact LOS and resource occupancy, like in Connelly and

Bair, 2004, Komashie and Mousavi, 2005, and Raunak et al., 2009. Similarly, some works propose DES as a decision support tool to make multi-scenario evaluations, by modeling staff, equipment and treatment spaces as input variables, to obtain a reduction of waiting times and overcrowding and an improved patient flow (Castanheira-Pinto et al., 2021; Duguay & Chetouane, 2007; Hung et al., 2007; Nahhas et al., 2017).

Some other authors have focused on the measurement and prediction of overcrowding. For example, Ahalt et al., 2018 use DES to evaluate and compare the efficacy of different indicators in detecting ED crowding, while Hoot et al., 2008 propose a DES-based framework for short-term overcrowding prediction.

The work of Ceglowski et al., 2007 combines data-mining techniques to identify patient treatment patterns, and a DES to detect bottlenecks in the passage between ED and inpatient wards, while Lim et al., 2013 emphasize the importance of modeling the interaction between physicians and delegates, and the impact these have on waiting times and LOS.

While these works focus on modeling ED under standard operational conditions, the work of Cimellaro et al., 2017 proposes a DES to assess the resilience of an ED after a disaster, which could lead to a sudden increase in demand.

Beyond simulation-based modeling, other works directly address patient scheduling and prioritization problems through OR techniques such as Mixed Integer Linear Programming (MILP) (Harzi et al., 2017), Stochastic Programming (He et al., 2019), Dynamic Programming (Elalouf & Wachtel, 2015), Queueing Theory (Huang et al., 2015), and heuristic approaches (Allihaibi et al., 2020), with the shared aim of reducing waiting times and improving patient flow.

The progressive increase in computational capacity has led to the diffusion of AI studies in healthcare, with ML and DL techniques being widely adopted for decision support in ED processes. An example is the use of these methods for diagnostic support, as in the works of Hwang et al., 2019 and Gustafsson et al., 2022. An extensive study presented by Kareemi et al., 2021 reveals that in these types of applications, ML in general presents a higher accuracy in diagnostic and prognostic prediction with respect to usual care.

Several studies also experimented with predictive models for forecasting clinical outcomes, often anticipating prediction at triage time (Goto et al., 2019; Hong et al., 2018; Joseph et al., 2020; Raita et al., 2019; Yao et al., 2021), and with a particular focus to predicting mortality (Cardosi et al., 2021; Naemi et al., 2021; Taylor et al., 2016; Tschoellitsch et al., 2023).

Apart from the clinical decision support, other studies present process-oriented

analyses, like in the case of models developed for predicting LOS (Etu et al., 2022; Kadri et al., 2023; Ricciardi et al., 2024; Shbool et al., 2023), clinical orders (Hunter-Zinck et al., 2019), overcrowding (Harrou et al., 2020; Vural et al., 2025), and variations in patient flow (Porto & Fogliatto, 2024; Sharafat & Bayati, 2021).

Recent literature increasingly proposes hybrid and integrated approaches that combine AI techniques with other methodologies traditionally belonging to the field of OR.

For instance, Nas and Koyuncu, 2019 address capacity planning by integrating a Recurrent Neural Network with Simulation to determine the optimal number of beds in an ED, with the goal of reducing LOS. Similarly, Hosseini Shokouh et al., 2022 combine DES, Neural Networks and Genetic Algorithms to optimize staff and equipment allocation.

In the work of Alenany and Cadi, 2020, a Decision Tree is used to predict hospitalization of patients from ED, and this information is employed in a DES to improve patient flow.

A particularly innovative approach was proposed by Furian et al., 2022, in which idle resources are assigned to patients through an ML model, the strategies of which are trained on near-optimal solutions obtained through a Simulated Annealing algorithm. The approach is then evaluated through a DES.

Comparable principles can be found in the works of Duma and Aringhieri, 2023 and Fabbri et al., 2024. Both exploit the acquired knowledge of patients clinical pathways to support an adaptive selection of patient prioritization strategies, embedding the strategy selection within a simulation model that reproduces the real system behavior. The former use process mining to discover patterns in the clinical pathways, while the latter use Natural Language Processing and a DNN-based approach for this purpose.

These works highlight the increasing need for adaptive and data-driven decision support tools, consistent with the nature of real-time decisions in ED. RL appears to be a promising field in this regard. However, current applications of RL to ED process management remain very limited.

One of the few relevant contributions is that of Lee and Lee, 2020, in which an RL model is used to define a patient scheduling policy, minimizing the patient waiting times, weighted on acuity levels. After highlighting the limitations of using a MILP formulation for responding to a dynamic environment as the ED, the authors propose a Markov Decision Process formulation for the problem and train a Deep Q-Network model to obtain a patient scheduling policy. However, no mention appears on the use of simulation to actually test the proposed approach in a dynamic environment,

but only on static test scenarios. Ajmi et al., 2023 also address the problem of patient scheduling in ED using Q-learning, embedding a set of metaheuristics in a collaborative multi-agent environment.

Prabu, 2025 also presents an application of RL on an ED setting, even if with a different scope than patient scheduling. In fact, they propose an Implicit Q-Learning agent for decision support to the triage nurses, comparing it with a Behavior Cloning agent used as a performance baseline.

Although Girishan Prabhu and Taaffe, 2024 discuss the potential integration of DES and RL for reducing LOS in the ED, their contribution in this sense remains merely conceptual, focusing on general design considerations rather than the development of a concrete RL model. Similarly, H. Jiang et al., 2020 only provide a research proposal.

Finally, despite not using a proper RL framework, the work of Kim, 2024 is conceptually close to RL principles, employing a simulation supported by an ML model that dynamically suggests resource scheduling policies to reduce LOS.

The originality of the contribution presented in this thesis lies in the development, for the first time, of a DRL model integrated in a DES of a real ED, enabling the training and evaluation of adaptive queue management and patient prioritization policies on a realistic and dynamic environment –an aspect still absent in the available research. In this regard, the policy learned by the DRL agents is based on a system-wise understanding of patient flow, rather than on static scenarios limited to subsets of patients. Furthermore, the framework incorporates a predictive DNN model providing patient-level insights about the progression of clinical pathways –particularly, the likelihood of being shortly discharged–, this way enriching the state representation used by the RL agent with probabilistic knowledge of future changes in patient flow. This prediction is also included as one of the patient prioritization criteria and embedded in the actions available to the agent. Moreover, the RL agents of this work have been trained using Policy Gradient methods, namely A2C and PPO, which are indicated for complex and stochastic environments, and whose application to ED process optimization has not yet been explored in the literature.

1.5 Decision Support Tool functional framework and thesis structure

The proposed model is designed as an operational Decision Support Tool. To train and evaluate this tool, a functional architecture was developed, consisting of three interacting components (illustrated in Figure 1.4), each serving a specific role within

the framework:

- **DES**: it replicates the dynamics of the real system, acting as a Digital Twin. It allows testing and comparing alternative strategies, assessing their impact on patient flow and performance indicators.
- **DRL Agent**: it receives information on the evolving system state from the simulation environment and, based on this representation, adaptively selects actions that impact system dynamics (decisions on patient prioritization). The agent also receives a reward feedback, guiding the learning process toward the selection of actions that improve the system's performance. The interaction between the DES and the agent follows a Markov Decision Process (MDP) formulation.
- **DNN-based Predictive Model**: it estimates the likelihood of discharge of each patient based on individual-level data (demographic and triage information, clinical pathway features). The predicted discharge probability is integrated into the system state representation, enriching the available information for the DRL agent. It also supports the implementation of actions proposed by the DRL agent, enabling the use of discharge likelihood as a patient prioritization criterion.

The interaction between the DRL agent and the DES unfolds through a sequence of discrete timesteps, each corresponding to a fixed time interval. At each timestep, the simulated system provides a state representation to the agent, which responds by selecting an action. The selected action is implemented in simulation until the following agent-environment interaction, deploying a queue management rule that impacts system-wise patient flow. After the environment evolution to the next timestep, the agent receives a numerical reward signal that quantifies the impact of the previously chosen action on system performance. Through repeated interaction, the agent accumulates experience, which is used to periodically update the decision policy, reinforcing improving actions and steering away from worsening ones.

Once trained on an accurate simulation of the real ED, the DRL agent can be detached and operate independently, directly interacting with the real system. The final output is a DNN-based decision-support framework that periodically receives information on the ED conditions, processes it into an encoded representation, selects the most appropriate action, and outputs it in a human-readable form, specifically as recommendations on how to prioritize patients in the waiting queues.

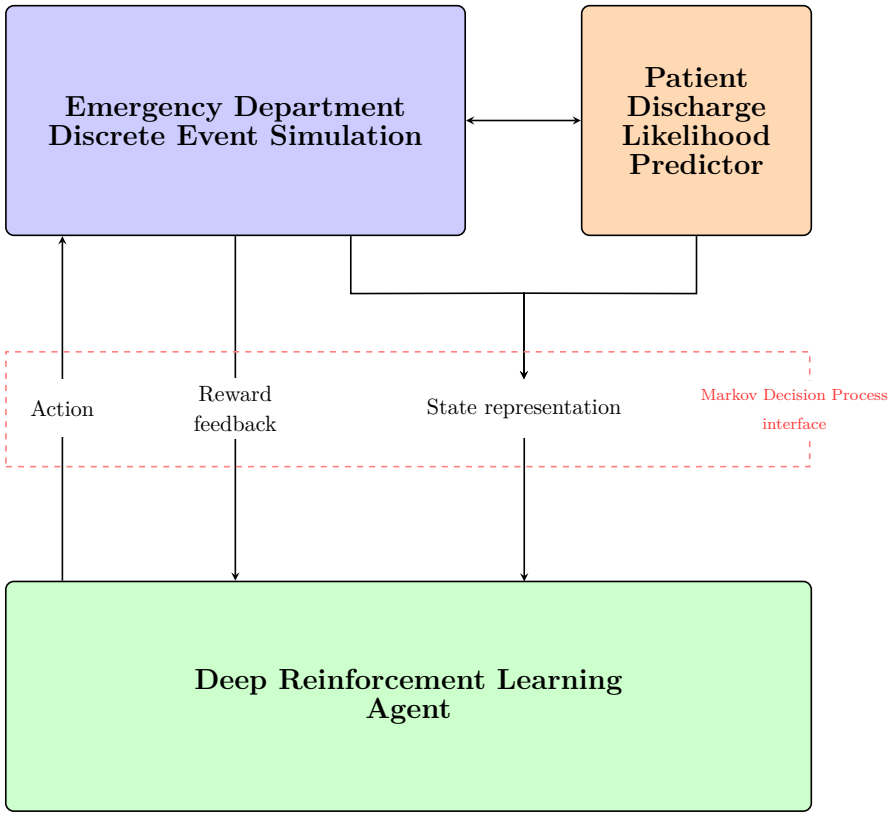

Figure 1.4: Functional representation of the components involved in the Decision Support framework

The remainder of this thesis presents how the functional parts presented were developed and integrated:

- Chapter 2 describes the data used in this work, together with the preprocessing steps that were required;
- Chapter 3 introduces the DES;
- Chapter 4 is dedicated to the mathematical formalization of the presented case study as an RL problem;
- Chapter 5 introduces the Deep Learning (DL) model for predicting the patient likelihood of discharge, a module that is integrated into the RL formulation;
- Chapter 6 describes the implementation of two RL algorithms chosen for the case study, along with a comparison between the two methods and a discussion of the results;
- Chapter 7 concludes this thesis with some considerations on the conducted work.

2 Data Sources and Preprocessing

This chapter is dedicated to describing the datasets used for the development of all the models in this thesis, starting from the presentation of the raw datasets, followed by a description of the cleaning and transformation operations that were required. All processed data were pseudonymized, so that no entity involved in ED processes could be directly recognized (forbidden access to de-anonymization keys), and treated according to the European GDPR 2016/679.

2.1 Available datasets

All the raw datasets that have been used originate from the IRCCS AOU Bologna information architecture, with particular reference to two distinct but connected databases:

- the **Emergency Department operational database**, which contains information about patients accesses to the ED, triage, visits and diagnostic exams or small procedures that have been performed;
- the **Radiology Information System database**, which enriches the previously mentioned database with some additional information on X-ray, CT scans and related exams.

All the extracted datasets were in tabular format. For each of these, a list of the variables of interest is provided, along with a brief description.

2.1.1 ED accesses

This dataset contains the main information about patient admission, type of arrival (walking / ambulance), patient discharge, and physician that has been assigned to the patient. It corresponds to a subset of the main table in the ED database, to which the majority of the other tables are linked through the *ED access ID* string, which corresponds to the primary key for each ED access (e.g., same patient is

labeled with different *ED access ID*s in case of multiple accesses). Table 2.1 lists the variables used in this work.

Table 2.1: Summary table of the ED Accesses dataset

Column name	**Type**
ED access ID	*String*
Arrival time	*DateTime*
Type of arrival	*String*
Assigned physician ID	*String*
Discharge time	*DateTime*
Discharge disposition	*String*

2.1.2 Patient demographics

Among all the available demographic variables, in this dataset only the age (computed with respect to ED access date) and the sex of the patient were kept, as shown in Table 2.2

Table 2.2: Summary table of the patient demographics dataset

Column name	**Type**
ED access ID	*String*
Sex	*String*
Age	*Int*

2.1.3 Triage

The triage dataset contains the data registered during the first triage evaluation, which is typically performed by a nurse. The registration of severity level and main identified problem is mandatory for all patients, while the vital signs, along with the Glasgow Coma Scale –an indicator of consciousness–, can be recorded for just some patients. Table 2.3 summarizes the variables used in this work.

Table 2.3: Summary table of the Triage dataset

Column name	**Type**
ED access ID	*String*
Severity	*String*
Main identified problem	*String*
Category of the main identified problem	*String*
Triage nurse notes	*String*
Body temperature	*Float*
Systolic pressure	*Float*
Diastolic pressure	*Float*
Isolated systolic hypertension	*String*
Heart rate	*Float*
Respiratory rate	*Float*
Oxygen saturation	*Float*
Glasgow coma scale	*Int*

2.1.4 Visits

Physicians are required to record a single timestamp for each visit. Although the ED database also includes information related to visits (e.g., clinical notes written by physicians), these additional data were not considered within the scope of this thesis. Table 2.4 presents the variables used in this work.

Table 2.4: Summary table of the Visits dataset

Column name	**Type**
ED access ID	*String*
Physician ID	*String*
Visit start time	*DateTime*

2.1.5 Diagnostics and treatments (services)

This dataset is fed by both data originating from the ED operational database and the Radiology Information System database. The latter contains a more detailed representation of the different phases of a service request and execution, as presented in Table 2.5.

Table 2.5: Summary table of the Exams dataset

Column name	**Type**	**Diagnostic imaging exams**	**Other services**
ED access ID	*String*	✓	✓
Service Description	*String*	✓	✓
Service Category	*String*	✓	✓
Service request time	*DateTime*	✓	✓
Service execution start time	*DateTime*	✓	×
Service execution end time	*DateTime*	✓	×
Service reporting time	*DateTime*	✓	✓

2.2 Data preprocessing

Some preprocessing steps were required to prepare a data structure suitable for use as input in the models presented in the following chapters. In this section, preprocessing operations are distinguished between those specifically designed for the simulation and those aimed at feeding the predictive model.

2.2.1 Preprocessing for the Discrete Event Simulation

Following the process described in Chapter 1, diagnostic and therapeutic services were divided into sequential sets, corresponding to the different stages of each patient's clinical pathway, each set associated with a physician visit. Algorithm 1 illustrates the logic used to group services based on their recorded request timestamps.

The underlying assumption is that services prescribed during the same physician visit do not necessarily share identical request timestamps, since they can be entered into the system at different moments throughout the visit. Conversely, noticeable gaps in timestamps may indicate separate and consecutive clinical pathway phases. Lines 10–20 of the algorithm implement this logic, identifying such temporal separations to assign services to distinct patient pathway stages.

Lines 21–28 address a specific inconsistency occasionally observed in the ED information system, where certain services –typically specialist consultations– appear with identical, or very close, timestamps for request and reporting. This situation typically occurs when a service request is made informally (e.g., by phone) and only

Algorithm 1 Separation of patient clinical pathway into sequential sets of services

Input: sequence of services $S = s_1, ..., s_n$, each having related ED access ID p_s, request datetime DTr_s and report generation datetime DTe_s; a larger time range (e.g., 30 minutes) γ; a smaller time range (e.g., 5 minutes) ε

Sort sequence S by patient p_s and request date DTr_s (from earliest to latest)
for $s \in S$ **do**
    $set\_id_s := 0$ ▷ assign a set id to each service
end for
$id := 1$
$DTsetstart := DTr_{s_1}$
$set\_id_{s_1}$:= id
for $i = 2, ..., n$ **do**
    $cs := s_i$ ▷ current service
    $ps := s_{i-1}$ ▷ previous service
    if $p_{cs} == p_{ps}$ **and** $(DTr_{cs} - DTsetstart \leq \gamma$ **or** $DTr_{cs} - DTr_{ps} \leq \varepsilon)$ **then**
        $set\_id_{cs} := id$
    else
        $id := id + 1$
        $set\_id_{cs} := id$
        $DTsetstart := DTr_{cs}$
    end if
end for
for $i = 2, ..., n$ **do**
    $cs := s_i$
    Be ps a service s_j in the list such that $set\_id_{ps} = set\_id_{cs} - 1$, when possible
    if $p_{cs} == p_{ps}$ **and** $DTe_{cs} - DTr_{cs} \leq \varepsilon$ **then**
        $set\_id_{cs} := set\_id_{cs} - 1$ ▷ If current service has a quite small time range between request and execution, it was probably requested by phone together with the previous set
        $DTr_{cs} := DTr_{ps}$
    end if
end for
Group services by set_id and p
Sort grouped services by set_id to define the patient clinical pathway and its stages

recorded at the time of its actual delivery. In these cases, the system generates nearly coincident request and execution times. To correct this artifact, these services are re-assigned to the most recent service group identified for the same patient. This adjustment ensures that services are consistently aligned with the appropriate stages of the patient's clinical pathway.

A second preprocessing operation concerns the estimation of ED physician visit durations. As presented in subsection 2.1.4, each visit is recorded in the information system with a single timestamp, corresponding to its registration time. Consequently, visit durations must be inferred from the available temporal data rather than directly observed. Algorithm 2 illustrates the procedure adopted to estimate these durations.

Initially, all visits are sorted by physician identifier and timestamp. The duration of each visit is then provisionally computed as the difference between the timestamp of the current visit and that of the subsequent visit performed by the same physician (lines 7–15). This approach also implicitly accounts for the time required for the physician to move between patients; however, it also tends to produce unrealistically long durations when the interval between two consecutive visits spans different physician shifts, such as between the last visit of one day and the first visit of the following day.

To correct these values, lines 16–27 introduce a refinement procedure. When an excessively large duration is detected, the algorithm searches for a timestamp that more plausibly marks the end of the visit. If the visit is associated with at least one prescribed service, the timestamp of the last service in the set of requests is used as a reference. Alternatively, when the patient's discharge occurs within a reasonable temporal proximity to the visit, the correspondent timestamp is used to adjust the estimated duration.

Subsequently, visits are sorted by ED access identifier and chronologically ordered (lines 28–38), allowing the construction of individual visit sequences. Each visit is then labeled with an incremental counter (*visit_number*). For each group, defined by triage severity level and by visit type (first visit, where $visit\_number == 1$, or check-up, where $visit\_number > 1$), the mean duration and the 99th percentile threshold are computed (line 40), based on the assumption that the two visit types follow different distributions.

The clinical pathways identified through Algorithm 1 are then used to detect potentially unrecorded visits, which are added to the patient's visit sequence, updating the numbering of subsequent visits as necessary (line 41). Physicians may occasionally omit to record check-up visits, while this rarely happens for first visits.

Algorithm 2 Extrapolation of visits duration

Input: sequence of visits $V = v_1, ..., v_m$, each having a related patient p_v, physician h_v, associated timestamp DT_v; sequence of type-2 services $S = s_1, ..., s_n$, each having related ED access id p_s, and request datetime DTr_s; a time range γ

Sort V by physician h_v and by timestamp DTv (from earliest to latest)
for $v \in V$ **do**
    $duration_v := 0$
end for
for $i = 1, ..., m-1$ **do**
    $cv := v_i$ ▷ current visit
    $fv := v_{i+1}$ ▷ following visit
    if $h_{cv} == h_{fv}$ **then**
        $duration_{cv} := DT_{fv} - DT_{cv}$
    else
        pass
    end if
end for
for $i = 1, ..., m$ **do**
    Based on the clinical pathways defined in Algorithm 1, identify the last service request linked to the current visit (if it exists), and be *last_request_time* its DTr
    Be *discharge_time* the time of discharge of patient p_{v_i}
    if $duration_{v_i} \geq \gamma$ **then**
        if $\exists$ *last_request_time* **then**
            $duration_i := last\_request\_time - DT_i$
        else if $discharge\_time - DT_i < \gamma$ **then**
            $duration_i := discharge\_time - DT_i$
        elsepass
        end if
    end if
end for
Sort V by patient p_v and timestamp DT_v
$counter := 1$
$visit\_number_{v_1} := 1$
for $i = 2, ..., m$ **do**
    if $p_{v_i} == p_{v_{i-1}}$ **then**
        $counter := counter + 1$
    else
        $counter := 1$
    end if
    $visit\_number_{v_i} := counter$
end for
Associate each visit with the severity code assigned to the patient during triage
Compute mean and 99 percentile of visits duration, separating severity levels and first visits ($visit\_number == 1$) from check-up visits ($visit\_number > 1$)
Based on clinical pathways defined in Algorithm 1, add possibly not recorded visits in the visit sequence, initializing their *visit_number* accordingly and their *duration* to 0 (updating other visits *visit_number* if required)
for $i = 1, ..., m$ **do**
    Be *perc* the 99 percentile and μ the mean of visits duration for the same group of v_i (severity and first visit / check-up)
    if $duration_{v_i} == 0$ **or** $duration_{v_i} \geq perc$ **then**
        $duration_{v_i} := \mu$
    end if
end for

This step helps preserve the consistency of each patient's pathway.

In the end, mean and percentile values are used to revise any remaining anomalous durations that exceed plausible limits, this way excluding outliers that could compromise the correct functioning of the simulation (lines 42–47).

A third, simpler, preprocessing operation, consisted in extracting physician shifts from the available data. This meant identifying, for each date in the period of interest and for each hour of the day, the number of physicians providing visits to patients, distinguishing them by the *Physician ID* variable.

Finally, some discrepancies were found between certain patients last activities (e.g., received visits) and their discharge time. This can happen when patient discharge does not entirely depend on the ED process itself, but other external factors may interfere, such as bed availability in inpatient wards. To keep track of these dynamics, a variable was added to each patient access, computing the difference in minutes between the last recorded activity and the time of discharge.

2.2.2 Preprocessing for the predictive model

The predictive model that will be described in Chapter 5 relies on two main categories of input information: (i) demographic and triage data, and (ii) patient's clinical pathway. For the latter category, most of the relevant preprocessing operations have already been detailed in subsection 2.2.1.

Additional preprocessing was instead required for the demographic and triage data. In particular, each patient's age was classified into age groups. Similarly, the vital signs measured during triage were discretized into clinically meaningful classes. Table 2.6 summarizes the classification criteria adopted for both demographic and vital signs variables.

Additionally, patient arrival to the ED, obtainable from the ED accesses table, was classified in four time slots: 00:00-5:59AM, 06:00-11:59AM, 12:00-17:59PM, 18:00-23:59PM.

Table 2.6: Classification of demographic and triage variables

Value	**Range**	**Correspondent class**
Age	0-14 years	*Paediatric age*
	14-22 years	*Adolescence*
	22-39 years	*Early adulthood*
	40-59 years	*Middle adulthood*
	60-75 years	*Third age*
	75-90 years	*Fourth age*
	>90 years	*Fifth age*
Body temperature	<37.5	*No fever*
	37.5-38°C	*Low-grade fever*
	38-39°C	*Moderate fever*
	>39°C	*High fever*
	-	*Not measured*
Systolic pressure	<120 mmHg	*Optimal*
	120-129 mmHg	*Normal*
	130-139 mmHg	*Normal-high*
	140-159 mmHg	*Stage 1 hypertension*
	160-179 mmHg	*Stage 2 hypertension*
	≥180 mmHg	*Stage 3 hypertension*
	-	*Not measured*
Diastolic pressure	<80 mmHg	*Optimal*
	80-84 mmHg	*Normal*
	85-89 mmHg	*Normal-high*
	90-99 mmHg	*Stage 1 hypertension*
	100-109 mmHg	*Stage 2 hypertension*
	≥110 mmHg	*Stage 3 hypertension*
	-	*Not measured*
Isolated systolic hypertension	systolic pressure≥140 mmHg and diastolic pressure≤90 mmHg	*Yes*
	other values	*No*
	-	*Not measured*

Heart rate	<60 BPM	*Bradycardia*
	60-100 BPM	*Normal*
	>100 BPM	*Tachycardia*
	-	*Not measured*
Respiratory rate	<12 breaths/min	*Bradypnea*
	12-20 breaths/min	*Normal*
	>20 breaths/min	*Tachypnea*
	-	*Not measured*
Oxygen saturation	≥95%	*Normal*
	91-94%	*Mild hypoxia*
	86-90%	*Moderate hypoxia*
	≤85%	*Severe hypoxia*
	-	*Not measured*
Glasgow coma scale	15	*Fully responsive*
	9-14	*Compromised consciousness*
	≤8	*Coma*
	-	*Not measured*

2.3 Summary of the processed datasets

Following the preprocessing operations described in this chapter, the available data were organized into the following structured tables:

- *patient_accesses*, containing the timestamps of arrival and discharge for each ED access, as well as the detected time between the last performed activity and the moment of discharge;
- *demographics*, including the age class and sex of each patient;
- *triage*, reporting the assigned severity level, main identified problem and corresponding category, as well as vital signs (each discretized into classes);
- *clinical_pathways*, describing the ordered sequence of service sets prescribed to each patient during their stay, with *Discharge* appended as the final stage for every patient;
- *visit_durations*, reporting the duration (in minutes) of each physician visit recorded for each patient;

- *service_durations*, reporting the execution time (in minutes) for type-2 services, and the report generation time for both type-1 and type-2 services;
- *staff_timetable*, derived from the *Visits* dataset, providing information on physician work shifts and availability.

Each time a variable is used in the subsequent chapters, its source will be referenced using these table names.

3 Discrete Event Simulation Model

This chapter is dedicated to presenting the DES of the ED. Section 3.1 introduces the simulation library that was used and its main features. Section 3.2 proceeds with the description of the different components involved in the simulation. Finally, section 3.3 concludes the chapter with a description of the tuning process that was necessary to make the simulation resemble the real system, along with the assumptions that were used during its development.

3.1 Simulation framework and its operating logic

The simulation model was developed using the Salabim library, a Python framework designed for DES (van der Ham, 2016). A simulated environment typically consists of multiple components, implemented according to the Object-Oriented Programming paradigm, and interacting with each other. The behavior of each component is defined by a dedicated procedure, called *Process*, which starts automatically right after an instance of a component is generated. The execution of a process is governed by the principles of the DES paradigm, where the occurrence of specific events drives the evolution of the modeled system.

At any time, a component can be in one of two states:

- **active**, in which the component's process is currently running;
- **passive**, in which the process is temporarily paused.

Through its process, a component can be put on hold, waiting either for a time interval to elapse or for a specific event to occur. When a component completes the algorithm that describes its process, it is discarded and removed from the simulation.

Resources are passive objects that are not governed by a process, but can be seized, held, and released by other components through their process.

Queues are objects that store components and allow their contents to be inspected, sorted, and removed.

Finally, all components are encapsulated in an environment that acts as the main process of the simulation.

3.2 Model structure and simulation components

This section introduces the set of components, resources and queues defined to model the ED process described in Chapter 1. The simulation involves the following components:

- **`Patient`**: instances of this class may occupy triage resources at the beginning of their process (waiting in the corresponding queue if the resource is temporarily unavailable). After triage, patients can either activate an idle physician if available, or interrupt an ongoing visit in case of severity 1 in need of a first assessment (provided that the interrupted patient is not themselves a severity 1 patient being visited for the first time). Otherwise, they must wait in the physician visit queues until a physician becomes available and selects them for the visit. The same logic applies for the following checkup visits, except that in this case interruptions are no longer permitted. After each physician visit, if the patient is not declared ready for discharge, they generate one or more instances of `ServiceRequest`, according to the exams and treatments assigned by the physicians.

- **`PatientGenerator`**: a single instance of this class is created, responsible for generating `Patient` instances, according to the arrivals recorded in historical data.

- **`ServiceRequest`**: instances of this class may occupy service resources (when applicable). Before occupying a resource, they must wait in its dedicated queue.

- **`Physician`**: instances of this class follow a continuous process alternating between patient selection from queues, medical visits, and actions to regulate the number of active instances of the same class (according to staff shifts).

- **`ServiceResourcesQueueManager`**: a single instance of this class is created. When activated, it scans the service resources to find if there is any available, and checks the corresponding queues to decide which patient will proceed.

Figure 3.1 summarizes the interactions between the described components, resources and queues. This section proceeds with a deeper presentation of each component class, along with their descriptive pseudocode.

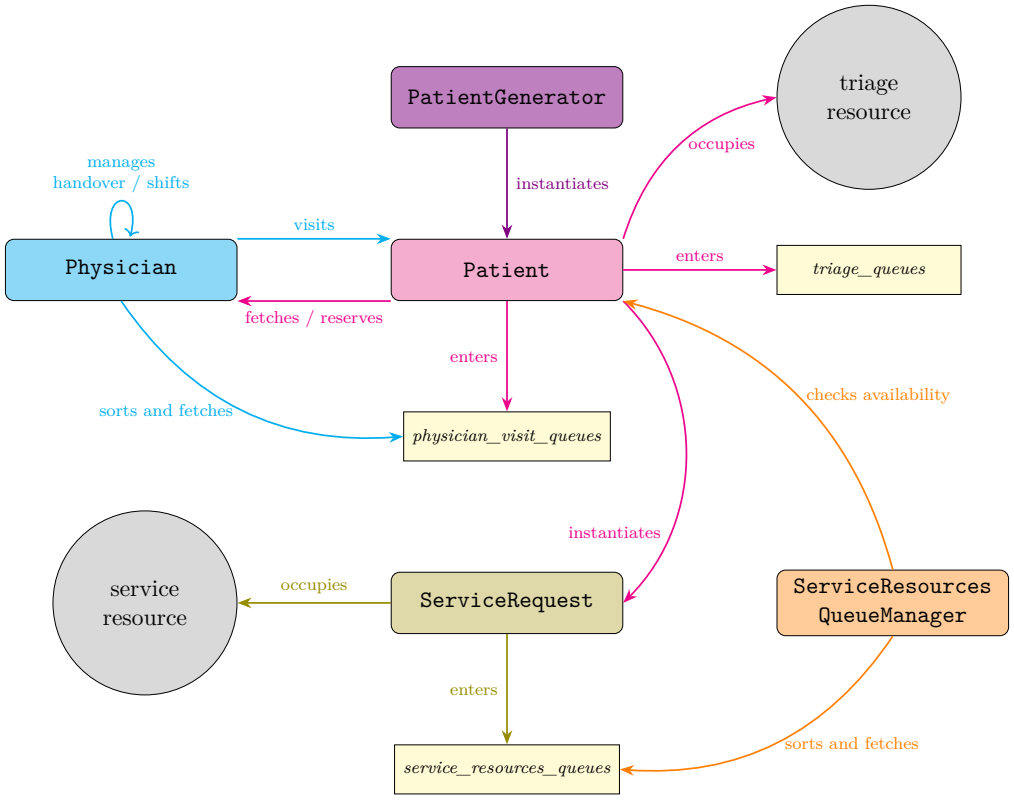

Figure 3.1: Interaction between DES components, queues and resources

3.2.1 Patient class

Algorithm 3 describes the `Patient` class. Each instance is created with a set of input parameters derived from ED access records, demographics, triage data, and datasets of visits and services (see Chapter 2). Lines 3–12 of the pseudocode describe the initialization stage, including the registration of the admission time and triage, together with the setup of the patient's attributes. In this phase, the parameter *additional_LOS_time* is subtracted from the patient's entrance time into the system. This adjustment accounts for patients who were already present in the ED before midnight but are nonetheless included in the simulation (patients instances generated in the middle of their clinical pathway). In addition, a prediction is computed for the likelihood of discharge after the following medical visit (line 9), using the DL model described in Chapter 5.

The procedure *TriageRequest* is then invoked, placing the patient in the appropriate triage queue (ambulance or walk-in). Since no information was available regarding triage duration, a uniform distribution between 5 and 10 minutes was adopted, based on field observations.

From line 13, the patient enters the main loop of the simulation, alternating between physician visits and the diagnostics and treatment phase. At each iteration,

the patient marks themselves as waiting for a physician (line 14). If no physician has been assigned (line 18), the patient enters the first-visit queue and invokes *RequestPhysician*, which attempts to activate an idle physician. If none are available, the patient may interrupt an ongoing visit provided that: (i) they are classified as severity 1, (ii) they are waiting for their first visit, and (iii) the patient currently being visited does not meet the same conditions. In this case, the new patient is flagged as emergent and linked to the physician (lines 73–74). Visit interruption dynamics are detailed in subsection 3.2.4.

If the patient already has an assigned physician (line 21), they enter the corresponding checkup queue. In both cases, the patient passivates until they are reactivated at the end of the visit (line 26). At that point, the visit counter is incremented (line 28), and the "waiting for physician" flag is reset (line 29).

The assignment of services at the end of a visit is modeled by retrieving the next set of diagnostics (or treatments) from the dedicated dataset. The clinical pathway is updated accordingly, and a new discharge prediction is computed (lines 32–33). For each assigned service, a `ServiceRequest` is instantiated with the appropriate execution and reporting times (lines 36–39). The `ServiceResourcesQueueManager` is then activated to scan queues and handle service executions (line 40). Before continuing in the loop, the patient must wait for all services to be completed (line 41).

When the services set includes a discharge declaration, the patient exits the loop. A waiting time parameter (*waiting_time_before_discharge*) is added to reflect the delay observed in the real system between the last recorded activity and the effective discharge (line 44). Finally, the patient exits the simulation, being removed from the patient population of the system (line 45).

Algorithm 3 `Patient` class

Input: Patient identifier *id*; walking/ambulance arrival *access_type*; age of the patient *age*; sex of the patient *sex*; patient severity level *severity*; main assigned problem *main_problem*; category of main assigned problem *main_problem_category*;clinical data and vital signs acquired during triage *triage_data*; sequential sets of services prescribed for the patient *clinical_pathways*; counter of how many physician visits the patient has received *visits_counter*; additional LOS time between the last recorded activity and discharge *waiting_time_before_discharge*; additional LOS time for patients already in the system at simulation start *additional_LOS_time*; visits dataset *visit_durations*; services dataset *service_durations*; clinical pathway progress, as a list of assigned services *clinical_pathway_progress*; discharge prediction network parameters $\theta_{discharge}$
procedure Process
    self.*arrival_time* = env.current_time - self.additional_LOS_time
    Classify self.*arrival_time_slot* accordingly
    if self.*visits_counter* == 0 **then**
        self.TriageRequest()
    end if
    Add self to env.*patients_crowd*
    self.*discharge_probability* := predict((self.*severity*, self.*arrival_time_slot*, self.*access_type*, self.*age*, self.*sex*, self.*triage_data*, self.*clinical_pathway_progress*); $\theta_{discharge}$)
    self.*waiting_for_physician* := 0
    self.*busy* := **false**
    self.*requested_services* := []
    while self.*requested_services* != [*Discharge*] **do**
        self.*services_state* := *no service requested*
        self.*waiting_for_physician* := 1
        self.*wait_visit_start* := env.current_time
        Obtain visit duration *v_t* from *visit_durations*, searching for self.*id* and self.*visits_counter*
        if (self.*visits_counter* == 0) **or** (self.*assigned_physician* == none) **then**
            enter *first_visit_queue*
            self.RequestPhysician()
        else
            Enter i.*assigned_phyisician.checkup_queue*
            if !(self.*assigned_physician*.active) **then**
                activate self.*assigned_physician*
            end if
        end if
        Passivate self
        self.*visits_counter* := self.*visits_counter* + 1
        self.*waiting_for_physician* := 0
        Obtain *next_services_set* from *clinical_pathways*, searching for self.*id*, self.*visits_counter*
        self.*requested_services* := *next_services_set*
        Update self.*clinical_pathway_progress* according to self.*requested_services*
        self.*discharge_probability* := predict((self.*severity*, self.*arrival_time_slot*, self.*access_type*, self.*age*, self.*sex*, self.*triage_data*, self.*clinical_pathway_progress*); $\theta_{discharge}$)
        self.*services_state* := *waiting for services reports*
        if self.*requested_services* != [*Discharge*] **then**
            for *e* ∈ self.*requested_services* **do**
                Be *e_t* the execution time and *r_t* the reporting time associated to service *e* (from *service_durations*)
                Instantiate `ServiceRequest`(patient := self, service_type := *e*, execution := *e_t*, reporting := *r_t*)
            end for
            Activate `ServiceResourcesQueueManager`
            Wait until (self.*services_state* == *all services completed*)
        end if
    end while
    Wait for *waiting_time_before_discharge*
    Remove self from env.*patients_crowd*
end procedure

```
47: procedure TriageRequest
48:     triage_duration := sample from Uniform(5, 10)
49:     if self.access_type == ambulance then
50:         if ambulance_triage_resource is not available then
51:             Wait for ambulance_triage_resource in the corresponding queue
52:         end if
53:         Occupy ambulance_triage_resource
54:         Wait for triage_duration
55:         Release ambulance_triage_resource
56:     else
57:         if walking_triage_resource is not available then
58:             Wait for walking_triage_resource in the corresponding queue
59:         end if
60:         Occupy walking_triage_resource
61:         Wait for triage_duration
62:         Release walking_triage_resource
63:     end if
64: end procedure
65:
66: procedure RequestPhysician
67:     idle_physicians := {i ∈ physicians: !(i.active) and (i.on_duty)}
68:     active_physicians := {i ∈ physicians: (i.active) and (i.on_duty)}
69:     if |idle_physicians| >0 then
70:         activate idle_physicians[0]
71:     else if (self.severity == 1) and (self.visits_counter == 0) then
72:         for i ∈ active_physicians do
73:             if (i.patient_being_visited.severity != 1) or
    ((i.patient_being_visited.severity == 1) and
    (i.patient_being_visited.visits_counter > 0)) then
74:                 i.emergent_patient := self
75:                 Interrupt i.patient_being_visited visit
76:                 break
77:             end if
78:         end for
79:     end if
80: end procedure
```

3.2.2 PatientGenerator class

Algorithm 4 describes the `PatientGenerator` class, which as a single component manages patient arrivals according to historical records. Once instantiated, this component remains active through the whole simulation.

Since each simulated instance is set to always begin at midnight, the generator first initializes the residual population, i.e., patients who were admitted before midnight but whose discharge occurs after midnight (lines 4–12). These patients are instantiated with the appropriate visit counters, partial clinical pathways, and additional LOS time.

Subsequently, the generator iterates through new arrivals, as reported in the *patient_accesses* table. For each patient, an instance of the `Patient` class is cre-

ated with the corresponding demographic, triage, and admission data. Generation continues in sequence until the simulation ends (lines 13–21).

Algorithm 4 `PatientGenerator` class

1: **Input**: patient accesses table *patient_accesses*; patient clinical pathways table *clinical_pathways*; patient data acquired during triage *triage*; patient demographic data *demographics*; table containing visits information *visit_durations*; table containing services information *service_durations*; discharge likelihood prediction network parameters *DNN_parameters*
2: **procedure** Process
3: Sort *patient_accesses* by admission date-time
4: *residual_patients* := patients in *patient_accesses* where admission time < env.*simulation_start* < discharge time
5: **for** *patient* ∈ *residual_patients* **do**
6: Be *v_c* the counter of the last recorded visit before env.current_time, obtained from *clinical_pathways*
7: Be *c_p_p* the encoded representation of the services assigned before env.current_time, obtained from *clinical_pathways*
8: Be *admission_date* the date and time of admission for the patient
9: *a_L* := env.current_time - *admission_date*
10: Be *patient_parameters* the list of all their other input parameters, obtained from *patient_accesses*, *demographics*, and *triage*
11: Instantiate `Patient`(visits_counter:=*v_c*, clinical_pathway_progress := *c_p_p*, additional_LOS_time := *a_L*, (id, access_type, age, sex, severity, main_problem, main_problem_category, triage_data, waiting_time_before_discharge) := *patient_parameters*, visit_durations := *visit_durations*, service_durations := *service_durations*, $\theta_{discharge}$:= *DNN_parameters*)
12: **end for**
13: Be *patients_list* the sublist of *patient_accesses* where admission time > env.*simulation_start*
14: **while true do**
15: Be *current_patient* the next patient on *patients_list*
16: Be *patient_parameters* the list of their input parameters, obtained from *patient_accesses*, *demographics*, and *triage*
17: Be *c_p_p* the encoded representation of an empty clinical pathway (no services assigned yet)
18: Instantiate `Patient`(visits_counter:=0, clinical_pathway_progress := *c_p_p*, additional_LOS_time := 0, (id, access_type, age, sex, severity, main_problem, main_problem_category, triage_data, waiting_time_before_discharge) := *patient_parameters*, visit_durations := *visit_durations*, service_durations := *service_durations*, $\theta_{discharge}$:= *DNN_parameters*)
19: Be *following_patient* the patient following *current_patient*
20: Wait for admission time of *following_patient* - admission time of *current_patient*
21: **end while**
22: **end procedure**

3.2.3 ServiceRequest class

Algorithm 5 describes the `ServiceRequest` class. Requests belonging to the type-2 category of services (see Chapter 1), i.e., those requiring the presence of the patient, must seize a dedicated service resource (line 3).

After joining the corresponding queue (line 4), the request remains passive until activated by the `ServiceResourcesQueueManager`. Upon activation, the request seizes the resource and marks the originating patient as busy, ensuring that a patient cannot undergo multiple simultaneous type 2 exams. Once the service is completed, the resource is released (lines 11–12), and the queue manager is reactivated (line 13) to assign the resource to another waiting request if available.

The request then waits for the reporting time to elapse. At its completion, the service is removed from the patient's pending list. When all assigned services are completed, the patient is notified and may resume their process (lines 17–19).

Algorithm 5 `ServiceRequest` class

```
1: Input: patient from whom the request is generated patient; indication of the type of service
   service_type; time required for service execution execution; time required for service reporting
   reporting
2: procedure Process
3:     if self.service_type ∈ type_2_services then
4:         Enter service_type resource associated queue
5:         self.wait_start := env.current_time
6:         Passivate self
7:         self.patient.busy := true
8:         Exit service_type designated queue
9:         Occupy service_type resource
10:        Wait for execution
11:        Release service_type resource
12:        self.patient.busy := false
13:        Activate ServiceResourcesQueueManager
14:    end if
15:    Wait for reporting
16:    Remove service_type from self.patient.requested_services
17:    if |self.patient.requested_services| == 0 then
18:        self.patient.services_state := all services completed
19:    end if
20: end procedure
```

3.2.4 Physician class

Algorithm 6 describes the `Physician` class. Each physician instance receives the staff timetable as input, which specifies how many physicians are expected to be on duty at each time. Each physician also receives a dedicated checkup queue for their assigned patients.

Once created, a physician remains in the system until the end of the simulation. Its process is structured as a loop alternating staff management, patient selection, and medical visits. A physician may be either *on duty* or *off duty*. Off-duty physicians remain inactive and are excluded from resource allocation, while on-duty physicians can be either active (engaged in a patient visit) or passive (waiting).

During the *StaffManagement* procedure, each physician verifies whether the current number of on-duty physicians matches the staff timetable. If fewer physicians than required are active, additional physicians are put on duty and activated (line 36). If there is an excess, a physician may put themselves off duty after redistributing their assigned patients to other on-duty physician instances (line 22). This ensures that the total number of active physicians remains consistent with the schedule.

The *SelectPatient* procedure performs a hierarchical search among patients. First, emergent severity 1 patients waiting for a first assessment are prioritized (line 50). If none is found, the physician checks for patients whose visits were previously interrupted by emergent cases. Finally, the physician selects from their personal checkup queue or the common first visit queue. Candidate patients are sorted according to the active queue management strategy (lines 58–62), and the first patient in line is selected (line 63). This way, the physician component allows the exploration and implementation of alternative queue management policies. If the candidate patients list is empty, the physician passivates until reactivated (line 14). Otherwise, the visit begins through the *Visit* procedure.

The *Visit* procedure receives the selected patient and the visit duration as inputs. If it is the patient's first visit, the physician-patient assignment is established (lines 73–76). If the visit is interrupted by an emergent patient, the current patient is moved to the temporary buffer *suspended_patients* (line 81), and the procedure *Visit* is called recursively to handle the emergent case.

Algorithm 6 `Physician` class

1: **Input**: time table indicating the number of physicians expected on duty for each day and for each hour of the day (physician shifts) *staff_timetable*; queue for patients assigned to the physician *checkup_queue*
2: **procedure** Process
3: self.*on_duty* := **true**
4: self.*assigned_patients* := []
5: self.*suspended_patients* := []
6: **while true do**
7: **if** |self.*suspended_patients*|==0 **then**
8: self.StaffManagement()
9: **end if**
10: self.*patient_being_visited* := self.SelectPatient()
11: **if** self.*patient_being_visited* != none **then**
12: self.Visit(patient := self.*patient_being_visited*, visit_duration := self.*patient_being_visited.v_t*)
13: **else**
14: Passivate self
15: **end if**
16: **end while**
17: **end procedure**
18:
19: **procedure** StaffManagement
20: Obtain the number of physicians expected to be on duty *expected_physicians* from *staff_timetable*, searching for current day and current hour of the day
21: δ := *expected_physicians* - $|\{i \in$ physicians: *i.on_duty*$\}|$
22: **if** $\delta < 0$ **then** ▷ Currently more physicians on duty than expected
23: **while** |self.*assigned_patients*| > 0 **do**
24: *candidate_physicians* := $p \in$ physicians : $p! =$ self **and** *p.on_duty*
25: *target_physician* := $\underset{p \in candidate\_physicians}{\arg\min}$ ($|p.assigned\_patients|$)
26: *target_patient* := self.*assigned_patients*[0]
27: Remove *target_patient* from self.*assigned_patients*
28: Add *target_patient* to *target_physician.assigned_patients*
29: *target_patient.assigned_physician* := *target_physician*
30: **if** *target_patient* $\in$ self.*checkup_queue* **then**
31: Move *target_patient* to *target_physician.check_up_queue*
32: **end if**
33: **end while**
34: self.*on_duty* := **false**
35: Passivate self
36: **else if** $\delta > 0$ **then** ▷ Currently less physicians on duty than expected
37: **while** $\delta > 0$ **do**
38: **for** $i \in$ physicians **do**
39: **if** !(*i.on_duty*) **then**
40: *i.on_duty* := **true**
41: Activate *i*
42: $\delta := \delta$ - 1
43: **break**
44: **end if**
45: **end for**
46: **end while**
47: **end if**
48: **end procedure**

```
49: procedure SelectPatient
50:     candidate_patients := {p ∈ first_visit_queue: p.severity == 1 and p.visits_counter==0}
51:     if |candidate_patients| == 0 then
52:         candidate_patients := self.suspended_patients
53:         if |candidate_patients| == 0 then
54:             candidate_patients := {p ∈ self.checkup_queue} ⋃ {p ∈ first_visit_queue}
55:         end if
56:     end if
57:     if |candidate_patients| > 0 then
58:         if reinforcement_agent is employed then
59:             Sort candidate_patients according to φ_currentAction
60:         else
61:             Sort candidate_patients according to φ_standardPolicy
62:         end if
63:         selected_patient := candidate_patients[0]
64:         Remove selected_patient from their current queue, or from self.suspended_patients
65:         return selected_patient
66:     else
67:         return none
68:     end if
69: end procedure
70:
71: procedure Visit
72:     Input: patient to visit patient, duration of the visit visit_duration
73:     if patient.assigned_physician == none then
74:         patient.assigned_phyisician := self
75:         Add patient to self.assigned_patients
76:     end if
77:     patient.t_start := env.current_time
78:     Wait for visit_duration or until visit is interrupted
79:     if visit was interrupted then
80:         patient.v_t := visit_duration - (env.current_time - t_start)
81:         Add patient to self.suspended_patients
82:         self.patient_being_visited := self.emergent_patient
83:         self.emergent_patient := none
84:         self.Visit(patient := self.patient_being_visited,
    visit_duration := self.patient_being_visited.v_t)
85:     else
86:         Activate patient
87:     end if
88: end procedure
```

3.2.5 ServiceResourcesQueueManager class

Algorithm 7 describes the `ServiceResourcesQueueManager` class. A single instance of this component is created and remains in the system until the end of the simulation. Its role is to manage the allocation of service resources by monitoring the corresponding queues of patient requests.

The process is structured as an infinite loop in which, at each cycle, the manager scans all queues associated with currently available service resources. For each of

these queues, it identifies candidate requests, i.e., instances of `ServiceRequest` whose originating patient is not already marked as busy (line 4). This condition ensures that patients cannot simultaneously be provided two type-2 services.

If at least one candidate request is found, the selection process is performed, sorting requests according to the active queue management strategy. The first request in the sorted list is then selected, removed from its queue, and activated, allowing it to seize the corresponding service resource.

Once all queues have been scanned, the manager passivates until reactivated by the completion of a service or by the generation of a new service request. In this way, the component ensures efficient allocation of resources while supporting the exploration and implementation of alternative queue management policies.

Algorithm 7 `ServiceResourcesQueueManager` class

```
1:  procedure Process
2:      while true do
3:          for queue ∈ queues associated to service resources if service resource is available do
4:              candidate_requests := {r ∈ queue: !r.patient.busy}
5:              if |candidate_requests|> 0 then
6:                  if reinforcement_agent is employed then
7:                      Sort candidate_requests according to ψ_currentAction
8:                  else
9:                      Sort candidate_requests according to ψ_standardPolicy
10:                 end if
11:                 selected_request := candidate_requests[0]
12:                 Remove selected_request from their current queue
13:                 Activate selected_request
14:             end if
15:         end for
16:         Passivate self
17:     end while
18: end procedure
```

3.2.6 Environment (main)

Algorithm 8 describes the `Environment`, which acts as the container for all the simulation components. It defines the simulation start, initializes the resources and queues, and manages the generation of all the components' instances. Providing different dates as input for the *simulation_start* variable, different instances of the ED environment can be created, corresponding to replicating different days in the real ED.

At the beginning of the process, the simulation clock is initialized to the specified starting time (line 3). The environment then determines and generates a number of `Physician` instances that allows to cover the maximum expected number of physi-

cians simultaneously on duty, obtaining this number from the staff timetable (line 4). For each of these physicians, a dedicated patient check-up queue is generated (lines 5–8).

After that, the two triage resources (ambulance and walk-in) are created, each with its associated queue (lines 9–12). Service resources are then iteratively initialized and linked to their respective queues (lines 13–16).

Finally, an instance of the `PatientGenerator` is created (line 17), responsible for creating `Patient` instances according to the historical access and clinical pathway data.

Algorithm 8 `Environment`

1: **Input**: the starting time of the simulation (date) *simulation_start*; time table indicating the number of physicians expected on duty for each day and for each hour of the day *staff_timetable*; list of service resources *service_resources*; patient accesses table *patient_accesses*; patient clinical pathways table *clinical_pathways*
2: **procedure** Process
3: Initialize simulation clock at midnight of *simulation_start*
4: Be N the upper bound on the number of required physicians in *staff_timetable*
5: **for** N times **do**
6: Generate a queue *c_q*
7: Instantiate Physician(*staff_timetable*, checkup_queue := *c_q*)
8: **end for**
9: Generate a resource *ambulance_triage*
10: Generate a queue associated to *ambulance_triage*
11: Generate a resource *walking_triage*
12: Generate a queue associated to *walking_triage*
13: **for** $r \in$ *service_resources* **do**
14: Generate a resource r with capacity = 1
15: Generate a queue associated to r
16: **end for**
17: Instantiate `PatientGenerator`(patient_accesses := *patient_accesses*, clinical_pathways:=*clinical_pathways*)
18: **end procedure**

3.3 Validation and comparison with the real system

This section presents the assumptions adopted in the development of the ED simulation model (3.3.1), together with the tuning operations performed to ensure that the model provides a reliable representation of the real system (3.3.2).

3.3.1 Assumptions of the model

As the simulation represents a simplified version of the real ED, several assumptions were introduced during model development. These can be grouped as follows:

Organizational structure. The real ED is divided into two separate areas (high-severity and low-severity). However, patient assignment to these areas does not follow strict rules: for instance, severity 3 patients may be placed in either area depending on their condition and current crowding. Moreover, physicians are not rigidly bound to a single area and a retrospective data analysis revealed that they may move between them during the same shift. For this reason, the model represents the ED as a single shared area. Another simplification concerns the physical allocation of patients to consultation boxes during medical visits. Although this is a concrete logistical aspect of ED operations, it was not explicitly modeled; instead, visits are assumed to take place as soon as a physician becomes available, without considering room occupancy.

Patient characteristics and clinical pathway. In the simulation, patients retain the same severity level throughout their length of stay. In reality, periodic nurse reassessments may lead to severity updates, though retrospective analyses suggest this is a rare event. Similarly, the patient's clinical pathway in the simulation is assumed to evolve uniquely through the decisions of ED physicians. In practice, specialist consultants may also prescribe additional services, which could further branch the patient's pathway.

Care and support staff. Care staff (like nurses) and other support staff were not explicitly modeled, despite their actual relevance in the real ED process. Their contribution, transversal to the whole patient's clinical pathway, was incorporated in the contribution of the other modeled resources.

Arrival processes and stochasticity. Patient arrivals are modeled exactly as observed in historical data, without introducing additional stochastic variability. This choice is intentional: by emulating realistic instances directly derived from the data, the model allows for the evaluation of how the techniques described in later chapters may impact the real system under actual operating conditions.

Timing simplifications. Physician walking times and patient transfers for diagnostic exams and treatments are not modeled explicitly, but are incorporated into visit and service durations. The duration of visits and services is assumed to be independent of both the time of request and the individual resource performing them. Physicians are modeled as homogeneous resources, without accounting for potential differences in skills or performance.

Resources and services. Laboratory and microbiology exams (type 1 ser-

vices) are modeled as black-box processes, with reporting delays assumed to be independent of the number of concurrent requests. This simplification is supported by the fact that such exams generally produce relatively long delays compared to other services, making queuing effects less relevant in practice. Specialist consultations, although resource-dependent in the real ED, were also modeled as category 1 services. This choice was motivated by the absence of reliable data on the actual start times of consultations and by the consistently long delays observed in practice, mainly due to the time required for consultants to arrive from their primary wards to the ED (often, after completing other ongoing activities in their ward). For type 2 service resources, capacity is set to a single unit per service type (e.g., one X-ray machine, one CT scanner), reflecting the actual ED infrastructure. Some diagnostic procedures are often performed with portable devices (e.g., Doppler ultrasound). These resources are also incorporated in the model as the other service resources, acknowledging the necessary simplification.

Queueing and decision dynamics. In the model, patients waiting for first assessment are assigned to the first available physician for whom they are ranked highest according to the adopted prioritization strategy. In reality, however, physicians also consider workload balancing when taking charge of new patients. Queue management for patient selection is modeled through rigid sorting rules, although real physicians may deviate based on clinical judgment and evolving patient conditions. Similarly, service queues are scanned sequentially, while in practice more complex prioritization may apply. For instance, if a patient is simultaneously assigned multiple type-2 services, physicians or care staff may prioritize one service over another depending on clinical judgment, urgency, or logistical considerations. Such decisions, which can influence both patient flow and resource utilization, are not explicitly represented in the model. The model also requires that all pending service reports are received before a patient can undergo the next checkup, whereas real physicians may occasionally proceed regardless.

Discharge constraints. The model does not explicitly model patient discharge outcomes, and it doesn't account for external factors that may delay discharge. Examples include the shortage of inpatient beds in hospital wards, or, in the case of frail patients, the need to coordinate with family members or caregivers before the patient can safely leave the ED. Patients with *Left ED before first medical assessment* as a discharge outcome (mostly low-severity patients), as well as the ones *Deceased before arrival*, were intentionally excluded from the model, due to the impossibility to accurately characterize their stay in the ED, and considering the assumption of them not occupying any limited resource.

These assumptions, while simplifying, are consistent with common practices in ED simulation and were deemed necessary to keep the model tractable while preserving its ability to reproduce key system dynamics.

3.3.2 Simulation tuning and validation

Although historical data were used as input for the simulation, additional parameter tuning was required to ensure that the simulated system accurately represented the real ED. The tuning process was carried out using a set of test instances, which will also serve as the evaluation environment for the RL framework presented in later chapters.

Staff shifts

Although the number of physicians on duty was derived from historical records, manual adjustments were necessary. Discrepancies between the number of physicians observed in the data and the number of physicians effectively active in practice may arise from several factors, such as physicians not recording certain visits or residents recording visits on behalf of their supervising physicians. To account for these inconsistencies, the staff schedules in the test instances were manually corrected to better reflect actual operating conditions.

Patient prioritization

In the real ED, patient prioritization is primarily based on triage severity codes. However, the process is not strictly deterministic: physicians continuously re-evaluate the waiting queues to prevent excessive delays for less severe patients. In practice, this means that while higher-severity patients are generally prioritized, lower-severity patients are not systematically postponed. Without such a dynamic adjustment, minor cases would almost always find patients of higher severity ahead of them, potentially leading to unacceptably long waiting times. To replicate this real patient selection dynamic, a hybrid priority strategy was implemented, combining elements of pure priority and accumulating priority (Cildoz et al., 2019). In the followed approach, the priority score of a patient p waiting in a queue at time t is defined as

$$\text{priority}(p, t) = B_{d_p} + k_{d_p} QT_p^t \tag{3.1}$$

where d_p denotes the severity level of patient p, B_{d_p} is a baseline priority assigned to all patients having that severity level, k_{d_p} is a severity-dependent coefficient, and QT_p^t is the time already spent by patient p in the queue at observation time

t. While 3.1 is formulated in reference to a patient, in the simulation model the formula is generalized to the instances of `ServiceRequest` generated by a patient and associated with them.

In this formulation, both B_{d_p} and k_{d_p} are tunable parameters that determine how severity and waiting time interact in shaping prioritization. Accordingly, the queue management strategies $\phi_{standardPolicy}$ and $\psi_{standardPolicy}$ (referenced in lines 61 and 9 of Algorithm 6 and Algorithm 7) are implemented by sorting patients (or service request) based on this priority(p, t).

Validation

Consistently with previous works on DES applied to ED and with validation standards (Doudareva & Carter, 2022), the primary validation metric used was the LOS of patients, compared with real-world observations, followed by throughput –the number of patients being discharged in the referenced period– as a secondary metric. Two weeks of historical data were selected as testing instances, spanning from 2024-04-08 to 2024-04-21. This same time interval will be used in Chapter 6 for the final evaluation of the entire framework.

Table 3.1 reports the mean LOS values observed in the real ED and in the simulated ED, together with the coefficients of variation ($CV = \frac{\sigma}{\mu}$, with μ the mean value and σ the standard deviation). Table 3.2 reports a comparative analysis on the throughput. Trends in terms of the number of patients simultaneously present in the system also showed consistent results. Given the small deviations between simulated and observed values, both at the aggregate level and stratified by severity, the comparison indicates that the developed simulation provides a reliable approximation of the real ED dynamics.

Table 3.1: Comparison of LOS values (minutes) between historical observations and DES outputs.

Severity level	**Mean LOS real ED**	**CV real ED**	**Mean LOS simulated ED**	**CV simulated ED**
All patients	397.2	63.9%	397.8	58.6%
1	302.6	71.9%	307.2	70.6%
2	339.6	74.1%	334.3	69.6%
3	439.9	61.9%	441.1	57.0%
4	423.4	54.4%	425.1	48.8%
5	296.5	64.2%	302.7	48.4%

Table 3.2: Comparison of throughput (number of patients discharged) between historical observations and DES outputs.

Severity level	**Discharged patients real ED**	**Discharged patients simulated ED**
All patients	2302	2288
1	97	95
2	528	524
3	828	822
4	715	709
5	134	138

4 Reinforcement Learning Problem Formulation

This chapter introduces the mathematical formulation of the problem under study as an RL task. Section 4.1 recalls the theoretical background on RL problems, starting from Markov Decision Processes (MDPs) and then presenting the concepts of policy and discounted return. Section 4.2 applies the MDP formulation to the ED case study, providing a precise definition of the state, action, and reward components. Finally, Section 4.3 describes how the resulting MDP formulation was embedded into the DES introduced in Chapter 3.

4.1 Reinforcement Learning problems

RL problems are typically modeled within the framework of MDPs. Building on this foundation, RL introduces the additional notions of *policy* and *discounted return*. This section recalls these concepts, which form the basis of the learning algorithms that will be discussed in Chapter 6, using the contents of Sutton and Barto, 2018 as a main reference.

4.1.1 Markov Decision Processes

An MDP provides a general mathematical framework for modeling sequential decision-making problems under uncertainty. In these types of problems, actions influence not only immediate rewards, but also subsequent situations (states), and through those also future rewards. RL methods are naturally formulated within this framework.

An MDP is defined by four main components:

- **State space** S: the space (continuous or discrete) of all possible configurations of the system. A state $S_t \in S$ represents the system condition at time t.

- **Action space** A: the set of all possible decisions available to the agent. An action $A_t \in A$ represents the decision taken at time t, which influences the system evolution.
- **Reward function** $R_a(s, s')$: a numerical feedback received after transitioning from state s to s' due to action a.
- **Transition probabilities** $P_a(s, s')$: the probability of transitioning from s to s' under action a.

The agent-environment interaction –defined by state observation, action selection and reward feedback– progresses through discrete timesteps, which typically correspond to fixed time intervals, or may be marked by specific events that require decisions. This type of formulation captures the structural elements of decision-making under uncertainty. Building on top of the MDP framework, RL introduces two additional key notions: the policy, which defines the agent's behavior, and the discounted return, which formalizes the long-term objective to be optimized.

4.1.2 Policy and discounted returns

The primary goal of an RL problem is to train an agent to learn a policy, formalized as a function

$$\pi : S \to A$$

mapping from states to a probability distribution over the action space. Under this definition, $\pi(a|s)$ denotes the probability of the agent selecting action $A_t = a$ when $S_t = s$.

A central challenge in designing and training policies lies in balancing the trade-off between *exploration* and *exploitation*. Exploitation refers to selecting actions that the agent already knows produce high rewards, whereas exploration involves trying less certain or previously untested actions to potentially discover better strategies. RL algorithms must balance these two aspects: focusing solely on exploitation may trap the agent in suboptimal behaviors, while excessive exploration may slow down learning by repeatedly trying poor actions.

An effective policy is one that maximizes not only the immediate reward, but also the cumulative reward collected over time. To capture this long-term perspective, RL introduces the concept of *return*, defined at time t as the sum of all subsequent rewards:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots$$

Since many RL environments are continuing tasks where rewards can accumulate indefinitely, a discount factor $\gamma \in [0, 1]$ is introduced to attenuate the contribution of future rewards:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Even in episodic environments with a finite horizon T, the discounted formulation is commonly adopted. In such cases, the return can be equivalently written as

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1},$$

where T denotes the terminal timestep of the episode. The choice of γ balances the trade-off between short-term and long-term gains: values of γ closer to 0 make the agent myopic, focusing mostly on immediate rewards, while values closer to 1 encourage planning further into the future. Moreover, the discount factor also plays a crucial role in the study of stochastic and noisy environments, as reducing γ can act as a regularization mechanism, effectively mitigating the variability of returns introduced by uncertain transitions (Parr et al., 2024).

In summary, the notions of policy and discounted return provide the mathematical backbone of RL, defining how agents evaluate and select actions over time. Building on these foundations, the next section introduces an MDP formulation for the ED case study, highlighting how states, actions, and rewards are represented in this context.

4.2 A Markov Decision Process formulation for the Emergency Department case study

In stochastic and highly complex environments such as an ED, the state transition probabilities cannot be accurately represented for each state and action, due to a high number of possible configurations of the system. For this reason, transition samples obtained through simulation are studied to handle the dynamics of the system. With regard to the other components described in 4.1.1:

- the **State** should encompass all the relevant information for monitoring the ED condition such as the patient count, their characteristics, the number of physicians, the time of the day;
- the **Action space** should reflect queue management decisions, such as rules and criteria for assigning priorities to different patients;

- the **Reward** should be a function of the desired objective, that is the reduction of LOS, and consequently ED crowding level.

The following sections provide further explanations and a mathematical formulation for these components, along with the required notation.

Notation

C_t system clock at timestep t
M_t number of physicians on duty at timestep t
P_t set of patients admitted and still in the system at timestep t
H set of service resources
q_h^t queue of patients waiting for resource h at time t
$d_p \in \{1, ..., D\}$ severity of patient p
$e_p \in \{1, ..., E\}$ triage anamnesis assigned to patient p (main problem category identified for the patient)
$f_p^t \in \{0, 1\}$ pathway phase of patient p at time t, taking value 0 if patient p still has not received the first medical assessment, and value 1 afterwards
$g_p^t \in \{0, 1\}$ variable taking value 1 if patient p is waiting for a physician visit at time t, 0 otherwise (patient waiting for other resources)
PD_p^t probability for patient p at time t of being declared ready for discharge after the following physician visit. This probability is obtained from the predictive model described in Chapter 5, and it is updated after each physician visit received by the patient
τ_p admission time of patient p
$w_i \in (0, 1]$ weight coefficient associated to a patient of severity i
TL maximum allowed LOS in ED, according to regional standards (see Chapter 1)
TA_i maximum permitted time for the first medical assessment for a patient of severity i, according to regional standards (see Chapter 1)

4.2.1 State

The definition of the state space plays a fundamental role in the MDP formulation, as it determines the extent to which the agent can perceive the conditions of the system and recognize patterns. A richer and more detailed state representation may increase predictive accuracy and provide the agent with more informative signals, but it also introduces complexity during training. Sometimes, it may also introduce additional noise and worsen the training process instead of improving it. Striking the right balance between expressiveness and tractability is therefore essential, and motivates

the formulation presented below. Several formulations were explored, starting from an initial highly detailed but impractical representation, which was progressively refined toward a more tractable state space that preserves essential information while maintaining a suitable balance between expressiveness and learning stability.

The first piece of information that is traced is the number of medical resources (M_t) and time of the day (C_t). The characteristics of patients (d_p, e_p, f_p^t, g_p^t) are then used to separate them in groups. While severity d_p and triage anamnesis e_p are considered as fixed parameters (assigned at patient admission), the pathway phase f_p^t may change and it is used to separate patients that still have to receive the first medical assessment ($f_p^t = 0$) from the ones that have been visited at least once ($f_p^t = 1$). Furthermore, the value g_p^t indicates whether the patient is waiting for a physician visit at time t. Patients in the system can therefore be separated into subsets, based on the value of these patient features:

$$\mathcal{P}_{ijkl}^t = \{p \in P_t : d_p = i, e_p = j, f_p^t = k, g_p^t = l\} \tag{4.1}$$

To support the state description, three tensors $\mathcal{N}_t$, $\widehat{\mathcal{N}}_t$, $\mathcal{L}_t$ are introduced, the values of which are defined by

$$\mathcal{N}_{ijkl}^t = \frac{|\mathcal{P}_{ijkl}^t|}{|P_t|} \tag{4.2}$$

$$\widehat{\mathcal{N}}_{ijkl}^t = \frac{|\{p \in \mathcal{P}_{ijkl}^t : (C_t - \tau_p > TL) \vee ((C_t - \tau_p > TA_{d_p}) \wedge (f_p^t = 0))\}|}{|P_t|} \tag{4.3}$$

$$\mathcal{L}_{ijkl}^t = \frac{\sum_{p \in \mathcal{P}_{ijkl}^t} (C_t - \tau_p)}{|\mathcal{P}_{ijkl}^t|\, TL} \quad \text{if } |\mathcal{P}_{ijkl}^t| > 0,\ 0 \text{ otherwise} \tag{4.4}$$

The tensor $\mathcal{N}_t$, defined by (4.2), separates portions of patients based on the four variables aforementioned, while the tensor $\widehat{\mathcal{N}}_t$ defined by (4.3) highlights patients who have exceeded either the target maximum waiting time for the first medical assessment, or the target maximum LOS. In addition, the tensor $\mathcal{L}_t$ defined by (4.4) is used to represent the mean time spent in the system (current LOS) for each group of patients as previously described.

With respect to the prediction of the discharge probability (PD_p^t), a tensor $\mathcal{D}_t$ is defined, the values of which represent, for each patient group, the probability of at least one patient being discharged after the following physician visit, as

$$\mathcal{D}_{ijkl}^t = 1 - \prod_{p \in \mathcal{P}_{ijkl}^t} (1 - PD_p^t) \quad \text{if } |\mathcal{P}_{ijkl}^t| > 0,\ 0 \text{ otherwise} \tag{4.5}$$

Finally, a tensor $\mathcal{Q}_t$ monitors the saturation of the queues associated with each service resource, the values of which are defined by

$$\mathcal{Q}^t_{ih} = |\{p \in q^t_h : d_p = i\}| \tag{4.6}$$

The state of the system at timestep t can therefore be represented as the tensor resulting from the concatenation of the previously described values and tensors:

$$S_t = \Big(C_t,\ M_t,\ |P_t|,\ \mathcal{N}_t,\ \widehat{\mathcal{N}}_t,\ \mathcal{L}_t,\ \mathcal{D}_t,\ \mathcal{Q}_t\Big) \tag{4.7}$$

4.2.2 Action

The action space gives the actor a set of options that can impact the priority assigned to patients, and consequently queue management. Experiments were conducted with different formulations, including both continuous and discrete action spaces. The final chosen formulation is a discrete space in which each action represents different ways in which system queues can be ordered, considering available patients features (severity, pathway phase, time spent in the system, probability of being discharged). The action space is represented through a set of integer numbers

$$A = \{0, ..., N\} \tag{4.8}$$

Section 4.3 will outline how the formulated action space was integrated into the simulated environment.

4.2.3 Reward

The design of the reward function represents one of the most critical aspects of the problem. In line with the formulated objective, the reward function must capture the level of crowding in the ED. To this end, a function $\lambda(p, t)$ is introduced, which quantifies the contribution of each patient $p \in P_t$ to the overall system pressure. The resulting reward function to be maximized is therefore defined as

$$R_t = -\sum_{p \in P_t} \lambda(p, t) \tag{4.9}$$

Several formulations for $\lambda(p, t)$ were tested, following an empirical approach through the training experiments described in Chapter 6. As a first step, simple versions of the weighting function, independent of the time component (formally $\lambda(p)$), were considered. Initially, all patients were assigned an equal weight of 1, so that

$\sum_{p \in P_t} \lambda(p) \equiv |P_t|$. Under this definition, the trained agent exhibited a biased behavior: it tended to prioritize patients with lower severity, since their typically shorter clinical pathways made it easier to accelerate their discharge and thus reduce the number of patients in the system. Even when different static weights based on patient severity were introduced, the resulting behavior of the agent remained unsatisfactory. It therefore became necessary to reformulate $\lambda(p,t)$ so that it accounted for both patient characteristics and their time spent in the system, with particular emphasis on compliance with:

- Regional target times for the initial medical assessment, dependent on patient severity;
- Regional target maximum LOS for each patient (see Chapter 1).

To model these aspects, two auxiliary variables are introduced:

$$\alpha(p,t) = \begin{cases} 1 & \text{if } (C_t - \tau_p > TA_{d_p}) \ \wedge \ (f_p^t = 0) \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

$$\beta(p,t) = \begin{cases} 1 & \text{if } C_t - \tau_p > TL \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

These variables are then used as coefficients to determine the pressure applied by a patient on the system, depending on their waiting time and LOS. Specifically, (4.10) is set to 1 when the patient has exceeded the maximum waiting time before the first medical assessment, while (4.11) is set to 1 when the patient has exceeded the maximum LOS.

Accordingly, the contribution of each patient to the reward can be reformulated as

$$\begin{aligned} \lambda(p,t) = &(1-\alpha(p,t))\,(1-\beta(p,t))\left(w_{d_p}\frac{C_t-\tau_p}{TL}\right) + \\ &+(1-\alpha(p,t))\,\beta(p,t)\left[w_{d_p}\left(2+2\frac{C_t-\tau_p-TL}{TL}\right)\right] + \\ &+\alpha(p,t)\,(1-\beta(p,t))\left[w_{d_p}\left(1+\frac{C_t-\tau_p-TA_{d_p}}{TL}\right)\right] + \\ &+\alpha(p,t)\beta(p,t)\left[w_{d_p}\left(3+2\frac{C_t-\tau_p-TL}{TL}\right)\right] \end{aligned} \tag{4.12}$$

To avoid the reward function assuming excessively large values –which could lead to training instabilities– the contribution of each patient is clipped. The final reward

Table 4.1: Summary of the target maximum waiting time for the first medical assessment, along with the chosen weight coefficients for each severity level

Severity (d_p)	TA_{d_p} **(minutes)**	w_{d_p}
1	0	1
2	15	0.8
3	60	0.6
4	120	0.525
5	240	0.5

function thus becomes:

$$R_t = -\sum_{p \in P_t} \text{clip}\left(\lambda(p,t);\ 0,\ 4w_{d_p}\right) \tag{4.13}$$

Figure 4.1, based on the weighting values reported in Table 4.1, illustrates the behavior of the function clip $\left(\lambda(p,t);\ 0,\ 4w_{d_p}\right)$ before and after the patient has been evaluated by a physician.

This formulation produces a piecewise-linear penalty that reflects clinical soft constraints, and it encourages the agent to balance the overall patient LOS with a timely initiation of care. By assigning progressively higher penalties once the predefined thresholds are exceeded, the reward function steers the agent towards strategies that keep the system within the regions of minimal penalty. In practice, this means that the optimal behavior does not simply minimize congestion, but rather ensures a compromise between efficiency and compliance with clinical time targets.

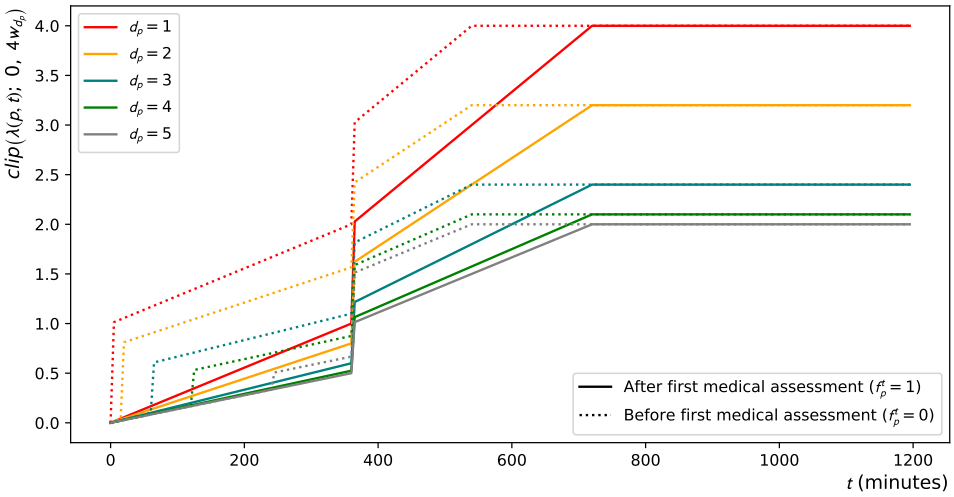

Figure 4.1: Comparison between penalty contributions, considering both patient severity and clinical pathway phase

4.3 Integration between the Discrete Event Simulation and the Reinforcement Learning model

The DES formulated in Chapter 3 already provides most of the data infrastructure required to serve as an environment for an RL agent. In Subsection 4.2.2, the action space is defined as a set of integer values from 0 to N. The queue management dynamics are now formalized by introducing two sets of sorting functions:

$$\Phi = \{\phi_0, ..., \phi_N\} \tag{4.14}$$

$$\Psi = \{\psi_0, ..., \psi_N\} \tag{4.15}$$

Functions in Φ take as input a set of patients and sort them according to different criteria (combinations of patient features). Similarly, functions in Ψ operate on service requests, sorting them based on the characteristics of the patients from which they originate.

The action space thus acts as an indexing set: each action $a \in \{0, ..., N\}$ corresponds to a pair (ϕ_a, ψ_a). In this way, the agent's choice directly influences the processes implemented in the simulation classes `Physician` and `ServiceResourcesQueueManager`, with reference to line 59 and line 7 of their respective pseudocodes Algorithm 6 and Algorithm 7.

Table 4.2 details the sorting functions included in the DES–RL framework. The first column lists the integer identifiers of the action space, while the following four columns correspond to the parameters used for sorting: w_{d_p}, PD_p^t, f_p^t and $C_t - \tau_p$.

Arrows indicate whether a parameter is considered in ascending order ($\Uparrow$) or descending order ($\Downarrow$). The number preceding the arrow specifies the sequence in which parameters are applied. A dash ($-$) indicates that the parameter is not considered in the sorting.

For instance, action $a = 16$ corresponds to a sorting function that prioritizes patients (or service requests) according to ascending values of f_p^t (i.e., patients not yet assessed are considered first), breaking ties by descending values of w_{d_p} (patients with higher severity are prioritized), and further breaking ties by descending values of $C_t - \tau_p$ (patients who have spent longer in the system are considered first). Reducing redundancy in the action space required the exclusion of sorting functions that, due to the predominance of a single parameter with high variability as a primary sorting element, would have produced essentially equivalent rankings to others already defined. In such cases, the contribution of secondary parameters as tie-breakers becomes negligible, since ties are statistically rare. Moreover, this choice also prevents the definition of an excessively large action space, which would increase training complexity and slow down convergence. The final set of actions therefore reflects a compromise between ensuring representative diversity, avoiding functional redundancy, and maintaining tractability in the learning process.

Table 4.2: Mapping between the action space and the corresponding sorting functions.

Selected action (a)	Φ_a / Ψ_a			
	w_{d_p}	PD_p^t	f_p^t	$C_t - \tau_p$
0	1 ⇓	2 ⇓	–	–
1	1 ⇓	2 ⇑	–	–
2	1 ⇑	2 ⇓	–	–
3	1 ⇑	2 ⇑	–	–
4	2 ⇓	3 ⇓	1 ⇑	–
5	2 ⇓	3 ⇑	1 ⇑	–
6	2 ⇑	3 ⇓	1 ⇑	–
7	2 ⇑	3 ⇑	1 ⇑	–
8	2 ⇓	3 ⇓	1 ⇓	–
9	2 ⇓	3 ⇑	1 ⇓	–
10	2 ⇑	3 ⇓	1 ⇓	–
11	2 ⇑	3 ⇑	1 ⇓	–
12	1 ⇓	–	–	2 ⇓
13	1 ⇑	–	–	2 ⇓
14	1 ⇓	–	–	2 ⇑
15	1 ⇑	–	–	2 ⇑
16	2 ⇓	–	1 ⇑	3 ⇓
17	2 ⇑	–	1 ⇑	3 ⇓
18	2 ⇓	–	1 ⇑	3 ⇑
19	2 ⇑	–	1 ⇑	3 ⇑
20	2 ⇓	–	1 ⇓	3 ⇓
21	2 ⇑	–	1 ⇓	3 ⇓
22	2 ⇓	–	1 ⇓	3 ⇑
23	2 ⇑	–	1 ⇓	3 ⇑
24	–	1 ⇓	–	–
25	–	1 ⇑	–	–
26	–	2 ⇓	1 ⇑	–
27	–	2 ⇑	1 ⇑	–
28	–	2 ⇓	1 ⇓	–
29	–	2 ⇑	1 ⇓	–
30	–	–	–	1 ⇓
31	–	–	–	1 ⇑
32	–	–	1 ⇑	2 ⇓
33	–	–	1 ⇑	2 ⇑
34	–	–	1 ⇓	2 ⇓
35	–	–	1 ⇓	2 ⇑

5 A Deep Learning Model for Estimating Patient Discharge Likelihood throughout their Clinical Pathway

This chapter describes the predictive model used to support the state description and action formulation of the MDP introduced in Chapter 4. Section 5.1 introduces the purpose of the model. Section 5.2 is dedicated to the description of the training process and model architecture. Finally, section 5.3 concludes the chapter with the evaluation of the trained model.

5.1 Purpose of the predictive model

The intuition behind the model is to obtain an estimation, at an individual level, of the probability that a patient –given their clinical characteristics and current stage in the clinical pathway– will be discharged following their next physician visit. According to the ED process introduced in Chapter 1, each physician encounter (either the first medical assessment or the following checkups) may result in the patient being declared ready for discharge, or in the assignment of additional services. The model was therefore designed to predict the probability of discharge after each visit. It is formulated as a binary classification task with two possible outcomes: DISCHARGE and NOT DISCHARGE. Patient clinical and demographic features, together with information on the current stage of their clinical pathway (i.e., previously completed exams), are provided as input to the model. The output represents the model's confidence in assigning the patient to one of the two classes (Bishop, 2006). Rather than limiting it to a strict binary classification, this output can be directly interpreted as a continuous probability. Such a formulation allows its integration into the framework introduced in Chapter 1, where it enriches the

state representation and implementation of actions with additional information on the likelihood of discharge of the patients crowding the system.

5.2 Model architecture, hyperparameters and training process

The model takes as input a snapshot of a patient's state in the system, consisting of demographic and clinical parameters together with the set of services already prescribed (clinical pathway stage). The reference datasets are *clinical_pathway*, *demographics*, and *triage* tables, whose preprocessing steps were described in Section 2.2.

Categorical features (*Admission time slot*, *Age*, *Body temperature*, *Systolic blood pressure*, *Diastolic blood pressure*, *Isolated systolic hypertension*, *Heart rate*, *Respiratory rate*, *Oxygen saturation*, *Glasgow Coma Scale*) were encoded using one-hot encoding, since their values were discretized into classes. The *Main identified problem* (see Table B.1), was one-hot encoded as well. Binary encoding was applied to *Sex* and *Type of arrival*. The *Severity* variable was normalized using a Min–Max scaler. To represent previously assigned exams, one column per exam type was added; for each patient instance, this column contained the cumulative count of exams of that type up to the considered pathway stage. This encoding resulted in a final input vector of size 289.

The binary classification task was implemented through a DNN in PyTorch (PyTorch contributors, 2016). The network consists of three fully connected hidden layers with 256, 128, and 64 neurons, respectively, each followed by a ReLU activation. The output layer consists of a single neuron with a Sigmoid activation, producing a value between 0 and 1 interpretable as the probability of imminent patient discharge.

The overall dataset comprised 402,015 samples, corresponding to different stages of patient pathways, considering admissions between January 1, 2021 and December 31, 2023. Samples beyond this time frame were excluded in this phase, as they were reserved for the RL experiments described in Chapter 6. The dataset was quite balanced, with 45% of the examples belonging to class DISCHARGE and the remainder belonging to NOT DISCHARGE. A test set (15% of the samples) was separated from the dataset. The remaining data were further split into training (85%) and validation (15%) subsets.

To mitigate overfitting, regularization techniques were applied, including Dropout on each hidden layer (Hertz et al., 1991; Hinton et al., 2012) and weight decay (L2

regularization) (Goodfellow et al., 2016). Multiple training experiments were conducted under different hyperparameter configurations, using the Adam optimizer (Kingma & Ba, 2017). Results were consistent across runs in terms of accuracy and loss. Table 5.1 summarizes the final network architecture and selected hyperparameters. Figure 5.1 reports the learning curves on the training and validation sets, showing that training was stopped after 15 epochs to prevent overfitting.

Table 5.1: Hyperparameters and network architecture

Network architecture	
Input dimension	289
Hidden layers	3 fully connected
Neurons per layer	256, 128, 64
Hidden layers activation function	ReLU
Output layer	Sigmoid
Training hyperparameters	
Learning rate	5×10^{-5}
Dropout	0.4 on each hidden layer
Weight decay	1×10^{-4}
Minibatch size	32
Training epochs	15

5.3 Model evaluation

Although the predictive model was ultimately integrated into the simulation framework as a probabilistic estimator –providing continuous confidence values between 0 and 1– it was trained as a binary classifier. Consequently, its performance can be assessed using standard evaluation metrics for binary classification tasks, as commonly reported in the literature (Canbek et al., 2022). This dual perspective ensures both interpretability within the RL framework, where continuous probabilities enrich the state description, and comparability with existing ML approaches.

Model performance was evaluated on the test set using accuracy, precision, recall, F1-score, and the receiver operating characteristic (ROC) curve with its corresponding area under the curve (AUC). These metrics derive from different combinations

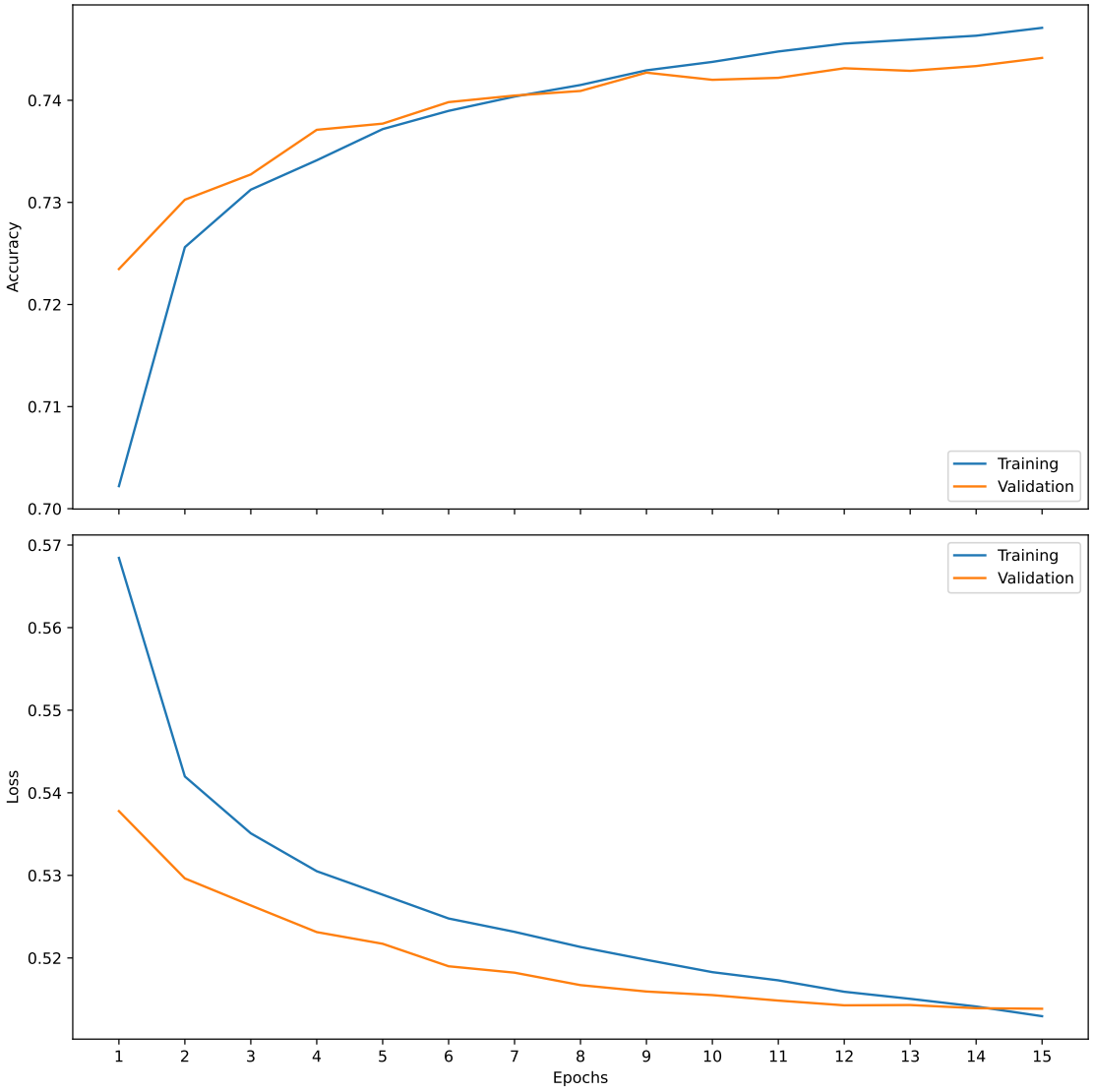

Figure 5.1: Training and validation curves for accuracy and loss. Early stopping was applied after 15 epochs to prevent overfitting.

Table 5.2: Performance metrics of the trained binary classifier

Metric	**Formula**	**Value**
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	74.5%
Precision	$\frac{TP}{TP+FP}$	78.9%
Recall	$\frac{TP}{TP+FN}$	73.6%
F1 Score	$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$	76.2%

of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).

Figure 5.2 reports the confusion matrix obtained by applying the trained model to the test set. The matrix was normalized row-wise to highlight, for each class, the proportion of correct and incorrect predictions. Table 5.2 summarizes the quantitative results for accuracy, precision, recall, and F1-score, while Figure 5.3 presents the ROC curve along with the achieved AUC.

Overall, the model achieves a balanced trade-off between precision and recall, with an F1-score of 76.2%. The ROC–AUC of 0.82 further confirms the model's discriminative ability between discharge and non-discharge classes. These results demonstrate that the trained classifier is effective in predicting patient discharge likelihood and is therefore suitable for integration into the RL framework.

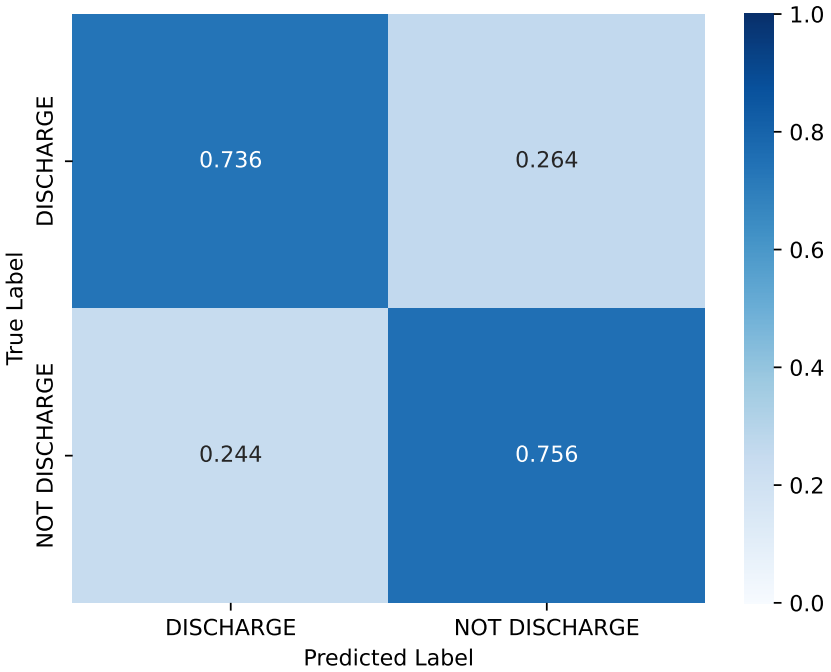

Figure 5.2: Confusion matrix describing the DL model performance over the test set (normalized row-wise).

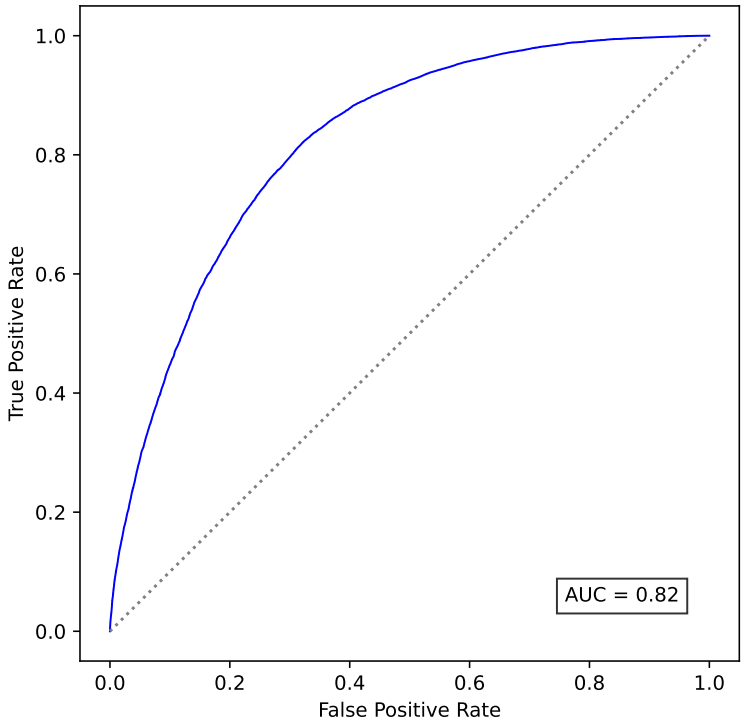

Figure 5.3: ROC curve of the trained DL model on the test set, with the dashed line indicating random classification.

6 Implementation of Reinforcement Learning Algorithms

The aim of this chapter is to describe the two classes of algorithms implemented and tested, using the DES described in Chapter 3 as an environment and following the MDP formulation introduced in Chapter 4 (supported and enriched by the predictive model described in Chapter 5). Section 6.1 opens the chapter by briefly recalling some theoretical concepts about the family of algorithms considered. Section 6.2 proceeds with the description of how the two selected algorithms have been applied to the case study, followed by section 6.3 in which a comparative analysis of the results is conducted. Finally, section 6.4 is dedicated to the discussion of the obtained results.

6.1 Policy gradient methods and Actor-Critic algorithms

This section serves as a brief theoretical background for the policy gradient methods and the Actor-Critic algorithms, using the concepts described in Konda and Tsitsiklis, 1999, Mnih et al., 2016, and Sutton and Barto, 2018 as the main reference.

Most classical RL algorithms are value-based, meaning they aim to estimate a value function –such as the action-value function $Q(s, a)$– which determines the following return that can be obtained by applying a certain action in a specific state. The policy in this case is derived indirectly by acting in a greedy manner with respect to these estimates. A well-known example of this family is Q-learning, in which the learning process focuses on approximating the expected long-term return of states or actions and then selecting those with maximal value.

Policy gradient methods, instead, parametrize the policy –for example, using a DNN– and adjust its parameters in a direction that maximizes the return. An important feature of policy gradient methods is that they naturally operate with

stochastic policies. Unlike deterministic policies, which are formulated to select the same action for a given state, stochastic policies define a probability distribution over the action space. This provides several advantages:

- Exploration intrinsically embedded into the learning process, since the agent acts by following a probability distribution over all the possible actions;
- Greater robustness in noisy or partially observable environments, as the risk of systematically committing suboptimal actions is reduced;
- The probabilistic and differentiable nature makes them particularly well-suited for gradient-based optimization.

A limitation of vanilla policy gradient methods is their often high variance in gradient estimates, which can lead to unstable learning and slow convergence. Actor-Critic methods were introduced to overcome this issue, by combining two complementary components:

- the **Actor**, which, by observing a state, is responsible for action selection, following the policy;
- the **Critic**, which, given a state as input, estimates a value function to guide and stabilize the Actor updates.

A key tool in these methods is the advantage function, defined as the difference between the discounted return –computed from the actual rewards obtained by following the policy– and the estimated value function. Intuitively, the advantage quantifies how much better (or worse) an action is compared to the average expectation from that state. Since policy gradient methods work with stochastic policies, the update rule does not deterministically replace one action with another. Instead, it shifts the probability distribution: actions with a positive advantage have their selection probability increased, while those with a negative advantage have their probability decreased. In this way, the agent gradually shifts its stochastic policy towards "good" actions while steering away from "bad" ones, reducing gradient variance in the process. The following section is dedicated to presenting the implementation of two different Actor-Critic algorithms to the problem discussed in the previous chapters.

6.2 Implementing Actor-Critic algorithms on the Emergency Department case study

The ED is characterized by an intrinsically stochastic nature:

- Arrivals of patients and demand peaks are only partially predictable;
- Patients with similar clinical characteristics may still present heterogeneous needs;
- In the same way, the duration of visits and other services may vary, depending on the patient and on the provider.

When reproduced in simulation, these dynamics produce a noisy and stochastic environment –even if, in the presented case, randomness is only emulated using heterogeneous historical data. As a result, both rewards and state transitions may differ, even when the RL agent is presented with similar initial conditions. This motivates the adoption of Actor-Critic methods, based on their characteristics described in Section 6.1.

Within this family, two methods were selected for this work: Advantage Actor–Critic (A2C) and Proximal Policy Optimization (PPO). Both are policy gradient approaches with variance reduction via advantage estimation, yet they differ in the way they deal with training stability. These two algorithms will be further detailed in the following subsections.

6.2.1 Advantage Actor-Critic (A2C) implementation

The A2C algorithm can be considered as the synchronous version of the Asynchronous Advantage Actor–Critic (A3C) (Mnih et al., 2016; Schulman et al., 2017). A3C employs multiple independent workers that interact asynchronously with separate environment instances, performing non synchronized updates on the policy. A2C, instead, aggregates trajectories collected synchronously before applying a centralized update.

In this work, a simplified single-agent configuration was adopted, representing a special case of the A2C framework. Some adjustments were introduced to stabilize policy updates. Actor and Critic updates are decoupled: the Critic is updated at the end of each episode, while the Actor is updated only after a fixed number of episodes (*updateFreq*). This strategy ensures that the Actor collects sufficiently diverse trajectories before each update, thereby reducing variance and preventing

oscillatory behaviors (Kochenderfer et al., 2022). During training, the Critic quickly adapts to new trajectories by updating at every episode, providing accurate value estimates. The Actor, on the other hand, performs batch updates after accumulating experience from multiple episodes. This formulation thus results in an A2C implementation where update frequencies are deliberately unbalanced between the two components. The instances of the environment presented to both the Actor and the Critic correspond to the simulation of two days in the ED, starting from a given date. Two days are presented instead of just one, to enrich the agent experience with a more detailed impact of the selected actions on a longer term. The starting date is updated after each episodic iteration, sampling it from a finite buffer.

Another technique that was employed for training the Actor is entropy regularization. By adding an entropy bonus as an additional term in the Actor loss function, exploration is encouraged, preventing premature convergence to suboptimal deterministic policies (Haarnoja et al., 2017, 2018, 2019; Liu et al., 2020). This ensures that the policy maintains sufficient stochasticity throughout training, balancing exploration and exploitation.

Algorithm 9 presents the implementation of A2C to the ED case study. The inputs include an episode limit E_{max}, a timestep limit T, Actor update frequency *updateFreq*, learning rates for Actor and Critic, discount factor γ and the entropy coefficient c. Actor and Critic are implemented as two DNNs, developed within the PyTorch framework. Table 6.1 summarizes the architecture of the networks, together with the hyperparameters used in training.

The Actor and the Critic networks are first initialized with random parameters (line 2). A set called *TrainingBuffer* is then created to store the starting dates for the simulation episodes, corresponding to the training period under consideration (line 3). To avoid exposing the Actor and the Critic to identical sequences of episodes, the buffer is shuffled (line 4). The algorithm proceeds within a while loop, running until the maximum episode limit is reached or until early stopping is triggered. If the *TrainingBuffer* becomes empty, it is refilled with the original dates, and reshuffled (lines 7–10). At each iteration, one date is drawn from the buffer and used to initialize a simulation instance. For a number of timesteps corresponding to 48 hours in the ED, the agent executes the state-action-reward cycle –with the agent-environment interaction occurring every 15 simulated minutes–, while also collecting value estimates, log-probabilities of the chosen actions and entropy values of the current policy (lines 14–23). At the end of each episode, discounted returns are computed for all timesteps, and advantages are obtained by subtracting the collected value estimates (lines 24–27). After every *updateFreq* episodes, an Actor

Algorithm 9 A2C implementation, with decoupled training of Actor and Critic and entropy bonus

Input: simulated ED environment with MDP interface, historical data from a selected period as input for the simulation, a timestep threshold T, an episode threshold E_{max}, an Actor updating frequency $updateFreq$, an Actor DNN (policy network), a Critic DNN (value network), learning rates for Actor lr_{actor} and learning rate for Critic lr_{critic}, discounting factor γ, entropy coefficient c
Initialize Actor DNN and Critic DNN with random parameters θ_{actor} and θ_{critic}
Initialize *TrainingBuffer* as a set of dates, corresponding to the selected period of historical data
Shuffle *TrainingBuffer*
Initialize episode counter $e := 1$
while $e \leq E_{max}$ **or until** early stopping **do**
    if *TrainingBuffer* is empty **then**
        Refill *TrainingBuffer* with the same dates as the first input
        Shuffle *TrainingBuffer*
    end if
    Extract one date from *TrainingBuffer*
    Starting from that date, generate a simulation instance from the corresponding historical data
    Initialize timestep counter $t := 1$
    while $t \leq T$ **do**
        Given S_t from the environment
        Get action A_t according to probability $\pi(A_t \mid S_t; \theta_{actor})$ under the current policy, and assign $logprob_t^e := \log \pi(A_t \mid S_t; \theta_{actor})$
        Get value $V_t^e = V(S_t; \theta_{critic})$
        Compute entropy $H_t^e = -\sum_{a \in A} \pi(a \mid S_t; \theta_{actor}) \log \pi(a \mid S_t; \theta_{actor})$
        Perform selected action in the environment, setting currentAction := A_t
        Run 15 minutes of simulated environment
        Observe reward R_t and next state S_{t+1}
        t:=t+1
    end while
    for t=1, ..., T **do**
        Compute discounted return $G_t^e = \sum_{k=t}^{T} \gamma^{k-t} R_k$
        Compute advantage estimate $\delta_t^e = G_t^e - V_t^e$
    end for

28: **if** e mod $updateFreq == 0$ **then**

29: Compute mean value μ_δ and standard deviation σ_δ over the batch of advantages collected in the last *updateFreq* episodes

30: Normalize advantages over the batch ($t = 1, ..., T$ and $i = e - updateFreq, ..., e$):

$$\delta_t^i := \frac{\delta_t^i - \mu_\delta}{\sigma_\delta + \varepsilon} \qquad \text{with } \varepsilon \text{ a small value}$$

31: Compute Actor loss

$$L_{actor} = -\frac{1}{updateFreq}\frac{1}{T}\left(\sum_{i=e-updateFreq}^{e}\sum_{t=1}^{T} logprob_t^i\,\delta_t^i \;+\; c\sum_{i=e-updateFreq}^{e}\sum_{t=1}^{T} H_t^i\right) \tag{6.1}$$

32: Update Actor parameters through a gradient descent step

$$\theta_{actor} := \theta_{actor} - lr_{actor}\nabla_{\theta_{actor}} L_{actor}$$

33: **end if**

34: Compute Critic loss

$$L_{critic} = \frac{1}{T}\sum_{t=1}^{T} \text{smooth}_{L1}\left(V_t^e,\; G_t^e\right) \tag{6.2}$$

35: Update Critic parameters through a gradient descent step

$$\theta_{critic} := \theta_{critic} - lr_{critic}\nabla_{\theta_{critic}} L_{critic}$$

36: e:=e+1

37: **end while**

(policy) update is required. Advantages from the most recent *updateFreq* episodes are normalized to reduce numerical instabilities (lines 29–30). The Actor loss is then computed (line 31), as the negative average, over all episodes and timesteps within the *updateFreq*-episodes batch, of the log-probabilities of the selected actions weighted by their advantages, plus the entropy bonus. Minimizing this loss increases the likelihood of improving actions, while discouraging disadvantageous ones, and simultaneously encourages policy stochasticity through entropy regularization. By computing the gradient on this loss with respect to θ_{actor}, the Actor update is performed through a gradient descent step (line 32). On the other side, the Critic is updated after each episode. Its loss is defined as the Smooth L1 distance between the predicted values and the realized discounted returns (line 34), a regression loss chosen for its robustness to outliers (Girshick, 2015). Minimizing this loss reduces discrepancies between the estimated value and the actual discounted return obtained following the policy. By computing the gradient with respect to θ_{critic} on this loss, the Critic update is performed through a gradient descent step (line 35).

Table 6.1: Hyperparameters and networks architecture used in implementing A2C.

Algorithm hyperparameters		
Discount factor γ	0.96	
Entropy coefficient c	0.001	
Networks architecture and training hyperparameters		
	Actor	**Critic**
Input dimension	State size $\|S\| = 1419$	State size $\|S\| = 1419$
Hidden layers	2 fully connected	2 fully connected
Neurons per hidden layer	2048, 512	2048, 512
Hidden layers activation function	ReLU	ReLU
Output layer	Softmax over $\|A\|$ actions	Linear (state value)
Learning rate	1×10^{-4}	1×10^{-3}
Parameters update frequency (episodes)	10	1

6.2.2 Proximal Policy Optimization (PPO) implementation

PPO extends the principles of Advantage Actor–Critic methods by introducing a more stable update mechanism for the policy network. In standard Actor-Critic implementations, even when variance reduction techniques are applied (e.g., collection of diversified episodes experience and entropy regularization), policy updates may still lead to abrupt changes, potentially destabilizing training and leading the agent to converge to suboptimal behaviors. PPO mitigates this problem by introducing a clipped surrogate objective that constrains the deviation of the updated policy from the previous one, enforcing conservative updates. In addition, PPO performs multiple epochs of stochastic gradient descent over minibatches sampled from the experience buffer, which improves data efficiency compared to single-pass updates. Through this combination of clipped objectives and minibatch optimization, PPO strikes a balance between learning efficiency, training stability, and robustness (Schulman et al., 2017).

Algorithm 10 presents the implementation of PPO in the ED case study. The inputs are aligned with those introduced for A2C in 6.2.1, with the addition of a clipping factor ϵ_{clip}, the minibatch size b and the number of training epochs K (used for the Actor updates). As with A2C, Actor and Critic networks were implemented

as two DNNs, developed within the PyTorch framework. Table 6.2 summarizes the architecture of the networks, together with the hyperparameters used in training.

The two networks are initialized as in the A2C setup, as well as the *TrainingBuffer* (lines 2–4). An additional memory, the *RolloutBuffer*, is introduced to store experience tuples (state - s, action - a, probability distribution - π, discounted return - G, advantage - δ), collected during interaction with the environment (line 5). As in the A2C implementation, the agent interacts with the environment for T timesteps, generating experience (lines 15–25). For each transition, the state, the chosen action, and the probability of that action under the current policy are stored in the *RolloutBuffer* (line 23). After the episode ends, discounted returns and advantage estimates are computed and appended to the *RolloutBuffer* as well (lines 26–30). Every *updateFreq* episodes, an Actor (policy) update is required (line 31). Advantages are first normalized (lines 32–33), then the policy update proceeds by repeatedly ($\frac{T\ updateFreq}{b}$ times) sampling minibatches of size b from the *RolloutBuffer*. For each element in a minibatch, the probability ratio between the current policy and the stored (old) policy is computed. The clipped surrogate objective is then calculated as

$$\min\Big(\text{ratio}_\tau\,\delta_\tau\ ;\ \text{clip}\Big(\text{ratio}_\tau; 1-\epsilon_{\text{clip}},\ 1+\epsilon_{\text{clip}}\Big)\delta_\tau\Big),$$

augmented with an entropy bonus. Minimizing the loss function –equal to the average of this contribution computed on the minibatch samples– corresponds to increasing the likelihood of improving actions while constraining policy updates within a trust region defined by ϵ_{clip}, thus avoiding destabilizing parameter jumps. By computing the gradient on this loss with respect to θ_{actor}, the Actor update is performed through a gradient descent step (line 42). At the end of the Actor update procedure, the *RolloutBuffer* is emptied to collect new trajectories. The Critic update follows the same procedure adopted in the A2C implementation (lines 47–48).

Algorithm 10 PPO implementation, with decoupled training of Actor and Critic and entropy bonus

Input: simulated ED environment with MDP interface, historical data from a selected period as input for the simulation, a timestep threshold T, an episode threshold E_{max}, an updating frequency $updateFreq$, an Actor DNN (policy network), a Critic DNN (value network), learning rates for actor lr_{actor} and learning rate for critic lr_{critic}, discounting factor γ, entropy coefficient c, clipping factor $\epsilon_{\text{clip}} \in (0, 1)$, minibatch size b, training epochs K
Initialize Actor DNN and Critic DNN with random parameters θ_{actor} and θ_{critic}
Initialize *TrainingBuffer* as a set of dates, corresponding to the selected period of historical data
Shuffle *TrainingBuffer*
Initialize *RolloutBuffer* as an empty memory for storing (s, a, π, G, δ) tuples
Initialize episode counter $e := 1$
while $e \leq E_{max}$ **or until** early stopping **do**
    if *TrainingBuffer* is empty **then**
        Refill *TrainingBuffer* with the same dates as the first input
        Shuffle *TrainingBuffer*
    end if
    Extract one date from *TrainingBuffer*
    Starting from that date, generate a simulation instance from the corresponding historical data
    Initialize timestep counter $t := 1$
    while $t \leq T$ **do**
        Given S_t^e from the environment
        Get action A_t^e according to probability $\pi(A_t^e \mid S_t^e; \theta_{actor})$ under the current policy, and assign $\pi_{\text{old}}^{te} := \pi(A_t^e \mid S_t^e; \theta_{actor})$
        Get value $V_t^e = V(S_t^e; \theta_{critic})$
        Compute entropy $H_t^e = -\sum_{a \in A} \pi(a \mid S_t^e; \theta_{actor}) \log \pi(a \mid S_t^e; \theta_{actor})$
        Perform selected action in the environment, setting currentAction $:= A_t^e$
        Run 15 minutes of simulated environment
        Observe reward R_t^e and next state S_{t+1}^e
        Add (S_t^e, A_t^e, π_{old}^{te}) to the *RolloutBuffer*
        t:=t+1
    end while
    for t=1, ..., T **do**
        Compute discounted return $G_t^e = \sum_{k=t}^{T} \gamma^{k-t} R_k$
        Compute advantage estimate $\delta_t^e = G_t^e - V_t^e$
        Add (G_t^e, δ_t^e) to the *RolloutBuffer*
    end for

if e mod $updateFreq == 0$ **then**

Compute mean value μ_δ and standard deviation σ_δ over the advantages collected in the last *updateFreq* episodes

Normalize advantages over the batch ($t = 1, ..., T$ and $i = e - updateFreq, ..., e$):

$$\delta_t^i := \frac{\delta_t^i - \mu_\delta}{\sigma_\delta + \varepsilon} \qquad \text{with } \varepsilon \text{ a small value}$$

for K epochs **do**

for minibatches $m = 1, ..., \frac{T}{b} \frac{updateFreq}{}$ **do**

Sample b random tuples from the *RolloutBuffer* (different for each minibatch)

Let τ be an identifier for the elements indexed t, e in the minibatch m:

for all elements in the extracted minibatch m **do**

Compute $\text{ratio}_\tau = \dfrac{\pi(a_\tau \mid S_\tau;\ \theta_{actor})}{\pi^\tau_{\text{old}}}$

end for

Compute minibatch Actor loss

$$L_{actor} = -\frac{1}{b} \sum_\tau \left(\min\Big(\text{ratio}_\tau\, \delta_\tau,\ \text{clip}\big(\text{ratio}_\tau\ ;\ 1 - \epsilon_{\text{clip}}\ ,\ 1 + \epsilon_{\text{clip}}\big)\, \delta_\tau\Big) + cH_\tau \right) \tag{6.3}$$

Update Actor parameters through a gradient descent step

$$\theta_{actor} := \theta_{actor} - lr_{actor} \nabla_{\theta_{actor}} L_{actor}$$

end for

end for

Empty *RolloutBuffer*

end if

Compute Critic loss

$$L_{critic} = \frac{1}{T} \sum_{t=1}^{T} \text{smooth}_{L1}\left(V_t^e,\ G_t^e\right) \tag{6.2}$$

Update Critic parameters through a gradient descent step

$$\theta_{critic} := \theta_{critic} - lr_{critic} \nabla_{\theta_{critic}} L_{critic}$$

e:=e+1

end while

Table 6.2: Hyperparameters and networks architecture used in implementing PPO.

Algorithm hyperparameters		
Discount factor γ	0.96	
Entropy coefficient c	0.001	
Clipping factor ϵ_{clip}	0.1	
Networks architecture and training hyperparameters		
	Actor	**Critic**
Input dimension	State size $\|S\| = 1419$	State size $\|S\| = 1419$
Hidden layers	2 fully connected	2 fully connected
Neurons per hidden layer	2048, 512	2048, 512
Hidden layers activation function	ReLU	ReLU
Output layer	Softmax over $\|A\|$ actions	Linear (state value)
Learning rate	1×10^{-4}	1×10^{-3}
Parameters update frequency (episodes)	12	1
Training minibatch size	256	full batch
Training epochs	3	1

6.3 Performance comparison and experimental results

This section presents the results obtained from training the A2C and PPO models, as described respectively in 6.2.1 and 6.2.2. First, a comparative analysis of the training performance is discussed, followed by the presentation of the results obtained by testing the trained models.

Both algorithms were trained over five independent runs with different random seeds. Sixty days of historical data, spanning from 2024-01-15 to 2024-03-14, were selected for the *TrainingBuffer*. This time interval was deemed sufficiently long to capture diversity in the ED environment dynamics, providing the agent with enough experience, while maintaining a tractable *TrainingBuffer* size facilitating performance monitoring during training.

Figure 6.1 compares the learning curves of A2C and PPO. The metric reported is the average reward per episode across the set of training episodes, aggregated over

the five training runs of each model. The horizontal axis therefore corresponds to the number of full passes through the set of training episodes. Solid lines indicate the mean, over the five runs, of the simple moving average (window size of 10 full iterations), while the shaded regions correspond to the 95% confidence interval ($\mu \pm 2\sigma$). The red dotted line shows the baseline performance obtained when applying the standard queue management policy (the emulation of the currently adopted strategy), applied to the same training set.

As presented in the figure, the two RL models show quite comparable performances in training, with both of them rapidly outperforming the baseline. A2C exhibits a steeper initial growth, likely due to the unconstrained policy updates. PPO, by contrast, shows slower improvement as a result of its clipped update rule, but in the long run converges to slightly better-performing policies. Although the plot reports results for 500 full iterations (corresponding to 30,000 overall simulated episodes), both models began showing signs of overfitting after approximately 250 iterations. In RL, environment overfitting occurs when the learning algorithm becomes overly adapted to the specific training instances of the environment, and it performs poorly when presented with different environment instances (Whiteson et al., 2011). Existing studies on the ability of RL models to generalize still seem to lack a common evaluation baseline, as they use different environments, configurations of the environment, and experimental protocols (Packer et al., 2019). Even benchmarks performed on the Arcade Learning Environment –a framework encompassing multiple Atari games, a widely adopted play field for RL algorithms–, struggle to fully capture the issue of the agents always encountering near-identical states –i.e., similar behaviors in the confronted environment– (Cobbe et al., 2019; Kirk et al., 2023). Among the strategies to mitigate overfitting, early stopping –limiting the number of policy update iterations– has been recognized as a simple yet effective approach (Ada et al., 2022; Kirk et al., 2023). In line with this evidence, the test performance results reported from this point onward refer to versions of the models trained with early stopping after 250 full iterations (15,000 simulated episodes).

The rest of the section is dedicated to the presentation of the results obtained on the test instances. Consistently with the time period on which the simulation was validated (section 3.3), the trained agents were tested on simulated instances spanning from 2024-04-08 to 2024-04-21. Unlike the training phase, which used two-days episodes, the test phase involved continuous two-weeks simulations, to better reflect the uninterrupted nature of real ED decision-making process.

For each trained agent, 20 test runs were performed, for a total of 100 runs for both A2C and PPO. Performance was compared against the standard policy using

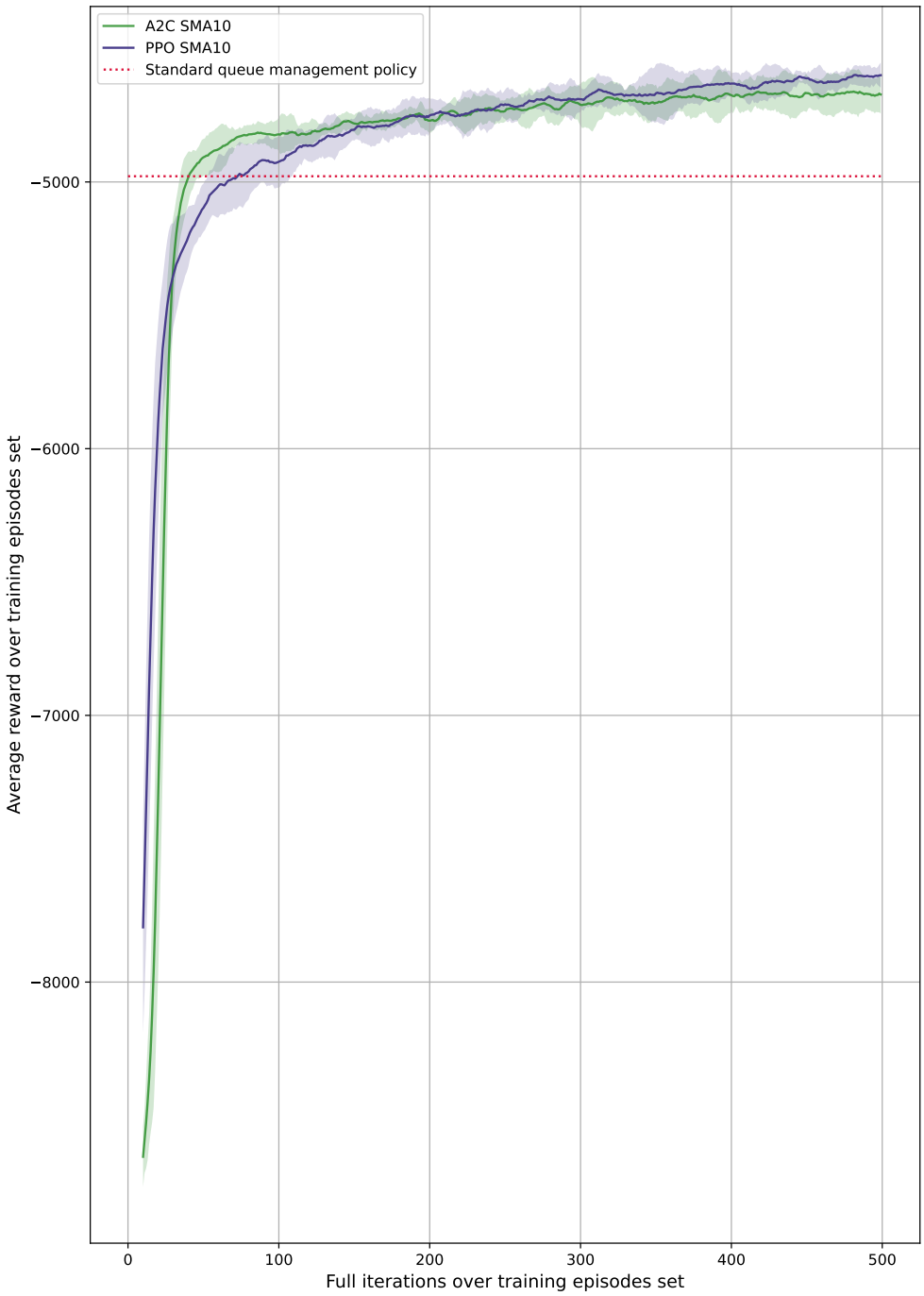

Figure 6.1: Training curves comparing A2C and PPO results

multiple metrics: mean and median LOS, mean and median time for first medical assessment (*door-to-doctor time*, DTDT), throughput, and the number of patients simultaneously crowding the system. Tables 6.3— 6.7 summarize the results reported as averages across the multiple runs, together with coefficients of variation (CV) to capture variability between the runs. Both A2C and PPO reduce mean LOS for high-severity patients, though their behavior slightly diverges at lower severity levels. Median LOS is reduced across all severity levels, suggesting that the slight increases in mean LOS for certain classes can be attributed to some outlier patients. Mean DTDT increases for both models, with PPO showing less worsening than A2C. Median DTDT follows the same trend, except for severity 3 (in both models) and severity 5 (PPO only), where reductions can be observed. Given the DES design, which systematically prioritizes severity 1 patients waiting for the first medical assessment, the small deviations observed in their mean and median DTDT are more likely due to slight differences in event-queueing effects (e.g., activation of physicians) rather than real policy-dependent consequences. Throughput remains almost unchanged across all policies. Figures 6.2 and 6.3 show the trends in the number of patients in the system, respectively comparing A2C and PPO performance with the one of the standard policy. Values were sampled every 15 simulated minutes, corresponding to the agent-environment interaction step. Table 6.8 complements these figures with the average number of patients per day. Both A2C and PPO consistently reduce patient load, with reductions most marked on days of high overcrowding.

In fact, while both agents show an average overall crowding reduction, the effect of the adaptive policies becomes particularly evident during peak congestion moments. This is confirmed by a closer inspection of the most critical days in the test period:

- On 2024-04-08, the maximum occupancy under the standard policy reaches 76 patients, while the adaptive policies reduce this peak to an average of 71.2 (A2C) and 71.6 (PPO). During the most congested time interval (14:00–19:00), the mean number of patients decreases from 72.7 (standard policy) to 68.8 (A2C) and 69.3 (PPO).

- On 2024-04-16, the system under the standard policy reaches a maximum of 84 patients, compared to 76.5 (A2C) and 76.3 (PPO). Over the extended peak period (11:00–20:00), the mean crowding level drops from 75.0 to 66.3 (A2C) and 66.9 (PPO).

- On 2024-04-17, the occupancy peak of 77 patients recorded under the stan-

dard policy is compared with 68.1 (A2C) and 69.4 (PPO). Between 15:00 and midnight, the mean number of patients is reduced from 70.6 to 64.5 (A2C) and 65.5 (PPO).

- On 2024-04-19, the peak under the standard policy reaches 75 patients, compared to 67.2 for both A2C and PPO. Between 13:00 and 20:00, the mean system load falls from 66.7 (standard policy) to 60.5 (A2C) and 61.5 (PPO).

To complete the presentation of the test results, a comparative analysis between the A2C and PPO policies is presented. Due to policy symmetry –where different actions combinations can produce similar system effects– average policies are difficult to represent. For this reason, the best performing agents among A2C and PPO trained models were chosen as representative for comparing their policies. Figure 6.4 presents a comparison between the average number of times an action was selected by each of the two agents throughout the test. As expected, A2C is characterized by a narrower set of preferred actions, whereas PPO distributes its choices more broadly across the action space. A2C focuses primarily on actions that prioritize patients after the first assessment or by severity and time spent in the system, whereas PPO diversifies between these strategies and actions oriented toward early discharge candidates.

The most frequently selected actions under the A2C policy are:

- 34, corresponding to prioritizing patients (and their service requests) that have already received the first medical assessment, breaking ties by descending time spent in the system;
- 12, corresponding to prioritizing patients (and their service requests) from most to least severe, breaking ties by descending time spent in the system;
- 20, corresponding to prioritizing patients (and their service requests) that have already received the first medical assessment, breaking ties by sorting them from the most to the least severe, further breaking ties by descending time spent in the system;
- 24, corresponding to prioritize patients (and their service requests) that are most likely to be discharged after the next physician visit.

Table 6.3: Mean length of stay (minutes) comparing standard queue management policy and the policies obtained from A2C and PPO agents

Severity level	**Mean LOS standard policy**	**Mean LOS A2C policy**	**Mean LOS PPO policy**
All patients	397.8	365.8 ($CV = 1.6\%$)	368.1 ($CV = 1.5\%$)
Severity 1	307.2	288.5 ($CV = 3.4\%$)	296.7 ($CV = 7.0\%$)
Severity 2	334.3	305.6 ($CV = 4.1\%$)	308.3 ($CV = 6.9\%$)
Severity 3	441.1	376.9 ($CV = 4.3\%$)	375.4 ($CV = 3.4\%$)
Severity 4	425.1	418.6 ($CV = 4.0\%$)	428.5 ($CV = 4.1\%$)
Severity 5	302.7	311.5 ($CV = 18.2\%$)	290.5 ($CV = 6.9\%$)

Table 6.4: Median length of stay (minutes) comparing standard queue management policy and the policies obtained from A2C and PPO agents

Severity level	**Median LOS standard policy**	**Median LOS A2C policy**	**Median LOS PPO policy**
All patients	379.0	313.2 ($CV = 4.7\%$)	312.3 ($CV = 3.7\%$)
Severity 1	251.5	235.0 ($CV = 5.1\%$)	242.8 ($CV = 9.1\%$)
Severity 2	274.2	255.8 ($CV = 4.3\%$)	258.7 ($CV = 7.9\%$)
Severity 3	430.3	325.0 ($CV = 6.6\%$)	322.3 ($CV = 4.8\%$)
Severity 4	433.1	368.7 ($CV = 6.4\%$)	375.3 ($CV = 3.5\%$)
Severity 5	313.6	284.5 ($CV = 19.5\%$)	273.5 ($CV = 7.5\%$)

Table 6.5: Mean time to first medical assessment (minutes) comparing standard queue management policy and the policies obtained from A2C and PPO agents

Severity level	**Mean DTDT standard policy**	**Mean DTDT A2C policy**	**Mean DTDT PPO policy**
All patients	90.4	119.9 ($CV = 4.0\%$)	114.3 ($CV = 4.9\%$)
Severity 1	10.2	10.5 ($CV = 2.3\%$)	10.5 ($CV = 2.3\%$)
Severity 2	39.1	43.7 ($CV = 14.9\%$)	38.9 ($CV = 10.2\%$)
Severity 3	100.5	109.2 ($CV = 10.6\%$)	97.9 ($CV = 7.7\%$)
Severity 4	122.9	194.4 ($CV = 9.3\%$)	197.8 ($CV = 8.5\%$)
Severity 5	110.6	165.2 ($CV = 33.2\%$)	139.4 ($CV = 12.2\%$)

Table 6.6: Median time to first medical assessment (minutes) comparing standard queue management policy and the policies obtained from A2C and PPO agents

Severity level	**Median DTDT standard policy**	**Median DTDT A2C policy**	**Median DTDT PPO policy**
All patients	65.3	58.5 ($CV = 11.7\%$)	50.1 ($CV = 10.3\%$)
Severity 1	8.2	8.9 ($CV = 2.6\%$)	8.9 ($CV = 2.8\%$)
Severity 2	22.6	27.6 ($CV = 11.3\%$)	25.3 ($CV = 12.4\%$)
Severity 3	63.8	58.5 ($CV = 24.0\%$)	49.4 ($CV = 16.7\%$)
Severity 4	103.6	133.1 ($CV = 14.2\%$)	124.2 ($CV = 9.4\%$)
Severity 5	112.0	136.3 ($CV = 45.2\%$)	104.8 ($CV = 14.5\%$)

Table 6.7: Number of discharged patients in the target period, comparing standard queue management policy and the policies obtained from A2C and PPO agents

Severity level	**Throughput standard policy**	**Throughput A2C policy**	**Throughput PPO policy**
All patients	2288	2291.1 ($CV = 0.1\%$)	2291.6 ($CV = 0.1\%$)
Severity 1	95	95 ($CV = 0.0\%$)	95 ($CV = 0.0\%$)
Severity 2	524	525.7 ($CV = 0.2\%$)	525.7 ($CV = 0.2\%$)
Severity 3	822	826.4 ($CV = 0.3\%$)	827.1 ($CV = 0.3\%$)
Severity 4	709	707.6 ($CV = 0.4\%$)	706.6 ($CV = 0.4\%$)
Severity 5	138	136.3 ($CV = 1.4\%$)	137.2 ($CV = 1.0\%$)

Table 6.8: Mean number of patients in the system for each day of the simulated test period

Day	**Mean number of patients standard policy**	**Mean number of patients A2C policy**	**Mean number of patients PPO policy**
2024-04-08	53.1	50.7	51.1
2024-04-09	46.8	43.6	44.1
2024-04-10	35.3	33.0	33.2
2024-04-11	37.2	34.1	33.9
2024-04-12	41.4	38.2	38.1
2024-04-13	42.7	39.3	39.0
2024-04-14	34.9	32.2	32.6
2024-04-15	51.7	49.1	49.6
2024-04-16	64.6	56.8	57.6
2024-04-17	56.7	52.0	52.6
2024-04-18	47.7	43.4	43.7
2024-04-19	45.1	41.2	41.7
2024-04-20	37.0	33.9	34.2
2024-04-21	35.9	32.2	32.3

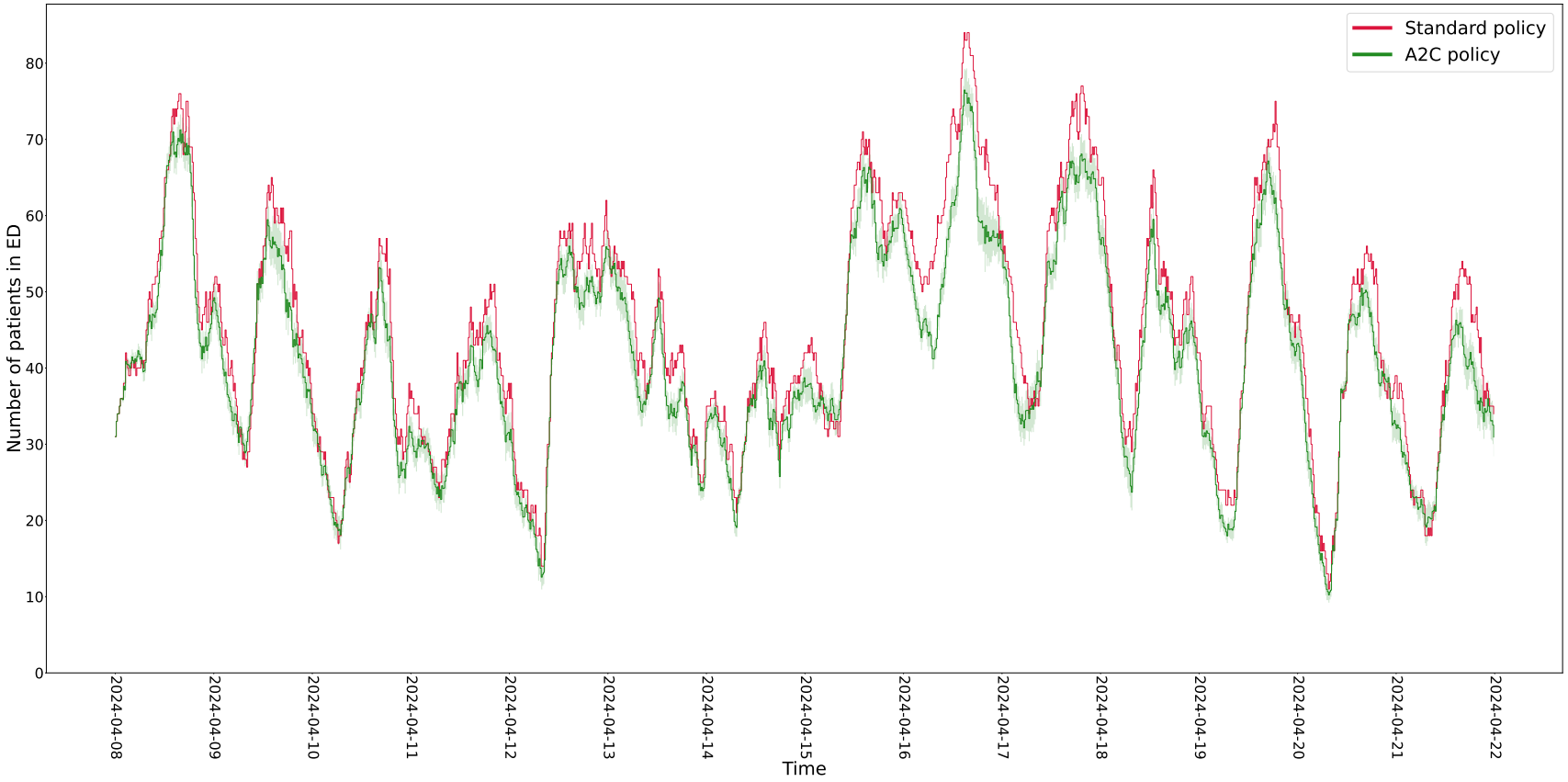

Figure 6.2: Comparison between the number of patients in the system under the standard queue management policy and following the policies of the A2C agents. The solid line represents the mean value across the performed runs, while the shaded area represents the standard deviation.

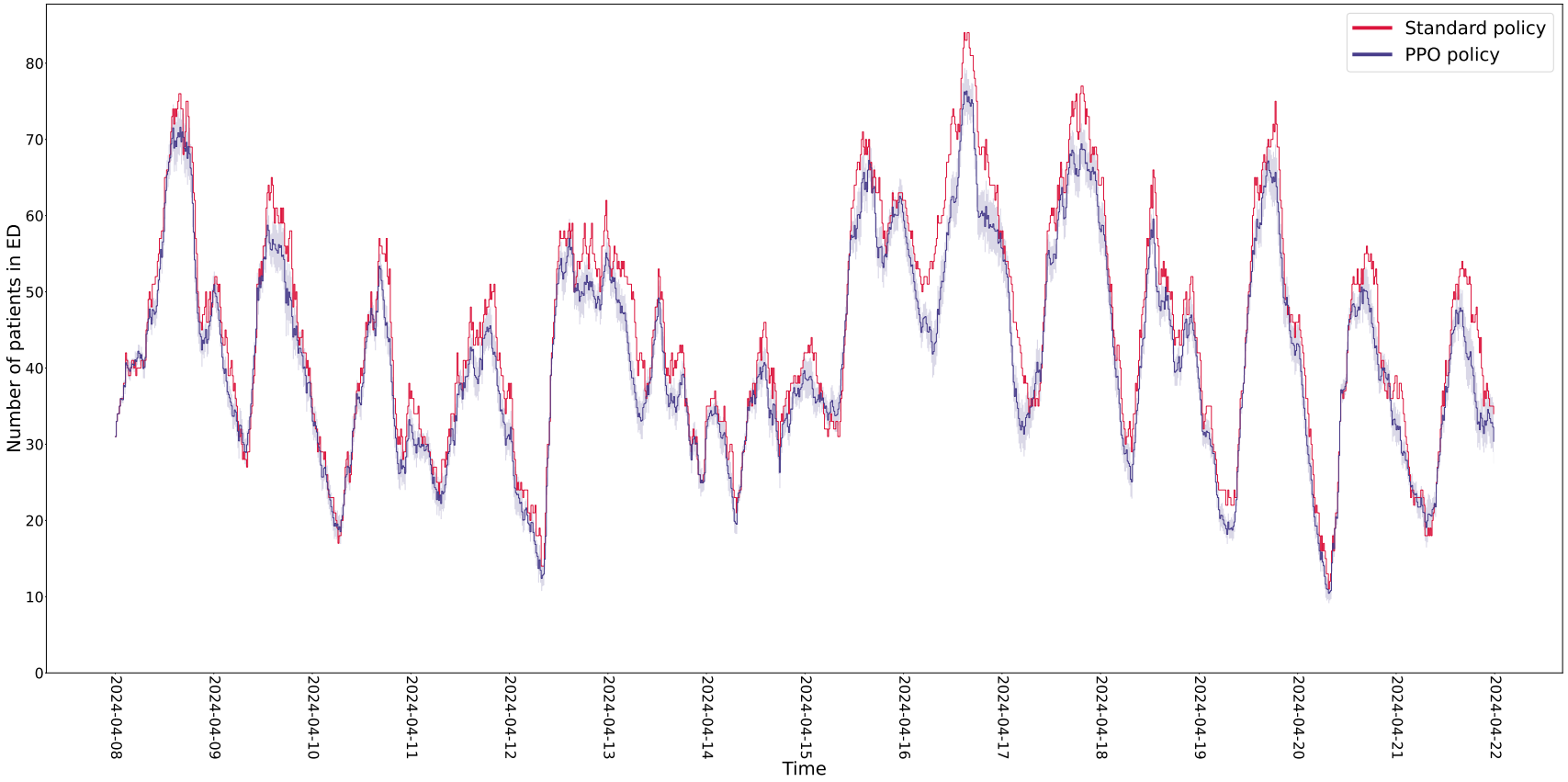

Figure 6.3: Comparison between the number of patients in the system under the standard queue management policy and following the policies of the PPO agents. The solid line represents the mean value across the performed runs, while the shaded area represents the standard deviation.

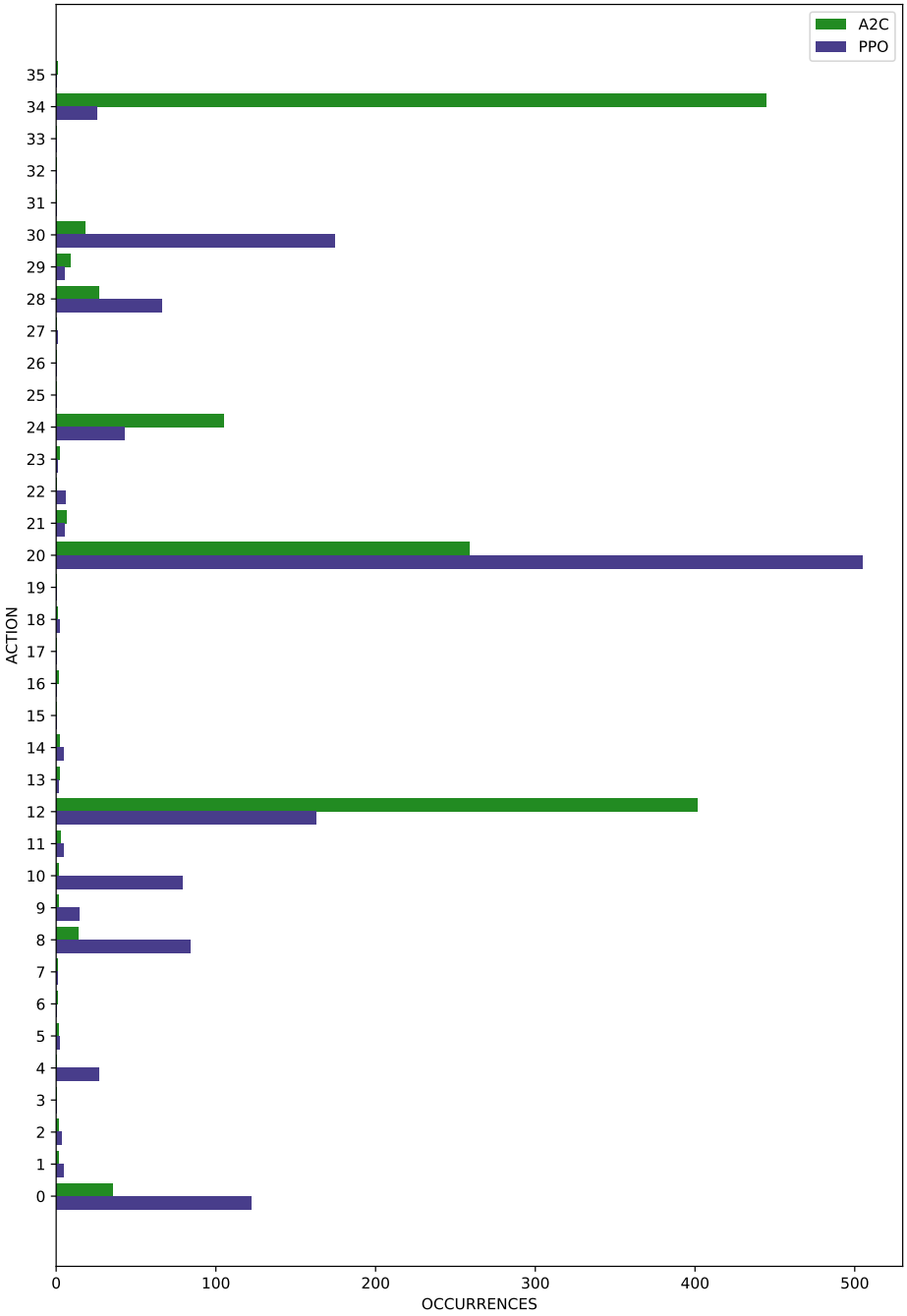

Figure 6.4: Average number of selected actions on the test instances, following A2C and PPO policies

On the other side, the most frequently selected actions under the PPO policy are:

- 20, corresponding to prioritizing patients (and their service requests) that have already received the first medical assessment, breaking ties by sorting them from the most to the least severe, further breaking ties by descending time spent in the system;
- 30, corresponding to prioritizing patients (and their service requests) by descending time spent in the system;
- 12, corresponding to prioritizing patients (and their service requests) from the most to least severe, breaking ties by descending time spent in the system;
- 0, corresponding to prioritizing patients (and their service requests) by sorting them from the most to the least severe, breaking ties by descending probability of them being discharged after the next physician visit;
- 8, corresponding to prioritizing patients (and their service requests) that have already received the first medical assessment, breaking ties by sorting them from the most to the least severe, further breaking ties by descending probability of them being discharged after the next physician visit;
- 10, corresponding to prioritizing patients (and their service requests) that have already received the first medical assessment, breaking ties by sorting them from the least to the most severe, further breaking ties by descending probability of them being discharged after the next physician visit;
- 28, corresponding to prioritizing patients (and their service requests) that have already received the first medical assessment, breaking ties by descending probability of them being discharged after the next physician visit;
- 24, corresponding to prioritizing patients (and their service requests) that are most likely to be discharged after the next physician visit.

6.4 Discussion

The observed results suggest that the RL-based policies can effectively improve ED patient flow by reducing congestion without sacrificing throughput. Given the nature of the system, in which patient arrivals cannot be influenced, the direct correlation between the reduction in crowding and LOS is consistent with Little's law (Little,

1961). The invariance of throughput, especially across severity levels, indicates that the trained agents do not develop absolute preferences for particular categories of patients. As a result, LOS and DTDT values can be compared meaningfully across the different policies.

Improvements in LOS –particularly in terms of medians– are consistent with the intended optimization objective, while more heterogeneous results on DTDT highlight some inherent trade-offs in the learned strategies. Specifically, the discrepancy between mean and median DTDT values indicates that, under the learned policies, most patients are assessed quickly, but a minority experiences longer delays. This behavior reflects the emergent strategy of the agents: rather than always immediately initiating clinical pathways for all new arrivals, pushing patients through the system, they sometimes delay the first assessment when the system is already saturated. By doing so, they prevent downstream bottlenecks, reduce overall LOS, and promote a more balanced resource allocation. This dynamic finds further confirmation in established queueing theory principles, such as Kingman's law (Kingman, 1961) and its extensions (Whitt, 1993), which emphasize how waiting times increase disproportionately as a system approaches saturation.

A closer inspection of the policies highlights a slight difference in emphasis between A2C and PPO. A2C tends to alleviate system pressure by prioritizing earlier discharges, whereas PPO develops a more diversified strategy that balances LOS improvements with a more contained increase in DTDT. This distinction reflects the structural differences between unconstrained and clipped policy updates, with PPO producing a broader distribution of actions.

The mentioned redistribution of waiting times particularly benefits high-severity patients, whose LOS decreases without significant impact on DTDT, while lower-severity patients are occasionally delayed. This outcome is coherent with the reward-driven design of the objective, but it also raises important considerations about balancing efficiency and fairness in real-world applications. In practice, the clinical risk associated with lower-severity patients is moderate, as they are periodically re-evaluated by care staff while waiting, ensuring that any deterioration in their condition can be promptly addressed.

In summary, the results demonstrate that RL-based queue management can reduce LOS and mitigate crowding without affecting throughput, by regulating the timing for patients entering resource-intensive phases of their pathway. These findings provide a solid foundation for further exploring adaptive queue management strategies in emergency settings.

7 Conclusions and Future Work

This final chapter is dedicated to summarizing the contributions of this research work, together with a discussion of the identified limitations. Finally, future research prospects are presented.

7.1 Summary of contributions

This thesis presented a Decision Support framework designed to mitigate ED overcrowding by reducing LOS and improving patient flow, while also monitoring DTDT. The case study focused on the General ED of the IRCCS AOU Bologna, one of the largest hospitals in Northern Italy.

A DES was developed and validated as a Digital Twin of the real system, capable of reproducing realistic and dynamic ED scenarios when provided with appropriate inputs, derived from historical data of the ED.

A DL model was designed to generate patient-level predictions on the likelihood of approaching the end of the clinical pathway, providing additional insight about patient flow dynamics.

Finally, DRL models were implemented following an MDP formulation of the problem. The DRL agents integrate information from the predictive model together with additional indicators computed from the simulation to interpret the current state of the system and propose actions, which are translated into queue management strategies. The agents were trained using Actor–Critic algorithms –specifically A2C and PPO– which represent well-established state-of-the-art methods for controlling complex and stochastic systems.

Experiments were conducted by testing the trained agents in simulation, by using an emulation of the current queue management policy adopted in the ED as a baseline for performance comparison. The results demonstrate the capability of the RL agents to improve system performance by reducing overall LOS and crowding levels. This improvement is achieved with only a slight increase in the average time for the first medical assessment (DTDT), a trade-off mainly affecting lower sever-

ity patients, while higher-severity cases mostly benefit from shorter LOS without significant delays in DTDT.

The proposed tool can be deployed within the real ED by detaching one of the trained RL agents –specifically, its Actor network– from the simulated environment and interfacing it with the hospital information system. For this purpose, an intermediate infrastructure would be required, consisting of (i) a state–encoding module able to translate real-time system conditions into the state representation expected by the agent, and (ii) a decoding module that converts the agent's selected actions into meaningful operational suggestions for the ED staff. The way the framework was formulated allows for a generalized implementation in other EDs, provided they have a data infrastructure similar to that of the case study.

This work highlights the potential of advanced data-driven approaches, based on AI and more specifically DRL, to support real-time decision-making in the ED and to improve patient flow, encouraging further research in this field.

7.2 Limitations

A first recognized limitation of the presented approach –common to all similar studies– lies in the use of a simulation model to reflect the dynamics of the real system. The famous aphorism *"All models are wrong, but some are useful"* (Box, 1976) highlights how no model can fully reproduce the complexity of reality; however, when key aspects are accurately represented, the model can still provide valuable insights.

In the present case, the only physical entities the DES implements are physicians, patients, and a limited set of other resources, while it omits the explicit representation of nurses and treatment spaces. This modeling choice was guided by direct observation of the real system, where physicians and diagnostic services were identified as the main bottlenecks in the process. Although the validation results confirmed the model's ability to realistically reproduce patient flow and LOS, further verification is still needed to confirm the impact on simulation reliability of excluding certain elements from the model and simplifying some dynamics.

Similarly, despite the model taking into account patient LOS, it does not explicitly consider the evolution of the patient's clinical condition, nor the potential impact of treatment delays on clinical outcomes. In real settings, clinical motivation, such as the deterioration in patient condition, may force physicians to deviate from the prioritization policy suggested by the agent. Therefore, future research should examine the effects of such deviations, quantifying a rejection rate threshold beyond

which the Decision Support Tool's effectiveness becomes minimal, and assessing how model performance degrades as this rate increases.

The propensity of physicians to accept such a model may also be highly influenced by the well-known intrinsic opacity of AI and, in particular, DL-based models (Chesterman, 2021). Although the trained agents successfully learn policies that improve overall system performance, it is not possible to extensively provide a human-interpretable explanation on why a certain state of the system leads to a specific action selection. Therefore, the perception of the model as a "black box" could affect the level of trust and acceptance among clinical staff.

A further limitation concerns the generalization of the trained models. Despite being trained and tested on multiple realistic instances, it cannot be guaranteed that the learned policy will remain optimal under every possible operational scenario: unforeseen conditions may lead to suboptimal or unstable behaviors. For this reason, the implementation of a Digital Twin operating in parallel with the real system is recommended, to monitor possible deviations and allow the activation of backup policies when necessary. Nonetheless, validation experiments, conducted during high-congestion periods (Section 6.3), show encouraging evidence of the model's ability to maintain stable performance over time.

Moreover, the computational burden of the proposed framework should be acknowledged. Training Actor-Critic agents requires significant processing power, ideally supported by dedicated GPU hardware, commonly recommended for DL-based models (Buber & Diri, 2018; Steinkraus et al., 2005). In this work, training was performed on a CPU-only system, resulting in training times of several days for a single agent. Adapting the model to different ED settings, with possible variations in system configuration and patient mix, may require revising the MDP formulation and conducting multiple experiments with a non-negligible computational effort.

Finally, as with all data-driven methods, the quality, completeness, and reliability of input data remain essential prerequisites to ensure the validity of the developed models.

7.3 Future developments

A first possible direction for future developments concerns further refinement of the model formulation, both in terms of state and action representation. The state description could be enriched by including a more detailed prediction of the evolution of the patient's clinical pathway, not limited to discharge likelihood but also anticipating diagnostic or therapeutic services. In this regard, additional informa-

tion could be extracted through Natural Language Processing techniques applied to clinical notes, as proposed by Fabbri et al., 2024, or process mining approaches, as in Duma and Aringhieri, 2020.

Regarding the action space, instead of restricting the agent to selecting queue-ordering criteria, future implementations may explore a more flexible formulation in which the agent directly selects specific patients based on their characteristics, enabling more complex and tailored policies.

A second development direction concerns the architecture of the RL components. The current work adopts a single agent, which receives system-wise information and whose centralized actions impact the whole environment. A natural extension would be the adoption of a multi-agent framework, in which physicians and other staff members are modeled as independent but cooperative decision-makers (Buşoniu et al., 2010), allowing for a closer alignment between simulated and real-world decision dynamics, and also in this case promoting the development of more sophisticated policies.

A third extension concerns expanding the model boundaries. The current framework focuses exclusively on the ED and does not explicitly account for what happens in the downstream process. However, ED patient flow is intrinsically linked to inpatient ward constraints, especially bed availability. A wider framework integrating inpatient ward and hospitalization/discharge dynamics would therefore provide a more comprehensive environment. Even without altering downstream policies, the inclusion of such information could increase the agent's awareness about ED patient flow. However, complete integration, where decisions at ED and ward-level are jointly optimized, would represent a significant advancement.

Finally, a promising research direction might involve the integration of Generative AI (GAI) models in the framework. GAI is increasingly recognized as a set of complementary tools in OR applications, particularly in the study of stochastic systems and simulation (Zhou & Sheu, 2025). Recent studies such as Sun et al., 2025 discuss the possibilities in terms of GAI-enhanced DRL frameworks, highlighting the advantages GAI models can produce both from a data perspective (e.g., data augmentation, extraction of relevant features, simulation support) and from a policy perspective (e.g., an improved flexibility in the way the agent manages data and operates). Such tools, an ongoing revolution in the field of AI, could therefore represent an important step forward in the design of a more powerful and adaptive Decision Support Tool for ED management.

Additionally, from an implementation perspective, and in relation to the computational effort discussed in the previous section, it is worth noting that institutions

such as IRCCS AOU Bologna are progressively evolving towards information system architectures suitable for integration with High Performance Computing (HPC) platforms. Such developments would considerably facilitate the application of solutions such as those presented in this thesis.

Bibliography

Ada, S. E., Ugur, E., & Akin, H. L. (2022). Generalization in transfer learning: Robust control of robot locomotion. *Robotica*, *40*(11), 3811–3836. https://doi.org/10.1017/s0263574722000625

Ahalt, V., Argon, N. T., Ziya, S., Strickler, J., & Mehrotra, A. (2018). Comparison of emergency department crowding scores: A discrete-event simulation approach. *Health Care Management Science*, *21*(1), 144–155. https://doi.org/10.1007/s10729-016-9385-z

Ajmi, F., Ben Othman, S., Ajmi, F., Zgaya-Biau, H., Renard, J.-M., Smith, G., & Hammadi, S. (2023). Self-adaptive agents based on reinforcement learning to optimize patient scheduling in emergency department. *IEEE Conference on Systems, Man, and Cybernetic (SMC 2023)*. https://hal.science/hal-04566621

Al Kuwaiti, A., Nazer, K., Al-Reedy, A., Al-Shehri, S., Al-Muhanna, A., Subbarayalu, A. V., Al Muhanna, D., & Al-Muhanna, F. A. (2023). A review of the role of artificial intelligence in healthcare. *Journal of Personalized Medicine*, *13*(6). https://doi.org/10.3390/jpm13060951

Alenany, E., & Cadi, A. A. E. (2020). Modeling patient flow in the emergency department using machine learning and simulation. https://arxiv.org/abs/2012.01192

Allihaibi, W. G. M., Cholette, M., Masoud, M., Burke, J., & Karim, A. (2020). A heuristic approach for scheduling patient treatment in an emergency department based on bed blocking. *International Journal of Industrial Engineering Computations*, *11*(4), 565–584. https://doi.org/10.5267/j.ijiec.2020.4.005

AOU Bologna. (2025). IRCCS Azienda Ospedaliero Universitaria di Bologna, Policlinico di Sant'Orsola [Last visited: 2 October, 2025]. https://www.aosp.bo.it

Asplin, B. R., Magid, D. J., Rhodes, K. V., Solberg, L. I., Lurie, N., & Camargo, J., Carlos A. (2003). A conceptual model of emergency department crowding. *Annals of Emergency Medicine*, *42*(2), 173–180. https://doi.org/10.1067/mem.2003.302

Badr, S., Nyce, A., Awan, T., Cortes, D., Mowdawalla, C., & Rachoin, J. S. (2022). Measures of emergency department crowding, a systematic review. how to make sense of a long list. *Open Access Emergency Medicine: OAEM*, *14*, 5–14. https://doi.org/10.2147/OAEM.S338079

Belsare, S., Badilla, E. D., & Dehghanimohammadabadi, M. (2022). Reinforcement learning with discrete event simulation: The premise, reality, and promise. *2022 Winter Simulation Conference (WSC)*, 2724–2735. https://doi.org/10.1109/WSC57314.2022.10015503

Bishop, C. (2006). Chapter 4: Linear models for classification. In *Pattern recognition and machine learning* (pp. 179–182). Springer New York, NY.

Box, G. E. P. (1976). Science and statistics. *Journal of the American Statistical Association*, *71*(356), 791–799. https://doi.org/10.1080/01621459.1976.10480949

Brennan, P., Perola, M., van Ommen, G.-J., Riboli, E., & behalf of the European Cohort Consortium, O. (2017). Chronic disease research in europe and the need for integrated population cohorts. *European Journal of Epidemiology*, *32*(9), 741–749. https://doi.org/10.1007/s10654-017-0315-2

Brown, A. F., & Cadogan, M. D. (2020). *Emergency medicine: Diagnosis and management* (8th ed.). CRC Press. https://doi.org/10.1201/9781003032618

Buber, E., & Diri, B. (2018). Performance analysis and cpu vs gpu comparison for deep learning. *2018 6th International Conference on Control Engineering & Information Technology (CEIT)*, 1–6. https://doi.org/10.1109/CEIT.2018.8751930

Buşoniu, L., Babuška, R., & De Schutter, B. (2010). Multi-agent reinforcement learning: An overview. In D. Srinivasan & L. C. Jain (Eds.), *Innovations in multi-agent systems and applications - 1* (pp. 183–221). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-14435-6_7

Calnan, M. (1984). The functions of the hospital emergency department: A study of patient demand. *The Journal of Emergency Medicine*, *2*(1), 57–63. https://doi.org/10.1016/0736-4679(84)90049-0

Canbek, G., Temizel, T. T., & Sagiroglu, S. (2022). Ptopi: A comprehensive review, analysis, and knowledge representation of binary classification performance measures/metrics. *SN Computer Science*, *4*(1), 13. https://doi.org/10.1007/s42979-022-01409-1

Capocchi, L., & Santucci, J.-F. (2022). Discrete event modeling and simulation for reinforcement learning system design. *Information*, *13*, 121. https://doi.org/10.3390/info13030121

Cardosi, J. D., Shen, H., Groner, J. I., Armstrong, M., & Xiang, H. (2021). Machine learning for outcome predictions of patients with trauma during emergency department care. *BMJ Health & Care Informatics*, *28*(1), e100407. https://doi.org/10.1136/bmjhci-2021-100407

Castanheira-Pinto, A., Gonçalves, B. S., Lima, R. M., & Dinis-Carvalho, J. (2021). Modeling, assessment and design of an emergency department of a public hospital through discrete-event simulation. *Applied Sciences*, *11*(2). https://doi.org/10.3390/app11020805

Ceglowski, R., Churilov, L., & Wasserthiel, J. (2007). Combining data mining and discrete event simulation for a value-added view of a hospital emergency department. *Journal of the Operational Research Society*, *58*(2), 246–254. https://doi.org/10.1057/palgrave.jors.2602270

Chesterman, S. (2021). Through a glass, darkly: Artificial intelligence and the problem of opacity. *The American Journal of Comparative Law*, *69*(2), 271–294. https://doi.org/10.1093/ajcl/avab012

Christ, M., Grossmann, F., Winter, D., Bingisser, R., & Platz, E. (2010). Modern triage in the emergency department. *Deutsches Ärzteblatt International*, *107*(50), 892–898. https://doi.org/10.3238/arztebl.2010.0892

Cildoz, M., Ibarra, A., & Mallor, F. (2019). Accumulating priority queues versus pure priority queues for managing patients in emergency departments. *Operations Research for Health Care*, *23*, 100224. https://doi.org/10.1016/j.orhc.2019.100224

Cimellaro, G. P., Malavisi, M., & Mahin, S. (2017). Using discrete event simulation models to evaluate resilience of an emergency department. *Journal of Earthquake Engineering*, *21*(2), 203–226. https://doi.org/10.1080/13632469.2016.1172373

Cobbe, K., Hesse, C., Hilton, J., & Schulman, J. (2019). Leveraging procedural generation to benchmark reinforcement learning. *CoRR*, *abs/1912.01588*. http://arxiv.org/abs/1912.01588

Connelly, L. G., & Bair, A. E. (2004). Discrete event simulation of emergency department activity: A platform for system-level operations research. *Academic Emergency Medicine*, *11*(11), 1177–1185. https://doi.org/10.1197/j.aem.2004.08.021

Derlet, R. W., & Richards, J. R. (2000). Overcrowding in the nation's emergency departments: Complex causes and disturbing effects. *Annals of Emergency Medicine*, *35*(1), 63–68. https://doi.org/10.1016/S0196-0644(00)70105-3

Di Leva, A., & Sulis, E. (2017). Process analysis for a hospital emergency department. *International Journal of Economics and Management Systems*, *2*, 34–41. https://www.iaras.org/journals/caijems/process-analysis-for-a-hospital-emergency-department

Di Somma, S., Paladino, L., Vaughan, L., Lalle, I., Magrini, L., & Magnanti, M. (2015). Overcrowding in emergency department: An international issue. *Intern Emerg Med*, *10*, 171–175. https://doi.org/10.1007/s11739-014-1154-8

Doudareva, E., & Carter, M. (2022). Discrete event simulation for emergency department modelling: A systematic review of validation methods. *Operations Research for Health Care*, *33*, 100340. https://doi.org/10.1016/j.orhc.2022.100340

Duguay, C., & Chetouane, F. (2007). Modeling and improving emergency department systems using discrete event simulation. *SIMULATION*, *83*(4), 311–320. https://doi.org/10.1177/0037549707083111

Duma, D., & Aringhieri, R. (2020). An ad hoc process mining approach to discover patient paths of an emergency department. *Flexible Services and Manufacturing Journal*, *32*(1), 6–34. https://doi.org/10.1007/s10696-018-9330-1

Duma, D., & Aringhieri, R. (2023). Real-time resource allocation in the emergency department: A case study. *Omega*, *117*, 102844. https://doi.org/10.1016/j.omega.2023.102844

El Morr, C., & Ali-Hassan, H. (2019). Analytics building blocks. In *Analytics in healthcare: A practical introduction* (pp. 16–18). Springer International Publishing. https://doi.org/10.1007/978-3-030-04506-7_2

Elalouf, A., & Wachtel, G. (2015). An alternative scheduling approach for improving patient-flow in emergency departments. *Operations Research for Health Care*, *7*, 94–102. https://doi.org/10.1016/j.orhc.2015.08.002

Etu, E.-E., Monplaisir, L., Arslanturk, S., Masoud, S., Aguwa, C., Markevych, I., & Miller, J. (2022). Prediction of length of stay in the emergency department for covid-19 patients: A machine learning approach. *IEEE Access*, *10*, 42243–42251. https://doi.org/10.1109/ACCESS.2022.3168045

Fabbri, C., Lombardi, M., Malaguti, E., & Monaci, M. (2024). On-line strategy selection for reducing overcrowding in an emergency department. *Omega*, *127*, 103098. https://doi.org/10.1016/j.omega.2024.103098

FitzGerald, G., Jelinek, G. A., Scott, D., & Gerdtz, M. F. (2010). Emergency department triage revisited. *Emergency Medicine Journal*, *27*(2), 86–92. https://doi.org/10.1136/emj.2009.077081

Furian, N., O'Sullivan, M., Walker, C., & Reuter-Oppermann, M. (2022). Machine learning-based patient selection in an emergency department. https://arxiv.org/abs/2206.03752

Girishan Prabhu, V., & Taaffe, K. M. (2024). Hybrid modelling approach using reinforcement learning in conjunction with simulation: A case study of an emergency department. In M. Fakhimi & N. Mustafee (Eds.), *Hybrid modeling and simulation: Conceptualizations, methods and applications* (pp. 295–318). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-59999-6_12

Girshick, R. (2015). Fast r-cnn. https://arxiv.org/abs/1504.08083

Golinelli, D., Campinoti, F., Sanmarchi, F., Rosa, S., Beleffi, M., Farina, G., Tampieri, A., Fantini, M. P., Giostra, F., & Santi, L. (2021). Patterns of emergency department visits for acute and chronic diseases during the two pandemic waves in italy. *The American Journal of Emergency Medicine*, *50*, 22–26. https://doi.org/10.1016/j.ajem.2021.07.010

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Chapter 7: Regularization for deep learning. In *Deep learning* (pp. 227–230). MIT Press. https://www.deeplearningbook.org

Goto, T., Camargo, J., Carlos A., Faridi, M. K., Freishtat, R. J., & Hasegawa, K. (2019). Machine learning–based prediction of clinical outcomes for children during emergency department triage. *JAMA Network Open*, *2*(1), e186937–e186937. https://doi.org/10.1001/jamanetworkopen.2018.6937

Gupta, D., & Denton, B. (2008). Appointment scheduling in health care: Challenges and opportunities. *IIE Transactions*, *40*(9), 800–819. https://doi.org/10.1080/07408170802165880

Gustafsson, S., Gedon, D., Lampa, E., Ribeiro, A. H., Holzmann, M. J., Schön, T. B., & Sundström, J. (2022). Development and validation of deep learning ecg-based prediction of myocardial infarction in emergency department patients. *Scientific Reports*, *12*(1), 19615. https://doi.org/10.1038/s41598-022-24254-x

Haarnoja, T., Tang, H., Abbeel, P., & Levine, S. (2017). Reinforcement learning with deep energy-based policies. https://arxiv.org/abs/1702.08165

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. https://arxiv.org/abs/1801.01290

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., & Levine, S. (2019). Soft actor-critic algorithms and applications. https://arxiv.org/abs/1812.05905

Harrou, F., Dairi, A., Kadri, F., & Sun, Y. (2020). Forecasting emergency department overcrowding: A deep learning framework. *Chaos, Solitons & Fractals*, *139*, 110247. https://doi.org/10.1016/j.chaos.2020.110247

Harzi, M., Condotta, J.-F., Nouaouri, I., & Krichen, S. (2017). Scheduling patients in emergency department by considering material resources [Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 21st International Conference, KES-20176-8 September 2017, Marseille, France]. *Procedia Computer Science*, *112*, 713–722. https://doi.org/10.1016/j.procs.2017.08.153

He, S., Sim, M., & Zhang, M. (2019). Data-driven patient scheduling in emergency departments: A hybrid robust-stochastic approach. *Management Science*, *65*(9), 4123–4140. https://doi.org/10.1287/mnsc.2018.3145

Hertz, J., Palmer, R. G., & Krogh, A. S. (1991). Introduction to the theory of neural computation. Perseus Publishing. https://dl.acm.org/doi/10.5555/574634

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, *abs/1207.0580*. http://arxiv.org/abs/1207.0580

Hong, W. S., Haimovich, A. D., & Taylor, R. A. (2018). Predicting hospital admission at emergency department triage using machine learning. *PLOS ONE*, *13*(7), 1–13. https://doi.org/10.1371/journal.pone.0201016

Hoot, N. R., LeBlanc, L. J., Jones, I., Levin, S. R., Zhou, C., Gadd, C. S., & Aronsky, D. (2008). Forecasting emergency department crowding: A discrete event simulation. *Annals of Emergency Medicine*, *52*(2), 116–125. https://doi.org/10.1016/j.annemergmed.2007.12.011

Hosseini Shokouh, S. M., Mohammadi, K., & Yaghoubi, M. (2022). Optimization of service process in emergency department using discrete event simulation and machine learning algorithm [Published online 2022-06-08, PMC9206831]. *Archives of Academic Emergency Medicine*, *10*(1), e44. https://doi.org/10.22037/aaem.v10i1.1545

Huang, J., Carmeli, B., & Mandelbaum, A. (2015). Control of patient flow in emergency departments, or multiclass queues with deadlines and feedback. *Operations Research*, *63*(4), 892–908. https://doi.org/10.1287/opre.2015.1389

Hulshof, P. J. H., Kortbeek, N., Boucherie, R. J., Hans, E. W., & Bakker, P. J. M. (2012). Taxonomic classification of planning decisions in health care: A struc-

tured review of the state of the art in or/ms. *Health Systems*, *1*(2), 129–175. https://doi.org/10.1057/hs.2012.18

Hung, G. R., Whitehouse, S. R., O'Neill, C., Gray, A. P., & Kissoon, N. (2007). Computer modeling of patient flow in a pediatric emergency department using discrete event simulation. *Pediatric Emergency Care*, *23*(1), 2. https://doi.org/10.1097/PEC.0b013e31802c611e

Hunter-Zinck, H. S., Peck, J. S., Strout, T. D., & Gaehde, S. A. (2019). Predicting emergency department orders with multilabel machine learning techniques and simulating effects on length of stay. *Journal of the American Medical Informatics Association*, *26*(12), 1427–1436. https://doi.org/10.1093/jamia/ocz171

Hwang, E. J., Nam, J. G., Lim, W. H., Park, S. J., Jeong, Y. S., Kang, J. H., Hong, E. K., Kim, T. M., Goo, J. M., Park, S., Kim, K. H., & Park, C. M. (2019). Deep learning for chest radiograph diagnosis in the emergency department [PMID: 31638490]. *Radiology*, *293*(3), 573–580. https://doi.org/10.1148/radiol.2019191225

Jacobson, S. H., Hall, S. N., & Swisher, J. R. (2013). Discrete-event simulation of health care systems. In R. Hall (Ed.), *Patient flow: Reducing delay in healthcare delivery* (pp. 273–309). Springer US. https://doi.org/10.1007/978-1-4614-9512-3_12

Javidan, A. P., Hansen, K., Higginson, I., Jones, P., & Lang, E. (2021). The international federation for emergency medicine report on emergency department crowding and access block: A brief summary (and International Federation Emergency Department Crowding, A. B. T. Force, A. Javidan, K. Hansen, I. Higginson, P. Jones, D. Petrie, J. Bonning, S. Judkins, E. Revue, D. Lewis, B. Holroyd, L. Mazurik, C. Graham, A. Carter, S. Lee, E. Cohen-Olivella, P. Ho, R. Maharjan, ... E. Lang, Eds.). *Emergency Medicine Journal*, *38*(3), 245–246. https://doi.org/10.1136/emermed-2020-210716

Jiang, H., Wang, W. Y. C., Goh, T.-T., & Zhu, J. (2020). Enhancing decision making with deep reinforcement learning in a context of novel coronavirus outbreak: An example in emergency department. *Proceedings of the 4th International Conference on Medical and Health Informatics*, 113–117. https://doi.org/10.1145/3418094.3418137

Jiang, L., & Giachetti, R. E. (2008). A queueing network model to analyze the impact of parallelization of care on patient cycle time. *Health Care Management Science*, *11*(3), 248–261. https://doi.org/10.1007/s10729-007-9040-9

Joseph, J. W., Leventhal, E. L., Grossestreuer, A. V., Wong, M. L., Joseph, L. J., Nathanson, L. A., Donnino, M. W., Elhadad, N., & Sanchez, L. D. (2020). Deep-learning approaches to identify critically ill patients at emergency department triage using limited information. *Journal of the American College of Emergency Physicians Open*, *1*(5), 773–781. https://doi.org/10.1002/emp2.12218

Kadri, F., Dairi, A., Harrou, F., & Sun, Y. (2023). Towards accurate prediction of patient length of stay at emergency department: A gan-driven deep learning framework. *Journal of Ambient Intelligence and Humanized Computing*, *14*(9), 11481–11495. https://doi.org/10.1007/s12652-022-03717-z

Kannampallil, T. G., Schauer, G. F., Cohen, T., & Patel, V. L. (2011). Considering complexity in healthcare systems. *Journal of Biomedical Informatics*, *44*(6), 943–947. https://doi.org/10.1016/j.jbi.2011.06.006

Kareemi, H., Vaillancourt, C., Rosenberg, H., Fournier, K., & Yadav, K. (2021). Machine learning versus usual care for diagnostic and prognostic prediction in the emergency department: A systematic review. *Academic Emergency Medicine*, *28*(2), 184–196. https://doi.org/10.1111/acem.14190

Kim, J.-K. (2024). Enhancing patient flow in emergency departments: A machine learning and simulation-based resource scheduling approach. *Applied Sciences*, *14*(10). https://doi.org/10.3390/app14104264

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization. https://arxiv.org/abs/1412.6980

Kingman, J. F. C. (1961). The single server queue in heavy traffic. *Mathematical Proceedings of the Cambridge Philosophical Society*, *57*(4), 902–904. https://doi.org/10.1017/S0305004100036094

Kirk, R., Zhang, A., Grefenstette, E., & Rocktäschel, T. (2023). A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, *76*, 201–264. https://doi.org/10.1613/jair.1.14174

Kochenderfer, M. J., Wheeler, T. A., & Wray, K. H. (2022). Chapter 13: Actor-critic methods. In *Algorithms for decision making* (p. 268). MIT Press. https://mitpress.mit.edu/9780262047012/algorithms-for-decision-making/

Komashie, A., & Mousavi, A. (2005). Modeling emergency departments using discrete event simulation techniques. *Proceedings of the Winter Simulation Conference, 2005.*, 5. https://doi.org/10.1109/WSC.2005.1574570

Konda, V., & Tsitsiklis, J. (1999). Actor-critic algorithms. In S. Solla, T. Leen, & K. Müller (Eds.), *Advances in neural information processing systems* (Vol. 12).

MIT Press. https://proceedings.neurips.cc/paper_files/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

Lee, S., & Lee, Y. H. (2020). Improving emergency department efficiency by patient scheduling using deep reinforcement learning. *Healthcare*, *8*(2). https://doi.org/10.3390/healthcare8020077

Leo, G., Lodi, A., Tubertini, P., & Di Martino, M. (2016). Emergency department management in lazio, italy. *Omega*, *58*, 128–138. https://doi.org/10.1016/j.omega.2015.05.007

Lim, M. E., Worster, A., Goeree, R., & Tarride, J.-É. (2013). Simulating an emergency department: The importance of modeling the interactions between physicians and delegates in a discrete event simulation. *BMC Medical Informatics and Decision Making*, *13*(1), 59. https://doi.org/10.1186/1472-6947-13-59

Little, J. D. C. (1961). A proof for the queuing formula: L = λw. *Operations Research*, *9*(3), 383–387. https://doi.org/10.1287/opre.9.3.383

Liu, J., Gu, X., & Liu, S. (2020). Policy optimization reinforcement learning with entropy regularization. https://arxiv.org/abs/1912.01557

Maninchedda, M., Proia, A. S., Bianco, L., Aromatario, M., Orsi, G. B., & Napoli, C. (2023). Main features and control strategies to reduce overcrowding in emergency departments: A systematic review of the literature [PMID: 36852330]. *Risk Management and Healthcare Policy*, *16*, 255–266. https://doi.org/10.2147/RMHP.S399045

Mans, R. S., van der Aalst, W. M., & Vanwersch, R. J. (2015). Chapter 2: Healthcare processes. In *Process mining in healthcare: Evaluating and exploiting operational healthcare processes* (p. 13). Springer International Publishing. https://doi.org/10.1007/978-3-319-16071-9

Mattiuzzi, C., Lippi, G., & Paolillo, C. (2025). Emergency department visits in Italy: 2017-2023 trends, decline and recovery. *Emergency Care Journal*, *21*(2). https://doi.org/10.4081/ecj.2025.13785

Ministero della Salute – Ufficio di Statistica. (2016, November). *Annuario statistico del servizio sanitario nazionale: Assetto organizzativo, attività e fattori produttivi del ssn. anno 2013* (Published November 2nd 2016; reference period 2013). Ministero della Salute. https://www.salute.gov.it/new/it/pubblicazione/annuario-statistico-del-servizio-sanitario-nazionale-assetto-organizzativo-attivita-2/

Ministero della Salute – Ufficio di Statistica. (2025, February). *Annuario statistico del servizio sanitario nazionale: Assetto organizzativo, attività e fattori*

produttivi del ssn. anno 2023 (Published February 3rd 2025; reference period 2023). Ministero della Salute. https://www.salute.gov.it/portale/documentazione/p6_2_2_1.jsp?id=3523&lingua=italiano

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. https://arxiv.org/abs/1602.01783

Morley, C., Unwin, M., Peterson, G. M., Stankovich, J., & Kinsman, L. (2018). Emergency department crowding: A systematic review of causes, consequences and solutions. *PLOS ONE*, *13*(8), e0203316. https://doi.org/10.1371/journal.pone.0203316

Naemi, A., Schmidt, T., Mansourvar, M., Naghavi-Behzad, M., Ebrahimi, A., & Wiil, U. K. (2021). Machine learning techniques for mortality prediction in emergency departments: A systematic review. *BMJ Open*, *11*(11). https://doi.org/10.1136/bmjopen-2021-052663

Nahhas, A., Awaldi, A., & Reggelin, T. (2017). Simulation and the emergency department overcrowding problem [RelStat-2016: Proceedings of the 16th International Scientific Conference Reliability and Statistics in Transportation and Communication October 19-22, 2016. Transport and Telecommunication Institute, Riga, Latvia]. *Procedia Engineering*, *178*, 368–376. https://doi.org/10.1016/j.proeng.2017.01.068

Nas, S., & Koyuncu, M. (2019). Emergency department capacity planning: A recurrent neural network and simulation approach. *Computational and Mathematical Methods in Medicine*, *2019*(1), 4359719. https://doi.org/10.1155/2019/4359719

Object Management Group (OMG). (2022, December). Business process model and notation (bpmn), version 2.0.2 [Accessed: 2025-10-02]. https://www.omg.org/spec/BPMN/2.0.2/

Packer, C., Gao, K., Kos, J., Krähenbühl, P., Koltun, V., & Song, D. (2019). Assessing generalization in deep reinforcement learning. https://arxiv.org/abs/1810.12282

Parr, R., Rudin, C., Chen, H., Boner, Z., Moshkovitz, M., & Semenova, L. (2024). Transition noise facilitates interpretability. *Workshop on Interpretable Policies in Reinforcement Learning @RLC-2024*. https://openreview.net/forum?id=1G8tdr6eFb

Pearce, S., Marchand, T., Shannon, T., Ganshorn, H., & Lang, E. (2023). Emergency department crowding: An overview of reviews describing measures, causes,

and harms. *Internal and Emergency Medicine*, *18*, 1137–1158. https://doi.org/10.1007/s11739-023-03239-2

Plaat, A. (2022). Chapter 1: Introduction. In *Deep reinforcement learning* (pp. 1–2). Springer Nature Singapore. https://doi.org/10.1007/978-981-19-0638-1

Plsek, P. E., & Greenhalgh, T. (2001). The challenge of complexity in health care. *BMJ*, *323*(7313), 625–628. https://doi.org/10.1136/bmj.323.7313.625

Porto, B. M., & Fogliatto, F. S. (2024). Enhanced forecasting of emergency department patient arrivals using feature engineering approach and machine learning. *BMC Medical Informatics and Decision Making*, *24*(1), 377. https://doi.org/10.1186/s12911-024-02788-6

Prabu, A. (2025). *A novel approach using implicit q-learning to optimize er patient triage* (tech. rep.). Stanford University. https://cs224r.stanford.edu/projects/pdfs/CS224R_Final_Project1.pdf

PyTorch contributors. (2016). Pytorch documentation [Accessed: 2025-09-18]. https://docs.pytorch.org/docs/stable/index.html

Rais, A., & Viana, A. (2011). Operations research in healthcare: A survey. *International Transactions in Operational Research*, *18*(1), 1–31. https://doi.org/10.1111/j.1475-3995.2010.00767.x

Raita, Y., Goto, T., Faridi, M. K., Brown, D. F. M., Camargo, C. A., & Hasegawa, K. (2019). Emergency department triage prediction of clinical outcomes using machine learning models. *Critical Care*, *23*(1), 64. https://doi.org/10.1186/s13054-019-2351-7

Raunak, M., Osterweil, L., Wise, A., Clarke, L., & Henneman, P. (2009). Simulating patient flow through an emergency department using process-driven discrete event simulation. *2009 ICSE Workshop on Software Engineering in Health Care*, 73–83. https://doi.org/10.1109/SEHC.2009.5069608

Regione Emilia-Romagna. (2019). Piano di miglioramento dell'accesso in emergenza-urgenza sanitaria — approvazione di linee di indirizzo alle aziende sanitarie. https://salute.regione.emilia-romagna.it/siseps/sanita/emergenza-urgenza/ps/files/anno-2019-dgr-1129-miglioramento-ps.pdf

Regione Emilia-Romagna. (2021). Linee di indirizzo regionali per il triage in pronto soccorso. https://salute.regione.emilia-romagna.it/siseps/sanita/emergenza-urgenza/ps/files/anno-2021-dgr-1230_nuovo-triage-ps.pdf

Ricciardi, C., Marino, M. R., Trunfio, T. A., Majolo, M., Romano, M., Amato, F., & Improta, G. (2024). Evaluation of different machine learning algorithms for predicting the length of stay in the emergency departments: A single-centre

study. *Frontiers in Digital Health, Volume 5 - 2023*. https://doi.org/10.3389/fdgth.2023.1323849

Rojas, E., Cifuentes, A., Burattin, A., Munoz-Gama, J., Sepúlveda, M., & Capurro, D. (2019). Performance analysis of emergency room episodes through process mining. *International Journal of Environmental Research and Public Health*, *16*(7). https://doi.org/10.3390/ijerph16071274

Sapra, A., Malik, A., & Bhandari, P. (2023). Vital sign assessment. In *Statpearls [internet]* (Updated 2023 May 1). StatPearls Publishing. https://www.ncbi.nlm.nih.gov/books/NBK553213/

Sartini, M., Carbone, A., Demartini, A., Giribone, L., Oliva, M., Spagnolo, A. M., Cremonesi, P., Canale, F., & Cristina, M. L. (2022). Overcrowding in emergency department: Causes, consequences, and solutions—a narrative review. *Healthcare*, *10*(9). https://doi.org/10.3390/healthcare10091625

Savioli, G., Ceresa, I. F., Gri, N., Bavestrello Piccini, G., Longhitano, Y., Zanza, C., Piccioni, A., Esposito, C., Ricevuti, G., & Bressan, M. A. (2022). Emergency department overcrowding: Understanding the factors to find corresponding solutions. *Journal of Personalized Medicine*, *12*(2). https://doi.org/10.3390/jpm12020279

Schneider, S. M., Zwemer, F. L., Doniger, A. S., Nagurney, J. T., Silverman, R., & Heller, M. B. (1998). Definition of emergency medicine. *Academic Emergency Medicine: Official Journal of the Society for Academic Emergency Medicine*, *5*(4), 348–351. https://doi.org/10.1111/j.1553-2712.1998.tb02720.x

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. https://arxiv.org/abs/1707.06347

Sharafat, A. R., & Bayati, M. (2021). Patientflownet: A deep learning approach to patient flow prediction in emergency departments. *IEEE Access*, *9*, 45552–45561. https://doi.org/10.1109/ACCESS.2021.3066164

Shbool, M. A., S. Arabeyyat, O., Al-Bazi, A., Al-Hyari, A., Salem, A., Abu-Hmaid, T., & Ali, M. (2023). Machine learning approaches to predict patient's length of stay in emergency department. *Applied Computational Intelligence and Soft Computing*, *2023*(1), 8063846. https://doi.org/https://doi.org/10.1155/2023/8063846

Steinbrook, R. (1996). The role of the emergency department. *New England Journal of Medicine*, *334*(10), 657–658. https://doi.org/10.1056/NEJM199603073341010

Steinkraus, D., Buck, I., & Simard, P. (2005). Using gpus for machine learning algorithms. *Eighth International Conference on Document Analysis and Recogni-*

tion (ICDAR'05), 1115–1120 Vol. 2. https://doi.org/10.1109/ICDAR.2005.251

Sterling, E., Marlowe, C., Ashdown, P., Fairfax, R., Ashbury, V., Ashcombe, L., Hargrave, E., Whitcombe, S., Blackthorn, I., Pembroke, A., Holloway, B., Ashford, N., Kingsley, R., Ellsworth, M., Ravenscroft, T., Ashworth, P., Montague, J., Thorne, G., Abeyk, M., & Wrentham, H. (2025, February). The role of vital signs in triage and emergency medicine. https://doi.org/10.13140/RG.2.2.16937.51040

Subrahmanya, S. V. G., Shetty, D. K., Patil, V., Hameed, B. M. Z., Paul, R., Smriti, K., Naik, N., & Somani, B. K. (2022). The role of data science in healthcare advancements: Applications, benefits, and future prospects. *Irish Journal of Medical Science (1971 -)*, *191*(4), 1473–1483. https://doi.org/10.1007/s11845-021-02730-z

Sun, G., Xie, W., Niyato, D., Mei, F., Kang, J., Du, H., & Mao, S. (2025). Generative ai for deep reinforcement learning: Framework, analysis, and use cases. *IEEE Wireless Communications*, *32*(3), 186–195. https://doi.org/10.1109/MWC.001.2400176

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA, USA.

Taylor, R. A., Pare, J. R., Venkatesh, A. K., Mowafi, H., Melnick, E. R., Fleischman, W., & Hall, M. K. (2016). Prediction of in-hospital mortality in emergency department patients with sepsis: A local big data–driven, machine learning approach. *Academic Emergency Medicine*, *23*(3), 269–278. https://doi.org/10.1111/acem.12876

Tschoellitsch, T., Seidl, P., Böck, C., Maletzky, A., Moser, P., Thumfart, S., Giretzlehner, M., Hochreiter, S., & Meier, J. (2023). Using emergency department triage for machine learning-based admission and mortality prediction. *European Journal of Emergency Medicine*, *30*(6). https://doi.org/10.1097/MEJ.0000000000001068

van der Ham, R. (2016). Salabim: A discrete event simulation library for python [Accessed: 2025-09-18]. https://salabim.org/

Vural, O., Ozaydin, B., Aram, K. Y., Booth, J., Lindsey, B. F., & Ahmed, A. (2025). An artificial intelligence–based framework for predicting emergency department overcrowding: Development and evaluation study. *JMIR Med Inform*, *13*, e73960. https://doi.org/10.2196/73960

Wang, H., Robinson, R. D., Bunch, K., Huggins, C. A., Watson, K., Jayswal, R. D., White, N. C., Banks, B., & Zenarosa, N. R. (2014). The inaccuracy of deter-

mining overcrowding status by using the national ed overcrowding study tool. *The American Journal of Emergency Medicine*, *32*(10), 1230–1236. https://doi.org/10.1016/j.ajem.2014.07.032

Weiss, S., Derlet, R., Arndahl, J., Ernst, A., Richards, J., Fernandez-Frackelton, M., Schwab, R., Stair, T., Vicellio, P., Levy, D., Brautigan, M., Johnson, A., Nick, T., & Fernández-Frankelton, M. (2004). Estimating the degree of emergency department overcrowding in academic medical centers: Results of the national ed overcrowding study (nedocs). *Academic emergency medicine : official journal of the Society for Academic Emergency Medicine*, *11*, 38–50. https://doi.org/10.1197/j.aem.2003.07.017

Whiteson, S., Tanner, B., Taylor, M. E., & Stone, P. (2011). Protecting against evaluation overfitting in empirical reinforcement learning. *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 120–127. https://doi.org/10.1109/ADPRL.2011.5967363

Whitt, W. (1993). Approximations for the GI/G/m queue. *Production and Operations Management*, *2*(2), 114–161. https://doi.org/10.1111/j.1937-5956.1993.tb00094.x

World Health Organization. (2025a). Ageing and health [Accessed: 2025-10-07]. https://www.who.int/news-room/fact-sheets/detail/ageing-and-health

World Health Organization. (2025b). Health systems governance [Accessed: 2025-10-07]. https://www.who.int/health-topics/health-systems-governance

Yancey, C. C., & O'Rourke, M. C. (2025). Emergency department triage. http://europepmc.org/books/NBK557583

Yao, L.-H., Leung, K.-C., Tsai, C.-L., Huang, C.-H., & Fu, L.-C. (2021). A novel deep learning–based system for triage in the emergency department using electronic medical records: Retrospective cohort study. *J Med Internet Res*, *23*(12), e27008. https://doi.org/10.2196/27008

Zaboli, A., Turcato, G., Brigiari, G., Massar, M., Ziller, M., Sibilio, S., & Brigo, F. (2024). Emergency departments in contemporary healthcare: Are they still for emergencies? an analysis of over 1 million attendances. *Healthcare*, *12*(23). https://doi.org/10.3390/healthcare12232426

Zhou, Q., & Sheu, J.-B. (2025). The use of generative artificial intelligence (genai) in operations research: Review and future research agenda. *Journal of the Operational Research Society*, *0*(0), 1–21. https://doi.org/10.1080/01605682.2025.2561762

A Additional Figures

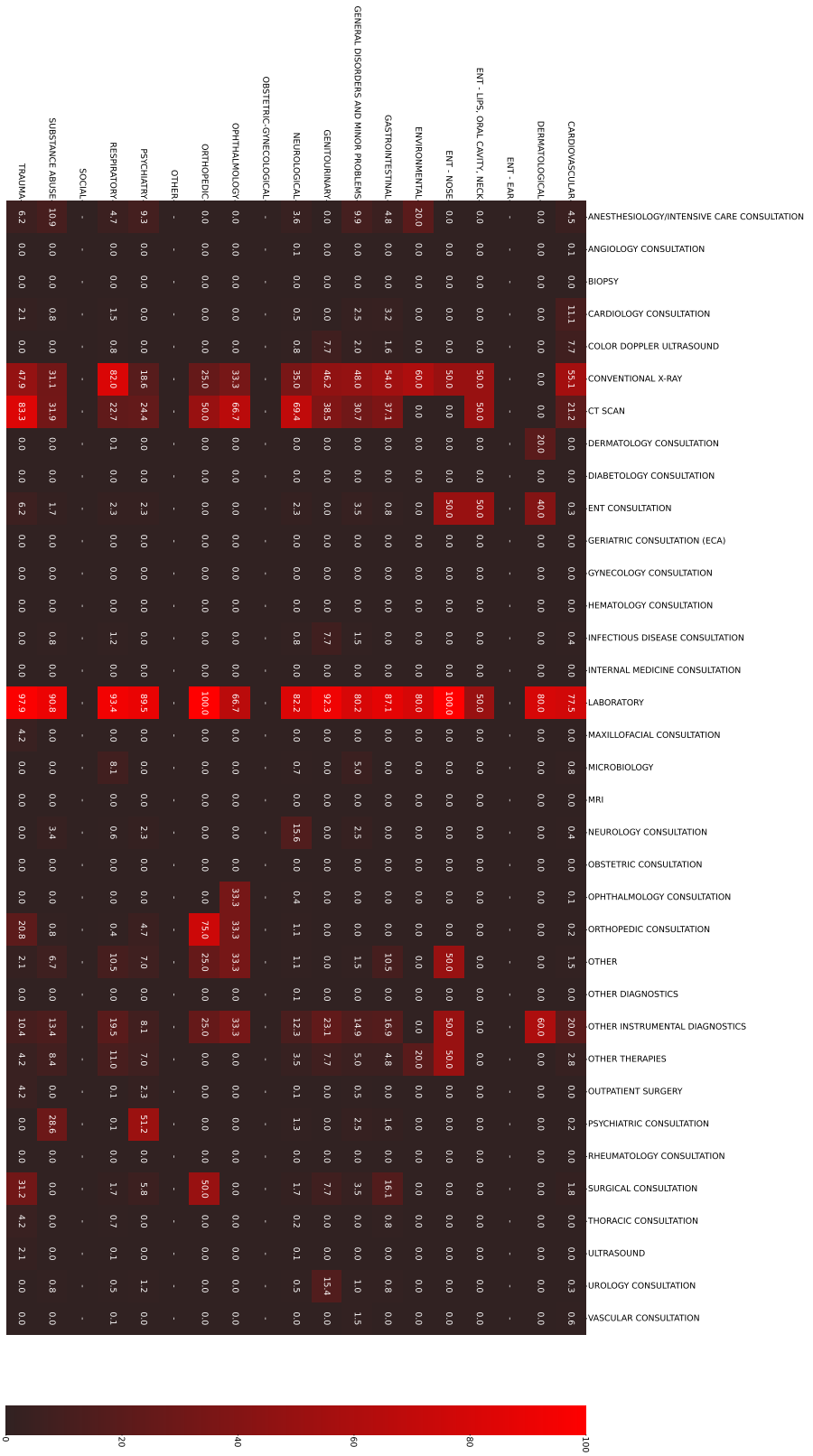

Figure A.1: Service patterns heatmap for patients with severity 1

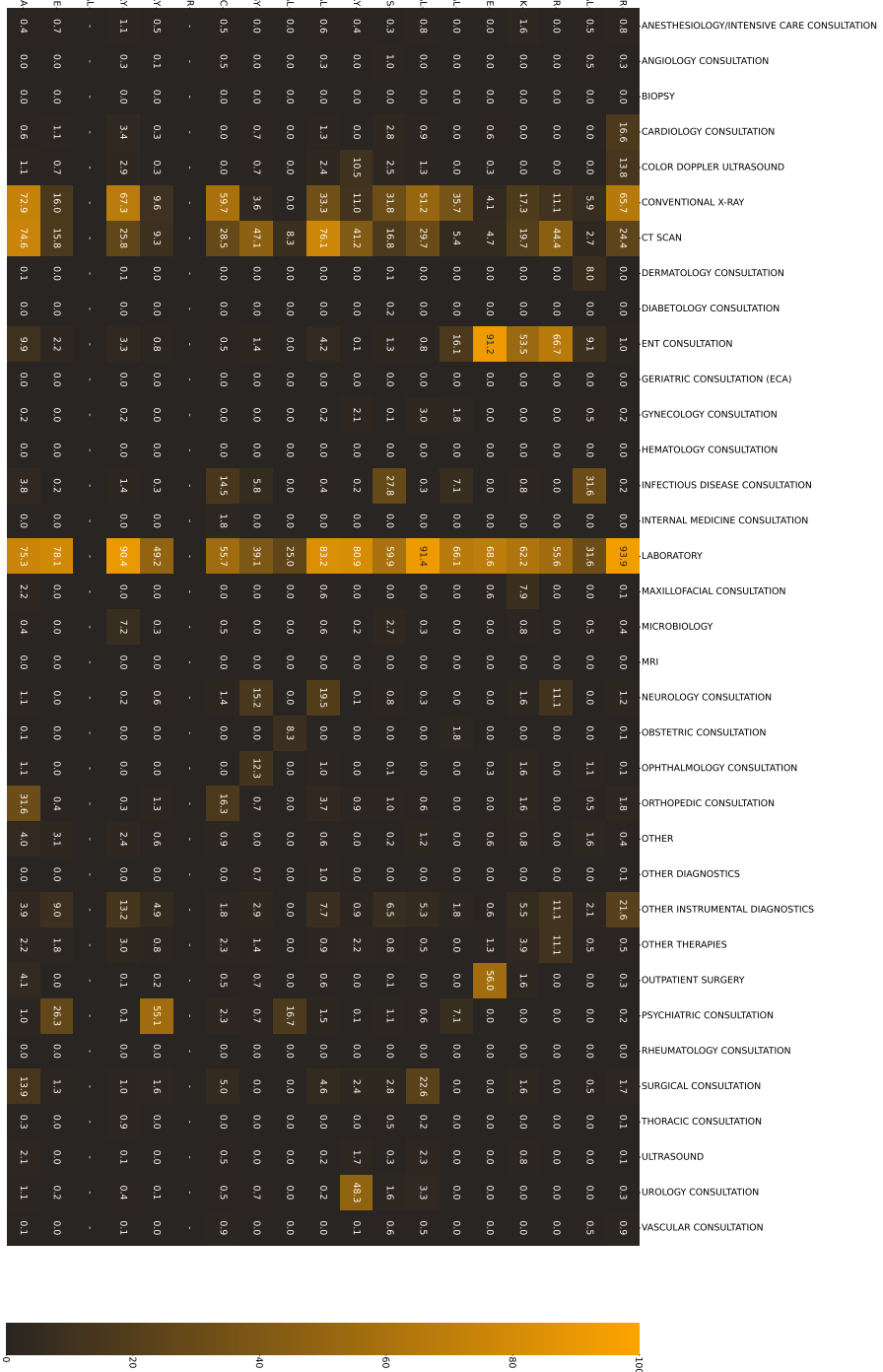

Figure A.2: Service patterns heatmap for patients with severity 2

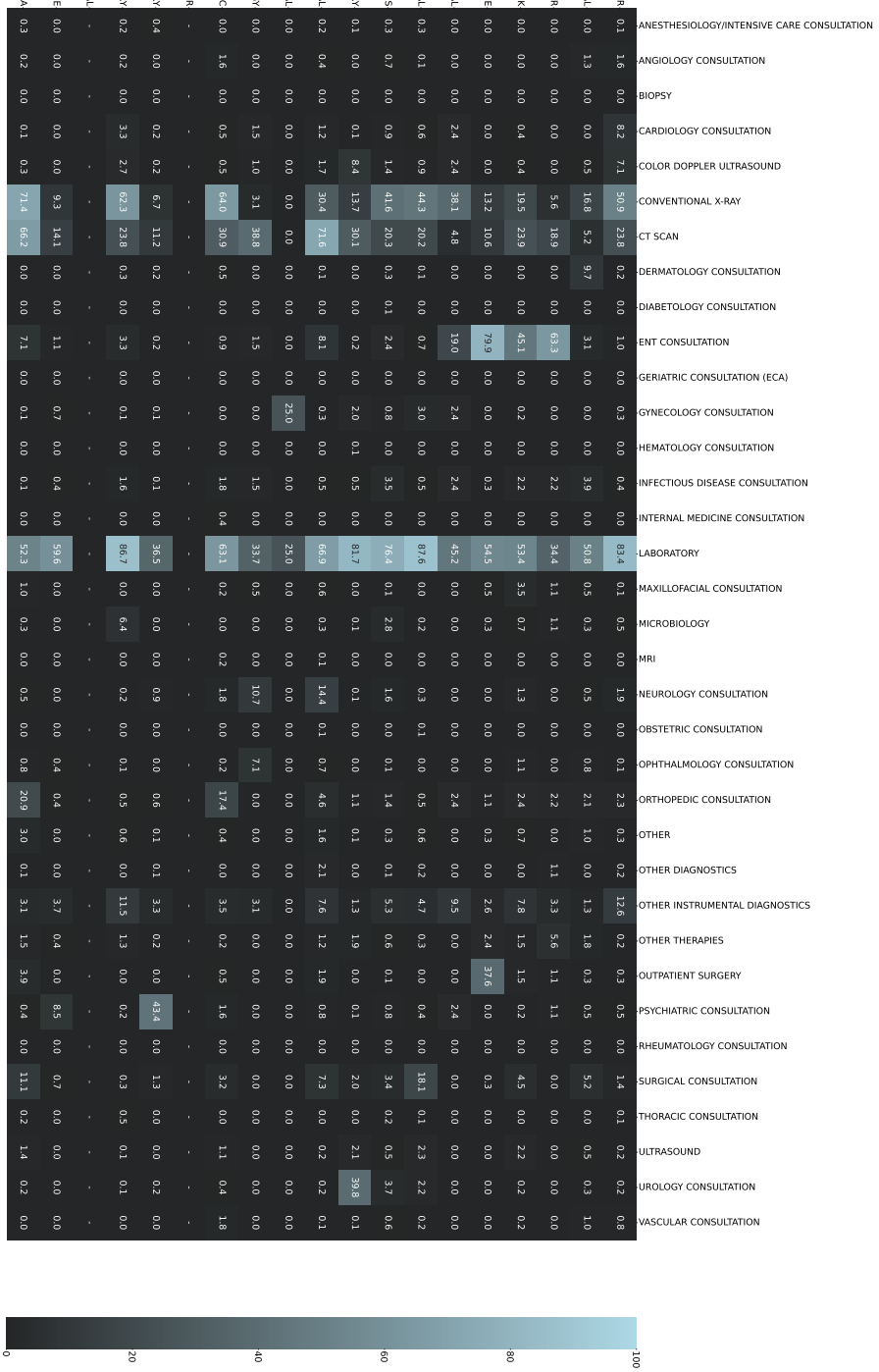

Figure A.3: Service patterns heatmap for patients with severity 3

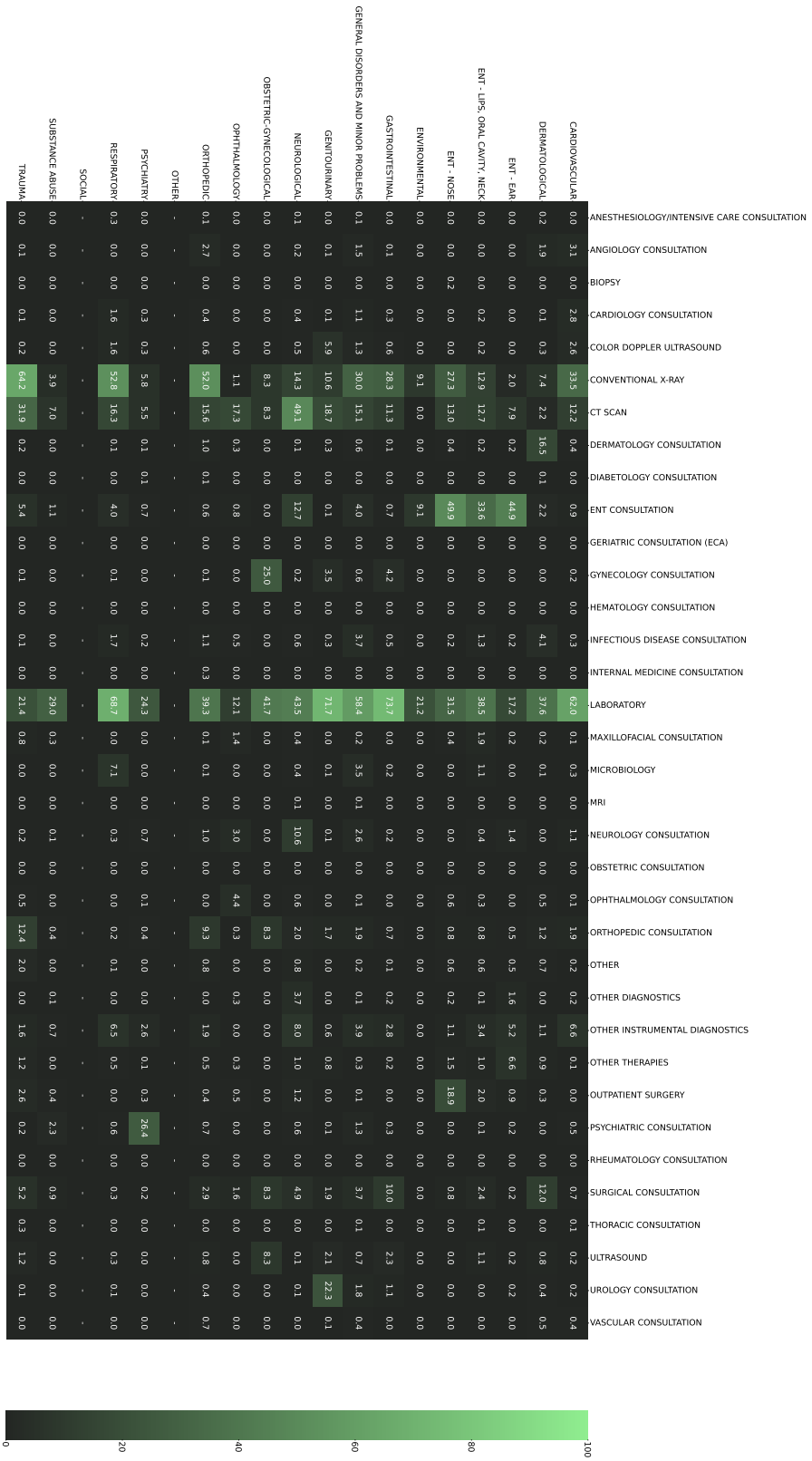

Figure A.4: Service patterns heatmap for patients with severity 4

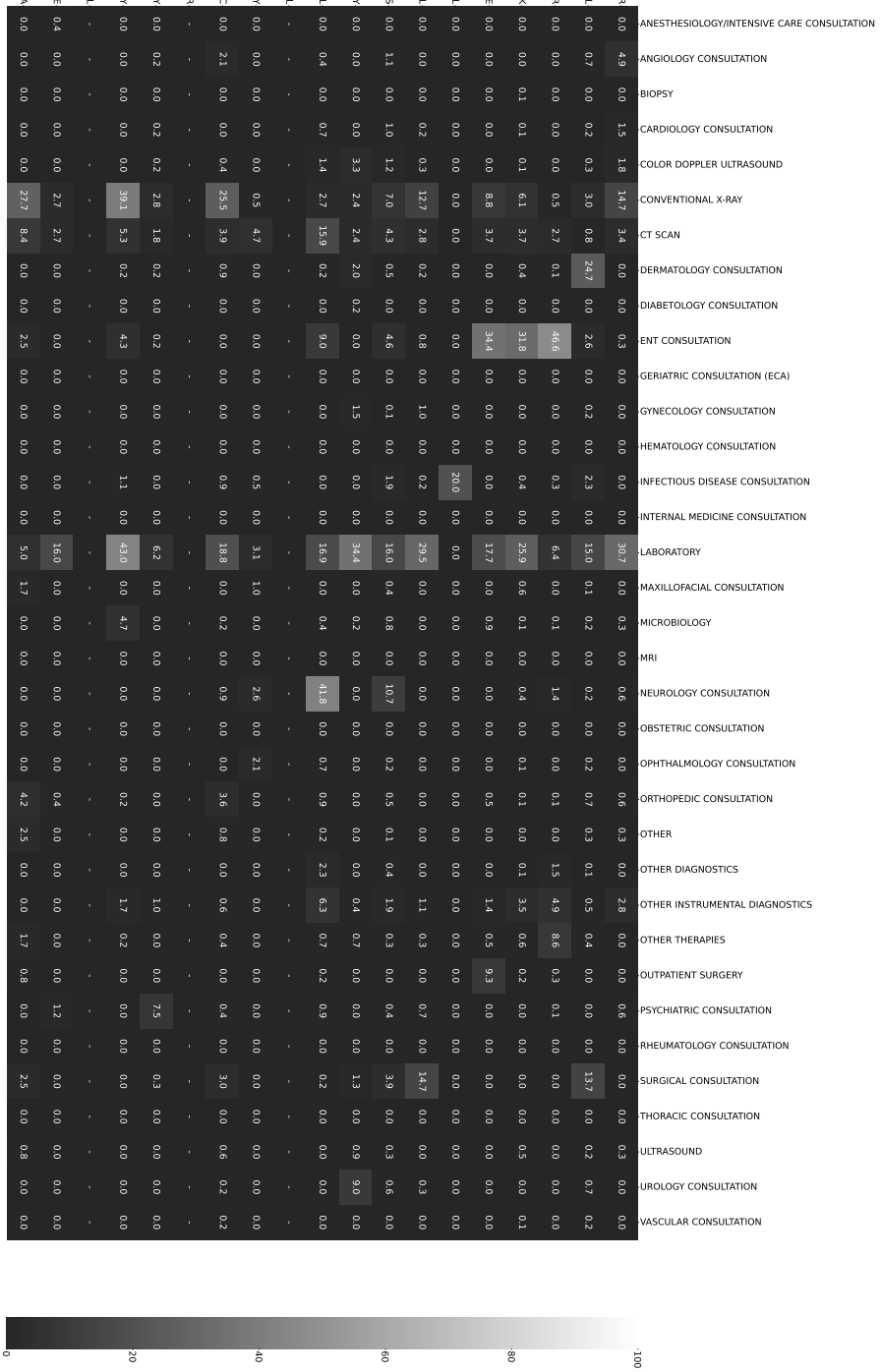

Figure A.5: Service patterns heatmap for patients with severity 5

B Additional Tables

Table B.1: Distribution of the main problems identified during triage, with a dash (−) if no patient has been assigned the problem in the target period

Identified problem	Frequency
Abdominal pain	8.95%
Dyspnea	6.61%
Minor non-specific disorders	4.65%
Polytrauma - blunt	4.65%
Chest pain of suspected cardiovascular origin	4.21%
Syncope / presyncope	4.01%
Flank pain	3.80%
Fever / hyperpyrexia / hyperthermia	3.79%
Chest pain not suspected cardiovascular	3.21%
Head trauma	3.15%
Palpitations / irregular pulse	3.09%
Headache / migraine	2.40%
Dizziness / vertigo	2.30%
Request for prescription or services	1.84%
Generalized weakness	1.61%
Nausea and/or repeated vomiting	1.57%
Hypertension	1.50%
Substance abuse / intoxication	1.44%
Cough / congestion	1.34%
Urinary tract infection symptoms	1.23%
Urgent request for specialist consultation	1.20%
Swollen leg / edema	1.20%
Anxiety / situational panic attack	1.17%
Confusional state	1.11%
Altered level of consciousness	1.10%
Gross hematuria	1.07%
Urinary retention	1.04%
Hematochezia / rectal bleeding / melena	1.03%
Sore throat / pharyngodynia	0.97%
Loss of sensation / paresthesias	0.91%
Limb weakness / cerebrovascular symptoms	0.83%
Thoracic trauma (single site) - blunt	0.80%
Epistaxis / nosebleed	0.77%
Lower limb pain	0.76%
Diarrhea	0.73%
Scrotal pain / swelling	0.71%
Imaging / laboratory tests	0.7%
Constipation	0.68%
Ear pain / otalgia	0.66%
Exposure to transmissible diseases	0.66%
Other skin problems	0.61%
Low back pain	0.58%
Pallor / anemia	0.55%
Depression / suicide / self-harm	0.54%
Social problems	0.50%
Bizarre behavior	0.50%
Seizures	0.48%
Upper limb pain	0.47%
Red swelling	0.47%
Erythema	0.46%
Warm and red limb	0.46%
Allergic reaction	0.43%
Clarification of imaging / lab reports	0.43%
Neck swelling / pain	0.40%
Facial trauma	0.39%
Swallowing disorder / dysphagia	0.37%
Medical device problems	0.34%
Upper limb injury	0.34%
Rectal / perineal pain	0.32%
Inguinal pain / mass	0.31%
Lower limb injury	0.30%
Hyperglycemia	0.26%
Unknown problem	0.24%
Nasal trauma	0.23%
Concerns about patient well-being	0.23%
Toothache / gum problems	0.21%
Post-surgical complications	0.21%
Hemoptysis	0.20%
Cold pulseless limb	0.19%
Itching	0.19%
⋮	⋮

Puncture wound	0.18%
Gait disturbance / ataxia	0.17%
Abdominal mass / distension	0.17%
Repeated hematemesis	0.17%
Oliguria	0.17%
Spinal trauma / vertebral injuries	0.16%
Generalized edema	0.16%
Violent / homicidal behavior	0.16%
Hallucinations / delusions	0.16%
Facial pain (non-traumatic, non-dental)	0.15%
Visual disturbances	0.15%
Genital discharge / lesions	0.15%
Hypoglycemia	0.14%
Tremors	0.14%
Burn	0.13%
Ocular foreign body	0.12%
Insomnia	0.11%
Penile swelling	0.11%
Jaundice	0.11%
Tinnitus	0.11%
Polytrauma - penetrating	0.11%
Direct eye trauma	0.10%
Hearing loss	0.10%
Nasal congestion / allergic rhinitis	0.10%
Bite injury	0.10%
Eye pain	0.10%
Wound check	0.09%
Dressing change	0.08%
Ear discharge / otorrhea	0.08%
Red eye with discharge	0.08%
Toxic substance inhalation	0.08%
Non-traumatic cardiac arrest	0.07%
Genital trauma	0.07%
Foreign body in mouth / esophagus	0.07%
Diplopia / double vision	0.06%
Anorexia	0.06%
Foreign body in external ear canal	0.06%
Ear trauma	0.06%
Nodules / protuberances / calluses	0.06%
Abdominal trauma (single site) - blunt	0.05%
Polyuria	0.05%
Wheezing / bronchospasm	0.05%
Patient left before completing triage	0.05%
Subsequent visit for therapy	0.04%
Upper respiratory tract inflammation	0.04%
Chemical contamination	0.04%
Breast redness / softening	0.04%
Periorbital swelling	0.04%
Laceration / puncture	0.04%
Substance overdose	0.04%
Spontaneous ecchymosis	0.03%
Withdrawal syndrome	0.03%
Loss of blood / body fluids	0.03%
Airway foreign body	0.03%
Joint swelling	0.03%
Neck trauma	0.02%
Exposure to chemical substances	0.02%
Abrasion	0.02%
Stridor	0.02%
Ophthalmologic follow-up	0.02%
Hiccup	0.02%
Nasal foreign body	0.01%
Hyperventilation	0.01%
Cutaneous foreign body	0.01%
Photophobia	0.01%
Sexual assault	0.01%
Electric injuries	0.01%
Exclusion of infestation	0.01%
Suture removal	0.01%
Child with disruptive behavior	0.01%
Thoracic trauma (single site) - penetrating	0.01%
⋮	⋮

Traumatic cardiac arrest	0.01%
Cyanosis	0.01%
Rectal foreign body	0.01%
Cast check	0.01%
Suspected heat stroke	< 0.01%
Gait disturbance / painful gait in child	< 0.01%
Respiratory arrest	< 0.01%
Vaginal pain / dyspareunia	< 0.01%
Abdominal trauma (single site) - penetrating	< 0.01%
Ring removal	< 0.01%
Inconsolable crying	< 0.01%
Hypothermia	< 0.01%
Pregnancy problems >20th week	< 0.01%
Menstrual problems	< 0.01%
Vaginal bleeding	< 0.01%
Rectal / anal trauma	< 0.01%
External genital swelling	< 0.01%
Neonatal apnea	< 0.01%
Neonatal feeding difficulty	< 0.01%
Postpartum problems	< 0.01%
Infant hypotonia	< 0.01%
Non-traumatic hemorrhage	< 0.01%
Other nervous system symptoms	< 0.01%
Self-harm	< 0.01%
Amputation	< 0.01%
Violence from others	< 0.01%
Urological symptoms or disorders	< 0.01%
Arterial hypertension	< 0.01%
Chest pain	< 0.01%
Arrhythmias	< 0.01%
Intoxication	< 0.01%
Pulmonology / respiratory disease	< 0.01%
Dermatological symptoms or disorders	< 0.01%
Acute neurological syndrome	< 0.01%
Oral / dental symptoms or disorders	< 0.01%
Social problem	< 0.01%
Shock	< 0.01%
Fall from standing height or less	< 0.01%
Coma	< 0.01%
Precordial pain	< 0.01%
Trauma >24h untreated	< 0.01%
ENT symptoms or disorders	< 0.01%
Psychiatric disorder	< 0.01%
Drowning	–
Newborn	–
Problems in children with congenital disease	–
Neonatal jaundice	–
Vaginal foreign body	–
Vaginal discharge	–
Frostbite	–
Pregnancy problems <20th week	–
Psychomotor agitation	–
Trauma >24h treated	–
Other symptoms or disorders	–
Suspected covid / covid	–
Ophthalmological symptoms or disorders	–
Fever	–
Obstetric / gynecological symptoms or disorders	–
Trauma within 24h	–
Forensic medical evaluation	–

Finanziato dall'Unione europea- Next Generation EU, Missione 4, Componente 2, Investimento 3.3 (D.M. 352/2022) CUP J33C22001460009

Funded by the European Union - Next Generation EU, Mission 4, Component 2, Investment 3.3 (Ministerial Decree 352/2022) CUP J33C22001460009