RESOURCE ALLOCATION OPTIMIZATION OF A DISRUPTED INTERDEPENDENT SYSTEM USING MACHINE LEARNING

by

Mohammed Talat Khouj

B.Sc. King Abdul Aziz University, 2001
M.Sc. King Abdul Aziz University, 2007

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

The Faculty of Graduate and Postdoctoral Studies

(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA
(Vancouver)

April 2014

Abstract

National safety and homeland security of an urban community rely heavily on a system of interconnected critical infrastructures (CI's). The interdependencies in such complex systems give rise to vulnerabilities, which must be accounted for by a proper disaster management. It is a proactive step that is needed to address and mitigate any major interruption in a timely manner. Only then will the management of CI's be able to appropriately reallocate and distribute the available scarce resources of an existing interdependent system.

In this research, we propose an intelligent decision making system that optimizes the allocation of the available resources following an infrastructure disruption. The novelty of our suggested model is based upon the application of a well-known Machine Learning (ML) technique called Reinforcement Learning (RL). This learning method is capable of using experience from a massive number of simulations to discover underlying statistical regularities. Two alternative approaches to intelligent decision-making are studied, learning by Temporal Differences (TD) and Monte Carlo (MC) based estimation. The learning paradigms are explored within the context of competing designs composed of simulators and learning agents architected either independently or together. The results indicate the best learning performance is obtained using MC within a homogeneous system. The goal here is to maximize the number of discharged patients from emergency units by intelligently utilizing the existing limited resources. We show that such a learning agent, through interactions with an environment of simulated catastrophic scenarios (i2Sim-

infrastrucutre interdependency simulator), is capable of making informed decisions in a reasonable time.

Preface

All of the work presented here was conducted in the Power and Energy System Research lab at the University of British Columbia, Point Grey campus. Some of the research results that are presented in this PhD dissertation have been published as conference proceedings, accepted for publication as journal articles or in the process of peer reviewing. For these publications that are listed below, I was responsible for developing the mathematical formulations, implementing the models, conducting the simulations, compiling the results and conclusions, as well as preparing the manuscripts. Both my Supervisor (Dr. José R. Martí) and Co-Supervisor (Dr. Sarbjit Sarkaria) have provided supervisory comments and corrections during the process of conducting this study and writing the manuscripts. The contributions of other co-authors for the published work or during the process of publication, if available, have been indicated and explained as follow:

A part of Chapter 4 was presented at a conference. Mohammed Khouj, César Loṕez, Sarbjit Sarkaria and José Martí, "Disaster Management in Real Time Simulation using Machine Learning", In proceedings of the 24th Canadian Conference on Electrical and Computer Engineering, 2011 CCECE, Niagara Falls, Canada. César Loṕez has provided modeling assistance, suggestions and constructive feedback.

A version of Chapter 4 has been accepted for publication in the International Journal of Critical Infrastructures. Mohammed Talat Khouj, Sarbjit Sarkaria and José R. Martí, "Decision Assistance Agent in Real-Time Simulation".

A part of Chapter 5 was presented at a conference. Mohammed Talat Khouj, Sarbjit Sarkaria, César Lopez and José Martí, "Reinforcement Learning using Monte Carlo Policy for Disaster Mitigation", In proceedings of Eighth IFIP WG 11.10 Conference on Critical Infrastructure Protection", 2014 IFIP, Arlington, Virginia, USA. César Lopez has provided modeling assistance and constructive feedback about the platform interface that was designed to establish a consistent intra-communication between the modeled environment and the proposed learning system.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

a	Action
ABS	Agent Based Simulation
ABM	Agent Based Modeling
AI	Artificial Intelligence
BC	British Columbia
CI	Critical Infrastructure
CPU	Central Processing Unit
CSI	Complex System Integration
CSQ	Cathedral Square Power Substation
CTI	Critical Transportation Infrastructure
DAARTS	Decision Assistance Agent in Real Time Simulation
DES	Discrete Event Simulation
DGR	Dal Grauer Power Substation
Dis	Distributor
DP	Dynamic Programming
DQ(λ)L	Distributed Multi-Step Learning
DQL	Distributed Learning
ER	Emergency Room
GB	Giga Byte
GHz	Giga Hertz
GIS	Geographical Information System
GM	General Motors
HIS	Hospital Information Server
HLA	High Level Architecture
HRT	Human Readable Table
i2Sim	Infrastructure Interdependency Simulator
ICU	Incentive Care Unit
IDS	Intelligent Decision System
IFI	Interdependency Failure Interdependencies
IRL	Individual Reinforcement Learning
JAVA	Object-Oriented Computer Programming Language (Sun Microsystems)
K	Kilo
KL	Kilo Liter
LUT	Look up Table
m^2	Meter Squared
m^3	Meter Cubed
MATLAB	Matrix Laboratory (Mathworks)

MC	Monte Carlo
MDP	Markov Decision Process
ML	Machine Learning
MUR	Murrin Power Substation
MW	Mega Watt
num	Number
OR	Operation Room
PC	Personal Computer
PM	Physical Mode
Q	Q-Learning Function
RAM	Random Access Memory
RCRSL	RoboCup Rescue Simulation League
RL	Reinforcement Learning
RL-MC	Reinforcement Learning using Monte Carlo Policy Estimation
RL-TD	Reinforcement Learning using Temporal Difference Policy Estimation
RM	Resource Mode
s	State
SOS	System-of-Systems
SPG	Sprling Power Substation
SPH	Saint Paul's Hospital
T	Terminal
TD	Temporal Difference
UBC	University of British Columbia
UOP	Ultimate Optimization Package
VGH	Vancouver General Hospital
XCS	Holland's Genetic Classifiers System

Glossary

Agent	An intelligent agent that captures knowledge using Reinforcement Learning.
Scenario	A series of predefined events occurs during simulation.
Event	A change in the operating conditions (PM's and RM's) within the system.
Episode	An emulation of a selected scenario for a given duration (Simulation time).
Run	A set of episodes where the learning continues from one episode to another.
Path	A learning route that is determined by RL-TD using step-by-step backup.
Trajectory	A track that is discovered by RL-MC using episode-by-episode backup.

Acknowledgments

It has been over five years to come at this moment. It is the final moment in my journey, which I have been waiting for since I have been admitted to the Department of Electrical and Computer Engineering at the University of British Columbia in January 2009. It is an adventurous journey that I will keep remembering as long as I live. It is a journey that I will keep telling to those who want to pursue their academic career. In fact, it is a journey that has been written from my own memories, and now it is the time for this journey to have a good ending. Thus, it is the time to write down the final pages of my doctoral dissertation.

Completion of this doctoral dissertation was possible with the support of several people. I would like to express my sincere gratitude and appreciation to both my Supervisor Dr. José R. Martí and my Co-Supervisor Dr. Sarbjit Sarkaria for their endless support, guidance and patience. Their purified spirits, manifest humbleness and obvious kindness have thought me a lot. Without them my recent academic successes and in particular this dissertation would not be alive.

Working within a research group is an opportunity that saves your effort and time. I was lucky to join a wonderful group called i2Sim back in 2009. This research group helped me to reshape my knowledge and enhance my background by conducting a weekly open discussion meeting, which is initiated by the members. This group is supervised by two wisdom philosophers Dr. José R. Martí and Dr. K.D. Srivastava, who inspired the graduate fellows by their topics and speeches. Thank you all and keep it up i2Sim.

I owe my deepest appreciation and regards to my wonderful parents, my father Talat Khouj and my mother Reema Khaled. Their continuous love, support, and prayers made this work possible. Their words of encouragement are still motivating me and always remind me that every problem has a solution, just remember do not give up.

I am also indebted to my love, to my soul, to my companion, to my wife Reem Jamalallail. The mother of the three magnificent children Leen, Wessam and Yousef has sacrificed to make my dream comes true. Because of her continued support, persistent encouragement and endless patients I was able to stand against the hardships and stressful moments during the years of my PhD.

Leen, Wessam and Yousef you are the future. Whatever achievements I have made throughout my entire life, they are just an investment that may contribute to your prosperous life. Always remember that you matter most to your parents.

I would like also to thank my brothers Ahmed Talat Khouj, Dr. Osama Talat Khouj and Abdullah Hussain Khouj for their support and prayers. You are a real representation of the brotherhood symbol. I still remember your supportive words that helped me to complete this long journey.

My family in-law deserves all appreciation and respect. Laila Khushifati (mother in-law), Dr. Mohammed Jamalallail (father in-law), Dr. Faisal Jamalallail (brother in-law) and my sisters' in-law all are ready to help and support whenever needed. It is an honor for me to have an extended family like you. Thank you all from the bottom of my heart.

My field study was made achievable because of the presence of my friends (in particular Haitham Eissa, Amr Alasaad, Ghassan Alshareef, Aiman Erbad, Tariq Alkhasip and Talal Alzeiby), colleagues (in particular to Abdullah Alsubaie, Khaled Alutaibi, César Lopez and Marco Gonzalez) and fellow graduate students (in particular Sina Chini Foroosh, Mehrdad Chapariha, Hamid Atighechi and Milad Fekri Moghadam) in the department. I would like to thank you all for your services, cooperation and encouragement.

The financial support for this research was made possible through the Ministry of Higher Education in the Kingdom of Saudi Arabia.

Above all, I owe praise and thanks to almighty God, the source of all good and guidance for granting me the wisdom, health and strength to undertake this task and enabling me to complete this long journey.

Dedication

To all humankind . . .

CHAPTER 1: INTRODUCTION

1.1 Motivation

Urban society relies heavily on a system of critical infrastructures (CI's) such as power and water systems. These systems are inherently complex in terms of interconnections and dependencies. That is why they are vulnerable to any major disruption, which can cascade to other dependent systems with possible disastrous consequences. For instance, the India Blackout in 2012 (considered to be the largest power blackout incident in the human history) caused a huge disruption to medical units, transportation system, water treatment plants and other interconnected infrastructures. This subsequently resulted in the loss of power to 600 million people, trapping miners, stranding train travelers and plunging hospitals into darkness [1]. Other catastrophic events like the Indian Ocean Earthquake-Indonesia in 2004, Hurricane Katrina-USA in 2005, the Great Eastern Japan Earthquake-Japan in 2011 and Typhoon Haiyan-Philippines in 2013, are horrible incidents in human history that caused massive loss of lives and economic activity. These incidents reveal the need for proficient preparedness, efficient planning and coordination at every level of the crisis response management process and particularly at the decision making level [2].

Effective management of the available, and often limited, resources in an urban community is essential to disaster mitigation. In natural or human-induced emergencies, the decisions that are made in the first minutes, hours and days are critical to successful mitigation,

damage management, death prevention, injury, structural loss, and control of financial costs and ultimately, the overall resolution of the disaster [3].

Thus, a series of carefully chosen decisions are vital in mitigating death and disaster following a catastrophe such as an earthquake. These decisions must be made on the basis of sound knowledge and experience. However, given that the worldwide frequency of such situations is, fortunately, low and that the likelihood of the same command and control personnel encountering similar scenarios over and over again is slim, it can be appreciated that opportunities to build up the necessary experience are severely limited. This is the context of this research work and we maintain that such decisions need to be carefully studied and pre-measured before implementation. In addition, they need to be monitored and modified as the situation evolves.

1.2 Background

Catastrophic events are well known for their rapidly changing behaviors. These behaviors occur within different time scales, which make the disaster mitigation process highly dynamic. Resource allocation decisions that are made by a decision maker within a dynamic environment can strongly affect the state of that environment. This state is recognized by its operating conditions that must be defined and analyzed beforehand. Only then will the decision maker be able to take the most effective resource allocation action that optimizes a targeted outcome in a realistic time. This outcome can be presented

financially, politically or numerically such as the total number of discharged patients in a hospital.

The difficulty in making effective decisions during disasters lies in the following reasons [4]:

1. Decision making during an emergency scenario typically has to be performed within a limited time frame because the crisis occurs suddenly and may evolve rapidly.
2. Decision making during an emergency response is often unstructured, fuzzy and unexpected in nature.
3. Information available to decision makers is often insufficient and/or inaccurate. A complete picture or understanding of the scenario cannot be collected within a limited time frame, thus decision makers can only rely at best, upon incomplete information.

The Resource allocation optimization problem is a research area that has been extensively investigated. In the following sections an overview of the resource allocation problems in the context of Critical Infrastructure (CI) has been addressed. Some studies have modeled CI's independently, whereas others have addressed the problem with interdependent CI's.

1.2.1 Resource Allocation Problem

Many studies and researches have addressed the resource allocation problem in attempts to assist the emergency responders to alleviate the impact of catastrophic incidents. For example, [5] proposed a solution, called "ARES", for solving the limited resource

reallocation problem. This approach consisted of a decision support system simulator and a linear optimization model to simulate an earthquake scenario. The objective was to find good aggregate assignments of tasks to the units involved in the affected areas using the available resources.

Similarly, [6] was able to minimize the number of fatalities by scheduling resources heuristically in order to optimize the search-and-rescue performance. To do so, a linear programming model called "ALLOCATE" was developed to determine the required operations in the affected areas using the allocated resources.

In another example, [7] investigated an integrated fuzzy linear programming and fuzzy clustering method to minimize the time needed for rehabilitation and relief distribution to affected areas. This was achieved by grouping affected areas according to travelling distance, locating distribution hubs and finally applying a fuzzy vehicle routing program.

Reference [8] presented a quick responsive emergency logistic distribution approach during the rescue period. Using a multi-objective optimization model, the approach predicted the time varying relief demand of an impacted area. Then, based on the level of damage and the degree of urgency, the affected areas were grouped using a fuzzy clustering technique. This enabled scheduling of relief supplies in priority order.

Lastly, [9] tried to maximize the service coverage of emergency vehicles in the critical transportation infrastructures (CTI) by using an analytical optimization approach (mixed integer linear programming model). This was obtained by studying the fluctuating

demands of the CTI nodes and the transportation network performance on the optimal coverage.

1.2.2 Resource Allocation Problem within Interdependent Infrastructures

The above models were able to present reasonable assistance to emergency decision makers, however, they do not give a clear vision of how those models would behave in the context of an interconnected infrastructure failure. Specifically, it was assumed that a facility will continue to provide service under severe circumstances [10].

CI systems are complex in terms of interconnections and dependencies. These systems are vulnerable to any major disturbance that would cascade to other dependent systems, which may lead to national disasters (Figure 1-1). For example, the Northeast Blackout in 2003 caused a huge disruption to medical units, telecommunication centers and financial institutions. This subsequently resulted in the loss of power to 50 million people, 11 deaths, and incurred costs in the order of $6 billion [11]. Interdependencies between infrastructures can be defined as "a bidirectional relationship between infrastructures through which the state of each infrastructure is influenced by or correlated to the state of the other" [12]. Therefore, it is important to address the resource allocation problem in the context of interdependent CI's.

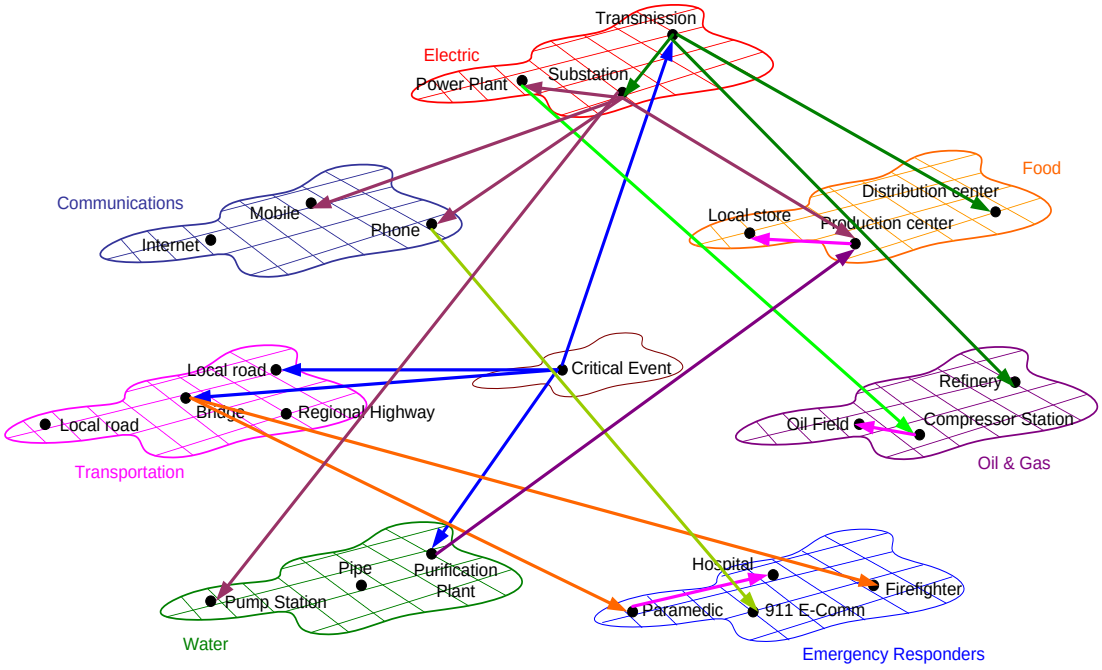

Figure 1-1 System of interconnected critical infrastructures (CI's)

Resources allocation for interdependent CI's is an area of research that has been revisited recently. A proposed conceptual framework [13] consists of a unique vast database developed to investigate the contribution of disrupted infrastructures to social and economic loss. This database helped in identifying the vulnerabilities associated with such systems in the case of natural or human-made disasters. This framework was designed using an empirical approach, which helped to characterize infrastructure failure interdependencies (IFIs) patterns and their impacts. The originality of this work relies on the ability to facilitate the characterization of IFIs in a basis of societal consideration. This framework was implemented by developing measures and characterizations of specific kinds of IFIs within a defined area of CI systems. The proposed framework was then used to illustrate the power outage incident that occurred inconsequence to the heavy Ice Storm

in 1998 at Eastern Canada. In this case study, an enriched database was developed and extensive analysis was conducted for this extreme event. The study indicated that the greatest impact was to local business. A two-week shutdown was reported by major employers south of Montreal, Canada, resulting in huge short-term monetary loss by manufacturing industry, estimated to be in the millions of dollars.

The work of [14] presented an integrated system to model the physical and financial impacts attributed to interdependencies between CI's. This framework consisted of a system dynamics model, functional modeling and a non-linear optimization model. The purpose of the system dynamics model was to analyze the interdependency between individual infrastructure components. The functional model was used to define the data requirements and the information exchanged between the models. The non-linear model enabled determination of the optimal values of the control variables. The purpose of this work was to enable officials to respond to any potential disruptions in a timely and effective manner.

A system dynamics model was also used by [15] to describe the interaction between interconnected infrastructures. They analyzed the impact of a telecom infrastructure failure on emergency services and concluded that the outcome of a medical emergency is negatively impacted and the costs of treating cases are significantly increased during loss of communications.

Similarly, [10] addressed the impact of failures between interdependent infrastructure components on the operation of healthcare facilities during a disaster. The goal was to determine the unsatisfied demand of the interconnected infrastructure systems and its cost using a network flow model. Linear programming was used to assess the level of interdependency between the healthcare facility and the primary infrastructure systems that are linked to it.

Similarly, [16] presented the internal operating capabilities of healthcare facilities in terms of their interaction between different service areas (emergency room – ER, intensive care unit – ICU, operation room – OR and wards). This was done using a system dynamics simulation model. The goal was to assess the vulnerability of a healthcare facility during a disaster compared to normal conditions. The approach enabled the identification of policies implemented within the facility to best mitigate the effects of a disruption.

Finally, [17] integrated the network flow model and the system dynamics model. This was done to simulate the impact of disruptions in the infrastructure systems on the operations of the healthcare service areas.

In this research, the importance of interdependency within an interconnected CI's system is considered and modeled using i2Sim. i2Sim (infrastructure interdependency simulator) is a time domain simulator that is used to simulate the impacts of major disruptions such as earthquakes on the modeled CI's [18]. Here, i2Sim is proposed as decision tool to assist in decision verification before the real implementation.

It is clear then that informed decisions that are made within an interdependent CI's system would assist significantly in disaster mitigation. These decisions have to be made on the basis of sound knowledge and experience. This is because optimal utilization of the available limited resources can have a strong influence in mitigation of death and devastation following a disaster. Luckily, the worldwide occurrence of such situations is minor. So the probability of the same command and control personnel encountering similar scenarios is too low. Thus, the ability for any one individual to compound experience in such scenarios is highly desirable but unfortunately limited. The following section attempts to addresses this issue, and describes an agent-based model where both knowledge and experience can be captured by an intelligent agent.

1.2.3 Agent Based Modeling for Disaster Mitigation

Agent-based modeling (ABM) refers to a system of single multiple autonomous decision-making entities called agents. These agents are capable of sensing and interacting with each other within a modeled environment based on a set of predefined rules. These rules determine the behaviors of the modeled agents and allow them to perform appropriate actions.

ABM is suggested to be used in many real-world applications because of the following advantages [19]:

1. It captures emergent behavior that results from the interaction of the individual entities (agents).

2. It provides a natural description of the modeled system by modeling and simulating the behaviors of the interacted entities.

3. It has a high flexibility, which allows for tuning the complexity of the individual entities. For example the number of the modeled agents and their behaviors can be increased and deployed as the system is extended.

These advantages have encouraged researchers within the disaster response community to implement ABM for disaster mitigation. For example, [20] proposed an agent-based model for patient information acquisition and real time decision making during emergencies. The goal was to offer an accurate diagnosis and exact treatment in a timely fashion for high-risk patients during emergency situations. This was accomplished by using a Reinforcement Learning (RL) agent model with an embedded Dynamic Programming (DP) learning mechanism to evaluate and improve the system value function and its policy. More will be discussed about RL based learning agents and their properties in section 3.2.3 (Chapter 3).

Another example, [21] proposed a mechanism to study and analyze the interdependencies between industrial control systems and the underlying critical infrastructures. This research assesses security and reliability in conjunction with various kinds of catastrophes and disasters. This was achieved by the so called coupled approach, in which a System-of-System (SOS) architecture is modeled using Agent Based Modeling (ABM) and High Level Architecture (HLA). The ABM was used to exhibit complex behavior patterns and provides valuable information about the dynamics of the simulated CI's, where the HLA was

considered to help in establishing the necessary communication between the CI level and the SOS level. To demonstrate the capability of the above-suggested approach, two designated experiments were conducted. The purpose of the feasibility experiment was to study the ability of the proposed approach to represents the CI's interdependencies. For the failure propagation experiment the idea was to realize the failure propagation between the modeled interdependent CI's due to a catastrophe. The outcomes revealed the ability of such approaches to investigate the interdependencies between modeled critical infrastructures, which helped to address the associated vulnerabilities.

In a third example, [22] proposed a risk assessment tool for seismic hazard that consisted of a multi-agent system. This system was designed with an adaptive knowledge base and different goals. This tool offered city stakeholders help in creating a risk management plan for better safety and response activities. The designed approach was used to simulate emergency response actions for a set of earthquake scenarios on different location within an urban community. The generated results when projected onto a Geographical Information System (GIS) map which would help to improve the quality of the decisions. The decisions were typically made post-event for restoration and recovery operations aimed at rehabilitating damaged infrastructures.

Finally, [23] proposed a comprehensive simulation model to assist decision makers in planning and managing responses to disastrous events. Clearly, it is not practical to attempt to guess the time, location and scale of a future event. Thus, pre-event planning and preparedness are needed for better disaster mitigation. To achieve this objective, a hybrid

discrete event simulation (DES) system was proposed consisting of an agent-based simulation (ABS) model, a geographical information system (GIS), databases and a rule based system. This proposed platform was able to offer realistic decision support in reasonable time using common Personal Computers (PC). The implementation of this innovative platform helped reduce the computational demands without compromising the accuracy of the system outcomes.

For this research, two agent based modeling approaches are used. Here, we explore two different ways in which to formulate the RL problem and deploy both, Temporal Difference (TD) and Monte Carlo (MC), as mechanisms for learning a policy. These approaches enable our learning agent to explore and exploit the available possible paths and trajectories that lead to an optimum goal in a reasonable time. These approaches differ in how and when the learning takes place, each having advantages and disadvantages that will be discussed later.

1.3 Scope and Goals of the Research

Disaster mitigation is a topic that has been investigated and developed in the past using different approaches, such as the framework presented by [24]. This framework was proposed for assessing empirical patterns of urban disaster recovery using statistical indicators. Alternatively, in the present research, we propose a decision system in which an intelligent agent will learn by experience through continuous interactions with a simulated environment. This interaction is defined in the form of sensing the physical operability and the resource availability of the critical infrastructures (CI's) and then learning to perform

those actions that intelligently reallocate the available limited resources, which ultimately lead to the best-case desired outcome (Figure 1-2).

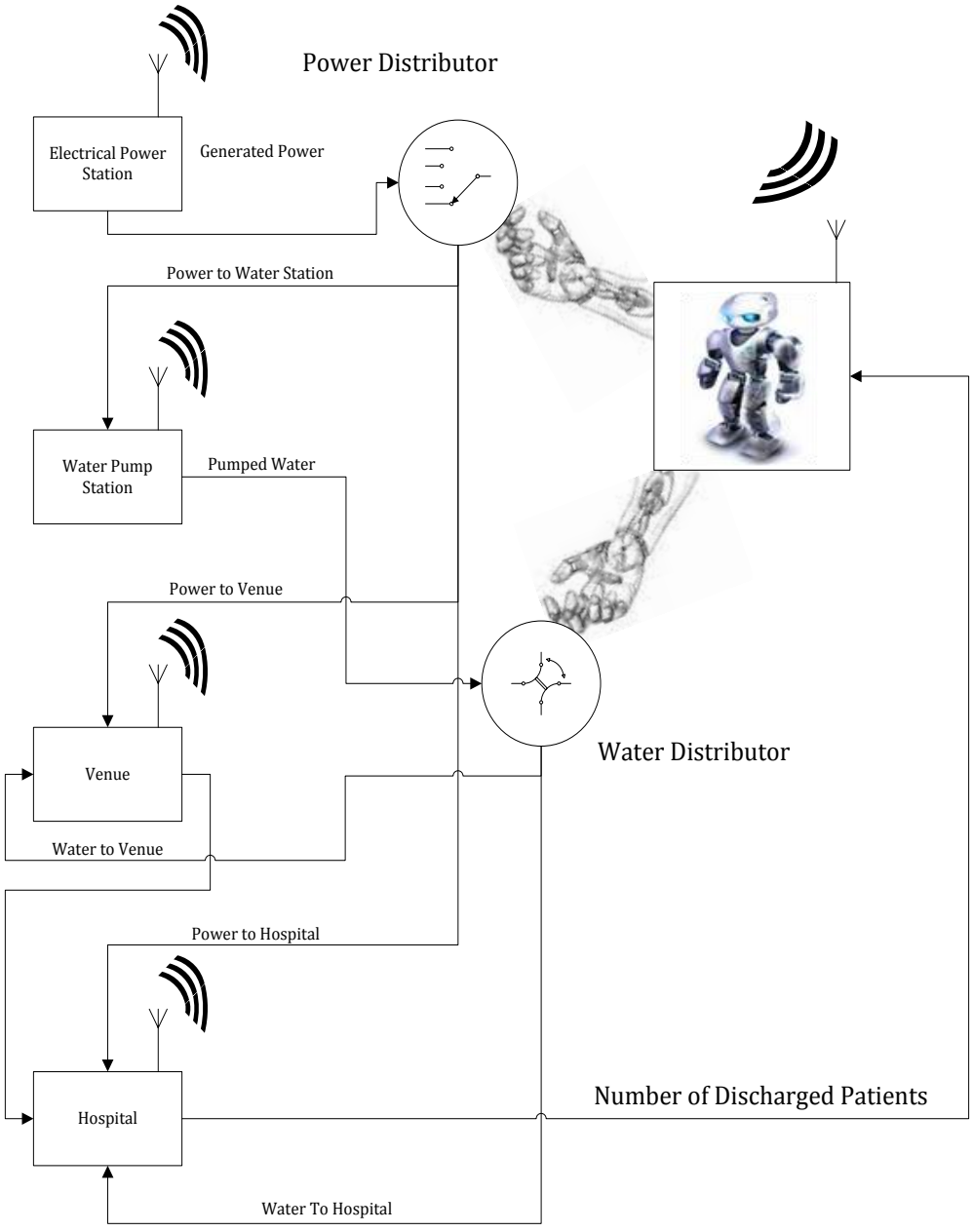

Figure 1-2 i2Sim and RL schematic diagram

Government policies typically consider the following entities as components of a region's CI (Table 1-1):

Table 1-1 Canada critical sectors

Energy	Water	Food	Finance	Communication
Transport	Health	Safety	Government	Manufacturing

All of these entities contribute variables that may be candidates for modeling within the scope of this research. However, in cases where the variable is unaffected by simulated disaster, it will for simplicity, be excluded. For example, related to health, it is anticipated that the stock of medication within a hospital is likely to last the duration of the disaster scenario and thus will not affect the outcome of the simulation. Whereas energy and water facilities will be disrupted by a simulated earthquake and changes in their disruption will significantly influence the outcome. Another example relates to the availability of health-care staff. It is understood that doctors and nurses live within walking distance of the hospital, thus are not affected by the simulated disaster. All these are considered within the scope of the simulation.

The primary objective of this work is to demonstrate the ability of intelligent machines to assist emergency responders during catastrophic events. Here, the goal is to reallocate the available limited resources optimally within an interdependent CI's system for better disaster mitigation. Such actions need to be made on a base of sound knowledge and experience, which humans may lack. In this study we investigate the possibility of pre-training an agent such that it may be made to exhibit intelligent behavior. The agent

behavior is trained using a well-known machine learning technique called Reinforcement Learning (RL) [25]. Prior work by [26] has shown that in principle this is possible. This work builds upon that previous work and shows a way of improving computation time by an order of magnitude. The core function of RL is to discover a policy that optimizes a long-term reward. RL is able to do this in the presence of uncertainties. Policy estimators are used to develop a pre-trained intelligent agent that has experienced a set of massive disaster scenarios.

This thesis explores three different learning models, referred to as DAARTS (Decision Assistance Agent in Real Time Simulation), IDS (Intelligent Decision System) and i2Sim-UOP. DAARTS uses a Temporal Difference (TD) based learning approach which is capable to achieve the optimum reallocation of the available compromised resources. However, its convergence times are relatively long. IDS on the other hand, uses a Monte Carlo (MC) based approach, which results in a significant improvement over DAARTS within a similar framework having convergence rates estimated to be around two times faster than DAARTS. Ultimately, a substantial advancement is obtained using an architecture in which the components are directly connected within a homogenous software framework. This single platform (i2Sim-UOP) is shown to optimize the desired outcome almost another order of magnitude faster than IDS.

1.4 Contribution and Organization of the Dissertation

This PhD dissertation addresses the resource allocation optimization problem within interdependent critical infrastructures (CI's) using Temporal Difference (TD) and Monte Carlo (MC) learning methods. A description of the practical differences between the TD and MC policy estimation is provided, and both methods are used to build a model of an intelligent Reinforcement Learning (RL) agent. This agent will suggest the actions that are needed to maximize the goal of mitigation death and injuries during disaster. This goal can be measured in terms of the number of discharged patients from hospitals or on site emergency units. We propose that by exposing such an intelligent agent to a large sequence of simulated disaster scenarios using i2Sim, we will capture enough experience to enable the agent to make informed decisions leading to an optimal outcome in terms of casualties or other damage. In this study we show that such optimality depends on the architecture of the modeled system and how the intelligent agent communicates or interacts with its environment. The most efficient architecture was determined to be that in which the agent and the simulation software co-exist within the same software framework, rather than as separate de-coupled entities.

The organization of this dissertation is as follows: The present chapter (Chapter 1) introduces disaster mitigation problems to highlight resource allocation optimization within an interconnected critical infrastructures system. Next, it outlines the research objectives of the dissertation and presents a literature survey to establish the related background work. Chapter 2 introduces the infrastructure interdependency simulator

(i2Sim) that supports modeling and simulation of a system of interconnected critical infrastructures. This simulator is used to model the targeted environment, which in this case is Downtown Vancouver. The RL agent will then be exposed to and interact with this environment. Chapter 3 provides a background to Machine Learning (ML) with an emphasis on RL and its real-world applications. Descriptions about the proposed response crisis management framework and its platform interface are presented. This interface will appear in subsequent chapters as a means of communication between the RL agent and the modeled environment. In Chapter 4 and Chapter 5, TD and MC are introduced respectively. Both methods are used to solve the resource allocation optimization problem within the environment that is presented in Chapter 2. System architectures, algorithms, models and assumptions of both RL agents (DAARTS and IDS) are presented and explained within these chapters. We consider these as competing models and attempt to instrument their learning performance. To this end, the suggested scenario is then applied to both systems and the outcomes analyzed. For Chapter 6, a third architecture referred to as the i2Sim-Ultimate Optimization Package (i2Sim-UOP) is proposed and described. In the design of this architecture, a version of i2Sim and the learning agent are realized within a single platform. Next, the performance of each of the competing system designs are compared and analyzed in Chapter 4 (DAARTS), Chapter 5 (IDS) and Chapter 6 (i2Sim-UOP). A comparative analysis has been conducted in terms of capacity and computation speed of the proposed learning systems. Finally, Chapter 7 summarizes the primary contributions of the dissertation and indicates several relevant issues and possible future work.

CHAPTER 2: INFRASTRUCTURE INTERDEPENDENCY SIMULATOR (I2SIM)

2.1 What is i2Sim?

i2Sim (infrastructure interdependency simulator) is a hybrid discrete time simulator that combines agent-based modeling with input-output production models. The simulator can model and play out scenarios within an infrastructure of an interdependent system. The simulator is able to model critical interdependencies under disaster circumstances between key facilities, such as power plants, water processing facilities and hospitals within a typical urban environment. i2Sim is being designed as a real-time simulator, which can also act as a decision advise tool while a disaster is actually happening by being able to "look ahead" at the predicted consequences of suggested decision actions before such decisions are actually made [27]. Although we used i2Sim to study and analyze the consequences of fast evolving catastrophes, i2Sim could also be used to simulate slower ones like those described in [28].

The i2Sim framework allows multiple infrastructures to be represented as a system of systems and explicitly solve for their external interactions (interdependencies) through a number of suggested scenarios such as an earthquake [18]. The results of these scenarios can then be used to optimize procedures and resources allocation.

As explained earlier, the ability to reallocate the existing resources of a modeled environment depends on the amount of knowledge and experience that the agent have acquired. Therefore, a Reinforcement Learning (RL) agent needs to interact continuously with a realistic simulated environment, like Downtown Vancouver, which consists of a number of interdependent systems. Only then the agent will be able to intelligently utilize the modeled resources that eventually help in optimizing a desired outcome.

2.2 i2Sim Ontology

The number of variables to model and simulate any real-world dynamic environment is massive due to the dimensionality of the problem. To overcome this issue, the i2Sim team at the University of British Columbia (UBC) has developed a novel ontology that allows a large diversity of infrastructures to be characterized using a common set of concepts while, at the same time, reducing the dimensionality of the representation [18].

The large geospatial and temporal extension of the physical systems to be represented during large disasters and the uncertainties in the data available has been a crucial motivation for the development of this unified ontological representation [29]. i2Sim's ontology defines a world space where the following entities exist (Figure 2-1):

• Cells (production units):

A cell is an entity (production unit) that performs a function. This function represents the relationship between the inputs and the outputs of this production unit. The inputs and

outputs are defined by a human readable table (HRT) that relates the available outputs to a number of possible operating states of the modeled system. The physical operating states are called “Physical Mode-PM” while the level of the output is called “Resource Mode-RM”. Both PM’s and RM’s are state variables. Knowing these variables will assist the RL agent in visualizing the surrounding environment, which will ultimately help in broadening its experience.

• Channels (transportation units):

A channel represents the means by which the resources (tokens) are transported from the generating source cell to the consuming load cell. Channels are characterized by a loss coefficient and by a time delay during the transportation process between the cells.

• Tokens (exchange units):

Tokens are the resource quantities that act as the inputs and the outputs of the cells, e.g., water is a token, electricity is a token and a patient is a token as well. It is the amount of resources what determines the state of the modeled cells, the more the resources the healthier the state is.

• Controllers (distributor and aggregator units):

Distributor and aggregator units are the points at which decision makers control the resources allocation. In the distributors, the tokens are sent from the generating source to

different consuming loads. Aggregators act as collectors that gather the same kind of tokens from different generating sources to be used as input to a cell.

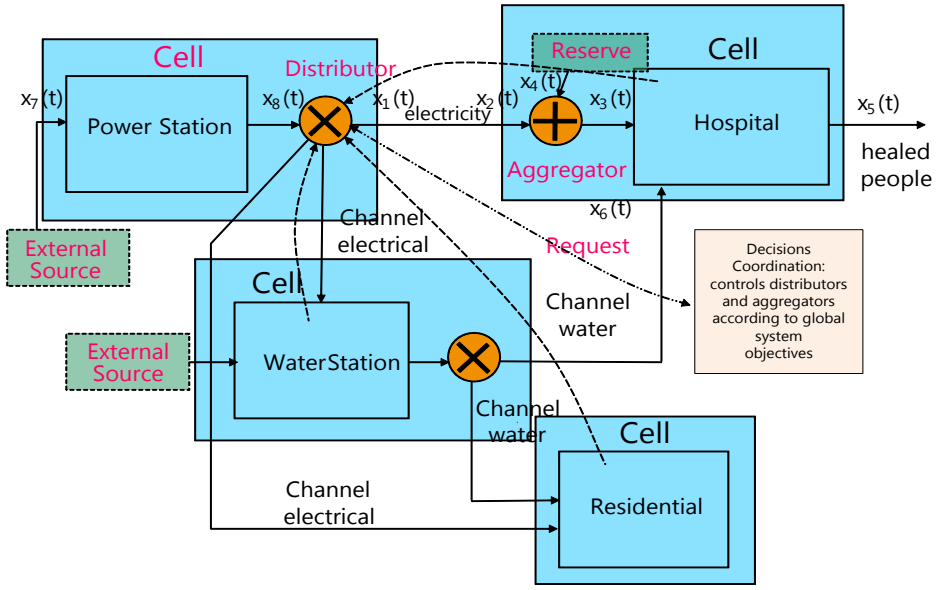

Figure 2-1 i2Sim Cells, Channels, Distributors and Aggregators

What makes the i2Sim model appear as a dynamic environment is the movement of tokens between the cells through designated channels. In fact, cells and channels are discrete notions of the real physical world. At every operating point along the event time line, the idealized i2Sim description corresponds to a system of discrete time equations called "transportation matrix" (Figure 2-2). This matrix relates input quantities that arrive at the cells with the quantities that are produced at the output of other cells.

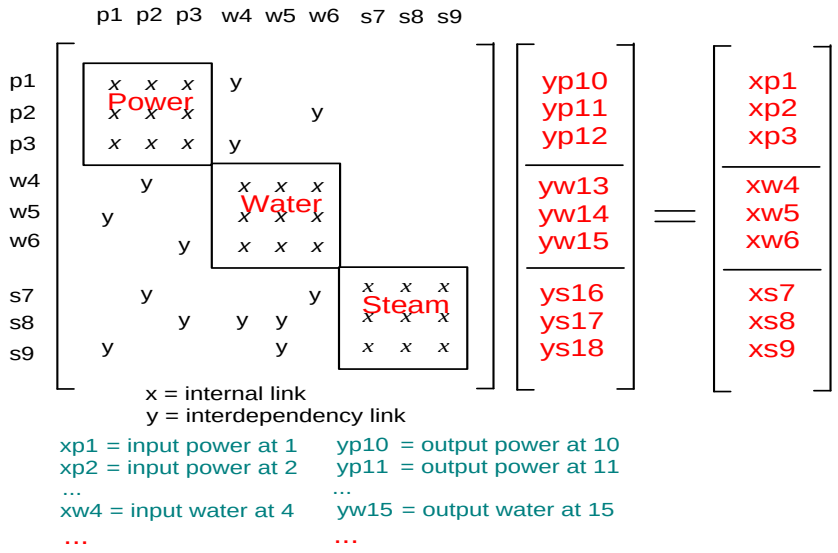

Figure 2-2 System transportation matrix showing interdependencies among infrastructures

In this proposed system, distributors are the entities that are used by the intelligent agent to perform actions. These actions define the distributions of the available tokens (resources) within interdependent systems based on the acquired knowledge and experience of the modeled agent.

It's worth noting that both the states (operating modes) and actions (distributors) of the modeled critical infrastructures (CI's) system are parameters that will be used to represent the knowledge needed by the Reinforcement Learning (RL) agent to improve its experience. This experience is presented by a table that ties every visited state with every performed action within the simulated environment. This table, known as a Look Up Table (LUT), is a probabilistic representation which attempts to capture the long term expected return associated with taking an action in the given state. It forms the core of an intelligent agent that will use this information in its decision making process.

2.3 Description of the Downtown Vancouver Model

The following section describes various details of the environment modeled using i2Sim (infrastructure interdependency simulator) and the resource allocation problem.

2.3.1 Crisis Management Framework Representation

Nine interdependent Critical Infrastructure (CI) cells are modeled and connected to each other by a number of lifelines (channels). The same kind of tokens will be aggregated and distributed along the presented scenarios. It is important to point out that some of the capacity and operating parameters of the production cells, distributors and channels were defined based upon public domain information, while others were obtained directly from the utility managers. The numbers in these tables are representative of particular operating conditions (Appendix A). The infrastructure models are defined as follows:

2.3.1.1 Hospitals (VGH and SPH)

The two-modeled hospitals are Vancouver General Hospital (VGH), the primary hospital, and Saint Paul's Hospital (SPH), the secondary one. At these hospitals, the Emergency Units are prepared to accept incoming patients sent for treatment. The inputs for these production cells come from the electrical power station (electricity), water pumping station (water), natural gas or steam (for heating purpose) and medical gas (for medical use). Based on the availability of these resources, the rate of discharged patients is known from experience in operating the hospitals to be as shown in Table 2-1 and Table 2-2. It is

assumed that 1/10 of the total hospitals' resources are needed by the Emergency Units (EU's).

Table 2-1 HRT for Vancouver General Hospital (VGH) production cell

Physical Mode # 1					
RM	Emergency Room Output	Hospital Inputs			
	Rate of Treatment [People/hour]	Electricity [MW/hour]	Water [KL/hour]	Natural Gas [m^3/hour]	Medical Gasses [%]
1	10.00	2.00	5.15	9.44	10.00
2	7.00	1.50	3.80	7.08	7.50
3	5.00	1.00	2.55	4.72	5.00
4	2.00	0.50	1.28	2.36	2.50
5	0.00	0.00	0.00	0.00	0.00

Table 2-2 HRT for Saint Paul Hospital (SPH) production cell

Physical Mode # 1					
RM	Emergency Room Output	Hospital Inputs			
	Rate of Treatment [People/hour]	Electricity [MW/hour]	Water [KL/hour]	Steam [m^3/hour]	Medical Gasses [%]
1	10.00	1.00	5.15	32.45	10.00
2	7.00	0.75	3.80	24.18	7.50
3	5.00	0.50	2.55	16.22	5.00
4	2.00	0.25	1.28	8.11	2.50
5	0.00	0.00	0.00	0.00	0.00

For a given amount of input resources, the outputs of the hospital production cells determine the patients per hour that can be treated. This determination is made by the utilization of the available resources within interdependent systems. These outputs are used to determine the maximum output rate for the waiting areas that are represented by storage cells. The patients arriving into these waiting area cells come from the modeled venues (GM Place and BC Place). As patients exit the waiting area cells, they encounter a

delay of 30 minutes, which is assumed to be the average time required to stabilize a trauma victim. Finally, patients that are input to the storage cells are used to count the total number of patients discharged from the hospitals during a given episode of the model. Additional columns may be added to tables 2-1 and 2-2 to account for human factors such as the fatigue level of the on-day doctors and nurses.

2.3.1.2 Electrical Power Substations (CSQ, DGR, SPG and MUR)

These four production cells represent the amount of power that is supplied by the external high-voltage system and delivered to the system under study. The amount of power delivered is determined based on the operating condition of the simulated substation. Normal power generation is expected if the state of modeled production cell does not suffer from physical damages or less incoming resources. The stations are geographically separated and each one supplies a specific amount of power to its interconnected infrastructures. For example, MUR is a 586 MW Power Substation that supplies power to five interconnected infrastructures: VGH-Hospital, Water Pumping Station and GM-Venue, DGR-Power Substation and residential load. What makes the MUR substation to appear critical is that, in addition to the amount of power that is delivered to the aforesaid infrastructures, it also delivers power to another power substation called DGR. DGR, 336 MW Power Substation, delivers the received amount of power to other CI's such as Saint Paul's Hospital.

The production cells are the main part of the electrical power substations model. They have human readable tables (HRT's) that represent the operating modes of the substation

transformers based on the inputs that come from the high-voltage feeders connected to these substations (Table 2-3 and Table 2-4).

The output power supplied by the power substations is distributed among the hospitals, the water pumping station and the venues. The available power distribution mode depends on the switching arrangement of the substations, the capacity of the outgoing feeders and the topology of the power distribution network. This table is provided by the power distribution company.

Table 2-3 HRT of Murrin (MUR) power substation production cell

Physical Mode # 1		
RM	Output	Input
	Electricity [MW]	Electricity [MW]
1	586.00	586.00
2	439.50	439.50
3	293.00	293.00
4	146.50	146.50
5	0.00	0.00

Table 2-4 HRT of Dal Grauer (DGR) power substation production cell

Physical Mode # 1		
RM	Output	Input
	Electricity [MW]	Electricity [MW]
1	336.00	336.00
2	210.00	210.00
3	105.00	105.00
4	84.00	84.00
5	0.00	0.00

The delivered power will be distributed as per the distribution table (Table 2-5 and Table 2-6). The "distributor" component in i2Sim is assumed to operate instantly after the decision of which distribution mode to use is made by the decision maker (Reinforcement Learning agent in this case). The distributed power is transmitted through the electrical "channels". Channels designated for power distribution are an abstraction of the real power mediums such as transmission lines and underground cables. No delay is incurred with electrical channels since electricity is carried at the speed of light.

Table 2-5 Power distribution table of Murrin (MUR) power substation production cell

Power Distribution Table					
Power Action #	**VGH**	**Water Station**	**GM**	**DGR**	**Terminated**
1	100%	0.00%	0.00%	0.00%	0.00%
2	1.12%	1.82%	2.1%	57.34%	37.62%
3	1.12%	1.28%	2.1%	57.34%	38.16%
4	0.00%	100%	0.00%	0.00%	0.00%
5	0.84%	1.82%	2.1%	57.34%	37.80%
6	1.12%	1.82%	1.58%	43.00%	52.48%
7	0.00%	0.00%	0.00%	100%	0.00%
8	0.56%	1.28%	1.58%	43.00%	53.58%
9	0.28%	0.85%	1.58%	28.67%	68.62%
10	0.00%	0.00%	0.00%	0.00%	100%

Table 2-6 Power distribution table of Dal Grauer (DGR) power substation production cell

Power Distribution Table			
Power Action #	**BC**	**SPH**	**Residential**
1	2.98%	2.98%	94.05%
2	100%	0.00%	0.00%
3	0.00%	2.98%	97.02%
4	0.00%	100%	0.00%
5	2.98%	0.00%	97.02%

2.3.1.3 Water Pumping Station

The water pumping station provides water to the hospitals (VGH and SPH). It receives power from the MUR power substation and water from an external source. Eventually, the output of this cell is high pressure pumped water that goes to a water distributor, which distributes the water through the water channels (water pipes) to both hospitals. The state of the modeled system determines how much water will be distributed among the EU's. The operation modes, Physical Modes (PM's) and Resource Modes (RM's), of the water station cell is determined in the human readable table (HRT), Table 2-7, and the distributor ratios is determined in Table 2-8.

The Reinforcement Learning (RL) agent is required to sense the state of the modeled system (including the state of the power and water pumping station cells) and apply the adequate distribution ratios of the available resources (including power and water).

Table 2-7 HRT of the water pumping station

Physical Mode # 1			
RM	Output	Inputs	
	Water [KL/hour]	Electricity [MW/hour]	Water [KL/hour]
1	103.00	0.01	103.00
2	77.00	0.008	77.00
3	52.00	0.005	52.00
4	26.00	0.003	26.00
5	0.00	0.00	0.00

Table 2-8 Water distribution table of the Water Pumping Station production cell

Water Distribution Table		
Water Action #	**VGH**	**SPH**
1	100%	0%
2	90%	10%
3	80%	20%
4	70%	30%
5	60%	40%
6	50%	50%
7	40%	60%
8	30%	70%
9	20%	80%
10	10%	90%
11	0%	100%

2.3.1.4 Venues (GM and BC)

These components represent buildings or facilities that contain large crowds of people. It is assumed that both venues are busy hosting two different events and are fully occupied. BC Place has over 65,000 m^2 and is the main venue hosting up to 60,000 people. GM Place is slightly smaller at over 44,000 m^2 and is the secondary venue capable of hosting up to 20,000 people. 480 injured people are assumed as casualties from the two venues. The majority of the casualties (336 injuries) will come from the BC-Venue and the rest (144 injuries) will come from the GM-Venue.

Each venue model consists of one production cell and one distributor. The inputs to these production cells are water and electricity. Both inputs and outputs are shown in the HRTs of these cells (Table 2-9 and Table 2-10). The outputs of these cells are the number of people exiting the venues. The better the allocation of the available resources is, the higher

the evacuation rate will be. This is something that the modeled intelligent agent has to learn or realize within the simulated environment. This means that the decisions made by the RL agent have to consider the need of other interconnected CI's.

People exiting the venues are classified based on their physical condition. The role of these distributors is to determine whether the exiting person is injured or is healthy.

Table 2-9 HRT of the GM-Venue

Physical Mode # 1			
RM	Output	Inputs	
	[KPeople/hour]	Electricity [MW/hour]	Water [KL/hour]
1	936.00	10.0110	28.00
2	702.00	7.5080	21.00
3	468.00	5.0005	14.00
4	234.00	2.5030	7.00
5	0.00	0.0000	0.00

Table 2-10 HRT of the BC-Venue

Physical Mode # 1			
RM	Output	Inputs	
	[KPeople/hour]	Electricity [MW/hour]	Water [KL/hour]
1	1,166.00	10.0000	41.00
2	875.00	7.5000	30.75
3	583.00	5.0000	20.50
4	292.00	2.5000	10.25
5	0.00	0.0000	0.00

The channels are modeled to represent the route to the hospitals (by road). The emergency vehicles use these channels to transport the injured people from the venues to the assigned

hospitals. A simple transportation model was considered in this work, consisting of a delay (channel latency) to represent the time the vehicles take to arrive at their final destination. The purpose of the learning agent will be to evaluate the surrounding environment (state) and find the best settings of the available distributors (action). These settings control the electricity and the water supply to the interconnected cells. The objective of doing so is to optimize (maximize) the number of discharged patients from the emergency units at both hospitals (VGH and SPH) within the simulation time.

CHAPTER 3: MACHINE LEARNING

3.1 Machine Learning: What and Why?

Machine Learning is a branch of Artificial Intelligence (AI) defined by Arthur Samuel in 1959 as a "Field of study that gives the ability to learn without being explicitly programmed" [30]. In 2012, Kevin Murphy defined Machine Learning comprehensively "as a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty" [31].

Machine Learning generally refers to any algorithm or algorithms that enable the analysis of data with the aim of discovering any underlying statistical regularities. Such algorithms can be used to model and generalize the behavior of a system from a set of data.

3.2 Taxonomy of Machine Learning

Machine Learning (ML) is divided into three main types, as seen in Figure 3-1: Supervised Learning, Unsupervised Learning and Reinforcement Learning [31]. The three types are presented as follow:

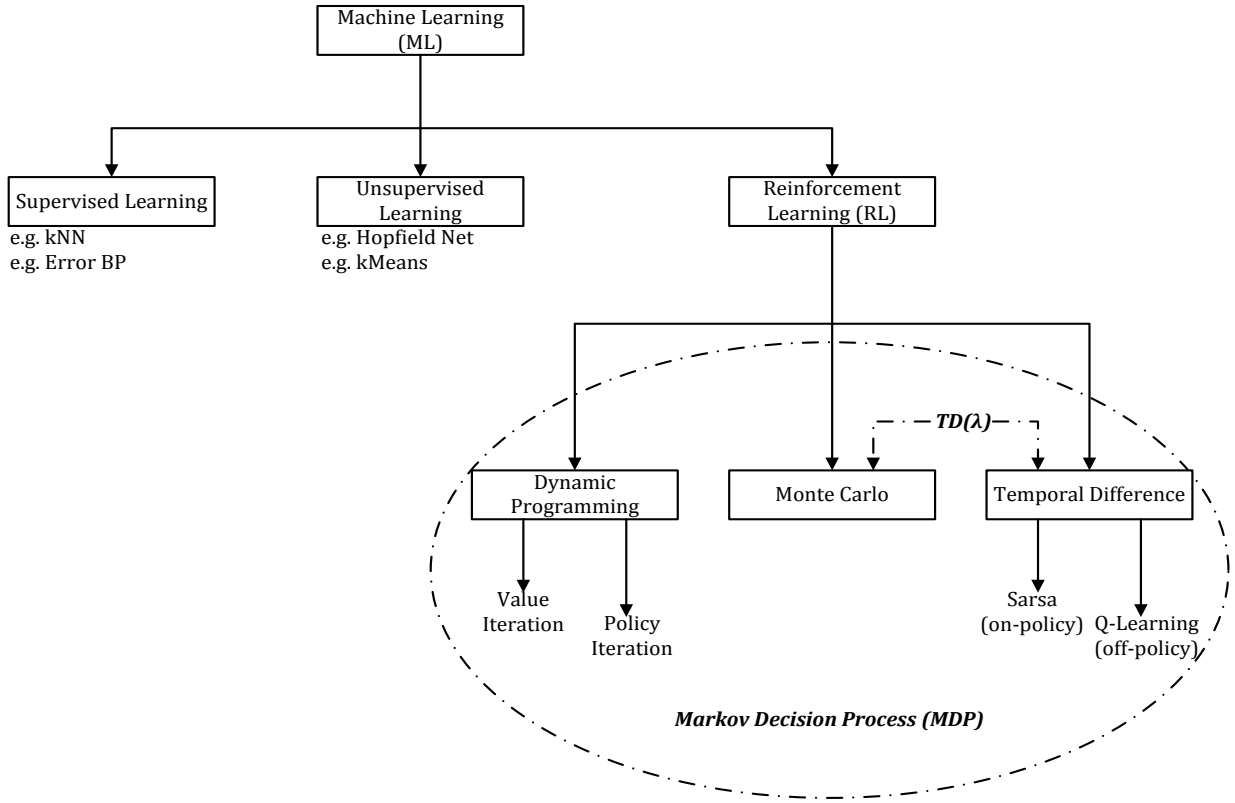

Figure 3-1 Taxonomy of Machine Learning (ML) [32]

3.2.1 Supervised Learning

Supervised Learning is a commonly used approach in Machine Learning (ML). The goal of this approach is to learn the mapping of inputs to outputs given a labeled set of input-output pairs.

$$D = \{(x_i, y_i)\}_{i=1}^{N} \quad (3\text{-}1)$$

Where:

x: Input data

y: Output data

D: Training set

N: Number of training examples

Such approach is adequate for solving classification problems (such as binary classification) where the response variables are discrete. Also, problems that are classified as regression problems, where the response variables are continuous, can be solved using this approach.

3.2.2 Unsupervised Learning

In Unsupervised Learning, the goal is to discover groupings or patterns inherent in the sampled data. These groupings can then later be used to solve classification tasks. Many such algorithms exist, each opting to identify clusters according to different metrics.

$$D = \{(x_i)\}_{i=1}^{N} \quad (3\text{-}2)$$

Where:

x: Input data

D: Training set

N: Number of training examples

3.2.3 Reinforcement Learning

Reinforcement Learning (RL) is neither supervised nor unsupervised. Its nature makes it a suitable approach for learning to associate an action to perform in a given state. It is different from Supervised Learning in that it does not attempt to learn a mapping of inputs to outputs. However, unlike Unsupervised Learning, there is feedback. RL is a mechanism that attempts to solve the assignment problem by sampling the environment. An RL agent will observe its surrounding environment and based upon that state, perform an action. In return, a feedback signal is produced and credit is assigned. The goal of RL is to learn those actions that maximize this feedback signal.

3.3 Reinforcement Learning

RL [25] is a machine-learning technique based on interactions between an agent and its surrounding environment (Figure 3-2). These interactions help the RL agent to maximize a (time delayed) goal in the presence of uncertainties.

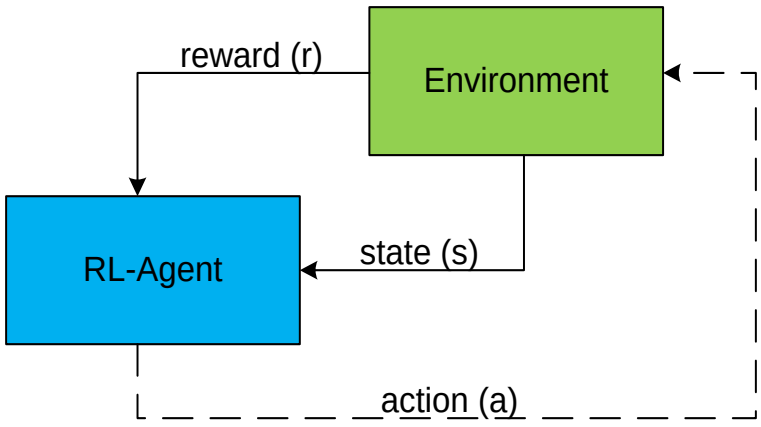

Figure 3-2 Structure of the Reinforcement Learning (RL)

The principle underlying RL is that the agent learns through accumulation of experience. The experience develops based on a feedback signal that the RL agent receives. This signal defines a performed action at a given state based on a return (reward). Thus, the goal of the RL agent is to learn which actions yield the largest long-term reward.

The first step in training an agent is to identify the target. Then the elements of the RL participating in reaching the target need to be identified. These elements are:

- **Policy:** defines the agent's behavior within its environment, i.e., how the agent determines which action should be selected at any given state.

- **Reward function:** defines the feedback signal (return) that is applied to the actions that have been performed by the agent at any given state. Rewards could be immediate or terminal and are observable at any time during the lifetime of the scenario.

- **Action-Value function (Q-function):** defines the predicted long-term reward that can be expected upon taking the chosen action at a given state. The $action-value\ function$ can be described in terms of the Bellman equation (a recursive definition that describes the current predicted reward in terms of a future predicted reward) [25].

$$Q^{\pi}(s,a) = E_{\pi}\{R_t|\, s_t = s, a_t = a\} = E_{\pi}\{r_{t+1} + \gamma V^{\pi}(s_{t+1})|\, s_t = s, a_t = a\} \qquad (3\text{-}3)$$

The goal of our learning agent will be to approximate an optimal $action - value\ function$ that leads to the best long-term reward, or in other words the actions that in our case best mitigate casualties. In general there are three ways of determining this: Dynamic Programming (DP), Monte Carlo (MC) or Temporal Difference (TD). Brief explanations about each one are given in the coming section.

In RL, the ability to optimize an $action - value\ function$ depends on the capability of the intelligent agent to discover its optimum policy. This policy represents the state transitions of the modeled environment and the expected associated returns. Only then will the RL agent be able to optimize the system outcome.

The state transitions in RL are typically modeled as a finite Markov Decision Process (MDP). This implies that the modeled problem must adhere to the following property [33]:

$$Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\} =$$

$$Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t \ldots . s_0, a_t \ldots . a_0, r_t \ldots . r_1\} \quad (3\text{-}4)$$

The above property states that the probability of transitioning from state s to state s' given action a does not depend on the history of state transitions prior to the current state s. This is an essential requirement for the application of any RL method. This is because, in many cases, the only information available for the agent is at the current state, which means that the RL agent is able to capture knowledge without prior history.

In a disaster management framework, for example, deciding how much of the limited power to route to a hospital or to a venue or to a water station, can be determined solely on knowledge of the current state. It is not necessary to know the trajectory of states that gave rise to the current situation.

3.4 Reinforcement Learning Policy Estimation

Reinforcement Learning (RL) is a learning system that is used to map situations to actions so as to maximize a numerical reward signal. For RL, a commonly used policy is $epsilon - greedy$. Here, action selections are often performed in a manner such that the agent will, most of the time, select that state or action that leads to the highest predicted long term reward. The remainder of the time, actions are selected *randomly*. The purpose of RL, thus, is to learn those values for the $action - value\ function$ that closely approximate the actual values for the long-term reward of a chosen system. The $action - value\ function$ can be implemented as a look up table (LUT) and in fact RL approach is proven to converge when implemented in this way [25].

In RL, the learning operation estimates the $action - value\ function$ of the policy being followed, for a Markovian or partially Markovian environment. This estimation is performed differently as per a chosen mechanism. These RL mechanisms are listed and defined as below [25]:

3.4.1 Dynamic Programming (DP)

Dynamic Programming (DP) refers to algorithms, well developed mathematically, to be used to obtain the optimal policy given a complete model of the environment as a Markov Decision Process (MDP). In DP, the $action - value\ function$ is used to organize and structure the search for an optimal policy. The optimal policy could be achieved once the optimal $action - value\ function$ is determined.

DP methods iteratively perform a full backup operation on each state. In order to do so, knowledge of the transition probabilities (also known as the model) associated with the environment is required.

$$Q^*(s,a) = E\{r_{t+1} + \gamma\, max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a\} =$$
$$\sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma\, max_{a'} Q^*(s', a')] \quad (3\text{-}5)$$

In (3-5) above, $P^a_{ss'}$ is the probability of transitioning to state s' after taking action a in state s. However, it is often the case that such transition probabilities of a system are not known $a - priori$.

3.4.2 Monte Carlo (MC)

The Monte Carlo (MC) method is a way of solving the RL problem based on averaging sampled returns. The approach applies specifically to problems that are episodic in nature. For MC, the estimation process as opposed to DP, does not require knowledge of transition

probabilities and all learning updates ($back - ups$) are performed upon completion of each episode.

In MC, the goal is to approximate the optimal $action - value\ function$ that leads to the best long-term reward that corresponds to the best trajectory. This trajectory identifies actions that best defines the long-term reward for a given policy. It is a recursive-learning algorithm that uses episode-by-episode back-ups to solve the Bellman equation.

3.4.3 Temporal Difference (TD)

Temporal Difference (TD) learning is a combination of DP and MC. Like DP, the state estimate in TD happens based on other state estimates, which makes it suitable for problems that are fully incremental by nature. Like MC, the learning process could be accomplished without a need for a complete model, only a subset of states of the modeled environment is enough to do so.

In contrast to MC, the recursive-learning algorithm of TD uses incremental step-by-step back-ups to solve the Bellman equation. This helps to determine the long-term reward based on the approximation of the optimal $action - value\ function$ that is correlated to the optimum path for a given policy.

In this study, an attempt to use both TD and MC has been made (Chapter 4 and Chapter 5). Although both RL policy estimators are capable of discovering the optimum policy that maximizes a desired outcome, in this case the total number of discharged patients, faster

convergence is expected in the case of MC over TD within our proposed platform. Figure 3-3 shows the differences between how back-ups (a step that updates a previously learned estimate) are performed in TD versus MC.

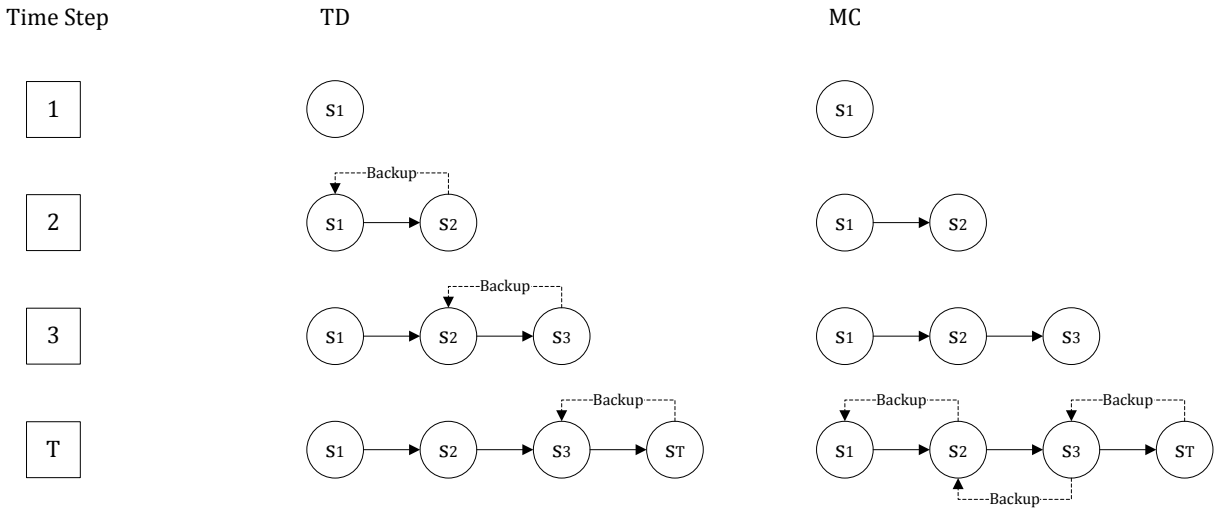

Figure 3-3 Backup process in TD and MC

3.5 Reinforcement Learning Successful Applications

The following describes some notable applications of Reinforcement Learning (RL). One of the most famous is the board game Backgammon. Gerald Tesauro applied a Temporal Difference (TD) learning mechanism to this board game [34]. The difficulty with Backgammon is the huge number of states that have to be considered, which is about 10^{20}. The challenge here is dealing with such a large state space. Clearly the use of a look up table is intractable and Tesauro successfully used a neural network to approximate the look up

table. The model was so successful that the machine was able to beat the world's best Backgammon players.

In another example, [35] presented a residential demand response application to reduce power energy cost and smooth its usage. Here, the learning model was able to capture the dynamic behavior of the consumer's preferences and the probabilistic structure of energy prices. Their system outcome not only demonstrated a significant reduction (up to 40%) on the average financial cost of the end-user but also the ability of avoiding pricing peaks and smoothing energy expenses.

Similarly, [36] demonstrated a distributed multi-step learning (DQ(λ)L) algorithm for solving a large-scale multi objective optimal power flow problem. The key solution for such a problem is to breakdown the complex system into subsystems where each subsystem can be solved independently. This is why Individual Reinforcement Learning (IRL) is proposed in which every agent in each subsystem executes a learning task individually and coordinates with other agents to achieve the global goal. With this backtrack updating rule model, fast convergence and precision are guaranteed compared to other model such as Distributed Learning (DQL). For verification, a simplified system of an electrical power network (9 *buses*) was suggested. The results indicated that DQ(λ)L converged faster than DQL model with a closer estimation of the outcomes. These results encouraged [36] to apply their proposed model to a large-scale power network (118 *buses*) and evaluate the results. Here, the comparison revealed that not only DQ(λ)L performed better than DQL in

terms of both objective value and average objective value but also it took less number of learning steps to get there, which means faster learning.

3.6 Reinforcement Learning Disaster Mitigation Applications

Disaster mitigation problems using Reinforcement Learning (RL) have received broad attention. Examples are presented below:

In [37], a framework of an intelligent system to allow decision makers to mitigate consequences of any natural or human made disaster (e.g. forest fires) is presented. This type of disaster is complex in nature because it involves many interacting sub-processes, making cost estimation too difficult for human experts. Using RL, the chosen behavior tells the agents what actions to choose at any visited states as per the optimum policy. An incremental approach is developed for a fire spread model to overcome the complexity of the modeled system. This addressed the limitations of the available hardware while maintaining a realistic model.

Similarly, [38] suggested an approach to find the optimal policy of a fire-extinguishing task using Temporal Difference (TD). The ability of TD in estimating the value of every visited state, which in turn is based on the estimation of other states, makes it suitable for such an application. The learning approach was implemented in two stages. In the first stage, the agent was asked to interact with a simplified rescue simulation model in an incremental way (lesson-by-lesson) for learning enhancement. In the second stage, the experienced agent was asked to interact with an official rescue simulation model to evaluate the ability

of achieving the optimum goal in a timely manner. Here, the trained agent was applied to the 2010 worldwide rescue simulation in the Singapore 2010 competition. The results revealed that the agent was able to discover a better rescue strategy, which scored significantly higher than the other three top ranked teams in the championship.

In another example, [20] proposed an agent-based model for patients in emergency circumstances. The goal of the study was to offer an accurate diagnosis and treatment in appropriate time for high-risk patients during emergencies. The algorithm was implemented using a RL agent model with an embedded Dynamic Programming learning mechanism to evaluate and improve the system value function and its policy. The process begins by sending an informative signal that contains the patient profile from a device that is attached to the patient body. This signal is then sent to the Hospital Information Server (HIS), where access to a database is granted to the agent. The acquired information (accumulated knowledge) is used to build a diagnosis history. The agent was able to successfully provide the physician with the right diagnosis and the proper treatment with a minimal time cost.

In 2006, [39] proposed an evolutionary RL model for search and rescue tasks. The goal of the study is to support the decision making of a central agent on a complex victim rescue problem. This problem was formulated within a RoboCup Rescue Simulation League (RCRSL) environment that is known by its complexity and uncertainty. To interact with such environment, a central agent using a learning mechanism called XCS (a version of Holland's Genetic Classifiers Systems) was modeled to deploy a required number of

ambulances for this critical task. This type of learning mechanism can offer a satisfactory solution in large state-space systems at low computational cost. This is due to its ability to search for an optimal policy directly over the space of the policies. To observe the performance of the agent, a rescue task of trained rules and random rules were compared using three maps from different cities. The outcomes revealed that the learning agent and its team were able to exhibit better performance than the other competed teams within the RCRSL environment.

The last example [40] proposed a path selection algorithm for disaster response management. This work was suggested for search and rescue activities in a dynamic dangerous environment. The proposed algorithm based on RL helps disaster responders to discover the fastest and the shortest path to the targeted locations. To do so, a learning agent interacts with a model of a two-dimensional geographic grid environment. These interactions are classified as four actions to choose at any visited states with respect to the status of this state. For example, the agent has to learn after a number of trials, how to avoid arriving at dangerous states and also to get around the inaccessible states. As expected, the experienced agent was able to find the optimal (or an approximately optimal) path in a finite number of loops.

3.7 Response Crisis Management Framework

3.7.1 Application of Reinforcement Learning in i2Sim

The prior section shows that the application of machine learning, specifically Reinforcement Learning (RL) in the context of response crisis management has potential to expedite relief efforts through improved decision making. We believe that there exists a significant opportunity to extend and enhance previous work in the direction of assisting decision makers following a disaster. Our initiative proposes the novel application of RL within the context of problems involving interconnected and interdependent Critical Infrastructures (CI's). The generality of RL permits its application to a wide variety of problem areas. The previous sections identify the main aspects of RL and the parameters available in i2Sim (infrastructure interdependency simulator). The application of RL to i2Sim is a matter of formulating those parameters in terms of states, actions and rewards.

We formulate the RL problem in i2Sim as follows: in i2Sim the operating modes (Physical Modes-PM's and Resource Modes-RM's) represent the states of each modeled infrastructure unit and the distributors represent the actions that the agent can perform. The reward is a function that is calculated based on the expected number of discharged patients from the Emergency Units (EU's) (Figure 1-2). A tabular form of the action-value function is implemented.

3.7.2 Q-Learning

$Q-Learning$ is an off-policy variant of the back-up mechanism used in Temporal Difference (TD) and Monte Carlo (MC). The $Q-Function$ is a utility function that estimates the probability of obtaining a long-term reward upon choosing an $action$ at a given $state$. Here the goal of our learning agent is to approximate an optimal $action-value\ function$, which maximizes a predicted reward, or in other words the actions that in our case best mitigate casualties. We choose to apply Q-learning and explore two of the RL algorithms, TD and MC learning mechanisms. These recursive algorithms, as stated earlier, use step-by-step and episode-by-episode backing-up respectively, to solve the Bellman equation.

The step-by-step backing-up, known as TD, is defined as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma\, max_a\ Q(s',a') - Q(s,a)] \tag{3-6}$$

Where:

$Q(s,a)$: Action-Value Function of current state, action pair

$Q(s',a')$: Action-Value Function of the next state, action pair

r: Reward associated with next state, action pair

α: Learning rate, determines to what extent the newly required information will override the old information. A value of 0 will result in an agent in which no learning takes place.

γ: Discount rate, determines how much influence future rewards have on the learning process. For example, in a game of Chess it might be preferable to win the overall game, rather than to avoid losing a piece in the current move.

Whereas the episode-by-episode backing up, known by MC, is defined as follows:

$Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma\, R_T - Q(s,a)]$ at the terminal state (3-7)

$Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma\, max_a\ Q(s',a') - Q(s,a)]$ at a non-terminal state (3-8)

Where

Q(s, a): Action-Value Function of current state-action pair

R_T: Total return at the terminal state

r: Immediate reward associated with next state-action pair

α: Learning rate, determines to what extent the newly required information will override the old information.

γ: Discount rate, determines how much influence future rewards have on the learning process.

3.7.3 Platform Interface

In realizing our goal of implementing a learning agent for i2Sim, one of the challenges was for this agent to be able to communicate with i2Sim. We chose to implement the learning agent in JAVA, whereas i2Sim is built using MATLAB. This imposes a technical requirement whereby the JAVA virtual machine must communicate with MATLAB. The information to be exchanged consists of states, rewards and actions. This led us to develop a MATLAB-JAVA interface, where a fluent connection between the two languages could be established at any time based on the learning mechanism that is chosen as shown in Figure 3-4. We use this platform to explore different RL agents implemented according to both TD and MC paradigms.

The purpose of the interface is to relay information between the two environments either at each time step, as in TD, or at the end of each episode, as in MC.

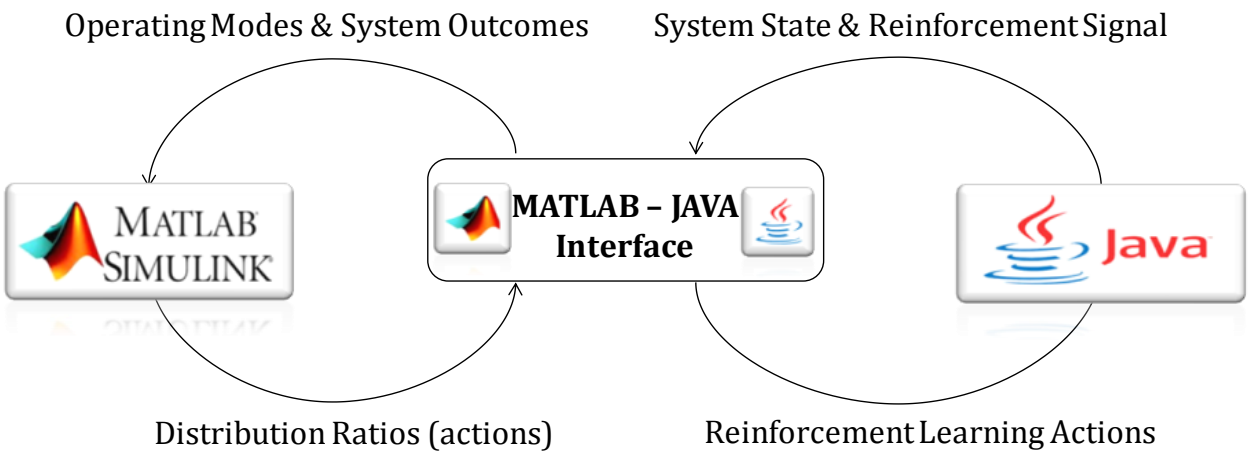

Figure 3-4 MATLAB-JAVA connection diagram

The RL agent needs to sense the environment and, if available, any reward in i2Sim. In turn, i2Sim needs to accept an action from the agent. The interface is based upon a proxy class within JAVA that synchronizes the communication. The implementation of this interface has been described in the coming chapters (4 and 5). In these chapters, the proposed RL agents, DAARTS using TD and IDS using MC, and their platforms, are presented and explained.

CHAPTER 4: RESOURCE ALLOCATION OPTIMIZATION USING TEMPORAL DIFFERENCE POLICY ESTIMATION

In this chapter the resource allocation optimization problem is addressed and solved using the Temporal Difference (RL-TD) learning mechanism. This learning mechanism is used to optimize an $action - value\ function$ in a step-by-step fashion. This proposed system is expected to maximize the total number of discharged patients of the simulated environment.

4.1 Introduction to Decision Assistance Agent in Real-Time Simulation (DAARTS)

In this thesis, DAARTS is used to refer to the learning agent implemented using RL-TD. DAARTS will sense the i2Sim (infrastructure interdependency simulator) environment and attempt to learn those actions that minimize physical damage and human casualties. A detailed explanation of the DAARTS-i2Sim system is presented below.

4.1.1 Application of RL-TD in i2Sim

Resource reallocation optimization in conjunction with an infrastructure interdependency system is a complex problem that can be solved using RL-TD. To achieve this, it is necessary to understand the schematic of the problem and to define the learning parameters needed

by the Reinforcement Learning (RL) agent. It is expected that DAARTS (the RL agent) will be able to optimize the system outcome (number of discharged patients) of the modeled system. Since Temporal Difference (TD) learns via incremental back-ups, DAARTS needs to communicate with i2sim at every time-step of the simulation (Figure 4-1). This communication is done through the platform interface as described in the previous chapter.

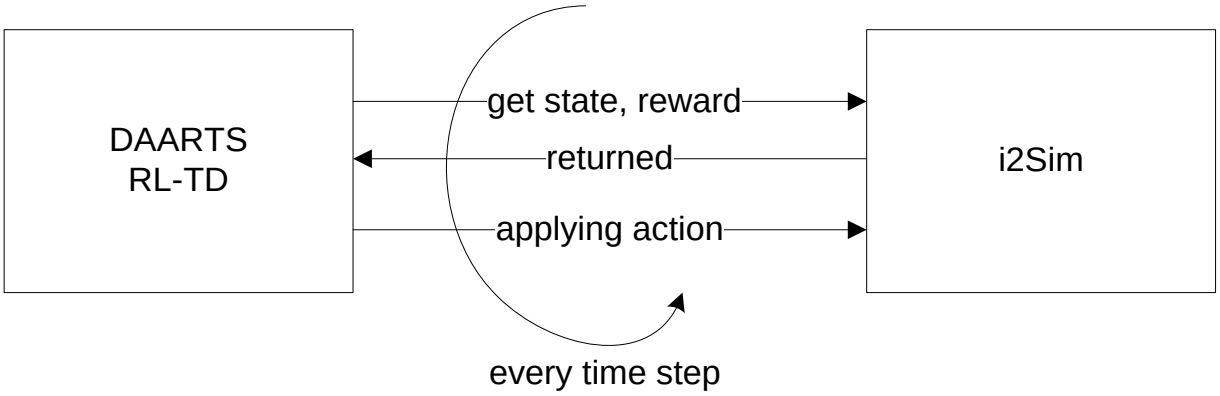

Figure 4-1 System interactions in DAARTS

The initial investigation was based on an Intel(R), Core(TM) i5 CPU, 2.8 GHz based computer with 8 GB RAM. The communication mechanism used by DAARTS to interact with i2Sim suggested that we may be faced with issues regarding learning speed. For example, if each communication step incurred a latency of t seconds, representing a simulated time increment of m minutes, then an $n - hour$ simulation would consume $(60 \text{ x } n \text{ x } t)/m$. For example if $t = 0.1\ seconds$, $m = 5\ minutes$, $n = 10\ hours$, then the total communication overhead $=$ $12\ seconds$. If we simulate 100 of these scenarios, then

the communication overhead is in the order of 20 *minutes*. In this example at least, this means that the ability of DAARTS to optimize a system outcome cannot be met in real time.

4.1.2 System Architecture Description

According to RL, DAARTS (the learning agent) needs to know the states representing the surrounding environment. These states are introduced by what is called in i2Sim, PM and RM. PM (Physical Mode) determines the physical operability of the modeled critical infrastructures (CI's), while RM (Resource Mode) determines the resources availability of theses interconnected CI's. The modeled CI's PM's and RM's relations are defined as below (Table 4-1). In this table, the percentages represent the availability of the input resources (RM) and the severity of the physical damages (PM) for every modeled CI. For example, in $RM_1 = 100\%$ the facility is able to operate at full capacity since its demand is fully satisfied and no physical damages have been observed ($PM_1 = 100\%$), whereas in the case of $RM_5 = 0\%$ two explanations are possible. Either, no enough inputs have been delivered to the facility or the facility is totally collapsed ($PM_5 = 0\%$).

Table 4-1 i2Sim PMs and RMs relations

	RM_1 (100%)	**RM_2 (75%)**	**RM_3 (50%)**	**RM_4 (25%)**	**RM_5 (0%)**
PM_1 (100%)	X	X	X	X	X
PM_2 (75%)	-	X	X	X	X
PM_3 (50%)	-	-	X	X	X
PM_4 (25%)	-	-	-	X	X
PM_5 (0%)	-	-	-	-	X

Here, the state of the simulated system is expected to change based on the action chosen by the RL agent. This state and action are then paired together and recorded in a Look Up Table (LUT). This can be presented and derived mathematically as follow:

State **(*s*):** Represents the total number of states of the modeled system.

Action **(*a*):** Represents the total number of combined actions of the two distributors (Power distributor and Water Distributor) as seen in Figure 4-2. The MUR power distributor has 10 different distribution ratios and the water distributor has 11 different distribution ratios.

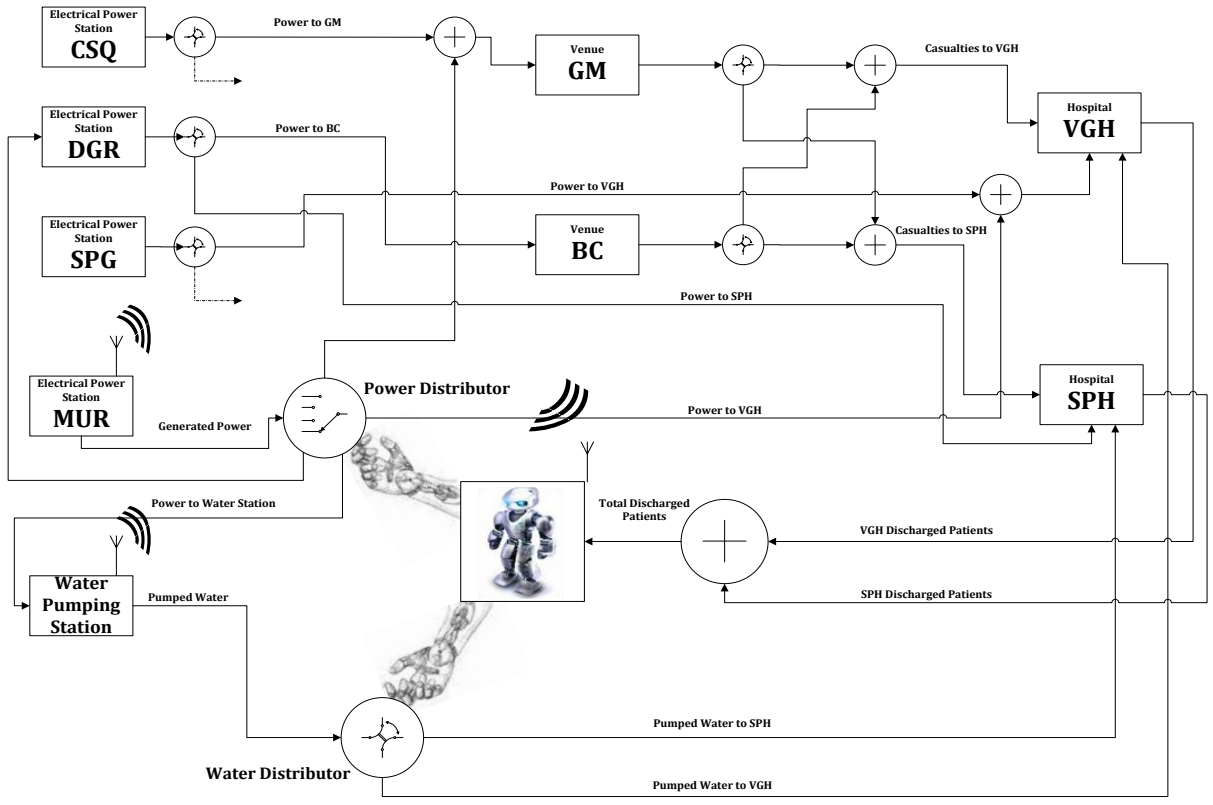

Figure 4-2 DAARTS within i2Sim Downtown Vancouver Model

The model has the following number of states:

$$num_{\ states} = x^{y} = 15^{2} = 225\ states \quad (4\text{-}1)$$

Where:

x: Represents the maximum possible number of resource modes associated with each physical mode preproduction cell.

y: Represents the number of production cells that are exposed to DAARTS.

$$num_{\ actions} = Dis._{\ MUR} \text{ x } Dis._{\ Water} = 10 \text{ x } 11 = 110\ actions \quad (4\text{-}2)$$

Where:

$Dis._{\ MUR}$ = Number of distribution ratios of the MUR Power Substation distributor.

$Dis._{\ Water}$ = Number of distribution ratios of the Water Pumping Station distributor.

The size of the LUT can be estimated as below:

$$num_{\ rows} = num_{\ states} \text{ x } num_{\ actions} = 225 \text{ x } 110 = 24{,}750\ elements \quad (4\text{-}3)$$

The states and actions, as described above, suggest a theoretical maximum size for the LUT of *24,750 elements*, which records the updated experience of the RL agent. Each row of the table represents a $state - action$ pair and the associated long term predicted reward or state-action value for taking that action when in that state.

Below is a sample of the LUT being used for this model:

Table 4-2 DAARTS's LUT (Last Episode)

#	$(< State >, < Action >)$ $(< \mathbf{PM_{MUR}}, \mathbf{RM_{MUR}}, \mathbf{PM_{W}}, \mathbf{RM_{W}} >, < \mathbf{Dis._{MUR}}, \mathbf{Dis._{W}} >)$	**Q(s, a)**
1	(<1, 1, 1, 1>, <1, 1>)	69.67
2	(<1, 1, 1, 1>, <1, 2>)	0.610
3	(<1, 1, 1, 1>, <1, 3>)	-0.020
4	(<1, 1, 1, 1>, <1, 4>)	0.389
5	(<1, 1, 1, 1>, <1, 5>)	0.192
.	.	.
.	.	.
24,749	(<5, 5, 5, 5>, <10, 10>)	0.419
24,750	(<5, 5, 5, 5>, <10, 11>)	0.128

Where:

PM_{MUR}: Represents the MUR electrical power station physical mode (state)

RM_{MUR}: Represents the MUR electrical power station resource mode (state)

PM_{Water}: Represents the Water Pumping station physical mode (state)

RM_{Water}: Represents the Water Pumping station resource mode (state)

$Dis._{MUR}$: Represents the MUR electrical power station distributor (action)

$Dis._{Water}$: Represents the Water Pumping station distributor (action)

The purpose of DAARTS will be to find the best settings of the two distributors appropriate for each possible state of the system (based on the gained knowledge and experience). These distributors control the electricity and the water supplies to all interconnected cells and directly influence the resources available to a hospital's emergency facility and by consequence, the ability of an emergency unit to discharge injured patients.

4.2 Training DAARTS

We ran 100 *episodes* per test, where each episode represents a scenario that simulated a $10 - hour$ disaster period following a disaster event. It is worthwhile to point out that the same scenario is conducted for all episodes, however each episode presents different decisions (actions) taken by the Reinforcement Learning (RL) agent regarding the allocation of the available resources. Each episode took about 6 $minutes$ of computer time to complete. Upon starting the scenario, the Physical Modes (PM's) representing damage to cells and channels were set to model a disaster. Following that, no further changes were made in the state of damage. However, the Resource Modes (RM's) of the associated infrastructures' cells change as the scenario evolves. The LUT was initialized randomly at the start of the first scenario and learning continued from one episode to the next.

4.2.1 Scenario Explanation

An earthquake disaster scenario was configured (Figure 4-2). The PM of MUR Electrical Power Substation was not affected, however the RM was reduced due to failure in one of its electrical feeders. Subsequently, due to reduced electrical power, the water facility was not

able to operate at full capacity and the amount of power delivered to the DGR Electrical Power Substation was also affected.

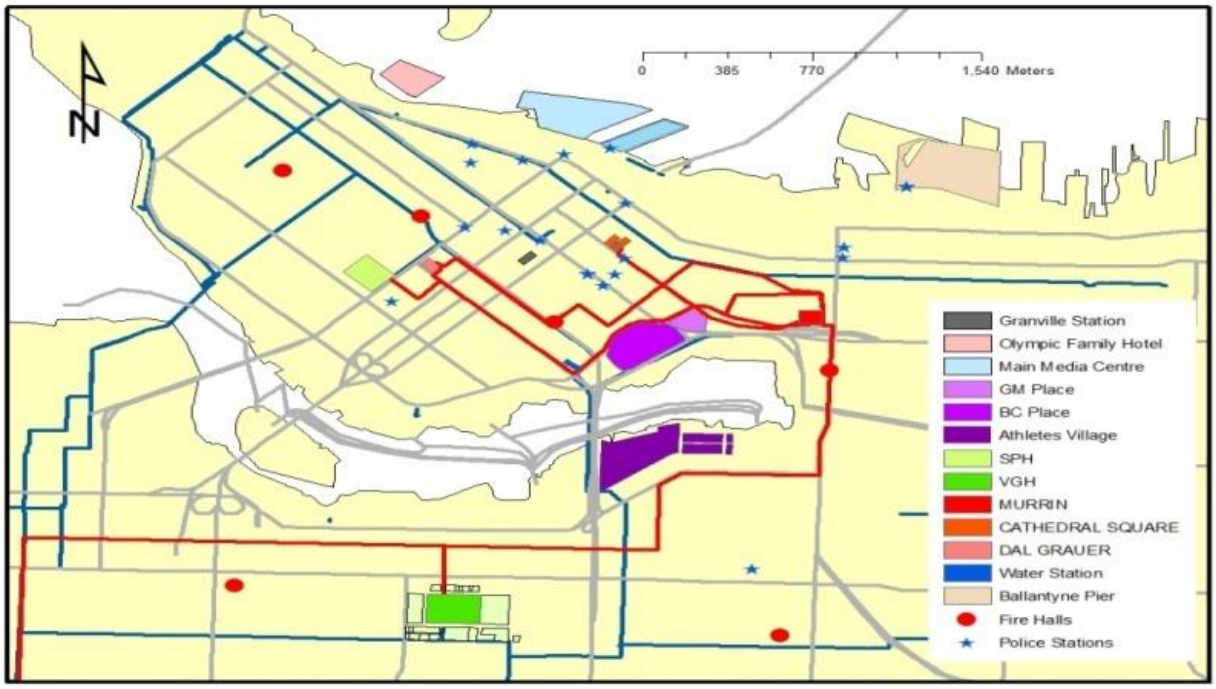

Figure 4-3 Downtown Vancouver geographical area [41]

The earthquake is presumed to lead to casualties due to panic and chaos as the crowd trying to escape from the GM and BC venues (Figure 4-2). It was assumed that medical triage at the venue took an average of 30 *minutes* per injured person. Upon completion of the assessment, emergency vehicles carried the injured people to either the VGH or SPH hospitals for treatment. Travel time was assumed to be 10 *minutes* simulation time (Figure 4-3). The severity of the injuries dictated which victims went to which hospital. A schematic of the modeled scenario is shown in Figure 4-4. The details of the modeled components were described in section 2.3 (chapter 2).

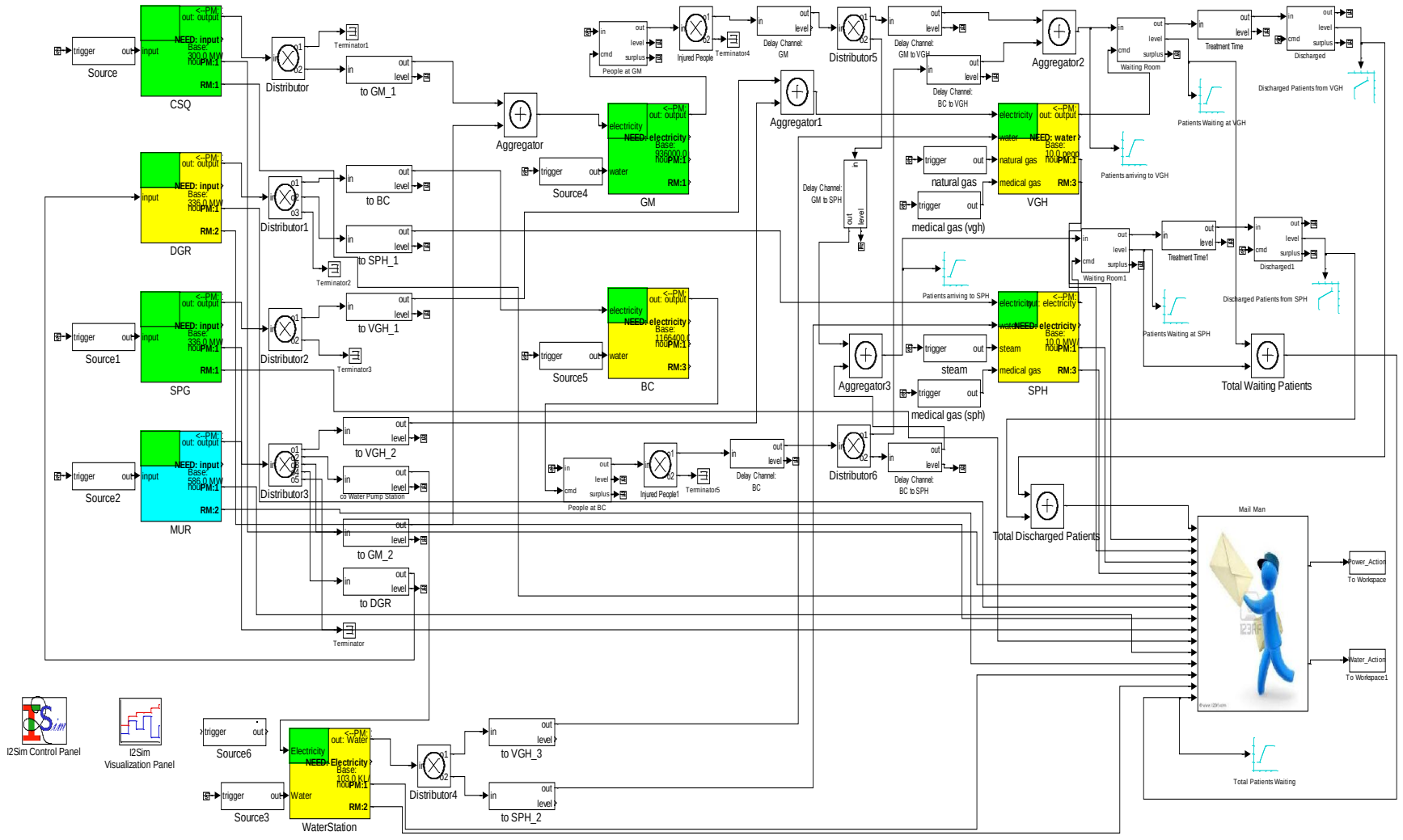

Figure 4-4 i2Sim Downtown Vancouver Model

The model simulated a period of 10 *hours* in five-minute time steps. The data used for i2Sim were taken from an internal report written by the Complex System Integration Group (CSI) at the University of British Columbia (UBC) [42]. The report helps to determine both the evacuation rates at the modeled venues, and the hospital discharge rates for both hospitals, used in this simulation.

4.2.2 RL-TD Algorithm

DAARTS implements the tabular form of the Q-Learning algorithm in which the state of the environment, as modeled by the Look Up Table (LUT), is used to determine the next action to take. Here the action is deciding how to set the two distributors (MUR-Power Distributor and Water Distributor) as per table 2-5 and 2-8. Given an initially untrained LUT, the goal of RL is to find the optimal action to perform in each state. If available, real world experience could possibly be used to initialize the LUT as a starting estimate of the optimum schedule.

$$a = \begin{bmatrix} PowerDistributor_{VGH} \\ PowerDistributor_{Water} \\ PowerDistributor_{GM} \\ PowerDistributor_{DGR} \\ PowerDistributor_{Others} \\ WaterDistributor_{VGH} \\ WaterDistributor_{SPH} \end{bmatrix} \text{X} \begin{bmatrix} p_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & p_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & p_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & p_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & p_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & w_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & w_2 \end{bmatrix} \quad (4\text{-}4)$$

The state of the environment is a vector representing the PM's and RM's of the production cells at any given time.

$$s = \begin{bmatrix} PM_1\ RM_{1_{MUR}} & PM\ _1RM_{1_{Water}} & a_1 \\ PM_1\ RM_{1_{MUR}} & PM\ _1RM_{1_{Water}} & a_2 \\ \vdots & \vdots & \vdots \\ PM_1\ RM_{1_{MUR}} & PM\ _1RM_{1_{Water}} & a_{10} \\ PM_1\ RM_{1_{MUR}} & PM\ _1RM_{1_{Water}} & a_{11} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ PM_1\ RM_{1_{MUR}} & PM\ _1RM_{1_{Water}} & a_{110} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ PM_5\ RM_{5_{MUR}} & PM\ _5RM_{5_{Water}} & a_{110} \end{bmatrix} \quad (4\text{-}5)$$

According to RL, DAARTS will:

1. Determine what state the environment is in. That is, determine the current state vector (Figure 4-5, 1).

2. Look up that state in the LUT. There will be multiple $state - action$ entries that apply. All for the same state but each with a different action (Figure 4-5, 2).

3. DAARTS will employ an epsilon-greedy policy. I.e. it will choose the $state - action$ that corresponds to the largest $Q(state, action)$ or pick it randomly if performing exploratory learning. The largest $Q(state, action)$ represents the action that is likely to lead to the most optimal long-term reward (Figure 4-5, 3).

4. Take the corresponding action and determine the reward. In DAARTS, the reward is a function of the number of patients discharged and in this model it is a terminal reward only, computed at the final time step (Figure 4-5, 4).

5. Update the previous $Q(state, action)$ according to equation (3-6) (Figure 4-5, 5).

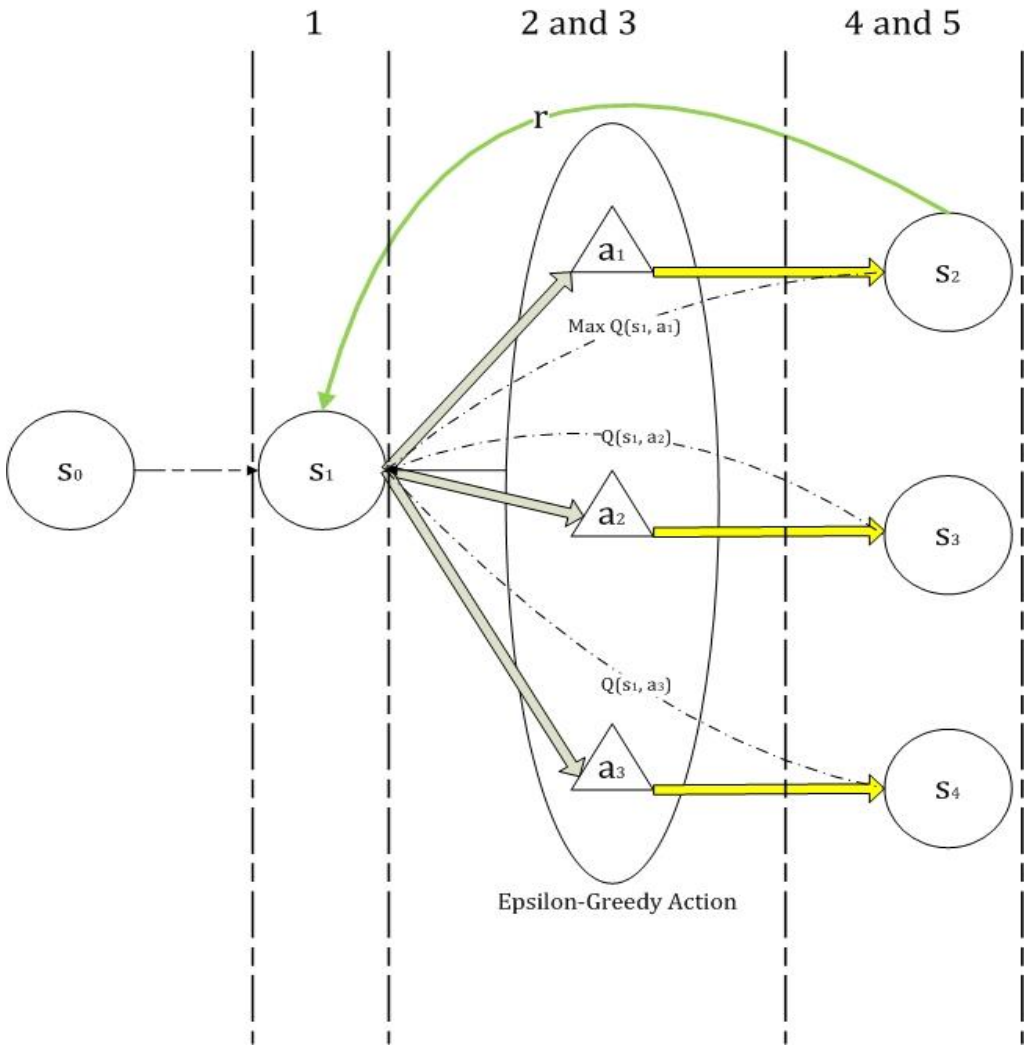

Figure 4-5 DAARTS back-up process using RL-TD

The effectiveness of learning is controlled by two parameters, the learning rate (α) and the discount factor (γ). Our findings suggest that these values have to be carefully chosen. For example with a small learning rate, the agent learning behavior is slow, whereas higher learning rates may lead to instability.

The exploration rate (ε), is used to control the frequency with which random actions are selected. This helps the learning agent to explore other paths that otherwise may not be experienced and which might lead to a more optimal solution.

To choose adequate parameters for our simulated model, sensitivity analysis was applied to a prototype model [26] and also summarized in Appendix B. This process was conducted before the actual implementation. The result suggested the following values:

$\alpha = 0.5$, $\gamma = 0.7$, ε = approximately one exploratory movement every 8 time steps

4.2.3 Training Results

We demonstrate learning in DAARTS as follows. The goal is to optimize the total number of discharged patients of the two emergency units at both hospitals (VGH and SPH) within a simulated $10 - hour$ scenario with updating every $5 - minute$ (simulation time-step). In this scenario, only terminal rewards were used. The following sections discuss the results of this scenario.

4.2.3.1 DAARTS Discharged Patients Optimization

For the discussed scenario, DAARTS was configured to optimize the total number of discharged patients from the Emergency Units (EU's) at both hospitals (VGH & SPH). The optimum goal is defined as the ability to discharge all injured people (maximum of 480) from both hospitals within the simulation time. The scenario was initialized to reflect a

disaster in which the resource mode of MUR (Power Substation) was set to operate at 75% capacity ($RM_{MUR} = 2$).

The terminal rewards were computed as follows:

$$r = num_{VGH} + num_{SPH} \quad (4\text{-}6)$$

Where:

num_{VGH}: Total number of discharged patients from VGH

num_{SPH}: Total number of discharged patients from SPH

The actions that are performed by DAARTS will reallocate the available limited resources among the interconnected infrastructure using the predefined distribution ratios as per Table 2-5 and Table 2-8.

After training, DAARTS was able to discharge all injured patients within $10 - hours$ of simulated time (Figure 4-6). A total of 366 patients (76.25%) were discharged in less than 9 $hours$ (Figure 4-7). In real time however, each episode took DAARTS about 6 $minutes$ to complete. If DAARTS is to be used to assist in a real emergency, this figure can be too long. For example, Figure 4-6 shows DAARTS reaching a stable discharge rate after about 22 learning episodes. Given 6 minutes per episode, this translates to about 2.2 $hours$ of real elapsed time. It is expected that having to wait 2.2 $hours$ before such a software support system is ready to offer recommendations in the mitigation of a disaster is unacceptable.

The largest portion of this computation time can be attributed to the communication overhead incurred by the platform interface. Further, the time taken for DAARTS to converge is not guaranteed. It may take longer than what was observed here.

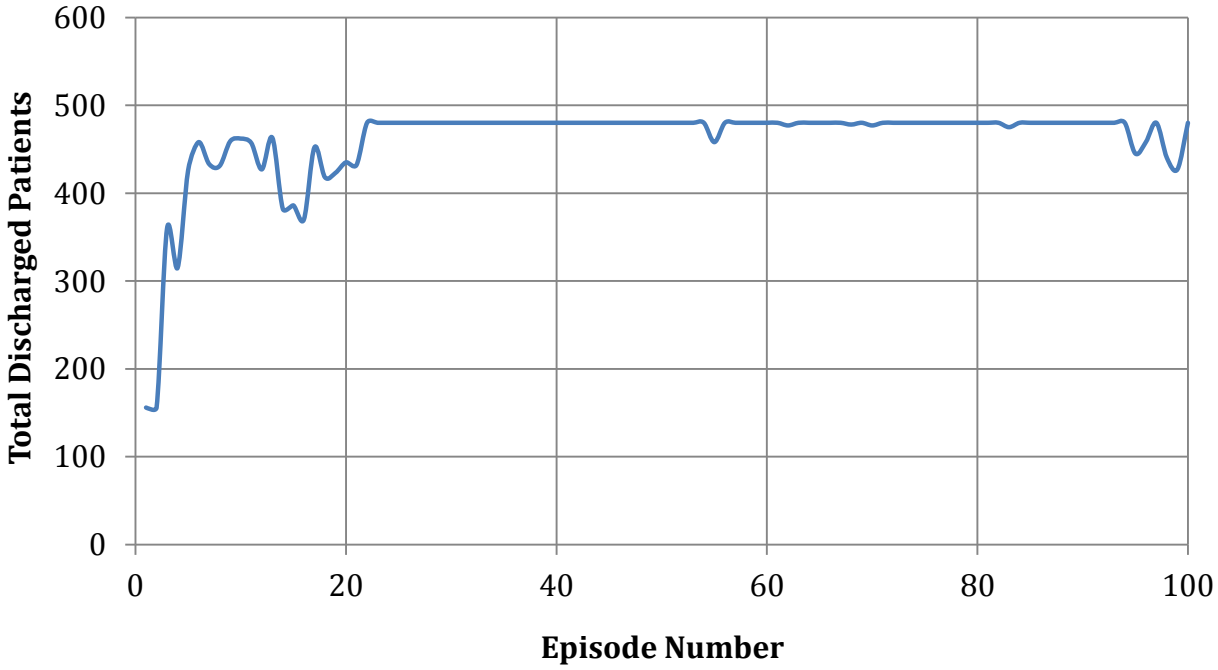

Figure 4-6 Total number of discharged patients from both emergency units

The following table (Table 4-3) summarizes the results and shows an improvement in DAARTS performance upon the training completion.

Table 4-3 DAARTS learning performance comparisons between first and last episodes

	Total Discharged Patients
First Episode	156
Last Episode	480

These results encouraged us to further analyze the reasons behind this success. A detailed analysis of the system outcome is presented in the following paragraph.

Figure 4-7 shows DAARTS behavior over a single simulated $10 - hour$ episode at the start of training. It is consistent with the expectation that, with no experience DAARTS' ability to discharge patients is effectively random. This can be seen from the left plot (VGH), in which the total number of discharged patients from VGH is only 12. For SPH (the right plot), we observe that all patients are discharged, albeit sporadically. This may be explained by the fact that some distributor settings will divert adequate resources to SPH and that the patient load is small enough and the simulation time long enough for even random selection to have an effect. This is not so for VGH.

The behavior of DAARTS after training is captured in Figure 4-8. The right plot (VGH) and the left plot (SPH) show that DAARTS was able to discharge all patients from both hospitals. The progress of DAARTS' learning itself can be measured by observing the changes in the $state - action\ value$ (ΔQ) of one of the most visited $state - action$ pairs in the final episode. Ideally, we expect ΔQ to gradually approach zero as a strong indication of convergence (Figure 4-9).

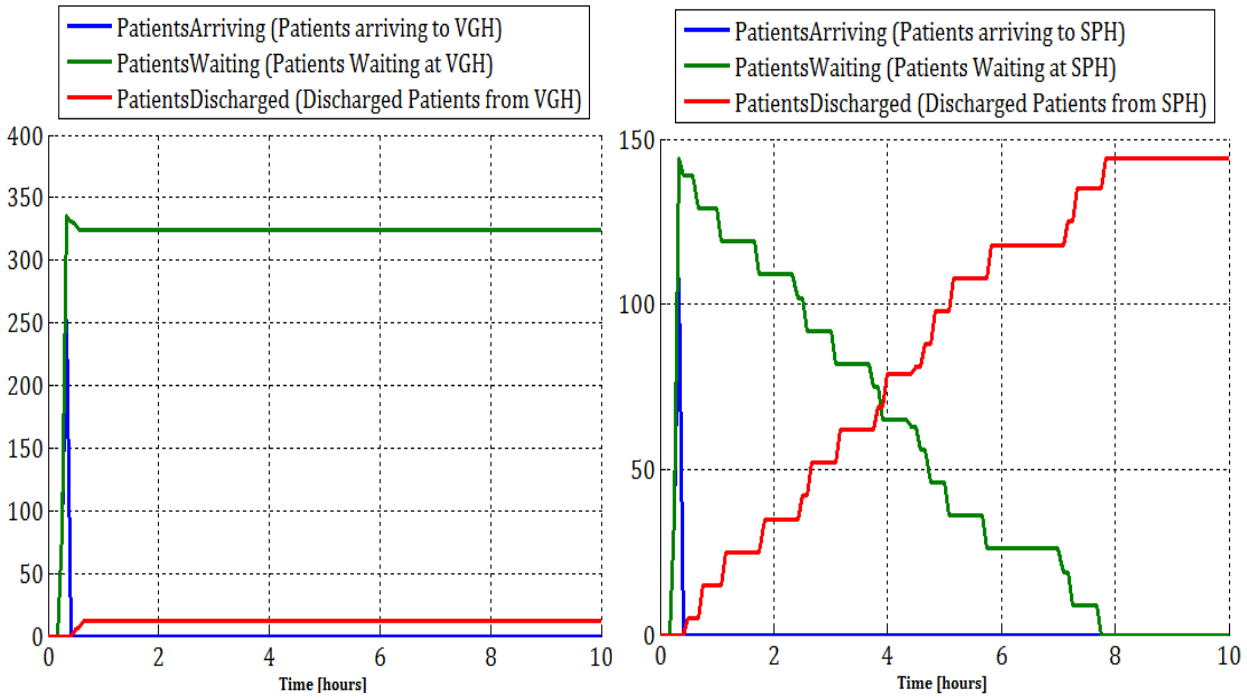

Figure 4-7 DAARTS (VGH and SPH) first episode

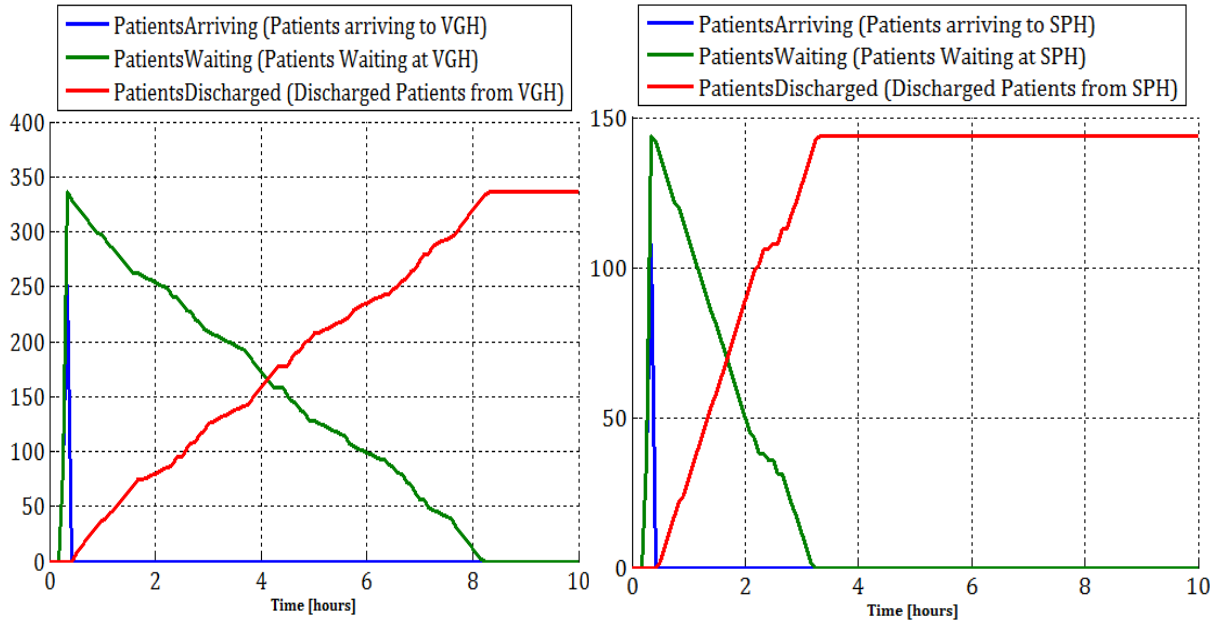

Figure 4-8 DAARTS (VGH and SPH) final episode

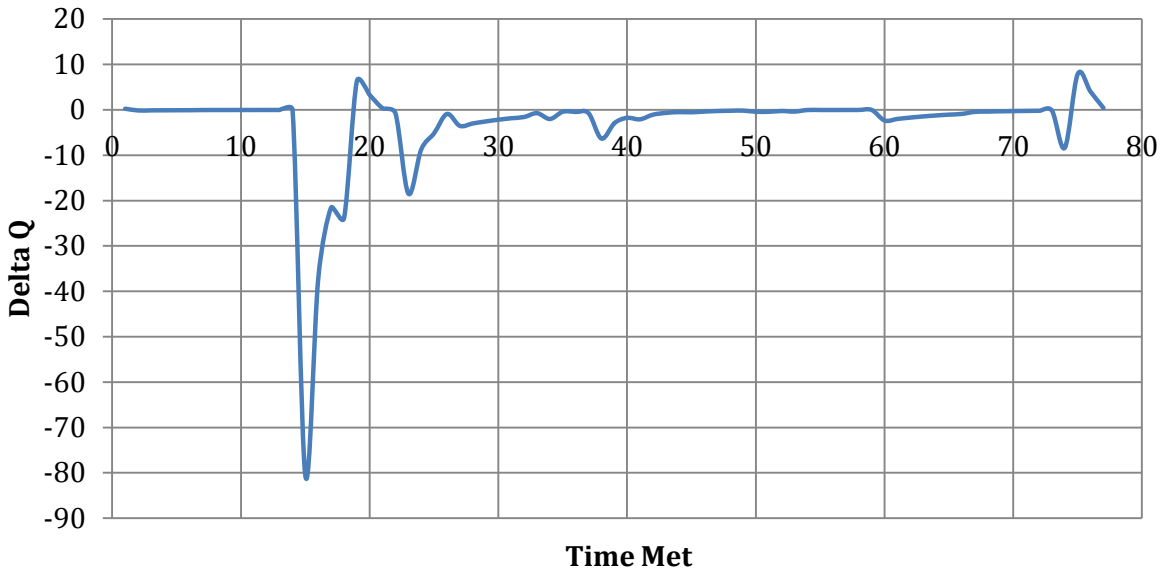

Figure 4-9 Delta Q behavior of the most visited state, action index of the final episode

The above results illustrate the ability of DAARTS to manage the available resources between hospitals intelligently. This observation is supported by Figures 4-10 and 4-11, where the RM's of both emergency units before and after the training were captured. These captions match what we observed in Figures 4-7 and 4-8 and, most importantly, the ability to discharge all patients in fewer time steps compared to the first episode. In Figure 4-10, DAARTS with no learning was not observed to make useful actions. This explains the frequent variations in the VGH resource mode behavior (blue line). An improvement is observed after training since the available compromised resources were set at higher discharged rates as per i2Sim HRT (red line). With i2Sim greater resources are expected to improve discharge rates. Figure 4-11, shows results for SPH. Note that higher discharge patients rates are not observed for SPH, since only 144 patients out of 480 patients are directed to SPH.

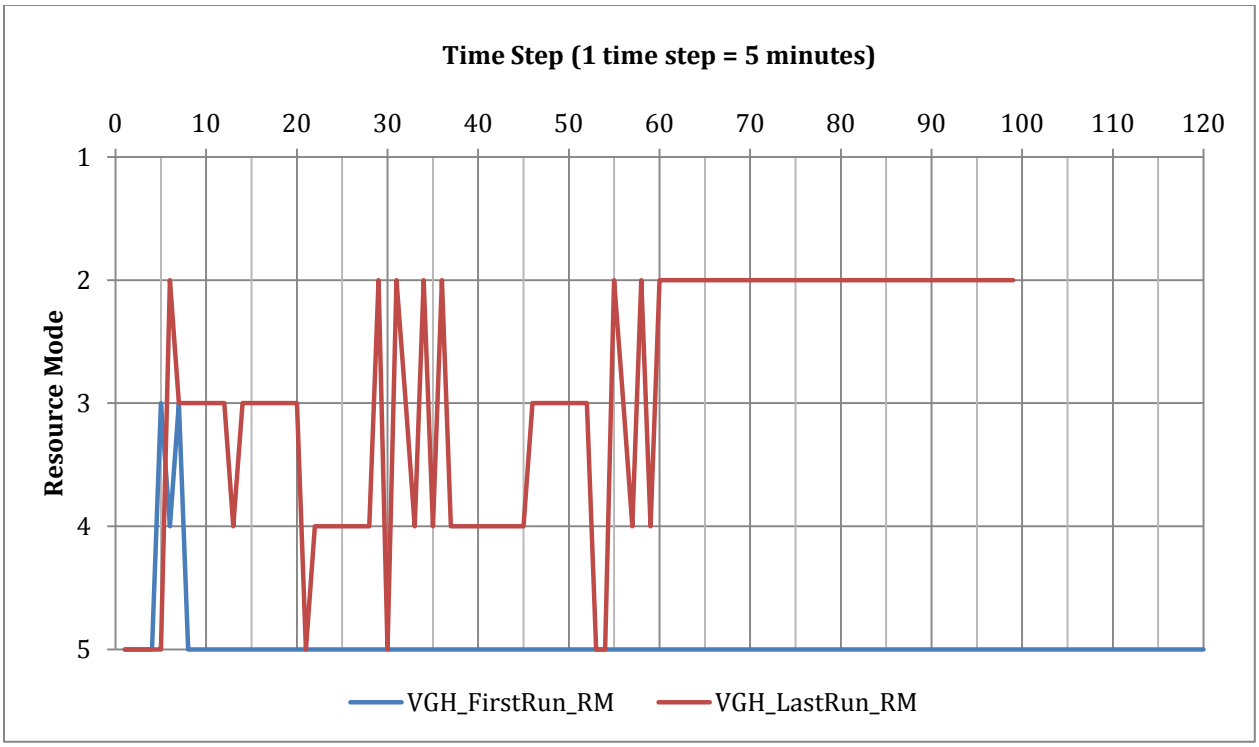

Figure 4-10 VGH resource mode behavior

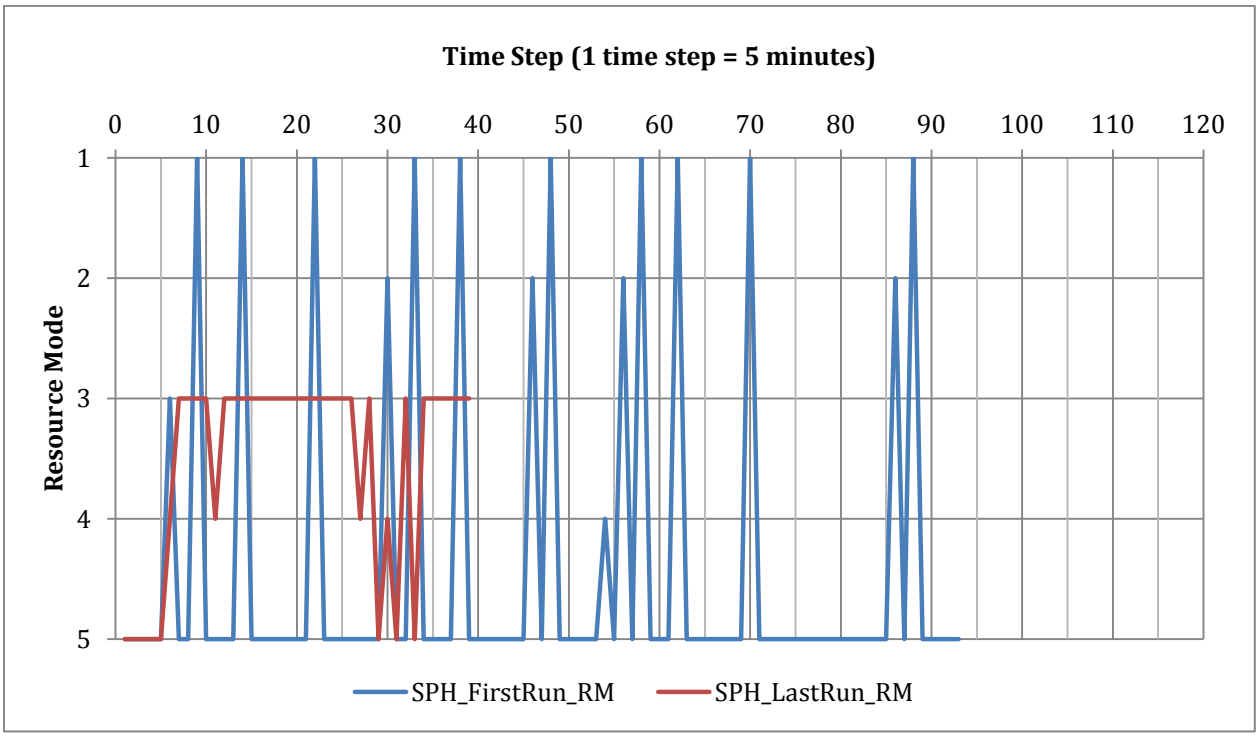

Figure 4-11 SPH resource mode behavior

The management of the available resources is a reflection of the actions that were performed by DAARTS. Thus, it is interesting to look at what actions DAARTS learned. Here, the most favored action distributes the available power and water at state, $PM_1\ RM_{2_{MUR}}$ and $PM_1\ RM_{2_{Water}}$, as seen below (Table 4-4):

Table 4-4 Actions correspond to the most visited state-action pair for final episode

Power Distribution Table					
Power Action #	VGH	Water Station	GM	DGR	Terminated
2	1.12%	1.82%	2.1%	57.34%	37.62%

Water Distribution Table		
Water Action #	VGH	SPH
7	40%	60%

For the water distributor, DAARTS chose the action that pumps more water to SPH than VGH. Surprisingly, DAARTS did not favor water distribution to VGH over SPH (as explained earlier most of the patients are discharged from VGH). This is because given the power disruption, VGH is not capable of consuming that amount of water. Through experience, DAARTS is able to observe this and learn there is no benefit in diverting large supplies of water when the target Critical Infrastructure (CI) is already compromised in terms of power. That is DAARTS through experience is able to optimize its actions in the context of a fully interdependent CI.

It is observed then that RL-TD, DAARTS, is able to learn how to maximize a desired outcome, in this case the total number of discharged patients. This result was obtained by

intelligently reallocating the available limited resources within a system of interdependent CI's. While the analysis shows that DAARTS was able to learn actions that lead to an optimal goal, it was not able to do so in real-time. For any decision support system, it is vital that such actions can be recommended in a reasonable amount of time. This is the context of the next chapter, where RL-MC (IDS-Intelligent Decision System) is asked to obtain the same goal but in a timely manner.

CHAPTER 5: RESOURCE ALLOCATION OPTIMIZATION USING MONTE CARLO POLICY ESTIMATION

Here, the resource allocation optimization problem is addressed and solved using a Monte Carlo (RL-MC) based Reinforcement Learning method. A feature of Monte Carlo (MC) based learning over Temporal Difference (TD) based learning is that learning is all performed upon completion of an episode rather than upon each time step of the episode as in TD. This is appealing as it has the potential to eliminate or reduce the extensive communication latencies experienced by DAARTS and thus lead to performance benefits. A detailed explanation of the proposed system is provided in the following sections.

5.1 Introduction to Intelligent Decision System (IDS)

This section provides a technical description of how RL-MC is realized within the i2Sim (infrastructure interdependency simulator) system. It is expected that IDS is able to maximize the desired system outcome in a reasonable time that is much faster than DAARTS. But before that, it is essential to clarify the learning sequence of this proposed architecture.

In RL-MC, the agent interacts with the modeled environment episodically. As explained in earlier chapters, the agent's behavior is based on following a policy. This policy helps the agent to experience trajectories possibly leading to an optimum goal. Such learning process

terminates at a terminal state (T). At the terminal state, the estimation of the $state - action\ value\ function$, $Q(state, action)$, happens by back-stepping equation (3-7) at the terminal state only and equation (3-8) at the non-terminal states, to all $state - action$ values in the visited trajectory as shown in Figure 5-1. This process averages the observed total returns of all visited states within the trajectory, moving towards discovery of an optimum one.

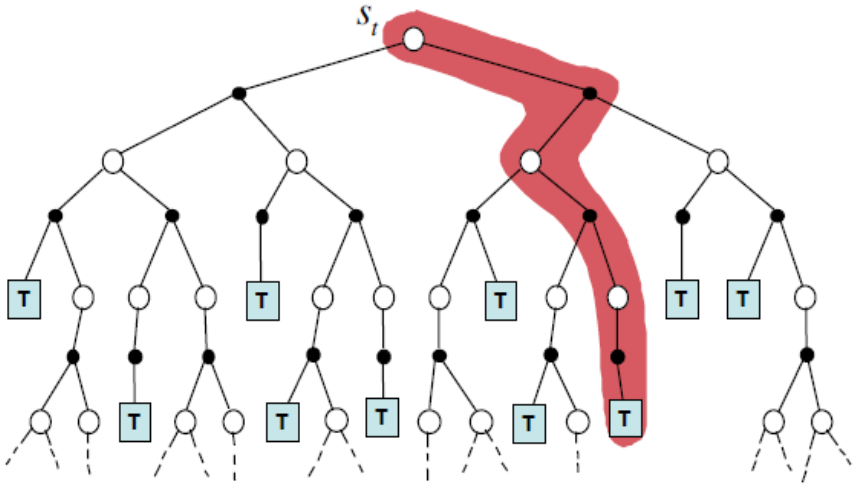

Figure 5-1 RL-MC estimation of the observed total returns [25]

5.1.1 Application of RL-MC in i2Sim

The research in this thesis addresses the problem of decision-making in the context of decision-making involving interconnected and interdependent critical infrastructures (CI's) in a timely manner. The objective of the RL-MC is to reduce the communication time of the information exchange between the two-modeled systems. This is possible since RL-MC does not perform bootstrapping. This aspect of RL-MC will help minimize the number of communication instances within the platform interface. For example, because of the

nature of RL-TD, DAARTS performs 120 communication requests to complete its learning episode. This causes the overall learning process to last over 600 $minutes$. When simulating a $10-hour$ scenario, this figure is too long. In the case of IDS, however, the number of communication instances has been reduced to just two. It is anticipated that the learning processing time of RL-MC will improve by an order of magnitude compared to RL-TD. Such improvement will allow the modeling to be extended to systems that include a larger number of state variables. From a decision-making point of view, this will generate the same optimum precise decisions but in shorter time.

The main aspects of RL-MC and the parameters available in i2Sim are similar to that in RL-TD. For example, the application of RL-MC to i2Sim is a matter of formulating those parameters in terms of states, actions and rewards.

In the RL-MC, the problem is formulated as follows: the operating modes (Physical Mode and Resource Mode) of each modeled infrastructure unit represent the $state$ of the modeled system. The ratios at the distributors represent the $actions$ that the agent can perform at each visited $state$. As before, the $state-action$ pairs are represented by a utility function that estimates the probability of obtaining the long-term reward upon choosing action a in state s and form the basis by which a Reinforcement Learning (RL) agent will make decisions. The probability estimate is determined by averaging sampled return. Similarly to DAARTS, the return is the expected number of discharged patients at the end of each episode but is applied only upon reaching the terminal state at the end of the simulation horizon, as depicted in Figure 5-2.

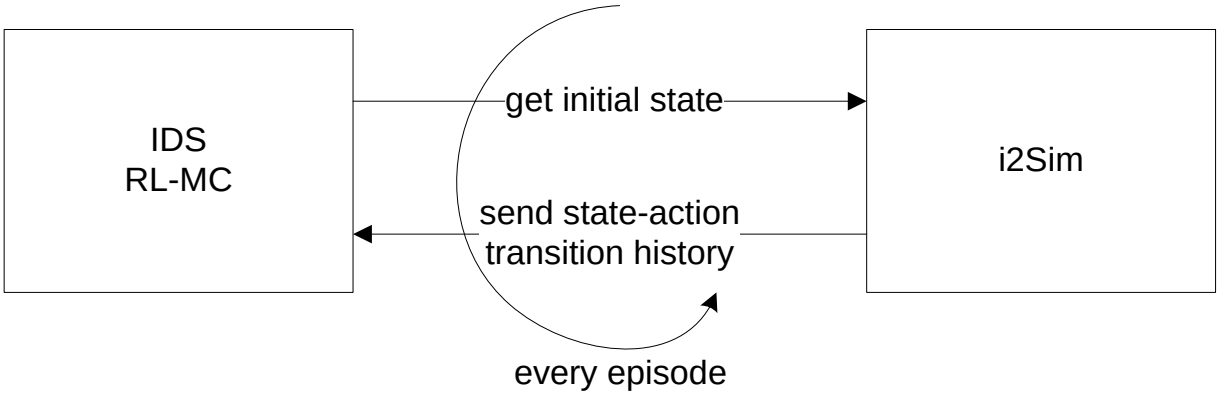

Figure 5-2 Systems interaction in IDS

5.1.2 System Architecture Description

Like DAARTS, the learning agent (IDS) is implemented in JAVA and communicates with the i2Sim environment in MATLAB. As explained in Chapter 3, the communication is performed over the MATLAB-JAVA interface, where the required information *states* (from i2Sim) and *actions* (from IDS) are exchanged. From i2Sim, IDS recognizes the Physical Modes (PM's) and the Resource Modes (RM's) of the modeled critical infrastructures (CI's) (state). Then at JAVA, IDS picks the best distribution ratio (action) that maximizes the total number of discharged patients.

In RL-TD (DAARTS) the communication between the two systems introduces an overhead that is incurred into every time step. In fact a significant portion of the computation time is due to this MATLAB-JAVA communication interface alone. With RL-MC (IDS), this communication overhead happens only twice per episode, once at the beginning and once

at the end. Therefore, the communication time is reduced by half. This is especially advantageous when modeling a more complex system.

5.2 IDS vs. DAARTS

In this section the technical differences between IDS and DAARTS are described. In [43], DAARTS is shown to converge to the optimum solution after certain number of episodes, as shown in Figure 5-3. In this case, DAARTS took about 22 *episodes* to discover the optimum path to the maximum number of discharged patients using RL-TD. The 480 *patients* were entirely discharged throughout the next episodes except for few episodes were DAARTS tended to explore more (dips in the curve). As explained in Chapter 4, using an Intel(R) Core(TM) i5 CPU 2.8 GHz, based computer with 8 GB RAM, each episode took around 6 *minutes* to complete, which makes the total learning time about 132 (22 x 6) *minutes* to discover the optimum number of discharged patients.

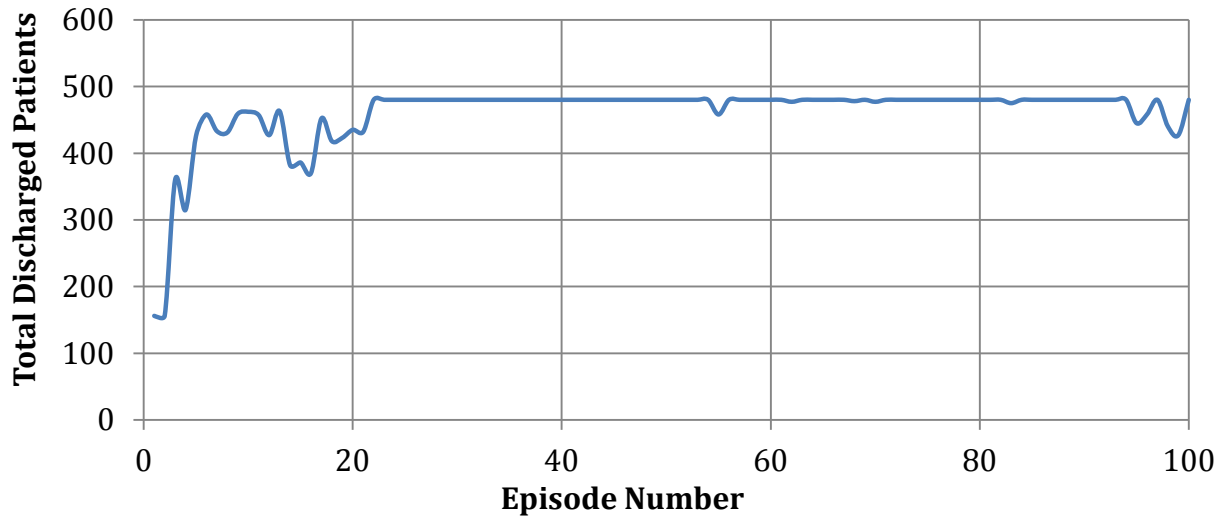

Figure 5-3 DAARTS learning behavior using RL-TD

For IDS, the agent is asked to interact with the same i2Sim (infrastructure interdependency simulator) environment used by DAARTS. The goal of IDS is to find the optimum trajectory that leads to an optimum goal using RL-MC. Our expectation is that this approach will show an ability to converge towards the maximum number of discharged patients much faster. As seen in Figure 5-4, for three consecutive runs (initialized independently) not only was IDS able to converge to the optimum solution in 17 *episodes* (as per the 3rd attempt) but each episode took around 3 *minutes* to complete.

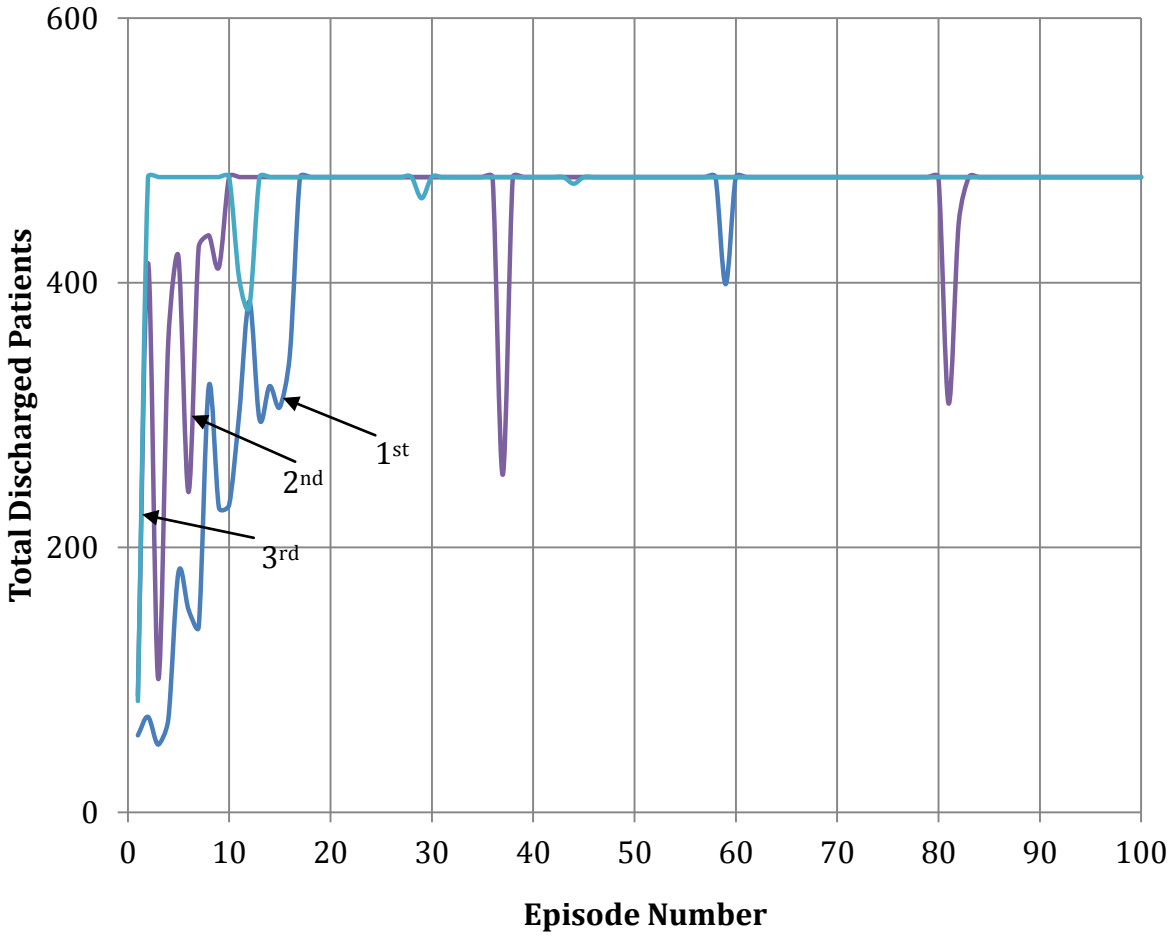

Figure 5-4 IDS learning behavior using RL-MC

Thus, using RL-MC the total required time to discover the optimum trajectory was about 51 *minutes* (Table 5-1). That makes it 2.6 times faster than DAARTS. This reveals a substantial improvement of IDS over DAARTS for the proposed system.

Table 5-1 IDS and DAARTS learning performance comparisons for first and last episodes

	Total Discharged Patients (IDS)	**Optimality Reaching in Real Time (min.)**	**Total Discharged Patients (DAARTS)**	**Optimality Reaching in Real Time (min.)**
First Episode	58	51	156	132
Last Episode	480		480	

Figures 5-5 and 5-6 show the behavior of IDS before and after training. These figures represent the level of input resources that are available for both hospitals at every time-step (the more the input resources the higher the RM). In the first episode, IDS attempted to explore the surrounding environment by reallocating the available resources randomly. Similar behavior was observed by DAARTS (Chapter 4) since no experience was captured early on. However, IDS discovered that more patients are discharged from VGH (Vancouver General Hospital). This explains why IDS elected to send more resources to VGH over SPH (Saint Paul Hospital). The actions that changed the distribution improved the resource mode of VGH to $RM_{VGH} = 2$, which means higher discharged patient rate as per i2Sim (infrastructure interdependency simulator) HRT (human readable table). This improvement resulted in IDS discharging all patients from VGH in less than 60 *time steps*, where in the case of DAARTS it took around 100 *time steps*. Moreover, the decisions from IDS enabled SPH to continue its service with fewer resources.

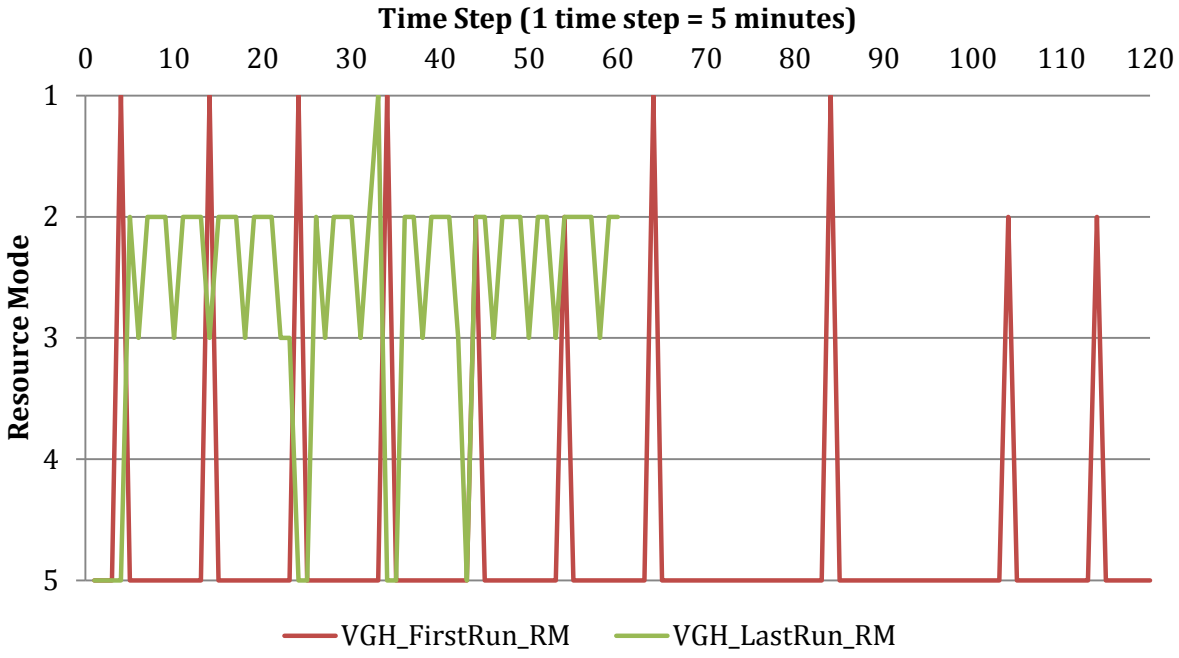

Figure 5-5 VGH resource mode behavior

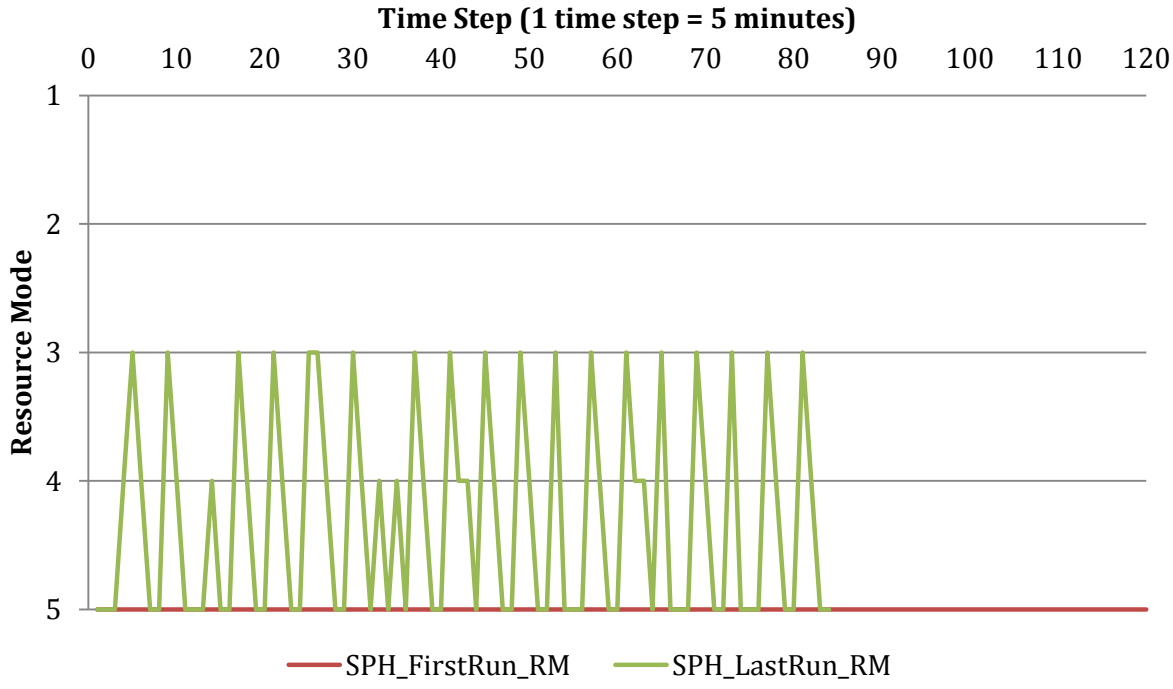

Figure 5-6 SPH resource mode behavior

5.3 IDS Downtown Vancouver Model Formulation

The previous section illustrated the potential of IDS as an assistance tool for Critical Infrastructure (CI) managers. This is because IDS is able to complete the simulation in much faster than DAARTS. A $10-hour$ simulation can be completed in about half the time of DAARTS. We are encouraged by these results and attempt to apply IDS to an extended city scale model such as Downtown Vancouver (Figure 5-7). Next, these extensions are mathematically derived, explained and compared to previous work.

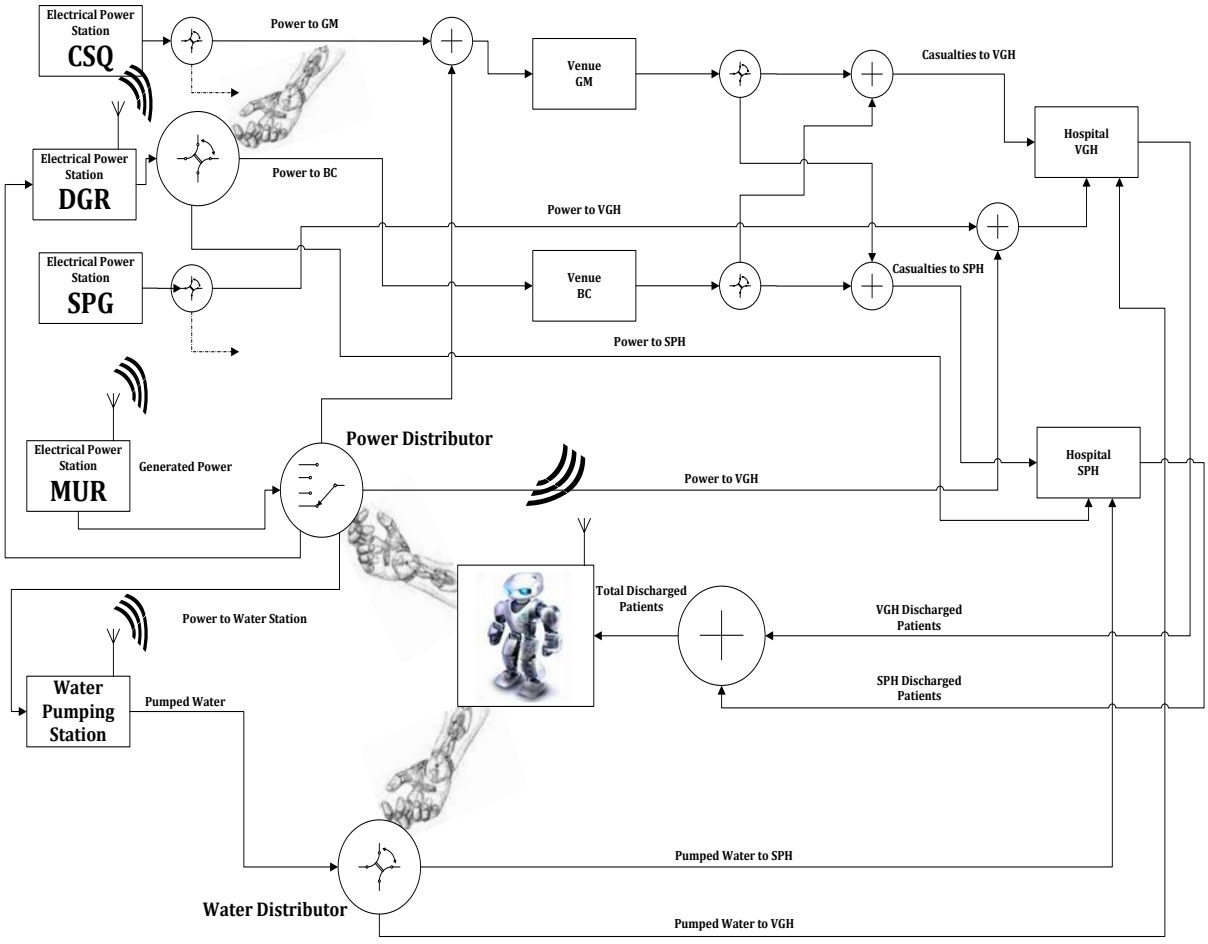

Figure 5-7 IDS within i2Sim Downtown Vancouver Model

Table 5-2 IDS and DAARTS state space system comparison

System Elements	DAARTS	IDS
State	num $_{states}$ = x^y =152 = 225	num $_{states}$ = x^y =153 = 3,375
Action	num $_{actions}$ = Dis. $_{MUR}$ x Dis. $_{Water}$ = 10 x 11 = 110	num $_{actions}$ = Dis. $_{DGR}$ x Dis. $_{MUR}$ x Dis. $_{Water}$ = 5 x 10 x 11 = 550
LUT Size	num $_{rows}$ = num $_{states}$ x num $_{actions}$ = 225 x 110 = 24,750	num $_{rows}$ = num $_{states}$ x num $_{actions}$ = 3,375 x 550 = 1,856,250
Real Time Needed	6.2054 minutes / episode	3.2519 minutes / episode

Where:

State (***s***)**:** Represents the total number of states (Physical Modes-PM's and associated Resource Modes-RM's) of the modeled system.

Action (***a***)**:** Represents the total number of combined actions of the controlled distributors.

x: Represents the maximum possible number of resource modes associated with each physical mode per production cell.

y: Represents the number of production cells that are exposed to the RL agent.

$Dis._{DGR}$: Represents the number of distribution ratios of the DGR power substation distributor.

$Dis._{MUR}$: Represents the number of distribution ratios of the MUR power substation distributor.

$Dis._{Water}$: Represents the number of distribution ratios of the Water Pumping station distributor.

Similarly to DAARTS (Table 4-2), the *states* and *actions* suggest a theoretical maximum size of 1,856,250 *elements* for the Look Up Table (LUT), as seen in Table 5-3. Each row of the table represents a $state - action$ pair and the associated long-term average sampled return. These $state - action$ values are updated at the end of every episode for a given policy.

Below is a sample of the LUT being used for this model:

Table 5-3 Sample of IDS's look up table

(<State>, <Action>)	**Q(s, a)**
$(< \mathbf{PM_{DGR}}, \mathbf{RM_{DGR}}, \mathbf{PM_{MUR}}, \mathbf{RM_{MUR}}, \mathbf{PM_{water}}, \mathbf{RM_{water}} >,$ $< \mathbf{Dis._{DGR}}, \mathbf{Dis._{MUR}}, \mathbf{Dis._{water}} >)$	
---	---
---	---

Where:

PM_{DGR}: Represents the DGR electrical power substation physical mode (state)

RM_{DGR}: Represents the DGR electrical power substation resource mode (state)

PM_{MUR}: Represents the MUR electrical power substation physical mode (state)

RM_{MUR}: Represents the MUR electrical power substation resource mode (state)

PM_{Water}: Represents the Water Pumping substation physical mode (state)

RM_{Water}: Represents the Water Pumping substation resource mode (state)

$Dis._{DGR}$: Represents the DGR electrical power substation distributor (action)

$Dis._{MUR}$: Represents the MUR electrical power substation distributor (action)

$Dis._{Water}$: Represents the Water Pumping station distributor (action)

The purpose of IDS will be to evaluate the environment and to find the best distribution settings of the three distributors (two for power, MUR and DGR, and one for water). These elements control the supplies of electricity and water that are distributed to the interconnected cells and significantly influence how effectively patients can be discharged from both Emergency Units (EU's) at the two hospitals.

5.3.1 Training IDS

A test of 200 episodes was conducted to train IDS. Each episode represented a $10 - hour$ (simulation time) period following a catastrophic event. This event caused sudden changes on the operating conditions of the targeted cells. These changes lead to deterioration in the operability with respect to nominal conditions. Following the event, no further changes

were made in the state of damage of the modeled CI's. However, the associated Resource modes (RM's) are altered as the situation evolves. These frequent changes in the operating modes of the interconnected CI's and the corresponding reallocation decisions of the available resources are captured by the Look Up Table (LUT). As learning proceeds, the contents of the LUT are updated as IDS converges towards an optimal policy.

5.3.2 Scenario Explanation

The same disaster scenario used in DAARTS (Figure 4-4) was applied to IDS. The simulated earthquake caused severe damage to the towers that are used to carry the power transmission lines. These lines transmit power to the MUR electrical power substation. Although, the PM of MUR was not affected, the RM was reduced due to this damage. Subsequently, because of the reduced electrical power, the water facility was not able to operate at full capacity and the amount of power delivered to the DGR Electrical Power Substation was also compromised.

Unlike DAARTS which was used to control two production cells, IDS will control three production cells DGR, MUR and the Water Pumping Station. This extended system required a much larger LUT than that of DAARTS. While this would have caused problems for DAARTS in terms of computation time, it is expected that IDS will be able to adequately cope with the increased overhead and still discover the optimal trajectory within reasonable time.

The triage time, ambulatory travel time, number of casualties and other parameters are as in DAARTS. The emergency units, at both VGH (Vancouver General Hospital) and SPH (Saint Paul Hospital), were asked to continue to function, even though compromised due to the limited water and power supplies. Here, the objective of IDS was to experience this scenario and to find a way to mitigate the impact to the emergency units at the hospitals.

Similarly to what we had in DAARTS, the model simulated a period of $10-hours$ in $5-minute$ time steps. The simulation parameters used were taken from the same report [42].

5.3.3 RL-MC Algorithm

IDS implements the tabular form of the $Q-Learning$ algorithm using RL-MC. LUT is used to determine the action for the next $state$ of the modeled environment. Here, the $action$ is to decide how to set the three distributors (MUR, DGR and Water Pumping Station). Given an initially untrained LUT, the goal of the RL-MC is to find the optimal action to perform in each state (optimal trajectory).

$$a = \begin{bmatrix} DGR_{distributor_{BC}} \\ DGR_{distributor_{SPH}} \\ DGR_{distributor_{Others}} \\ MUR_{distributor_{VGH}} \\ MUR_{distributor_{Water}} \\ MUR_{distributor_{GM}} \\ MUR_{distributor_{DGR}} \\ MUR_{distributor_{Others}} \\ Water_{distributor_{VGH}} \\ Water_{distributor_{SPH}} \end{bmatrix} \quad X \quad \begin{bmatrix} p_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & p_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & p_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & p_4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & p_5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & p_6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & p_7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_2 \end{bmatrix} \tag{5-1}$$

The state of the environment is a vector representing the PM's and RM's of the production cells at any given time.

$$s = \begin{bmatrix} PM_1\ RM_{1_{DGR}} & PM_1\ RM_{1_{MUR}} & PM\ _1RM_{1_{Water}} & a_1 \\ PM_1\ RM_{1_{DGR}} & PM_1\ RM_{1_{MUR}} & PM\ _1RM_{1_{Water}} & a_2 \\ \vdots & \vdots & \vdots & \vdots \\ PM_1\ RM_{1_{DGR}} & PM_1\ RM_{1_{MUR}} & PM\ _1RM_{1_{Water}} & a_{10} \\ PM_1\ RM_{1_{DGR}} & PM_1\ RM_{1_{MUR}} & PM\ _1RM_{1_{Water}} & a_{11} \\ \vdots & \vdots & \vdots & \vdots \\ PM_1\ RM_{1_{DGR}} & PM_1\ RM_{1_{MUR}} & PM\ _1RM_{1_{Water}} & a_{550} \\ \vdots & \vdots & \vdots & \vdots \\ PM_5\ RM_{5_{DGR}} & PM_5\ RM_{5_{MUR}} & PM\ _5RM_{5_{Water}} & a_{550} \end{bmatrix} \quad (5\text{-}2)$$

According to RL-MC, a set of chosen actions will be initialized by IDS. The information in IDS' LUT has to be passed to i2Sim (infrastructure interdependency simulator) in order for IDS to know how to act during each episode. This is done using a list that instructs to i2Sim on what action to perform at any given state within the episode. This list is sent through the interface to i2Sim. At the end of every episode, i2Sim must collect the transition history or trajectory of states and actions that were performed. Also collected is the total number of discharged patients associated with this episode. This information is then returned back to IDS and is used to update the LUT. This is the point at which the full back-ups are performed to incorporate the returns sampled from the previous episode. The LUT, as explained earlier, will represent the state-action pairs and its associated state-action values. There will be multiple $state - action$ entries that apply (same state but each with a different action). Then IDS will employ an epsilon-greedy policy. I.e. it will choose the

$state - action$ that corresponds to the largest $Q(state, action)$ or pick a policy randomly if performing exploratory learning. The largest $Q(state, action)$ represents the action that is likely to lead to the most optimal trajectory. Then IDS will take the corresponding action and determine the reward. Here, the reward is a function of the number of patients discharged and in this model terminal and immediate reward are applied. The immediate reward is applied at every time step by determining the difference of discharged patients between the current and previous states. For the terminal reward it is calculated and applied at the final time step only (terminal state) based on the total number of discharged patients from both Emergency Units (EU's). Finally, IDS updates the previous $Q(state, action)$ according to equation (3-8) for non-terminal states and according to equation (3-7) for the terminal one.

The effectiveness of the learning operation is controlled by three parameters, learning rate (α), discount factor (γ) and exploration rate (ε). These values have to be carefully selected for best convergence.

To choose an adequate set of learning parameters for this model, a form of sensitivity analysis was employed prior to the implementation. This approach adopted the heuristic of fixing one parameter while observing variations in another. For example, to determine the best α and γ, we fix ε (to one exploratory movement every 10 time steps) and observe learning performance as α and γ are varied (in a nested loop). From Figure 5-8, episodes that fall between episode number 612 and 684 are seen to exhibit stable behavior in terms of maximizing discharged patients. This range is bounded as follows: $\alpha = [0.4104, 0.3528]$

and $\gamma = [0.5896, 0.6472]$. Values for α and γ were selected as the simple averages of these bounds. Thus, the recommended values are: $\alpha = 0.3816, \gamma = 0.6184$.

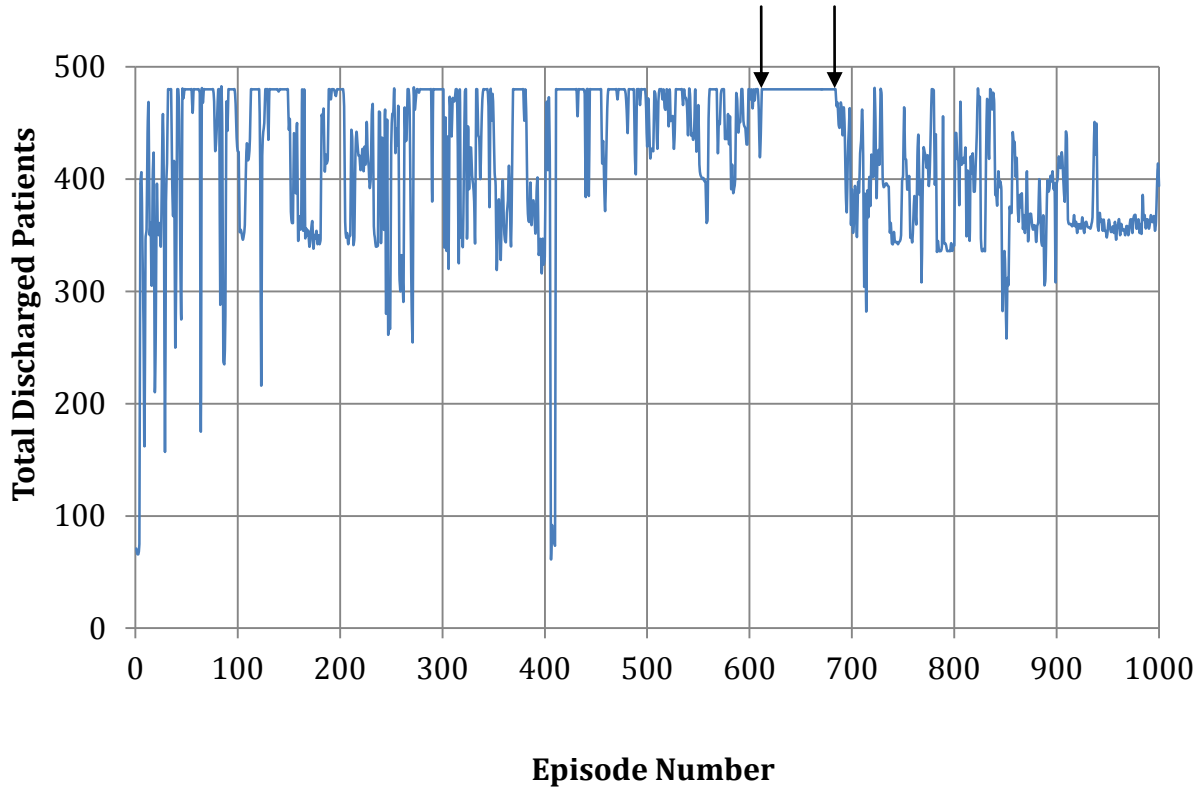

Figure 5-8 Sensitivity analysis for better learning parameters

5.3.4 Training Results

The learning behavior of IDS is demonstrated as follows. The goal is to optimize the total number of discharged patients from the emergency units at VGH and SPH. Here, we want to show that IDS is able to do so for a realistic system.

In this scenario the immediate and terminal rewards were computed as follow:

$$r_{immediate} = (num_{VGH} + num_{SPH})_{current} - (num_{VGH} + num_{SPH})_{previous} \quad (5\text{-}3)$$

$$r_{terminal} = num_{VGH} + num_{SPH} \quad (5\text{-}4)$$

Where:

num_{VGH}: Total number of discharged patients from VGH

num_{SPH}: Total number of discharged patients from SPH

A detailed discussion of the system outcome is presented in the following section.

5.3.5 IDS Discharged Patients Optimization

In this scenario, IDS was configured to optimize the total number of discharged patients from both EU's at the two hospitals. The goal is defined as the ability to discharge all injured people (maximum of 480) from both units within the simulation time (10 $hours$). The scenario was initialized to reflect a disaster in which the resource mode of MUR was set to operate at 75% capacity ($RM_{MUR} = 2$).

The actions that IDS will perform, will reallocate the available limited resources among the interconnected infrastructure using the predefined distribution ratios, as per Table 2-5, 2-6 and 2-8.

An indication of good learning is when all the patients are discharged within $10 - hours$ simulation time. IDS was able to achieve this goal in under $10 - hours$ simulation time. It is

worth mentioning that this achievement used a LUT that is 75 *times* larger than the LUT that was used in DAARTS [43]. That is, IDS is actually simulating a more complex system than DAARTS. Recall that IDS can simulate a $10-hour$ episode in 3.2519 *minutes* compared to 6.2054 *minutes* by DAARTS for a less complex system.

The following table summarizes the results and shows the improvement in IDS performance upon learning completion.

Table 5-4 IDS learning performance comparisons between first and last episodes

	Total Discharged Patients
First Episode	83
Last Episode	480

Figure 5-9 shows IDS behavior over a single $10-hour$ (simulation time) scenario at the start of training. It is consistent with the expectation that, with no experience IDS ability to discharge patients is effectively random as shown in the first 67 *episodes*. It took about 3.6 *hours* for IDS to stabilize to an optimum trajectory as observed in episodes 67 and after.

Figure 5-10 shows which $state-actions$ (x-axis) where visited at each time step (y-axis) during the final episode of training. For example, state number 243 was visited at 77 different time steps and action number 26 was the most performed action at state 243. Being the last episode, these actions represent those that lead to the best performance.

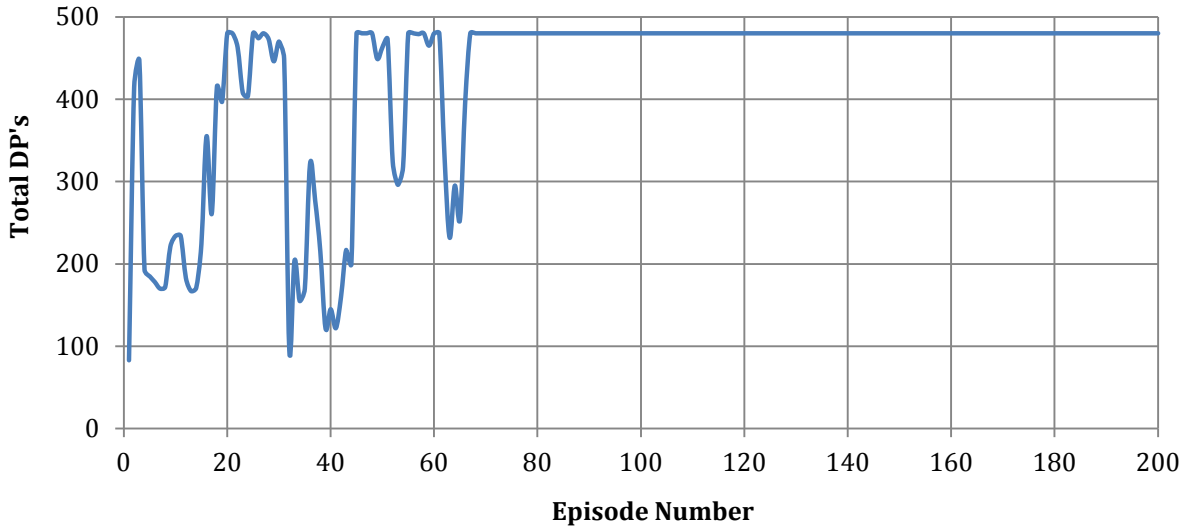

Figure 5-9 Total numbers of discharged patients from both emergency units

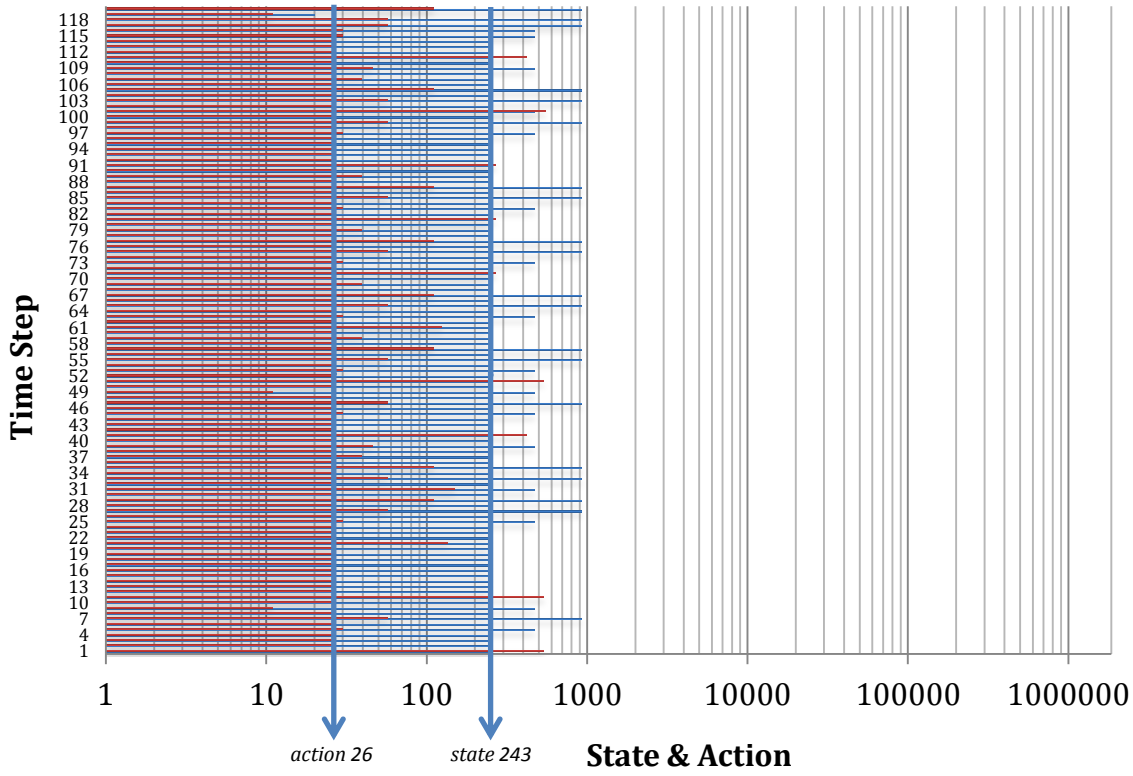

Figure 5-10 State-Action visits of the final episode

It is interesting to look at what actions IDS has learned to pick. The most favored action is the action that distributes the DGR and MUR power, and the Water Pumping Station water, as seen in Table 5-5.

Table 5-5 Action corresponds to the most visited state-action pair of the final episode

DGR Distribution Table			
Power Action #	BC	SPH	Terminated
1	2.98%	2.98%	94.05%

Water Distribution Table		
Water Action #	VGH	SPH
4	70%	30%

MUR Distribution Table					
Power Action #	VGH	Water Station	GM	DGR	Terminated
3	1.12%	1.28%	2.1%	57.34%	38.16%

IDS realized the reduction in the supplied power of MUR because of the disruption. That is why all corresponding actions should be made with respect to the available power. The 439.5 MW of power were distributed in a way to satisfy the demands of the interconnected CI's. Here, DGR received around 252 MW of power that helps to function BC and SPH. It is the same action that delivered 5.63 MW to the Water Pumping Station. Because of the limited amount of water that is available for both VGH and SPH, IDS decided to distribute 70% and 30% of water respectively. IDS favored VGH over SPH since more patients are admitted to VGH.

Surprisingly, IDS was able to recognize the needs of the interconnected CI's by reallocating the available resources without squandering these limited resources. IDS was able to do this based on the minimum requirements needed by the CI's, which helped achieve the

optimum goal without compromising their functionality. Here, IDS realized how critical VGH is but in the mean time IDS did not ignore the need of SPH since a reasonable number of patients were directed to it. In this case, IDS assigned 36.4 $KL/hours$ of water to VGH because there was no need to send more water if the received power gets compromised. Subsequently, a reasonable amount of water was assigned to SPH in order to help SPH perform its service. Through experience, IDS was able to observe this and learn that there was no benefit in diverting large supply of water when the target CI is already compromised in terms of power. That, is IDS through LUT was able to observe that the averaged sampled long-term reward (total discharged patients) was linked to the optimum trajectory, which eventually led to saving human lives and natural resources in the context of a fully interdependent CI's system.

As seen above, IDS was able to successfully discharge the available patients completely. This goal was met within a time scale that is better than the one observed in the previous chapter. This improvement was the result of an architectural feature in IDS that optimized communication between i2Sim and the learning agent. This feature was not compatible, however, with the incremental Temporal Difference (TD) learning and required an episodic form of Reinforcement Learning (RL), in this case a Monte Carlo (MC) solution. Even though this learning performance is encouraging, a team managing a real disaster would require almost immediate assistance. Therefore it is important to simulate and learn from the real situation as rapidly as possible, in fact faster than real time.

The goal of the next chapter is to further improve simulation and learning time. An ultimate optimization package (UOP) is proposed, the goal of which is to learn to make optimal decisions for a $10-hour$ simulation an order of magnitude much faster than IDS.

CHAPTER 6: I2SIM ULTIMATE OPTIMIZATION PACKAGE (I2SIM-UOP)

The previous chapters show that both DAARTS and IDS are able to optimize a targeted outcome within an interdependent critical infrastructures (CI's) system such as Downtown Vancouver. In the case of DAARTS, the (RL-TD) agent was able to discharge the available patients from both Emergency Units (EU's) completely, but could not achieve that within a reasonable time period. On the other hand, an over two times improvement was observed in the case of IDS. However, this solution time is still not fast enough. During natural or human-induced disasters, delays of even a few minutes could have significant negative consequences.

Here, in this section we demonstrate the same resource allocation optimization problem within a system of interdependent CI's, using RL-MC but this time implemented in a single software platform. Again, the goal is to maximize the total available number of discharged patients within an acceptable timeframe by making effective decisions. This proposed platform is referred to as i2Sim-UOP. The key architectural feature is the implementation of both simulation and learning components within a single platform. That is, both i2Sim (infrastructure interdependency simulator) and the Reinforcement Learning (RL) agent are built into a single JAVA program. Therefore, it is expected that the i2Sim-UOP will converge faster than the aforementioned learning systems (DAARTS and IDS), which would be a

significant improvement when it comes to assisting emergency responders during catastrophic events.

6.1 Why i2Sim-UOP is Needed?

The main factor that restricts the convergence speed of DAARTS and IDS is the communication bottleneck that exists between the simulator (i2Sim) and the learning mechanism (agent). An episodic RL-MC approach performs better than the incremental Temporal Difference (TD) learning approach as it reduces the overhead of communication. However, it does not eliminate it. As indicated in the prior chapters, both the simulator and the Reinforcement Learning (RL) agent are built using different programming languages. The i2Sim simulator is built using MATLAB and the RL agent is built using JAVA. The i2Sim and its platform interface (Figure 6-1), contribute to a large portion of the overall simulation and learning time.

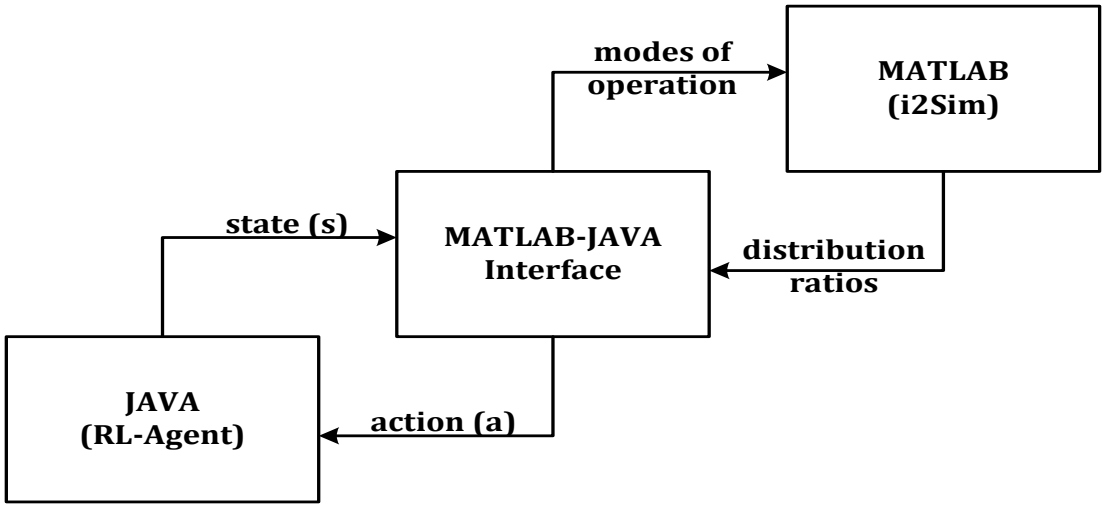

Figure 6-1 MATLAB-JAVA **intra-communication**

Further, MATLAB, as interpreted language, is uncompetitive in terms of processing efficiency compared to compiled programming languages like JAVA [44]. So MATLAB in general does not lend itself very well to modeling large systems for real-time simulation.

6.2 Monte Carlo Policy Estimation Using i2Sim-JAVA

Here, the IDS system architecture (section 5.3) is adopted for i2Sim-UOP. The i2Sim simulator is rebuilt using an object oriented programming language. The production cells, channels, storages, control elements (distributors and aggregators) and other i2Sim entities are recoded using JAVA. Similar to i2Sim (MATLAB version), a proper synchronization between the i2Sim-UOP modeled entities is required. Each entity must perform its function according to the simulation, which means that each entity is required to synchronously compute its output variables given input variables at every time step.

It should be noted that while the original i2Sim, running in MATLAB is a comprehensive piece of software written with the purpose of allowing a user to easily create, configure and simulate a system of critical infrastructures, i2Sim-UOP is limited in its feature set. For example, it does not have a graphical user interface and the infrastructures to be simulated have to be configured at the source code level. In fact, i2Sim-UOP was written with the main intent of duplicating the scenario simulated for IDS in a JAVA environment. i2Sim-UOP is not yet ready to fully replace the i2Sim system however, but already demonstrates a clear performance advantage. Progress is already underway to build a full version of i2Sim in a modern compiled language.

6.3 System Architecture Description

Similar to DAARTS and IDS, i2Sim-UOP captures the Physical Modes (PM's) and Resource Modes (RM's) of the modeled critical infrastructures (CI's) from the environment (state). Accordingly, i2Sim-UOP picks and performs the best distribution ratio (action) that maximizes the total number of discharged patients. State recognition and action implementation are both made within a single platform built using JAVA, which means no communication latency is observed (Figure 6-2). Thus, it is expected that i2Sim-UOP is able to process a comparatively massive number of simulations in a very short time.

To observe the performance of the modeled system, a test of 1000 ten-hour, five-minute time step episodes was used to train i2Sim-UOP using RL-MC. A catastrophic scenario was emulated by appropriately setting the RM's of the targeted cells. This change determines the availability of the required resources of the modeled CI's with respect to its nominal conditions. Following the incident, no changes were made in the state of damage (PM's) of the modeled CI's. However, the associated RM's that represent the available resources were altered as the situation progressed. These frequent changes in the operating modes of the interconnected CI's and the corresponding reallocation decisions of the available resources were captured and recorded by the Look Up Table (LUT). The contents of the LUT were updated as learning developed.

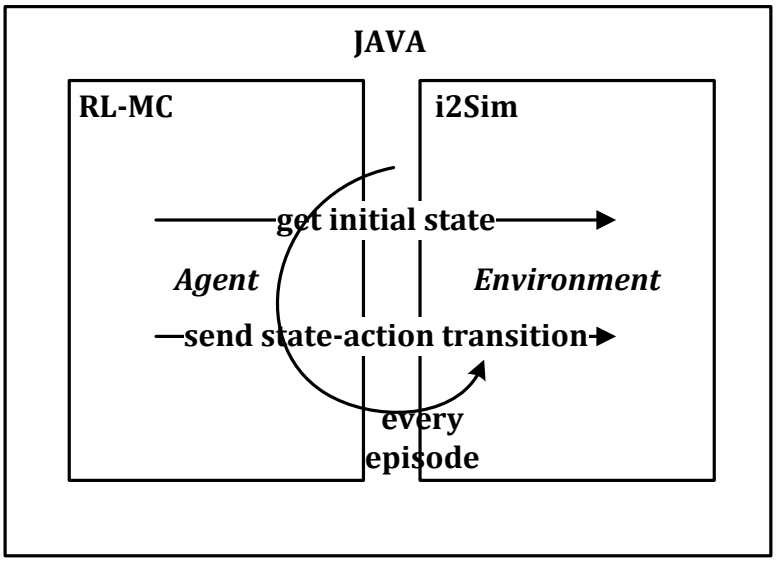

Figure 6-2 i2Sim-UOP system architecture

As a sign of optimality, the learning behavior of i2Sim-UOP is verified based on the total number of patients that are discharged from both hospitals. Like IDS, immediate and terminal rewards are applied.

6.4 i2Sim-UOP Discharged Patients Optimization

In this scenario, i2Sim-UOP is asked to optimize the total number of discharged patients from both Emergency Units (EU's) at the both hospitals (VGH and SPH). Here, we want to demonstrate the ability of the modified system to discharge all patients but in a time-scale that is much better than in DAARTS and IDS. Again, the scenario is initialized to reflect a natural disaster (earthquake) in which the Resource Mode (RM) of MUR is set to operate at 75% capacity ($RM_{MUR} = 2$).

As in IDS, the immediate rewards and terminal rewards are computed as follow:

$$r_{immediate} = (num_{VGH} + num_{SPH})_{current} - (num_{VGH} + num_{SPH})_{previous} \quad (6\text{-}1)$$

$$r_{terminal} = num_{VGH} + num_{SPH} \quad (6\text{-}2)$$

Where:

num_{VGH}: Total number of discharged patients from VGH

num_{SPH}: Total number of discharged patients from SPH

The actions that the i2Sim-UOP agent will choose and perform are the action that best distribute the available limited resources among the interconnected critical infrastructures (CI's). These actions are predefined distribution ratios that are available for the agent to choose from (Tables 2-5, 2-6 and 2-8).

As expected, the i2Sim-UOP agent was able to discharge the available patients within the simulation time, as seen in Figure 6-3. The figure shows a rapid improvement in i2Sim-UOP performance upon learning completion for two independent runs. It is consistent with our expectation that, with no experience i2Sim-UOP's ability to discharge patients is effectively random, as shown in the first episodes for both runs (Figure 6-4). It took around 10 episodes for i2Sim-UOP to discover the optimum trajectories in both cases. These are the trajectories that lead to the maximum number of discharged patients. This explains

why the agent stabilizes to those particular trajectories throughout the simulation in each run.

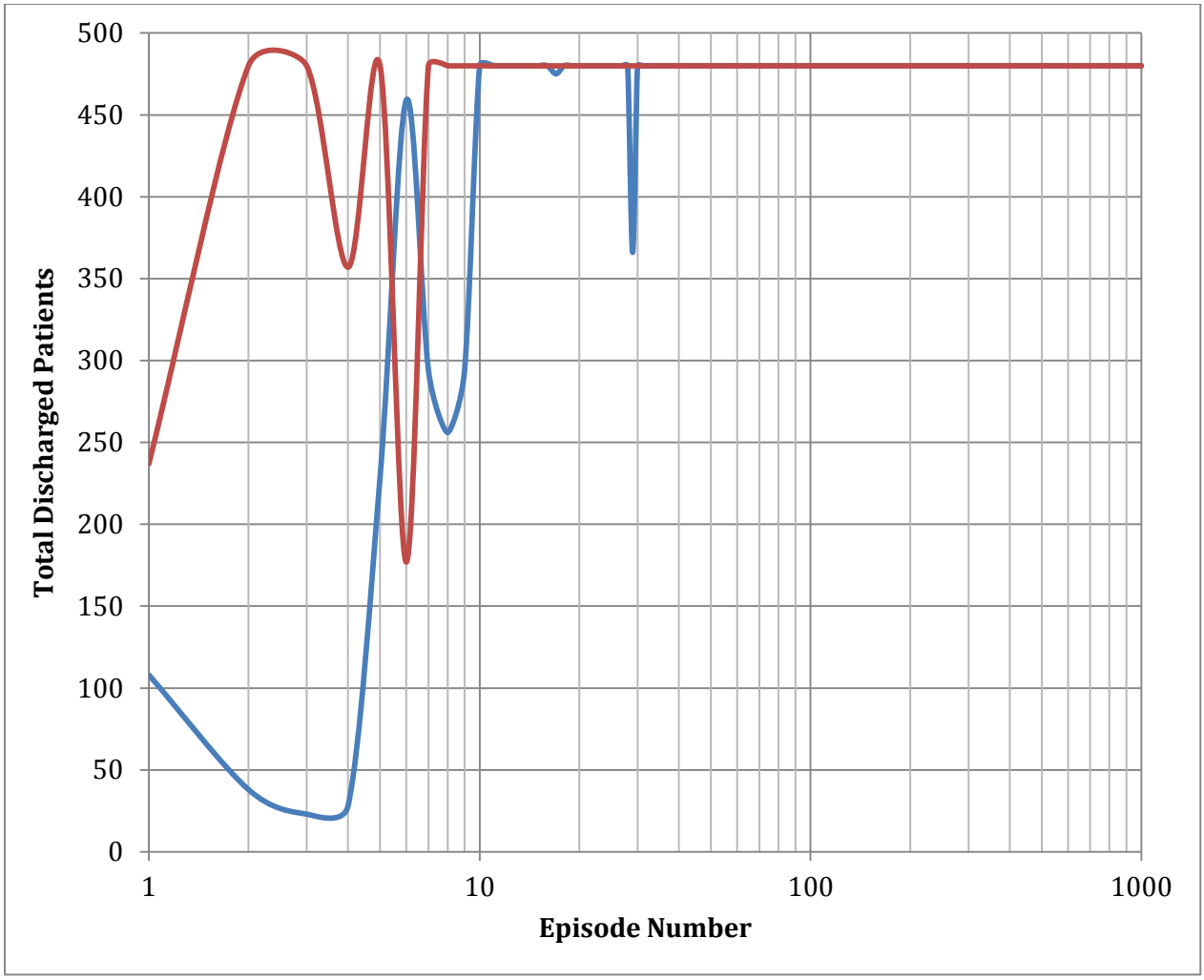

Figure 6-3 Numbers of discharged patients from both emergency units for two consecutive runs

Here, the explanation of the performed action that was chosen by i2Sim-UOP in both runs is presented. This will help explain how the optimum trajectories were discovered and why. For the first run, the agent observed the disturbance in the electrical system and the

impacts of this disturbance on other interconnected systems such as water. The agent behavior during the initial 10 *episodes* was effectively random. As learning proceeds, improvements are observed in the agent's behavior. This behavior tends to appear more stable as the learning capability of the i2Sim-UOP agent converges.

The results show that i2Sim-UOP accurately optimized the number of discharged patients. Notably, the learning was completed in about 40 *seconds*. This is far better than the time that was needed for IDS, which was about 3.6 hours. That is an improvement factor of about 324 times.

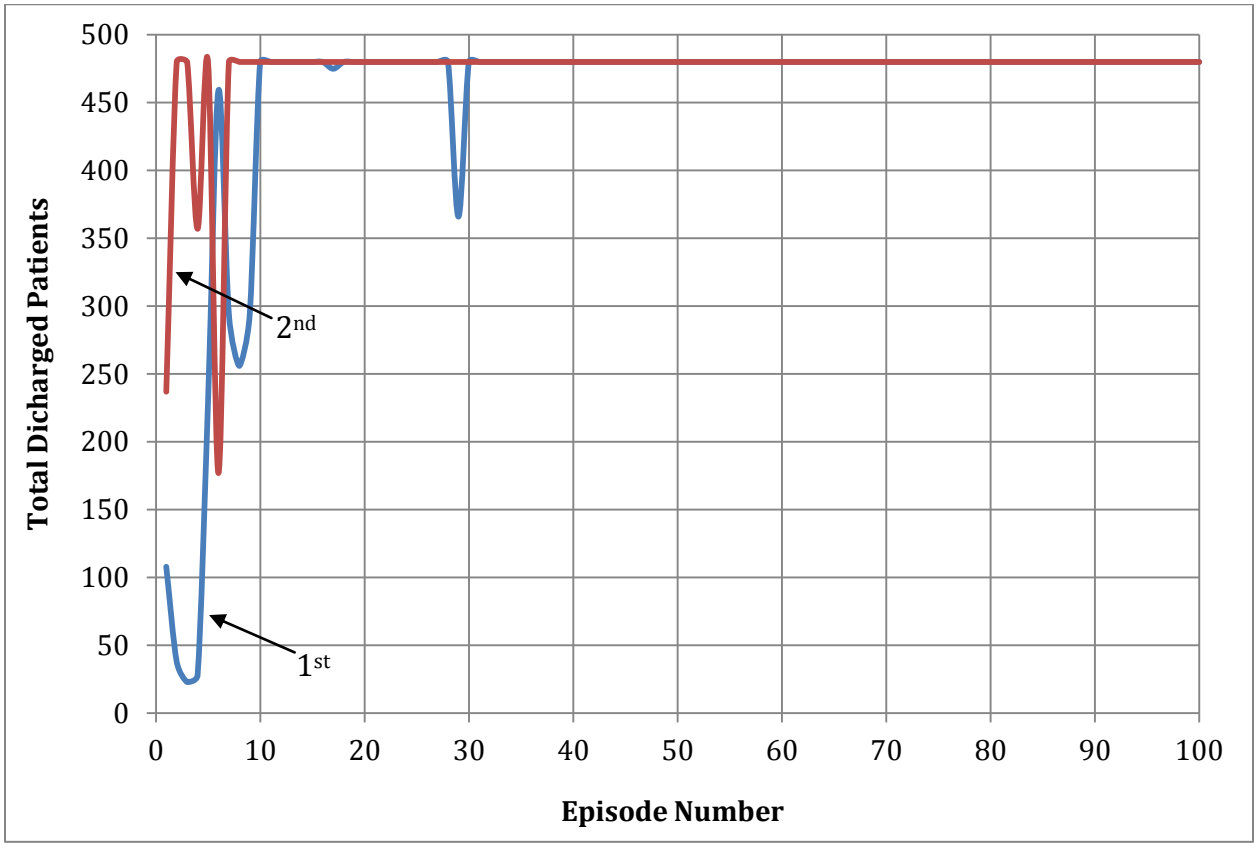

Figure 6-4 First 100 episodes of the both modeled runs

A closer look at the chosen trajectories reveals the followings. The i2Sim-UOP agent was able to optimize the system outcome in less than a minute. We note that the agent exhibited two different types of behaviors. From Table 6-1, it is observed that the Reinforcement Learning (RL) agent performed the actions that deliver the maximum available power to the other interconnected CI's.

After a series of episodes, the RL agent realized that more patients were being directed to VGH. That is why more resources were assigned to it over SPH. This was not the case in Table 6-2, where the agent decided to favor one over the other in every time step. That is both executed actions were applied evenly (excluding exploratory actions) throughout the episode. The action that was chosen first (Table 6-2, a) favored VGH. This is seen in the amount of power and water that were supplied to VGH. On the second time step (Table 6-2, b) the agent diverted the resources to favor SPH by sending less resources to VGH. In this time step the agent correctly realized that the SPH patient's discharged rate would be improved only if the RM of the hospital gets improved. That is why the agent distributed more power to the water pumping station as well as more water to SPH.

Table 6-1 Action corresponds to the most visited state-action pair of the last episode in the 1st run

DGR Distribution Table			
Power Action #	BC	SPH	Terminated
1	2.98%	2.98%	94.05%

Water Distribution Table		
Water Action #	VGH	SPH
5	60%	40%

MUR Distribution Table					
Power Action #	VGH	Water Station	GM	DGR	Terminated
2	1.12%	1.82%	2.1%	57.34%	37.62%

Table 6-2 (a) Action corresponds to the most visited state-action pair of the last episode in the 2nd run

DGR Distribution Table			
Power Action #	BC	SPH	Terminated
1	2.98%	2.98%	94.05%

Water Distribution Table		
Water Action #	VGH	SPH
9	20%	80%

MUR Distribution Table					
Power Action #	VGH	Water Station	GM	DGR	Terminated
3	1.12%	1.28%	2.1%	57.34%	38.16%

Table 6-2 (b) Action corresponds to the most visited state-action pair of the last episode in the 2nd run

DGR Distribution Table			
Power Action #	BC	SPH	Terminated
1	2.98%	2.98%	94.05%

Water Distribution Table		
Water Action #	VGH	SPH
5	60%	40%

MUR Distribution Table					
Power Action #	VGH	Water Station	GM	DGR	Terminated
5	0.84%	1.82%	2.1%	57.34%	37.80%

Our observations are consistent with the expectation that simulation and learning time will be substantially improved over IDS. i2Sim-UOP is able to perform the same optimization approximately two orders of magnitude faster, as seen in Figure 6-5. In this figure, it is clear that i2Sim-UOP was able to implement the learning of a Look Up Table (LUT) matching the same size of the LUT in IDS in 4 seconds. Compare this to 3.259 *minutes* for IDS. This makes i2Sim-UOP about 49 *times* faster than IDS. With i2Sim-UOP we are able to trade–off time against complexity if so desired.

Again, from Figure 6-5, a comparison of performance between the different learning systems is presented. In our initial attempts that involved MATLAB and DAARTS, the overall capacity of these systems in terms of LUT sizes is small compared to IDS and i2Sim-UOP. By moving towards a single platform software system, the intra-communication bottlenecks present in DAARTS and IDS were eliminated leading to improvements in learning speed. Here, i2Sim-UOP was able to optimize the system objective function (maximizing the total number of discharged patients) in just a few minutes. These results are very encouraging in terms of i2Sim-UOP as a tool for use by human emergency responders since the ability to rapidly simulate a developing scenario is crucial in mitigating death and disaster.

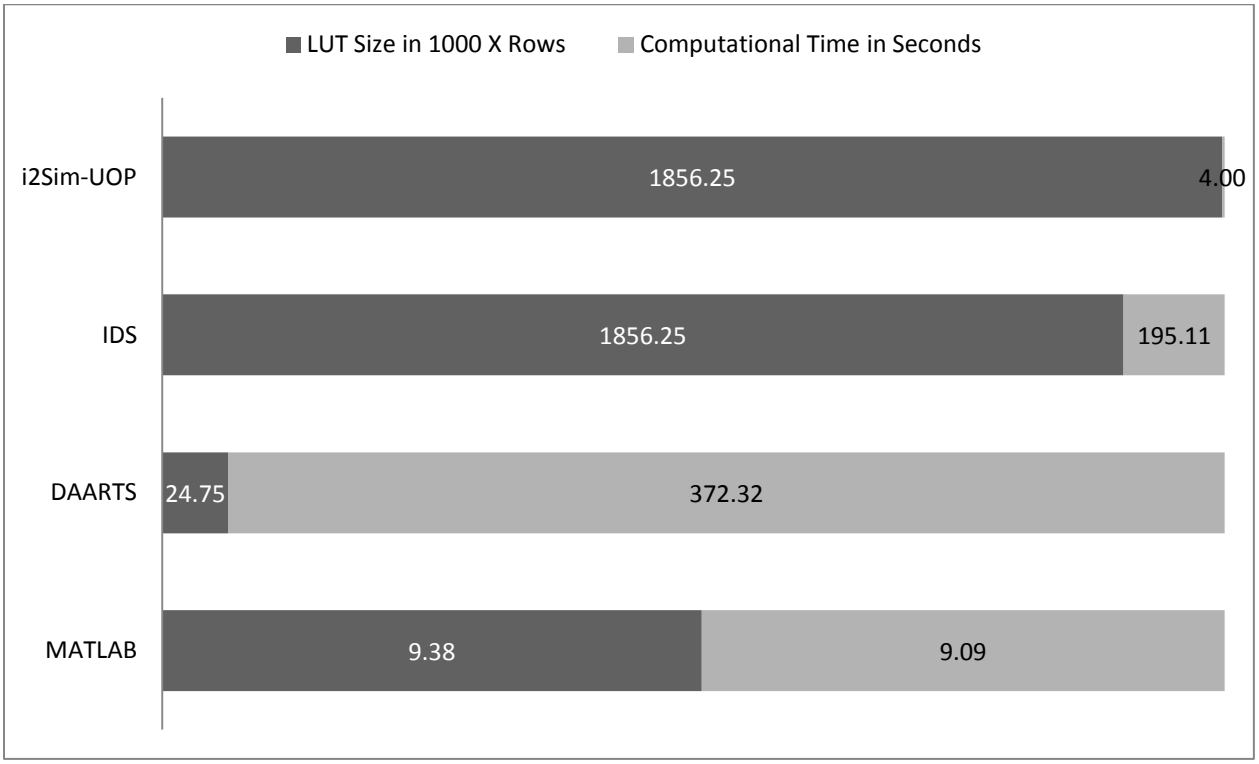

Figure 6-5 RL platforms comparison

CHAPTER 7: CONCLUSION AND SUGGESTIONS

In this thesis we presented an innovative modeling and analysis framework to study the impact of a natural or human-made disaster on a complex interconnected critical infrastructures (CI's) system and its associated resources. The framework is based on i2Sim and Reinforcement Learning (RL) using two different policy estimators, Temporal Difference (TD) and Monte Carlo (MC). i2Sim permitted the simulation of complex interconnected CI's, where RL enabled us to learn by experience how to best reallocate the available limited resources during a catastrophic event. It is expected that the decisions taken by the agent are consistent with those that would optimize the system output, which in this case is the total number of discharged patients from medical units. The primary research contributions made in this thesis are summarized in the next section. The subsequent sections discuss the limitations of the current work, and indicate possible improvements that could be made in future work.

7.1 Primary Contribution

A Reinforcement Learning (RL) agent was implemented within the i2Sim (infrastructure interdependency simulator) framework to experience and propose the best actions to undertake for a hypothetical disruption of the system of critical infrastructures (CI's). In the demonstration, we used a realistic model that interconnects a number of CI's to allow the RL agent to interact intelligently with. The experimental results revealed that the

proposed framework can reduce the devastating impact of disaster disruption. The loosely coupled nature of RL enables its application to a wide variety of optimization scenarios.

In our first model, a learning agent called DAARTS using RL-TD was implemented to interact with the i2Sim framework to propose the best actions to perform for a simulated disruption of the CI's. This agent, when asked to interact with a simulated environment called Downtown Vancouver, was able to maximize the system outcome but within a computational time that was far from real-time.

In the second model, an intelligent disaster management system, called IDS, trained using RL-MC was applied to a simulation of a natural disaster. In this study, both agents (IDS and DAARTS) were compared. In particular, the study highlighted the difference between policy evaluation performed in an incremental episode-by-episode manner, typical of Monte Carlo (MC) methods, and that performed by Temporal Difference (TD) methods that learn step-by-step. This learning system when used in conjunction with i2Sim was shown to provide effective informed decisions faster than was possible with DAARTS. Further the MC based agent improved communication efficiency between the agent and the simulator and achieved a doubling in computational performance.

In our last model, an initiative toward a faster intelligent disaster management system using RL was proposed. The resource allocation optimization problem within a system of interdependent CI's using MC was implemented within a single software platform called i2Sim-UOP. Minimal computational latency was observed within i2Sim-UOP due to the

complete elimination of intra-platform communications. The efficiency and capability of this system was demonstrated and compared to the previous learning systems, DAARTS and IDS (both modeled using a separate interface platform). i2Sim-UOP was able to mitigate casualties with a computational speed of about 93 times faster than DAARTS, and 49 times faster than IDS. The i2Sim-UOP platform timings are suitable for real time within the complexity of the simulated scenarios.

7.2 Limitations and Suggested Future Research

The overall goal of this research was to investigate ways to provide decision assistance, based upon learning from simulation, to an emergency response manager as fast as possible. Reinforcement Learning (RL) applications using discrete representations of an $action - value\ function$ become rapidly limited in their ability to model environments of increasing complexity. This is commonly known as the "curse of dimensionality" and is best addressed by restricting the number of modeled parameters. Strategies to address this problem include partitioning a problem having a large number of parameters to multiple problem of smaller complexity [45]. Pre-training off-line of a complex problem can also help to minimize the computational time taken for on-line learning.

We demonstrated how learning performance and complexity could be addressed by the use of appropriate RL learning techniques, which can help to mitigate communication inefficiencies in multi-platform systems. Further improvements were obtained by re-architecting the platform to remove the inherent bottlenecks. These were achieved while

still using Look Up Table (LUT) based utility functions. Ultimately, to support increasingly real world scenarios, LUT based approaches can become impractical and must be replaced in favor of function approximation techniques. Thus, it is necessary to study and analyze the computational variations of the learning system. A speed versus accuracy trade-off exists between approaches that use a conventional implementation of the action-value function (LUT) and other ways, which use function approximation techniques. Another approach to solve more complex scenarios is using computer parallelization techniques. With a Personal Computer (PC) clusters parallel scenarios can be run simultaneously in multiple threads. Given the available computational power of PC clusters (the cluster in our research lab has 336 processors) this should also be a line of future research.

REFERENCES

[1] F. J. Daniel, “India power cut hits millions, among world's worst outages,” REUTERS, 2012.

[2] David Rubens Association, “Great Eastern Japan Earthquake,” Disaster Response Management, London, 2011.

[3] K. M. Kowalski-Trakofler, C. Vaught and T. Scharf, “Judgment and decision making under stress: an overview for emergency managers,” *International Journal of Emergency Management,* vol. 1, no. 3, pp. 278-289, 2003.

[4] H. Shen and J. Zhao, “A Quick Group Decision-making Planning,” *IEEEXplore,* 2007.

[5] G. G. Brown and A. L. Vassiliou, “Optimizing disaster relief: real-time operational and tactical decision support,” *Naval Research Logistics,* vol. 40, no. 1, pp. 1-23, 1993.

[6] F. Fiedrich, F. Gehbauer and U. Rickers, “Optimized resource allocation for emergency response after earthquake disasters,” *Safety Science,* vol. 35, no. 1, pp. 41-57, 2000.

[7] J.-B. Sheu, Y.-H. Chen and L. W. Lan, “A novel model for quick response to disaster relief distribution,” in *Proceedings of the Eastern Asia Society Transportation Studies*, 2005.

[8] J.-B. Sheu, “An emergency logistics distribution approach for quick response to urgent relief demand in disasters,” *Science Direct-Transportation Research,* vol. 43, no. 6, pp. 687-709, 2007.

[9] Y. Huang, Y. Fan and R. L. Cheu, “Optimal allocation of multiple emergency service resources for critical transportation infrastructure protection,” in *Critical Transportation Infrastructure Protection Committee (ABE40)*, 2006.

[10] C. A. Arboleda, D. M. Abraham, J.-P. P. Richard and R. Lubitz, “Impact of

interdependencies between infrastructure systems in the operation of health care facilities during disaster events," in *Joint International Conference on Computing and Decision Making in Civil and Building Engineering*, Montreal, 2006.

[11] J. Minkel, "The 2003 Northeast Blackout - five years later," Scientific American, 2008.

[12] S. M. Rinaldi, "Modeling and simulating critical infrastructure and their interdependencies," in *37th Hawaii International Conference on System Sciences*, Piscataway, 2004.

[13] S. E. Chang, T. L. McDaniels, J. Mikawoz and K. Peterson, "Infrastructure failure interdependencies in extreme event: power outage consequences in the 1998 Ice Storm," *Natural Hazards,* vol. 41, no. 2, pp. 337-358, 2006.

[14] H.-S. J. Min, W. Beyeler, T. Brown, Y. J. Son and A. T. Jones, "Toward modeling and simulation of critical infrastructure interdependencies," vol. 39, no. 1, pp. 57-71, 2007.

[15] G. O'Reilly, H. Uzunalioglu, S. Conrard and W. Beyeler, "Inter-infrastructure simulations across telecom, power and emergency services," Design of Reliable Communication Networks-5th International Workshop, 2005.

[16] C. A. Arboleda, D. M. Abraham and R. Lubitz, "Simulation as a tool to assess the vulnerability of the operation of a health care facility," *Journal of Performance Constructed Facilities,* vol. 21, no. 4, pp. 302-312, 2007.

[17] C. A. Arboleda, D. M. Abraham, J.-P. P. Richard and R. Lubitz, "Vulnerability assessment of health care facilities during disaster events," *Journal of infrastructure system,* vol. 15, no. 3, pp. 149-161, 2009.

[18] J. Marti, C. Ventura, J. Hollman, K. Srivastava and H. Juarez, "I2Sim modeling and simulation framework for scenario development, training, and real-time decision support of multiple interdependent critical infrastructures during large emergencies,"

in *How is modeling and Simulation Meeting the Defense Challenges out to 2015?*, Vancouver, Canada, 2008b.

[19] B. Eric, "Agent-based modeling: Methods and techniques for simulating human systems," *Proceedings of the National Academy of Science of the United States of America,* vol. 99, no. 3, pp. 7280-7287, 2002.

[20] D. Thapa, I.-S. Jung and G.-N. Wang, "Agent based decision support system using reinforcement learning under emergency circumstances," Springer Lecture Notes in Computer Science, Changsha, 2005.

[21] I. Eusgeld, C. Nan and S. Dietz, "System-of-system approach for interdependent critical infrastructures," *Reliability Engineering and System Safety,* vol. 96, no. 6, pp. 679-686, 2011.

[22] G. M. Atanasiu and F. Leon, "Agent-based risk assessment and mitigation for urban public infrastructure," *Forensic Engineering 2012,* pp. 418-427, 2013.

[23] S. Wu, L. Shuman, B. Bidanda, M. Kelley, K. Sochats and C. Balaban, "Agent-based discrete Event simulation modeling for disaster responses," in *The Proceeding of the 2008 Industrial Engineering*, 2008.

[24] S. E. Chang, "Urban disaster recovery: a measurement framework and its application to the 1995 Kobe earthquake," *Disasters,* vol. 34, no. 2, pp. 303-327, 2010.

[25] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning), Library of Congress Cataloging-in-Publication Data, USA, 1998.

[26] M. T. Khouj, C. Lopez, S. Sarkaria and J. Marti, "Disaster management in real time simulation using machine learning," in *IEEE Canadian Conference of Electrical and*

Computer Engineering, Niagara Falls, Ontario, Canada, 2011.

[27] J. R. Marti, J. A. Hollman, C. E. Ventura and J. Jatskevich, "Dynamic recovery of critical infrastructures," *International Journal of Critical Infrastructures,* vol. 4, no. 1/2, pp. 17-31, 2008a.

[28] B. Y. Calida and P. F. Katina, "Regional industries as critical infrastructures: a tale of two modern cities," *International Journal of Critical Infrastructures,* vol. 8, no. 1, pp. 74-92, 2012.

[29] J. R. Marti, "Multisystem simulation: analysis of critical infrastructures for disaster response," in *Networks of networks: the last frontier of complexity*, G. D'Agostino and A. Scala, Eds., Heidelberg, Springer, 2014, pp. 255-277.

[30] P. Simon, Too big to ignore: the business case for big data, Hoboken: John Wiley and Sons, Inc. , 2013.

[31] K. Murphy, Machine learning: a probabilistic perspective (adaptive computation and machine learning), Massachusetts Institute of Technology, 2012.

[32] S. Sarkaria, "Reinforcement Learning," September 2013. [Online]. Available: https://courses.ece.ubc.ca/592/. [Accessed 8 October 2013].

[33] S. Nissen, *Large scale reinforcement learning using Q-SARSA(λ) and cascading neural networks,* Department of Computer Science, University of Copenhagen, 2007.

[34] G. Tesauro, "Practical issues in temporal difference learning," *Machine Learning 8,* pp. 257-277, 1992.

[35] D. O'Neil, M. Levorato, A. Goldsmith and U. Mitra, "Residential demand response using reinforcement learning," in *IEEE International Conference on Smart Grid*

Communications, Gaithersburg, Maryland, USA, 2010.

[36] T. Yu, J. Liu, K. W. Chan and J. J. Wang, "Distributed multi-step Q(λ) learning for optimal flow of large-scale power grids," *International Journal of Electrical Power and Energy Systems,* vol. 42, no. 1, pp. 614-620, 2012.

[37] M. Wiering and M. Dorigo, "Learning to control forest fires," in *Proceedings of the 12th International Symposium on Computer Science for Environmental Protection*, 1998.

[38] A. Abdolmaleki, M. Movahedi, S. Salehi, N. Lau and L. P. Reis, "A reinforcement learning based method for optimizing the process of decision making in fire bridge agents," in *Progress in Artificial Intelligence, 15th Portuguese Conference on Artificial Intelligence*, Lisbon, 2011.

[39] I. Martinez, D. Ojeda and E. Zamora, "Ambulance decision support using evolutionary reinforcement learning in robocup rescue simulation league," in *RoboCup: Robot Soccer World Cup X*, 2006.

[40] Z. P. Su, J. G. Jiang, C. Y. Liang and G. F. Zhang, "Path selection in disaster response management based on Q-learning," *International Journal of Automation and Computing,* vol. 8, no. 1, pp. 100-106, 2011.

[41] "Entire Downtown, Westend, Coal Harbour Map from 2002 Map Book," Vancouver, 2002.

[42] M. T. Khouj and J. Marti, "Modeling critical infrastructure interdependencies in support of the security operations for the Vancouver 2010 Olympics," Technical, Defense R&D Canada-CORA, Vancouver, 2010.

[43] M. T. Khouj, S. Sarkaria and J. Marti, "Decision assistance agent in real-time simulation," *International Journal of Critical Infrastructures,* 2012.

[44] S. J. Chapman, MATLAB programming for engineers, fourth edition, Toronto: Chris Carson, 2008.

[45] H. Abdur Rahman, M. Armstrong, D. Mao and J. Marti, "i2Sim: a matrix-partition based framework for critical infrastructure Interdependencies simulation," in *Electric Power Conference, 2008. EPEC 2008. IEEE Canada*, 2008.

[46] D. Thapa, I. S. Jung and G. N. Wang, “Agent based decision support system using reinforcement learning under emergency circumstances,” Springer lecture notes in Computer Science (LNCS), Changsha, China, 2005.

[47] H. Shen and J. Zhao, "A Quick Group Decision-making Planning Method for Emergency Response," in *Intelligence and Security Informatics*, 2007.

Appendices

Appendix A: i2Sim, Downtown Vancouver Model, Physical Layers [41]

What follows is an example of the physical layers that are involved within an I2Sim model. This particular example reflects the model developed for an investigation into affecting patients' discharge times from hospitals during a catastrophic event.

Hospitals

The Hospital model used in this scenario focuses solely on the ability to receive and process trauma victims during an emergency. Of course there are many other functions provided by hospitals, even in relation to emergency scenarios, but for the purposes of this demonstration, the primary focus is the ability to provide treatment to trauma victims exiting venues. For this reason, the functionality of the hospital model was simplified to: receiving trauma victims, housing them until they can be treated, treating them, and releasing them.

The main component of the hospital model is the production cell that determines the rate at which patients can be treated. This production cell takes as inputs electricity, water, natural gas, and medical gasses. Based upon the availability of these resources and the state of damage of building and lifelines (Physical Mode) which was also determined by a source cell external to the production cell, the rate at which patients can leave the waiting area to be treated is determined. These inputs the two hospitals considered (Vancouver General Hospital and St. Paul's Hospital), except that for St. Paul's the Natural Gas input is replaced

by an input called Steam. This is because St. Paul's is located in Downtown Vancouver, where steam heating is provided directly from a centralized heat distribution.

The electricity input includes an aggregator that combines inputs from the appropriate substations. The water to each hospital is provided from the Water Pumping Station described above. All other resources (Natural Gas, Steam, and Medical Gasses) are provided by source cells that are unique to each hospital and are used to represent a constant supply that can be adjusted by the user.

The output of the hospital production cell is a limiting rate at which patients can be treated, in terms of patients per minute. This output is used to determine the maximum output rate for the waiting area of the hospitals, which is represented by a storage cell. The input of patients to this cell comes from the city transportation system, which is modeled as a simple time delay. As patients are output from the "Waiting Area" cell, they encounter a delay of 30 minutes, which is assumed to be the average time required to stabilize a trauma victim. The final step is for the patients to enter a storage cell that is used only to count the total number of patients released from the hospital during a given run of the model. The modeled hospitals are described next.

Vancouver General Hospital (VGH)

Vancouver General is the primary trauma receiving hospital in Vancouver. Information obtained from conversations with hospital operators, employees, and administrators have provided the assumptions used regarding the resources required to maintain operations in

the emergency rooms, and the maximum rate at which trauma victims can be treated (10 patients per hour on the average). These requirements and capabilities for Vancouver General can be seen in the human readable table (HRT) of Table 2-1.

It is worth noting that the Medical Gasses resource is measured in percentage of required gasses, rather than in any physical unit (liters or bottles, for example). This unusual case is the result of there not being reliable estimates available to the modelers on the exact physical amount required of these gasses. As a result, it was simply represented in terms of the percentage required for optimal operation.

St. Paul's Hospital (SPH)

Based upon information provided by employees at St. Paul's and Vancouver General Hospitals, it was determined that the required inputs for St. Paul's are similar to those of Vancouver General, the difference being the use of Steam rather than Natural Gas to provide heat. As mentioned above, under normal conditions, Vancouver General is the primary trauma receiving hospital, but due to the emergency conditions, it was assumed that St. Paul's would transfer sufficient resources to their emergency room to be able to temporarily treat patients at the same rate as Vancouver General. The requirements and capabilities of St. Paul's hospital are shown in the HRT Table 2-2.

Electrical Substations

The scenario model includes a total of four electrical substations of the BC Hydro Vancouver Downtown electrical distribution system: Cathedral Square (CSQ), Dal Grauer

(DGR), Sperling (SPG), and Murrin (MUR). Each of these operates independently of one another, except for Dal Grauer, which is fed by Murrin. The information on how these substations are fed, the amount of power they transfer, and what other infrastructures are supported by them was obtained, at the time the case was created, from the experience of UBC professors and students along with their interactions with BC Hydro and other organizations.

As shown hereafter, each substation in the model is represented by an individual block. This block is a production cell containing an HRT representing the operation of the substation transformers based upon the input from feeders connecting the substation to upstream power generation facilities (or in the case of Dal Grauer, to another substation). Attached to these cells there are Distributors that use Distribution Tables to determine how the power from the substation can be allocated through channels among the various users in the model.

Cathedral Square (CSQ)

The Cathedral square substation is located underground in Downtown Vancouver near the intersection of Dunsmuir St. and Cambie St. There are three input feeders coming into Cathedral Square.

According to publicly available information, the Cathedral Square Substation contains two transformers, each rated at 150 MW, for a total nominal rating of 300 MW. This rating is reflected in the HRT shown in the table below:

	Output	Input
RM	Electricity [MW]	Electricity [MW]
1	300	300
2	200	200
3	100	100
4	0	0

As for all substations, for each RM, the input is exactly equal to the output. Issues such as losses within transformers, etc. are not dealt with in this model.

As shown in the i2Sim model (Figure 4-4), GM Place receives power from the Cathedral Square Substation.

Dal Grauer Substation (GDR)

The Dal Grauer substation is located on Burrard Street in Downtown Vancouver. In the scenario considered it provides power to BC Place and to St. Paul's Hospital. As previously mentioned, Dal Grauer is unique among the substations in this model in that its only supply of power comes from another substation in the model (Murrin).

According to publicly available information, Dal Grauer Substation now contains four transformers, each rated at 84 MW, for a total nominal rating of 336 MW. This rating is reflected in the HRT shown in Table 2-4.

As previously mentioned, two loads within the model receive power from Dal Grauer Substation: BC Place and St. Paul's Hospital. The amount of power assigned for each of them is defined in Table 2-6.

Sperling Substation (SPG)

The Sperling Substation is located in Greater Vancouver near the intersection of Arbutus Street and Kingsway. Like Cathedral Square, it is fed by three feeders that provide power from upstream sources not modeled in this scenario.

According to publicly available information, the Sperling substation contains two transformers, each rated at 168 MW, for a total nominal rating of 336 MW. This rating is reflected in the HRT can seen below:

	Output	Input
RM	Electricity	Electricity
1	336	336
2	210	210
3	105	105
4	0	0

As shown in the modeled system, Vancouver General Hospital (VGH) is supplied from the Sperling Substation.

Murrin Substation (MUR)

The Murrin substation is located just to the East of Downtown Vancouver, near the intersection of Quebec St. and Union St. According to publicly available information, the Murrin Substation contains seven transformers: three rated at 84 MW, two rated at 8 MW, one rated at 150 MW, and one rated at 168 MW, for a total nominal rating of 586 MW. This rating is reflected in the HRT in Table 2-3.

The Murrin Substation provides power to four blocks within the model. These include Vancovuer General, GM Place, the Water Pumping Station, and Dal Grauer substation. The amount of power assigned to each of them is defined in Table 2-5.

The specific values here were selected based upon energy usage data received from engineers on site at the venues, or from assumptions made by the modelers. Note that both Murrin and Cathedral Square Substations provide power to GM Place. This is part of the redundancy scheme put in place by BC Hydro to avoid loss of power.

Water Pumping Station

The Water Pumping Station in the model provides water to the two primary hospitals: St. Paul's and Vancouver General. The amount of water that is actually put out by the pumping station to all users in the city was unavailable, so as a simplifying assumption, the HRT only shows the amount required for these two hospitals. The water requirements for the hospitals ($51\ KL/hour$) are based upon estimates provided by engineers at the hospitals.

The central block of the Water Pumping Station model is a production cell that takes in electricity from Murrin Substation and water supply from a source block to produce water pumped to the downstream users.

GM Place (GM)

GM Place is an indoor sports arena located on Griffiths Way in Downtown Vancouver. The $44,129\ m^2$ area is surrounded by 10 main gates and can host 20,000 $fans$.

BC Place

BC Place is a multipurpose stadium and Canada's first domed arena. BC Place is the largest air-supported stadium in the world. It is located on the north side of False Creek and is owned and operated by BC Pavilion Corporation, a Crown Corporation in the Province of British Columbia.

The 65,032 m^2 area can host up to 60,000 *people* has been home to the CFL's BC Lions since 1983.

Appendix B: Sensitivity Analysis for DAARTS Learning Parameters (RL-TD) [26]