FORMULATION AND APPROXIMATE SOLUTIONS FOR PLANNING PROBLEMS WHICH CAN BE DECOUPLED

A Dissertation

by

DIPTANIL CHAUDHURI

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Dylan A. Shell
Committee Members,	Jason M. O'Kane
	Guni Sharon
	Suman Chakravorty
Head of Department,	Scott Schaefer

August 2023

Major Subject: Computer Science

ABSTRACT

Planning — developing strategies that an autonomous agent can use to achieve some desired goals — is a critical sub-part of the Artificial Intelligence literature. In general, planning problems involve single or multiple autonomous agents performing a series of actions that influence the agents' environment to bring desired changes that help progress toward the agents' goals. The computational challenge of identifying the progress-bringing actions depends upon the size of the planning problem of interest. This dissertation deviates from this traditional view of the agent-environment interaction and studies planning problems where components of the problem formulation exist that have minimal or no causal influence from the agent(s) actions. Such planning problems of this form have opportunities to realize computational efficiencies via decoupling.

At first glance, the premise of weak causation in actions might appear antithetical to the very idea of what plans (and actions) are for. After all, what will be the point of planning if actions do not cause changes in the environment? In fact, such situations often arise in existing problems and situations that model real-world scenarios, such as automated narrative generation, event narration, active perception, and situation depiction, to name a few. When causal influence is minor, traditional approaches that do not embrace this fact will treat the problem as a single component; this is a missed source of potential computational savings. Thus, approximate solutions are needed that consider each component separately. The approach taken here formulates the problem in a manner that essentially respects separate components, some under the agent's influence while others are not. The research then utilizes the intrinsic properties of formulation to decouple the solution and, thereby, present efficient approximate solution techniques.

The research begins by tackling problems involving a single component of the environment uninfluenced by the agent's action while having other elements that the agent has control over. We do this through example scenarios involving generation of structured narratives. First, we study the structured videography narrative capture problem where an autonomous videographer robot is tasked with observing its environment and later selectively summarizing what it saw as

a vivid structured narrative. Second, we investigate the multi-agent version of the same, where a team of agents must coordinate to record events that fit some narrative structure. Next, we study problems involving multiple components that are not influenced by the agent's action, and we do so in the guise of a scenario from the multi-commodity logistics problems. It involves an autonomous operations agent responsible for routing multiple commodities within a logistic network comprising multiple retail and storage units.

The three problem scenarios allow different opportunities for decoupling that this dissertation takes advantage of to provide efficient approximate solutions. For the structured videography narrative capture problem, the decoupling is in the form of separating the substance "what to capture?" from the style "how to capture?" allowing for an approximate solution. The multi-agent variation of the same decouples an agent's decisions from the others. And lastly, the multi-commodity logistic scenario is solved by decoupling the analysis of parallel aspects of consumption.

For the problem scenarios mentioned above, the research conducts a thorough comparison of the various approaches to decoupling with the traditional solution. While the traditional approach — solving the problem jointly by searching over the state space and action space formed by the product of all the components — quickly becomes intractable, even with a few components. The approximate algorithms via decoupling are efficient and tractable and, in some specific cases, superior to the traditional approach. A multi-robot implementation of the structured videography narrative capture is also provided, showing the feasibility of the approximate solution approaches in the real world.

DEDICATION

To my mother and father.

ACKNOWLEDGMENTS

My six years as a Ph.D. student at Texas A&M University were full of ups and downs. It is with the constant support from everyone around me that I was able to complete this journey.

First, I would like to thank my advisor, Dr. Dylan Shell, whose unwavering support and help made this dissertation possible. As an advisor, he was always there as my most prominent critic and my greatest supporter. I learned something new from him whenever we met. Working with him throughout these years allowed me to develop academic skills holistically.

I thank my committee members, Dr. Jason O'Kane, Dr. Guni Sharon, and Dr. Suman Chakravorty, for their helpful instructions and constructive criticism.

I would also like to thank my former committee members, Dr. Korok Ray, Dr. Dilma Da'Silva, and Dr. James Caverlee, for their insightful comments on my proposal document. I want to thank Dr. Korok Ray for funding the first two years of my Ph.D. and allowing me to work with him at the Mays Innovation Research Center.

I am also thankful to Dr. Jason O'Kane and Dr. Hazhar Rahmani from the University of South Carolina and Dr. Aaron T. Becker, Rhema Ike, and Omar Romero from the University of Houston for their contribution toward completing my dissertation.

I am grateful for having such amazing lab mates (Reza Oftadeh, Reza Hosseini Teshnizi, Yulin Zhang, Ya-Chun Hsu, and Grace McFassel) at Distributed A.I. Robotics Lab. I want to especially thank Reza Oftadah and Reza Teshnizi for always being there whenever I needed them and for all the fun memories they helped build over the years.

I am also thankful for the blessings of my parents and grandparents, which made it possible to reach where I am today. I want to thank my uncle, Atanu Bhattacharya, who helped me and guided me throughout my stay here in the United States.

Lastly, I am grateful to Shivangi Dwivedi, who has always been there for me and is my unwavering supporter.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Dylan A. Shell [advisor], Professor Jason M. O'Kane, and Professor Guni Sharon of the Department of Computer Science and Engineering and Professor Suman Chakravorty of the Department of Aerospace Engineering.

The hardware implementation for Chapter 4 was provided by Dr. Aaron T. Becker, Rhema Ike and Omar Romero of the University of Houston, in coordination with Dr. Jason M. O'Kane of Texas A & M University, and Dr. Hazhar Rahmani of the University of South Carolina.

Funding Sources

Graduate study was supported by the National Science Foundation (NSF) through grants IIS-1849303, IIS-1849249, and IIS-1849291 and the Mays Innovation Research Center.

NOMENCLATURE

A2C	Advantage Actor-Critic
DFA	Deterministic Finite Automaton
EGMC	Element-generating Markov chain
FMA	Fundamental Matrix Analysis
GPS	Global Positioning System
JA	Joint Action Automaton
JPG	World Story Joint Progress Graph
LWD	Logistics with Demand
MDP	Markov Decision Process
MRRCM	Multi-Robot Recording Cost Minimization
MRSVC	Multi-Robot Styled Video Capture
NFA	Nondeterministic Finite Automaton
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
SPA	Style Planning Automaton
SVC	Styled Video Capture
TEG	Time/Event Graph
UAV	Unmanned Aerial Vehicle
VA	Valid-action Automaton

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Planning — devising a plan of action to achieve some desired goals — is one of the fundamental problems in robotics literature. In the generic form, planning problems assume an initial description of an environment, a description of a goal, and a possible set of actions along with their influence on the environment. The objective, then, is to generate a plan that can achieve the desired goal. Researchers have studied planning problems extensively, and there exist numerous variations. The complexity of the planning problem and their solution approaches vary vastly depending on how the problem is specified: (a) the specification of the environment differs by the level of observability (fully observable to hidden) and even the state space on which the environment is defined (discrete to continuous); (b) the influence of the actions on the environment might be deterministic or even stochastic; (c) the problem might involve a single agent or multiple agents working cooperatively/competitively (see [1]; Chapter 2). However, a common assumption among the different formulations is the ability of the agent(s) actions to induce a direct influence over the environment. Though this assumption seems to be the basis of any plan, in real-world scenarios, planning may also be needed where the agent has no or minimal direct influence over the environment.

To understand how and where agents have no or minimal influence, consider the following two scenarios. A grains wholesale company tasks an autonomous agent with transporting rice and wheat between its storehouses and retail stores. The agent's objective is to sell all the items in the storehouses as soon as possible. To do so successfully, the agent needs to predict the consumers' future demands based on current market research and take routing actions preemptively. The items will only sell if there is a demand for the item at the retail store and the item is available. For the second scenario, imagine a videographer robot, tasked with producing videos from events that occur within a wedding reception. The wedding reception might host several events, both predictable and unpredictable, such as (a) dancing, (b) drinking, (c) vows, and even (d) a drunken ruckus. Not all events are likely to happen, but the videographer agent needs to predict what might

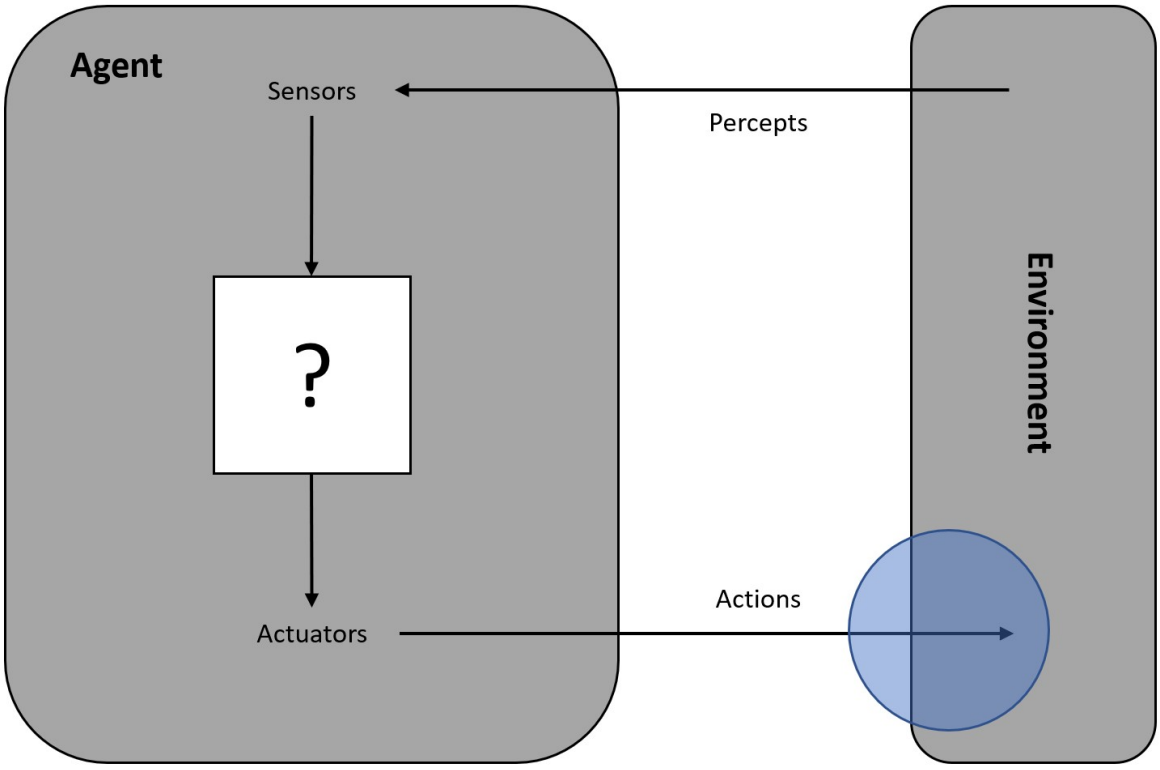


Figure 1.1: An AI agent interacts with its environment by perceiving inputs through sensors and taking actions via actuators (adapted from [1]; Chapter 2). This dissertation focuses on the interaction between the agent's action and the environment (shown in the figure via the blue circle).

happen and position itself preemptively to capture the videos to be successful.

Though the examples mentioned above might at first seem completely unrelated at first glance, there is a common theme that is present in both. In both problems, there exist aspects in the agent's environment that are not under the influence of the agent's action. In the first scenario, the agent can only transport the items within the storehouses and retail stores, with consumer behavior oftentimes driven entirely by external factors, with the dynamics being uninfluenced or only weakly affected by happenings within the network. For the second scenario, even though the agent's objective, recording a video narration, is under the agent's own influence — it decides which events needs to be recorded — it does not influence the guests. And therefore, when and where each event will occur. In both cases, the agent needs to anticipate the behavior in its surroundings and take actions preemptively to achieve its goal successfully.

The research presented in this dissertation studies planning problems where there exists no or minimal causal influence between the agent's actions and its environment. The premise, at first, might appear antithetical; after all, we plan to act — to bring desired changes within the environment to achieve some goal. But as is evident from the examples above, problems often occur in the real world where the agent cannot influence aspects of its environment. In this dissertation, instead of assuming a direct causal influence of the agent's action over the whole environment, the research studies the overarching question, "When an agent's actions have a weak influence on environment dynamics, how can this be exploited?"

The research explores the previous question by providing specific problems involving structured narrative generation and logistics. We provide formulations that allow these planning problems to be defined via modular components and then utilize the intrinsic properties of the formulation to present decoupled approaches to solving the problem efficiently. The decoupling allows the problems to be broken down into smaller sub-problems, which we then solve by formulating them as Markov Decision Processes (MDP) [5].

1.1 Research objective and approach

In this dissertation, we aim to formulate planning problems with components in the problem formulation that are not under its direct influence yet still having some temporal or state-based structure. This allows the approach to be applied to circumstances where there is structured noise, and disturbances, but also opportunities and other desirable elements. The research then aims to utilize the intrinsic properties of the problem to solve the problem efficiently. The key factor that the agent needs to deal with in such a problem is not only about managing uncertainty or risk management, but to anticipate in advance the future state of the system and take preemptive actions.

The approach taken in this research takes advantage of the clear separation that is present in these problems to model the problems into two separate components. One component — like the vehicle mechanics or the narrative sequence — under the influence of the agent's action. And the other — like the dynamics of the other pedestrians/cars or the behavior dynamics of the guests in the wedding — not influenced by the agent's action.

A conventional approach to solve the problem would be to produce a Markov Decision Process [5] by taking the product of the two components, resulting in a factored state space with some elements in the state keeping track of the causal part of the system while other components track the non-causal part. Indeed, such a solution would give the optimal result. But, for real-world applications, the state space of the MDP might become large and computing a solution intractable. Therefore, efficient approximate solutions are needed.

This dissertation provides two variants of approximation approaches that can be used to solve the problem. The first approach uses the decomposition of the problem into components to construct smaller MDPs, each with much smaller state space and action space than the product MDP, allowing for tractable solutions that are also computationally efficient. The other approach analyzes each of the components separately, reducing the states of each component which in turn also reduces the state space of the product.

1.2 Chronicle of events

The notion of using autonomous robots to record events, similar to the wedding example provided previously, that obey a narrative structure was first proposed by Shell et al. [6]. The solution to the problem was the foray of the paper [7]. The paper formulates the problem by modeling the stochastic process of event generation by a variant of Markovian structure (Markov chain [8]) and specifying the event sequence of interest as a regular language via a specification automaton (deterministic finite automaton (DFA) [9]), dividing the problem into two components.

The solution approach taken in the paper involves constructing a product between the Markovian structure and the specification automaton, creating a product Markov Decision Process, which can be solved optimally to get the policy for the agent. Since this formulation has only two components, this approach can solve the problem efficiently without the need of any form of approximation.

The research presented in this dissertation expands on the formulation presented in this paper, by gradually adding more components. As more components are added the size of the product between the components also increases, making the product intractable. To tackle this problem, the research develops and utilizes approximate solution techniques to solve them efficiently.

1.3 Stylized video chronicle of events

The first extension of the work studied in [7], that is presented here, involves extending the objective of recording events in the structured narrative by introducing the concept of cinematic styles. The aim of the extension is to increase the realism of the model, as a mere concatenation of event clips gives a rather poor and jiggered result.

To understand the importance of style, consider the following sporting example. Figure 1.2 shows activity from footage of a cricket match; the annotated sequence illustrates how (non-automated) professionals present the play of a single ball. It follows an established, recognizable structure: an overall situation is shown and then bursts of action quickly become localized in space for spans of concentrated time. Compared to the sequence in green, the one marked in yellow —

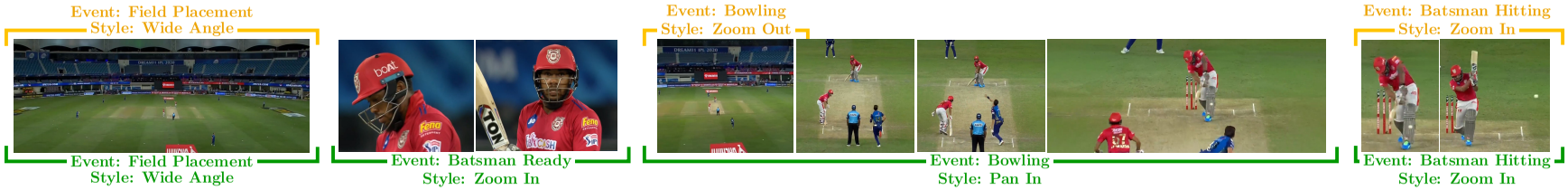


Figure 1.2: A sequence of events occurring as part of the broader drama of a cricket match. The unfolding action, for a single ball, is depicted as: ⟨1⟩ the field placement is set; ⟨2⟩ the batsmen make ready; ⟨3⟩ the bowler delivers the ball; and ⟨4⟩ the batter strikes the ball. Effective presentation shows important action along with suitable choices for camera position, i.e., events and styles.

imagine it was captured by a hypothetical robot — is less effective. Some events are absent: yellow lacks $\langle 2 \rangle$. But the delivery of the ball implies that the batsman was ready, so, strictly $\langle 2 \rangle$ need not appear. However, its absence would likely be felt because the interleaving of signals, innate to the game itself, has been violated. But absence of an event is just one cause for a dramatic arc to seem incomplete; the simple inclusion of would not necessarily yield a satisfactory result. An appropriate stylistic choice, such as rapid panning to show release of the ball, can focus attention on the impending nexus of the bat and ball or connote culmination and climax. Thus, to produce persuasive results, our video capturing robot must reason about events and also style choices.

The problem of the stylized structured narrative is modeled in this research by four distinct components. The first two components being a specification automaton and a modification of the Markovian structure presented in [7] to model the narrative sequence of interest and the stochastic process of event generation, respectively. The third component specifies constraints on event-style pairs. While the fourth quantifies the qualitative nature of styles.

With solution approaches based on underlying Markov Decision Processes, we present two algorithms: one that creates a product and therefore planning jointly over the event-style pairs and other, an approximate approach, that prescribes style conditioned on the events. Through simulations we show the approximate approach to be effective and much faster to compute.

1.4 Multi-agent event chronicle

For the second extension, we again take into consideration a problem involving event chronicle and extend it by considering multiple agents. The problem involves a team of robots that move about the environment, each recording what they observe. If anyone manages to capture some event according to the specification, they communicate that fact with the entire group. In the end, all events from all the robots are collected to provide a final video.

The formulation for this problem contains multiple components. Apart from the Markovian structure and the specification automation, each robot is associated with a component of its own, encapsulating a means for expressing constraints on the robots' capabilities in terms of what events they may capture in succession.

As with the stylized structure narrative, we provide two solution approaches here. A product treatment involving the optimal selection of joint choices, i.e., choosing the next events to attempt to capture by all robots, is formulated where costs are minimized in an expected sense. Since such plans are prohibitive to compute, the second solution approach provides an approximation scheme based on solving a sequence of individual planning problems, one for each robot. This scheme sacrifices some solution quality but requires far less computational expense.

To showcase the feasibility of the approximate solution approaches in real-world scenarios. The dissertation also provides a multi-robot implementation of the stylized structured narrative done in collaboration with teams from the University of Houston* and the University of South Carolina† To validate the policies generated by the solution, results of a field test is provided where a team of heterogenous robot attempts to record a narrative for a race.

1.5 Multi-commodity logistics

Next, we consider multiple aspects of the environment that are beyond the influence of the agent. The problem studied under this extension shifts away from the domain of structured narratives and applies the formulation to a multi-commodity flow problem with logistics domain. It formulates the planning problem for an autonomous logistic operations agent responsible for routing multiple commodities within a logistic network.

The components involved in the formulation consist of a logistic network containing nodes acting as either storage unit or retail unit, and the transportation of commodities among these units are under the direct influence of the agent. The study assumes stochastic demand is present for each retail unit. The stochastic demands are modeled via a Markovian structure similar to the ones used for the structured narrative problems. When a demand for a commodity arises with a retail node, if the commodity is present, it is assumed to be consumed; otherwise, the opportunity is lost, and the commodity remains unconsumed. The agent's objective is to predict future demands and route commodities preemptively to empty the storage units as fast as possible. Unlike the structured

*Dr. Aaron T. Becker, Rhema Ike, and Omar Romero
†Dr. Jason M O'Kane and Dr. Hazhar Rahmani.

narrative problems where a videographer agent needs to anticipate one step ahead for the events that might happen, here the agent might need to predict further into the future for commodities to be available within retail nodes when demand arises.

Unfortunately, the presence of such dynamic models of demand greatly increases the state space of a full-product MDP solution, rendering this approach intractable for most applications. The research tackles this challenge by developing an approximate solution that reduces the stochastic models via causal decoupling, giving a spectrum of solutions where weakening the time correlations in the stochastic demand process affords faster optimization.

1.6 Summary of contributions

Stemming from the overarching question, “When an agent’s actions have a weak influence on environment dynamics, how can this be exploited?”, the dissertation makes the following contributions:

1. Introduces the concept of an Element-generating Markov chain, a special type of Markov chain [8] with each state within the Markov chain having a special element-generating property.

2. Presents an application by formulating a stylized video narrative problem, where we quantify the qualitative nature of cinematic techniques used in videography via a structure inspired by Natural Language Processing called the style-gram. We develop a just-in-time decoupled approximate solution that allows for the solution of the stylized video narrative problem to take place in parts, with the part of “what to capture?" (content) being solved first and then using the solution to solve for “how to capture them?" (style). The just-in-time approach allowed utilization of the extra information of where the content is captured as a basis for which style to use for capturing it. This approach of conditioning styles based on where the content was captured, in some cases, produces policies that perform better than that of an MDP that solves the problem jointly.

3. The dissertation extends the formulation above by incorporating the concept of multiple

agents, with each agent having action constraints, where the agents work in coordination within an environment (agents' actions have no influence over the environment dynamics) to capture a structured narrative. The dissertation provides an approximate solution for solving this problem iteratively by solving for each agent separately using the policies for the agents solved before it. The iterative process vastly reduces the state space compared to solving a generic MDP that solves for all the agents simultaneously. This process reduces the solution time significantly. We also investigate the effect of the ordering of the agents being solved and provide an analysis of different heuristics for selecting the order.

4. The study provides a formulation of a multi-commodity network flow problem via a factored formulation, where the demand within the nodes of the network are modeled as stochastic processes that are uninfluenced by the actions of an agent (as customer behavior is oftentimes uninfluenced or only weakly affected by happenings within the network.) acting as an overseer of the supply change network. We provide an approximate solution approach that solves by analyzing the statistical information of the demands and the information present within the network structure. This approach allows for the reduction of the state space of the problem compared to an MDP, leading to a reduction in the computational time for solving the problem.

5. The dissertation provides details of an implementation of a multi-robot system that combines the multi-agent formulation of the structured narrative and the quantitative formulation of cinematic styles to show the formulation's feasibility in real-world scenarios where multiple robots cooperatively attempt to capture a stylized video narrative. We also provide a comparison of all the solution techniques presented here with state-of-the-art RL techniques.

1.7 Organization of the dissertation

This dissertation is organized as follows: Related works are presented in Chapter 2. Chapter 3 provides the formulation for a stylized video narrative problem along with the quantification model of cinematic styles. The chapter also provides approximate solutions required for solving the prob-

lem efficiently and provides results comparing the approximate solution with the traditional MDP solution and other neural network approximate solutions. Chapter 4 extends the formulation to incorporate multiple agents with action constraints associated with each agent. Approximate solutions are offered. Results are provided comparing the approximate approach with traditional MDP and neural network approximate techniques. The chapter also details a multi-robot implementation showing the feasibility of the solution in the real world [‡]. Chapter 5 extends the formulation for multiple components present within the agent's environment that are unaffected by the agent's actions and presents the formulation in the logistic domain. As with the other chapters, this chapter also provides an approximate solution approach and provides case studies comparing the solution with traditional MDP and neural network approximations. Finally, conclusions and possible future works are provided in Chapter 6.

[‡]work done in collaboration with teams from the University of Houston (Dr. Aaron T. Becker, Rhema Ike, and Omar Romero) and the University of South Carolina (Dr. Jason M O'Kane and Dr. Hazhar Rahmani)

2. RELATED WORK

The work presented here aims to study planning problems involving agents whose actions do not directly influence the dynamics of the environment. Such a problem often occurs in the real world, and the dissertation studies the problem via scenarios from automated cinematography and logistics. We attempt to solve the problem by using approximate solutions for Markov Decision Processes [5].

This chapter presents a background on the research conducted in these fields and their relation to the work presented here.

2.1 Markov Decision Processes

A popular choice for optimal planning involving decision-theoretic agents in stochastic environments is to represent the problem via a Markov Decision Process [5]. Based on the observability of the state space, there exist various generalizations. The Partially Observable Markov Decision Process (POMDP) [10] is a generalization that assumes the system dynamics to be determined by a regular MDP while the states of the MDP are not directly observable by the agent. Mixed Observability Markov Decision Process (MOMDP) [11] generalizes the POMDP formalism by considering part of the MDP state to be fully observable, putting the formulation in between MDPs and POMDPs. Researchers have also studied special cases of MDPs with non-observability [12]. While the different formulations consider different models of observability for the MDP state space, the research presented here assumes the MDP state space to be fully observable. Semi-Markov Decision Processes (SMDPs) [13, 14, 15] model continuous time discrete-event systems with actions in SMDPs taking a variable amount of time to execute. In this research, we assume discrete actions, with all actions taking the same amount of time to be executed.

A classical approach for solving MDPs optimally involves dynamic programming approaches such as value iteration and policy Iteration that utilize Bellman equations to solve for the optimal solution iteratively [5]. As mentioned previously, for real-world problems, the state space of

the MDPs can often become large, and therefore solving them optimally can become intractable. Consequently, researchers have sought efficient ways to solve large-scale MDPs via various approximation techniques.

One method involves reducing the state space/action space by defining a set of features that succinctly describe the dynamic process. This process of defining states via a set of features has been extensively studied in the field of factored MDPs [16, 17]. Although these representations allow large and complex MDPs to be represented compactly, the complexity of exact solution algorithms for such MDPs can grow exponentially in the representation size [17] (making finding solutions intractable), unlike the formulations presented in this dissertation that utilize the problem structure to solve large MDPs tractably. Various approximate solution approaches that can be used to solve these factored MDPs can be found in [17, 18, 19, 20, 21].

Other methods use function approximation techniques to solve for the MDPs approximately. Linear programming is a function approximation technique representing the value function as a linear combination of basic functions [22]. Whose solution can then be obtained by optimizing the coefficients of the linear function. Neural network algorithms such as Advantage Actor-Critic (A2C) [23] or Proximal Policy Optimization (PPO) [24] can learn complex and non-linear representations of value functions from datasets. Decision trees can also be used to approximate the value function in a piecewise-linear manner allowing for efficient computations [25]. While the approach taken in this dissertation decomposes larger MDPs into smaller ones, solving the smaller ones optimally using dynamic programming; these approximations rely on large-scale datasets and sampling techniques to solve large MDPs approximately.

2.2 Autonomous cinematography and chronicle of events

The research area of autonomous cinematography has become an active research area in robotics in recent years. Studies like [26, 27] focus on vision-based UAV, particularly multi-UAV, cinematography and provide a taxonomy of various shot types based on framing types (long shot, close-up) and camera motion (orbit, fly-by). Sabetghadam et al. [28] studied optimal trajectory planning algorithm for autonomous UAV cinematography with decoupled gimbal control and drone control.

Alćantara et al. [29] extended the previously mentioned work for multi-UAV cinematography. All the work mentioned above assumes that the agent already knows what to capture and what shot type to use to capture them. On the other hand, the work proposed here is to predict for multiple agents which event will occur in the future and thus to plan for which to record. The work also plans which shot type to use while recording such events using a data corpus of the sequence of shots that are used in different films. Wu et al. [30] provides an open database of annotated films along with an analysis of the style related to shot composition, shot transition, and shot arrangement. Their paper also provides a tool (Insight) to generate the style-related analysis.

There exist other related problems that bear similarities to the event chronicle problem. For instance, the problems of selecting effective viewpoints [31, 32, 33], and active perception generally [34, 35, 36], have long histories. The underlying objective of these works is informativeness and usually is applied to scenes that are understood to be static.

The research presented here also studies a multi-agent formulation of the chronicle of events problem. Work on multi-agent planning is vast, and various algorithms exist to solve these problems [37, 38, 39]. These problems focus on studying scenarios such as task assignments and motion planning. Researchers have also studied multi-agent variations of active perception problem [40, 41], which aims to increase informativeness by exploring complex environments. The work presented here differs as the objective here lies in anticipating future events.

Among others, the work studied in [42, 43] studies the story validation problem, the aim of which is to validate whether a sequence of recorded events by a set of sensors is consistent with a given story or not, can be considered as the inverse of the problem studied here. Video summarization is the problem of summarizing a given video based on some selection criteria such as identifying important objects [44], finding interesting events [45], selection using supervised learning [46], and finding inter-frame connections [47]. Among the summarization problems, the vacation snapshot problem [48, 49], considered a problem in which the goal is to summarize the data observed by a mobile robot via an online algorithm, is the closest to the work proposed here. However, the summarization problems essentially focus on post-processing a collection of images that have al-

ready been captured, which is different from what is proposed here. A survey on the various work done in video summarization can be found in [44, 50, 51, 52, 53, 54, 55]. Other research seeks to generate commentary [56, 57] or produce narrative text *de novo* [58, 59, 60, 61, 62, 63, 64, 65, 66].

2.3 Logistic planning

Distinct from videography, another domain where the work proposed here utilizes the formulation involves logistic planning for multiple commodities in a transportation network. Rapid developments with automated vehicles have spurred work on the routing of such vehicles in transportation networks [67, 68, 69]. When one thinks of these vehicles as enablers, they then form part of logistic networks within which the automated routing of goods and commodities becomes feasible.

The literature involving the flow of multiple commodities within a logistic network is vast and has been an important area of study since the first works of Ford and Fulkerson [70], and Hu [71] in the beginning of the 1960s, with a current review appear in [72]. Lately, work has sought to understand the multi-commodity flow problem in the presence of stochasticities. Given a variety of uncertainties present with the supply, demand, and transportation network, the problem of designing a multi-commodity distribution network has been tackled in the recent work including [73, 74, 75, 76]. Other work, like [77], considers a multi-commodity logistics problem with stochastic flow, taking into consideration the effects of transportation time, distance, and the steps involved in the transportation process along with stochastic supply and demand. Ding [78] investigates the multi-commodity flow problem in the presence of uncertain edge cost and edge capacity. Both the above studies pose the problem as a linear programming problem, with [77] employing a multi-objective genetic algorithm and [78] using the Dantzig–Wolfe decomposition method to solve it; the approach we propose here is a dynamic programming–based approach, providing a solution that can adapt/respond to the changes in the demand over time.

The work that most closely resembles the proposed problem is studied in [79], which utilizes a dynamic programming approach to solve the multi-commodity flow problem in the presence of stochastic demand. That study assumes the commodities as reusable and represents the stochastic

demand via a random variable at each vertex, as opposed to the proposed work where the demands (for non-reusable commodities) at each vertex are assumed to be generated by a stateful stochastic process.

Research has sought to understand the reliability of networks in the presence of stochastic damage and disruption [80, 81]. Those authors have studied the design of the network model and its reliability, while the work here considers a given network—those studies, thus, can complement our work, providing a way to select a robust network before computing plans which manage its operation.

The solution approach in this work involves reducing the state space of the MDP. There exists various research on state abstractions for MDPs such as bisimulation [82], homomorphism [83], utile distinction [84], and policy irrelevance [85]. While these abstraction techniques aim to reduce the MDP directly, the work presented here constructs a new smaller MDP by taking advantage of the non-causality of the components within the environment.

3. DECOUPLING STYLES FROM SUBSTANCE*

Chapter 1 introduced the general premise of this dissertation and provides the overarching problem studied "When an agent's actions have a weak influence on environment dynamics, how can this be exploited?" In this chapter, the idea is made concrete by introducing a problem in the domain of autonomous videography involving a single agent whose actions have no influence over the dynamics of the environment. To motivate the problem, consider the scenario described below.

Imagine Alice and Bob, being excited to participate in the Boston Marathon, hire an autonomous agent to produce a custom video of their race. Afterward, the agent will assemble a video clip from the events it recorded in order of occurrence to tell the tale of their day. The video might show them neck-and-neck, with one crossing the finish line just moments before the other, or perhaps they were widely separated and while Alice was sprinting past Boston College, Bob was crossing the Johnny Kelly statue (Figure 3.1). Beyond the mechanics of autonomously navigating, tracking, and shooting video, the agent needs to be strategic about what it tries to capture. Various events will occur simultaneously, and it is possible that events might fit multiple narrative arcs. Clearly, the form of the final story depends on how the day actually unfolded, with both predictable structure (start-middle-finish) but also unexpected, serendipitous detail. Therefore, to make the video successfully, the agent needs to predict the events that will occur in the future and take actions appropriately.

Though the notion of using an autonomous agent to record events that obey a narrative structure was first proposed in [6]. And, given a stochastic model of the environment and a narrative structure, the problem of which event to capture was the foray of the paper [7]. This chapter extends the work in two aspects. Firstly, it defines the environment by introducing a special Markovian structure, an enhancement from previous work. And lastly, here we introduce the concept of cinematic techniques in the formulation of structured narrative.

*Reprinted with permission from "Conditioning Style on Substance: Plans for Narrative Observation" by Diptanil Chaudhuri, Rhema Ike, Hazhar Rahmani, Dylan Shell, Aaron T. Becker and Jason M. O'Kane. *2021 Proceedings of IEEE International Conference on Robotics and Automation*. Copyright 2021 by *IEEE*.

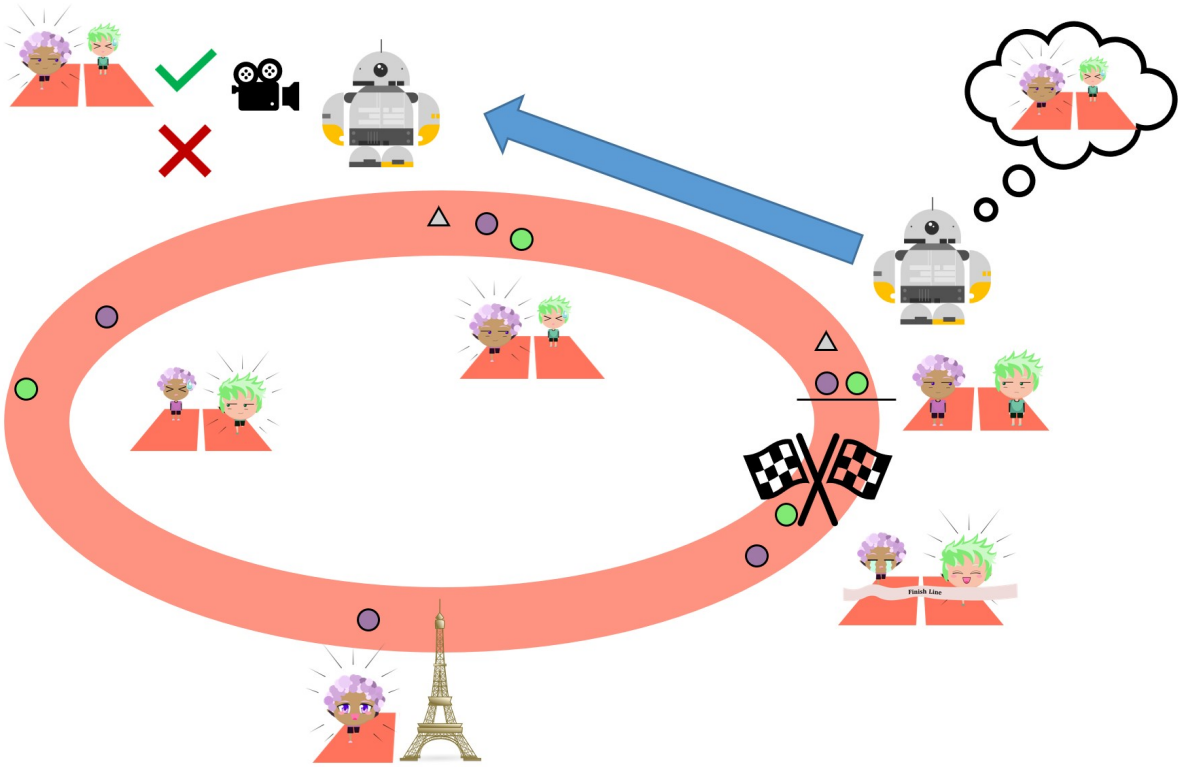


Figure 3.1: Alice (person with purple hair) and Bob (person with green hair) take part in a marathon and hire an autonomous agent (gray robot) to produce a custom video of their race. The race track is shown via an orange ellipse with a purple circle for Alice, a green circle for Bob, and a gray triangle for the robot. They start at the start line (black line) on the right and run the track anticlockwise till the finish (finish flag). Various events occur along the way — Alice overtaking Bob, Bob overtaking Alice, Alice passing by a landmark, and Bob winning the race. The robot starts along with them and predicts which event might happen in the future (thought bubble). The robot then moves to the location of its prediction and sets up to capture the event. If the prediction is correct (green tick), the robot records the event; otherwise (red cross), it fails. To make the video successful, the agent needs to correctly predict all the events of interest that will occur in the future and take action appropriately. (Clipart credit – Rhema Ike.)

A mere concatenation of a sequence of clips, one for each event, gives a rather poor result. Indeed, videography is a sophisticated craft involving a wide set of cinematic choices that include framing and positioning, camera focus and depth of field, filters, and motion. These choices are complex and constrained (it may only be physically feasible to place cameras in some few positions); the choices have semantic consequences, and their suitability depends on the subject and scene (e.g., reinforcing the action or portraying a contrast for rhetorical effect). Finally, the choices are made within the context of a broader flow, as transitions, cuts, and sequencing will alter the overall result. Throughout, we refer to all these aspects under the single umbrella term 'style.' Distinct from style, the substantive elements of the narrative are formed by sequences of events.

The chapter provides two solution approaches to solving the problem. First, a traditional approach to solve for the event-style pair jointly. Next, the chapter presents a decoupled method that first makes choices for events, then, based on those decisions, solves for style. We formulate the second step in a way that permits extra flexibility, settling on the style choice late so that additional information may be revealed.

Next, the chapter defines some structures essential for the problem definition and then provides the solution approach taken to solve it.

3.1 Problem definition

There are many structures needed to specify our problem completely. This section is in preparation for the formal problem statement.

3.1.1 Element-generating Markov chain

Before we continue defining the structures required for this problem, we define a special Markovian structure called the *Element-generating Markov chain* (EGMC) as a Markov chain with an element-generating property corresponding to each state of the Markov Chain. Formally,

Definition 1 (EGMC)**.** *An EGMC is a 5-tuple* $\mathcal{M} = (W, \mathbb{E}, T, w_0, g)$*, where*

- W *is a nonempty finite set constituting the state space of the model;*
- $\mathbb{E}$ *is the alphabet set;*

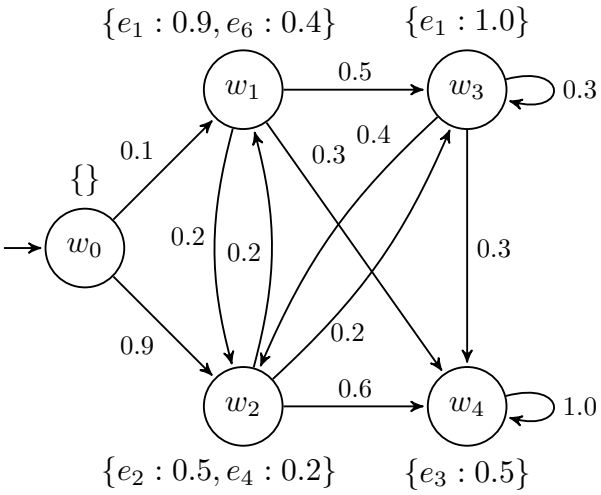


Figure 3.2: Example of modelling the world via an Element-generating Markov chain. The alphabets and their occurrence probability are shown inside curly braces along with the state that generate them. Note, some state like w_0 might not generate any events (element in the alphabet $\mathbb{E}$). (Reprinted with permission from [2].)

- $T : W \times W \to [0, 1]$ *is the transition probability function of the model, such that* $\forall w \in W$, $\sum_{w' \in W} T(w, w') = 1$*;*
- $w_0 \in W$ *is the initial state; and*
- $g : W \times \mathbb{E} \to [0, 1]$ *is an occurrence labeling function, such that for any state* $w \in W$ *and an element* $e \in \mathbb{E}$*,* $g(w, e)$ *is the probability that* e *occurs, or 'goes on', in the state* w*. We assume that* $\forall e \in \mathbb{E}$*,* $g(w_0, e) = 0$*.*

Starting with $w(0) = w_0$, an EGMC transitions from state to state in accordance with the probabilities $T(w(t), w(t+1))$, as time t progresses, in the conventional way for Markov chains. As $\mathcal{M}$ enters state $w(t)$, each alphabet $e \in \mathbb{E}$ either happens or does not, determined independently for each with probability $g(w(t), e)$.

3.1.2 The world and the narratives

In our scenario, the elements to capture are atomic events whose occurrence in the world is assumed to be structured via an EGMC, with $\mathbb{E}$ being the set of all possible events. Thus making

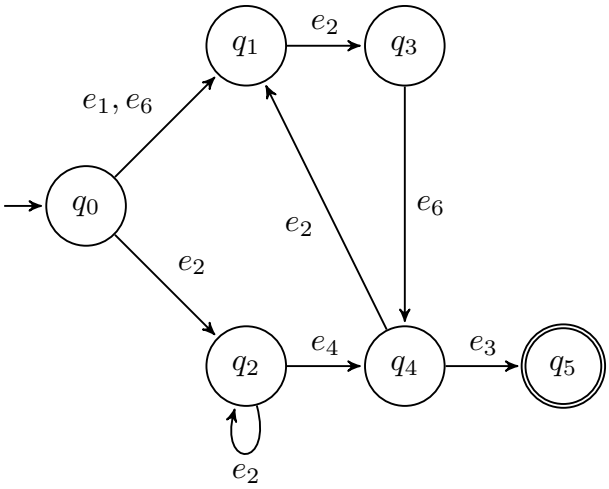


Figure 3.3: Example of narrative specification as a Story Automaton. (Reprinted with permission from [2].)

the world model essentially an Element-generating Markov chain. For the remainder of the chapter, we refer to this EGMC as the *event model* (for example, see Figure 3.2).

As the world evolves and events happen, the agent will attempt to capture events and produce a story portraying what occurred. The story is a sub-sequence of captured events selected to match a specification of suitable stories. These are given in the form of an automaton [9], we call the *story automaton* (for example, see Figure 3.3):

Definition 2 (Story Automaton)**.** *A story automaton is a deterministic finite automaton* $\mathcal{D} = (Q, \mathbb{E}, \delta, q_0, F)$ *with:*

- Q *a nonempty finite set of states;*
- $\mathbb{E}$*, its alphabet, a set of all possible events;*
- $\delta : Q \times \mathbb{E} \to Q$ *its transition function;*
- $q_0 \in Q$*, the initial state; and*
- $F \subseteq Q$*, the set of final (accepting) states.*

Let $\mathcal{L}(\mathcal{D})$ denote the set of event sequences accepted by $\mathcal{D}$, i.e., those sequences reaching some element of F, starting at q_0, and after tracing transitions via δ.

3.1.3 Style: constraints and sequential structure

Each time the agent attempts to capture an event, it must choose values for the various parameters that influence the videographic details of the recording. Let Γ be the set of all possible parameter values, or *styles*, available to the agent. Physical hardware and positioning constraints will typically mean that not all events can be captured in all possible styles. Let the *style catalogue*, $\kappa : \mathbb{E} \to 2^{\Gamma}$, map each event to the set of styles that may be used in capturing it. The style catalogue, despite being simple, impels the agent to plan ahead: the agent seeks to optimize properties of style sequences, but the style catalogue —as a constraint— binds choices about styles to decisions it makes about events as well.

Next, we introduce a measure for the efficacy of a sequence of style choices. Watching a film with high production value, the choices made for consecutive shots possess a temporal structure having a flow, which helps establish or reinforce a cinematic style. Taking inspiration from bi-grams, tri-grams, and N-grams as statistical models in natural language processing [86], we formalize this via the structure called the *style-gram* as follows:

Definition 3 (Style-gram)**.** *For $k \in \mathbb{Z}_+$, a k-order style-gram is a triple $\mathcal{S} = (\Gamma, \sigma, \hat{\omega})$, such that:*

- Γ *is the set of all available styles;*
- $\sigma \in \Gamma$ *is a special 'empty' symbol;*
- $\hat{\omega} : \Gamma^{k-1} \times \Gamma \to [0, 1]$ *is the* efficacy*, so, for each $s_1 s_2 \dots s_{k-1} \in \Gamma^{k-1}$ and $s' \in \Gamma$, it holds that $\sum_{s' \in \Gamma} \hat{\omega}(s_1 s_2 \dots s_{k-1}, s') = 1$.*

The intuition is that, for some cinematic style, a style-gram essentially encodes the probability of a shot with style $s \in \Gamma$ conditioned on the $k - 1$ immediately preceding style choices. One imagines that the oeuvres of Alfred Hitchcock, or Quentin Tarantino or Spike Jonze, might be summarized by style-grams that are quite different. Of course, much detail and many various

artistic factors are discarded by such a model. But, much as in natural language processing, the approach offers a valuable approximation for various more complex aspects and affords practical advantages:

1. the model can be induced from representative corpora;
2. via k, the range of temporal correlation is parametrizable; and
3. it enables direct incorporation into planning considerations.

To aid in combining the style-gram with other elements of our planning formulation, where Markov assumptions give straightforward sufficient statistics as states, we flatten the style-gram into a graph:

Definition 4 (Style-graph)**.** *For a k-order style-gram $\mathcal{S} = (\Gamma, \sigma, \hat{\omega})$, its associated* style graph *is the weighted directed graph $\mathcal{G}_{\mathcal{S}} = (V_{\mathcal{S}}, \tau_{\mathcal{S}}, \omega_{\mathcal{S}})$, where*

- *$V_{\mathcal{S}} \subseteq \Gamma^{k-1}$ is the set of states, with $\sigma^{k-1} = \underbrace{\sigma\sigma\ldots\sigma}_{k-1}$ being the initial state;*
- *$\tau_{\mathcal{S}} \subseteq V_{\mathcal{S}} \times V_{\mathcal{S}}$ is the set of edges such that: $\forall s_k \in \Gamma$, $(s_1 s_2 \ldots s_{k-2} s_{k-1}\ ,\ s_2 s_3 \ldots s_{k-1} s_k) \in \tau_{\mathcal{S}}$;*
- *$\omega_{\mathcal{S}} : \tau_{\mathcal{S}} \to [0, 1]$ is the edge weight constructed from $\hat{\omega}$,*

$$\omega_{\mathcal{S}}(s_1 \ldots s_{k-1}\ ,\ s_2 \ldots s_k) = \hat{\omega}(s_1 \ldots s_{k-1}, s_k).$$

3.1.4 Connecting the pieces: the agent and its capture choices

Up until time t, the agent keeps the sequence of events that it has recorded as ξ_t, the sequence of styles that the agent used to record the event sequence as ζ_t, along with the (unique) story automaton state q_t reached by tracing those events on $\mathcal{D}$. Just before $t + 1$ commences, the agent predicts a single event $e_{t+1} \in \mathbb{E}$ that it will attempt to capture. Additionally, the agent picks an

appropriate style $s \in \kappa(e_{t+1})$ for its capture. If the prediction was correct and the event indeed happens, the capture is successfully made and in the associated style. If the prediction was incorrect (that is, if the event does not occur), then nothing is captured. The sequence of captured events ξ_t, the sequence of styles ζ_t, and the story automata state q_t, are updated, all as follows:

$$\xi_{t+1} = \begin{cases} \xi_t e_{t+1} & \text{if } e_{t+1} \text{ was captured in state } w_{t+1} \\ \xi_t & \text{otherwise;} \end{cases}$$

$$\zeta_{t+1} = \begin{cases} \zeta_t s & \text{if event } e_{t+1} \text{ was captured with style } s \\ \zeta_t & \text{otherwise;} \end{cases}$$

$$q_{t+1} = \begin{cases} \delta(q_t, e_{t+1}) & \text{if } e_{t+1} \text{ was captured in } w_{t+1} \\ q_t & \text{otherwise.} \end{cases} \tag{3.1}$$

Initially, $\xi_0 = \epsilon$, the empty sequence, and $\zeta_0 = \sigma^{k-1}$. When $q_t \in F$, the agent terminates its execution.

Note that the agent traces *all* events captured, stopping when that sequence is in $\mathcal{L}(\mathcal{D})$; one might ask about the selection of sub-sequences of events. This seeming shortcoming in (3.1), in fact, is not one. As examined in the earlier work [7], edits to the captured sequence, such as selecting a sub-sequence (and other sorts of edits as well), can be encoded by mutating the story automaton, producing a new one. Hence, without losing generality, we will assume that whatever post-production steps are permissible to have already been expressed in $\mathcal{D}$.

3.1.5 Capture criterion and optimization problem

For a story automaton $\mathcal{D}$, let $\mathcal{L}_{\text{pre}}(\mathcal{D})$ denote those sequences in $\mathcal{L}(\mathcal{D})$ containing no proper prefixes that are themselves in $\mathcal{L}(\mathcal{D})$. Also, for the style graph $\mathcal{G}_{\mathcal{S}}$, we define *sequence efficacy* via

function $\bar{\nu}_{\mathcal{G}_\mathcal{S}} : \Gamma^{\mathbb{Z}_+} \to \mathbb{R}$, so that for $\zeta_= s_1 s_2 \ldots s_{|\zeta|}$,

$$\bar{\nu}_{\mathcal{G}_\mathcal{S}}(\zeta) = \prod_{i=k}^{|\zeta|} \omega_\mathcal{S}(s_{i-k+1} \ldots s_{i-1}, s_{i-k+2} \ldots s_i).$$

Further, we define *accepted sequence efficacy* for a pair of sequences of events ξ and styles ζ, and automata $\mathcal{D}$ as

$$\nu_{\mathcal{G}_\mathcal{S}}(\xi, \zeta) = \begin{cases} \bar{\nu}_{\mathcal{G}_\mathcal{S}}(\zeta) & \text{if } |\zeta| = |\xi| \text{ and } \xi \in \mathcal{L}_{\text{pre}}(\mathcal{D}), \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

Now, we use (3.2) as an optimization criterion.

Optimization Problem: Styled Video Capture (SVC)

Given: Event model $\mathcal{M} = (W, \mathbb{E}, T, w_0, g)$ over event set $\mathbb{E}$, a DFA $\mathcal{D} = (Q, \mathbb{E}, \delta, q_0, F)$, a set of available styles Γ, a style-graph $\mathcal{G}_\mathcal{S} = (V_\mathcal{S}, \tau_\mathcal{S}, \omega_\mathcal{S})$, a style catalogue $\kappa : \mathbb{E} \to 2^\Gamma$, and a finite horizon $N \in \mathbb{Z}_+$.

Output: Some prescription for events and styles to capture so the expected accepted sequence efficacy $\mathbf{E}_N[\nu_{\mathcal{G}_\mathcal{S}}(\xi, \zeta)]$ is maximal, expectation being taken over sequences ξ and ζ arising from at most N opportunities to capture an event.

One can anticipate that unless you have an exceptionally lucky agent, generally $|\xi|$ will be less than N.

For SVC to be formally defined, detail of what is meant by 'prescription' must be determined. Different choices lead to different solutions in an interesting way.

3.2 Solution approaches

In this section, we present the various solution techniques that can be used to solve the problem via different 'prescriptions'. We start by first solving the problem via a traditional product MDP that prescribes the joint action involving an event-style pair, and next, we present a proficient approach that utilizes the intrinsic property of the problem to solve the problem efficiently by prescribing which event to capture first then, based on which event model state the event is captured, the style to capture it.

3.2.1 Traditional solution

A quite natural choice, perhaps one already anticipated by the reader, is for the agent's predictions to be governed by a *capture policy*, $\pi_c : W \times Q \times \Gamma^{k-1} \to \Delta(\mathbb{E} \times \Gamma)^*$, that uses the state of the world (event model), the story captured so far (story automaton state), and history of recent styles (encapsulated as a state of the style graph) to select a pair comprising an event and a style.

A conceptually straightforward solution approach is to search over a joint action space, with choices comprising such pairs. This yields a Markov Decision Process [5] that we use the moniker *Monolithic* to describe:

Definition 5 (Monolithic MDP)**.** *Given event model $\mathcal{M} = (W, \mathbb{E}, T, w_0, g)$, automaton $\mathcal{D} = (Q, \mathbb{E}, \delta, q_0, F)$, available styles Γ, and a style graph $\mathcal{G}_\mathcal{S} = (V_\mathcal{S}, \tau_\mathcal{S}, \omega_\mathcal{S})$, construct the Monolithic MDP $\mathbb{M}_{mon} = (X_{mon}, x_0, \mathcal{A}_{mon}, P_{mon}, R_{mon})$, where*

- *$X_{mon} \subseteq W \times Q \times V_\mathcal{S}$ is the finite state space;*
- *$x_0 = (w_0, q_0, \sigma^{k-1})$ is the initial state;*
- *$\mathcal{A}_{mon} = \mathbb{E} \times \Gamma$ is the action space comprising pairs of events and sanctioned styles: $(e, s) \in \mathcal{A}_{mon}$ iff $s \in \kappa(e)$;*
- *$P_{mon} : X_{mon} \times \mathcal{A}_{mon} \times X_{mon} \to [0, 1]$ is the transition probability function, such that, for*

*$\Delta(X)$ denotes the set of probability distributions over X.

$(w_i, a_i, \zeta_i), (w_j, q_j, \zeta_j) \in X_{mon}$, $(e, s_k) \in \mathcal{A}_{mon}$, *with* $\zeta_i = (s_1 s_2 \ldots s_{k-1})$, $\zeta_j = (s_2 \ldots s_{k-1} s_k)$,

$$P_{mon}((w_i, q_i, \zeta_i), (e, s_k), (w_j, q_j, \zeta_j)) = \begin{cases} T(w_i, w_j) \cdot g(w_j, e) & \text{if } q_j = \delta(q_i, e) \text{ and } g(w_j, e) > 0, \\ T(w_i, w_j) \cdot (1 - g(w_j, e)) & \text{if } q_j = q_i, \\ 0 & \text{otherwise.} \end{cases}$$

and, for all other inputs, $T(\cdot, \cdot, \cdot)$ *is* 0.

- $R_{mon} : X_{mon} \times \mathcal{A}_{mon} \times X_{mon} \to \mathbb{R}$ *is a reward function such that for* $(w_i, q_i, \zeta_i), (w_j, q_j, \zeta_j) \in X_{mon}$, $(e, s_k) \in \mathcal{A}_{mon}$, *with* $\zeta_i = (s_1 s_2 \ldots s_{k-1})$, $\zeta_j = (s_2 \ldots s_{k-1} s_k)$, *we have*

$$R_{mon}((w_i, q_i, \zeta_i), (e, s_k), (w_j, q_j, \zeta_j)) = \log \omega_{\mathcal{S}}(\zeta_i, s_k),$$

and all other inputs $R_{mon}(\cdot, \cdot, \cdot)$ *takes some constant value* r_- *with* $r_- < \min_{\zeta, s} \log \omega_{\mathcal{S}}(\zeta, s)$.

An optimal policy $\pi^* : X_{\text{mon}} \to \Delta(\mathcal{A}_{\text{mon}})$ for this MDP provides a capture policy. Such a policy can be obtained using standard finite-horizon solution techniques; a deterministic policy may be sought, but it is not strictly required.

3.2.2 Decoupled solution

Though, in the problem, the choice of event and style are coupled via κ, one might posit that the separation of substance from style is typically quite clean and often useful. Thus, we might approach the problem by decomposing a capture policy into two functions, one to decide events (π_e), another for styles (π_s). Earlier work [7] can compute an event policy of the form $\pi_e : w \times q \to \Delta(\mathbb{E})$ efficiently. Then, given those event choices, one might ask the restricted

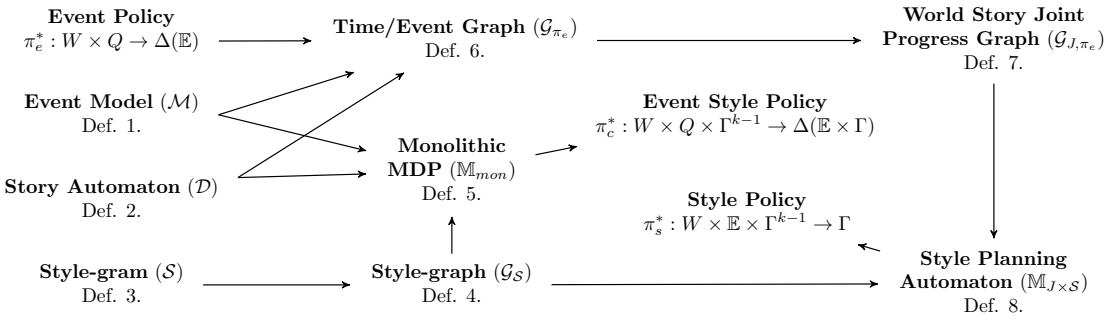


Figure 3.4: Flow diagram showing the series of constructions required for the two solution approaches presented here.

question of how to pick a suitable style. This decomposition, however, also spurs thoughts about new opportunities.

Capturing an event often entails a large-scale activity, needing time to execute —e.g., moving into position to film Becky passing a statue. For this reason the model has the agent predict what will occur, rather than merely discovering what is occurring and then quickly trying to capture it. In contrast, style choices are smaller and more local, so requiring predictions seems less necessary for style choices.

Suppose that after time t, the agent predicts event e_{t+1} might occur, but delays committing to a style for it. As the agent begins executing actions to capture e_{t+1}, the world evolves to w_{t+1}. Suppose that during the course of this execution, the agent learns w_{t+1}. So long as $g(w_{t+1}, e_{t+1}) > 0$, e_{t+1} can still occur and knowing w_{t+1} helps inform the choice of style. For instance, the guess that e_{t+1} will happen may turn out to be true, though perhaps the agent expected the event to happen in w_{t+1}, but the world actually evolved to w'_{t+1} instead. When the events that may occur subsequent to w'_{t+1} differ from those of w_{t+1}, a markedly different choice of style may be warranted. To make such late-breaking style selections, we compute a π_s that uses the state of the world, current progress in the story, and the last $k-1$ styles to make a conditional style selection. This conditional style selection is a function from $W \times \mathbb{E} \to \Gamma$, where the first input would now be w_{t+1}, i.e., the newly realized world state. In other words, we solve a planning problem for $\pi_s(w_t, q_t, (s_{t-k+1} \ldots s_t))$ whose output itself can be seen as a sort of local policy.

We attack this problem via a series of constructions, each summarized, along with their inter-relationships, in Fig. 3.4.

3.2.2.1 Joint evolution of the world and narrative

A source of complexity in thinking about the agent's interaction with its world is that it only progresses toward a story when it guesses events correctly. With a π_e in hand, one can ignore the detail of the robot making guesses, and model instead the (stochastic) progression of the successful event captures. We do this via the *Time/Event Graph (TEG)*, which essentially abstracts away the time-based evolution for a progression based on story automaton transitions (events). The formal definition is as follows:

Definition 6 (Time/Event Graph)**.** *For event model $\mathcal{M} = (W, \mathbb{E}, T, w_0, g)$, story automaton $\mathcal{D} = (Q, \mathbb{E}, \delta, q_0, F)$, and given event policy $\pi_e : W \times Q \to \Delta(\mathbb{E})$, construct $\mathcal{G}_{\pi_e} = (V_{\pi_e}, v_0, \tau_S, \tau_U, \omega_{\pi_e})$, where*

- *$V_{\pi_e} \subseteq W \times Q$ are the vertices of the graph;*
- *$v_0 = (w_0, q_0)$ is the starting vertex;*
- *$\tau_S \subseteq V_{\pi_e} \times \mathbb{E} \times V_{\pi_e}$, are the successful transitions, such that $((w_i, q_i), e, (w_j, q_j)) \in \tau_S$ iff $T(w_i, w_j) > 0$, $e \sim \pi_e(w_i, q_i)$, $g(w_j, e) > 0$, and $\delta(q_i, e) = q_j$;*
- *$\tau_U \subseteq V_{\pi_e} \times e \times V_{\pi_e}$, are the unsuccessful transitions, such that $((w_i, q_i), e, (w_j, q_j)) \in \tau_U$ iff $T(w_i, w_j) > 0$, $e \sim \pi_e(w_i, q_i)$, and $\delta(q_i, e) \neq q_j$;*
- *$\omega_{\pi_e} : \tau_S \cup \tau_U \to [0, 1]$ is the edge weight function, such that for $((w_i, q_i), e, (w_j, q_j)) \in$*

$\tau_S \cup \tau_U$,

$$\omega_{\pi_e}((w_i, q_i), e, (w_j, q_j)) =$$

$$\begin{cases} \pi_e(w_i, q_i)(e) \cdot T(w_i, w_j) \cdot g(w_j, e) & \\ \qquad\qquad\qquad\qquad\qquad\qquad \textit{if } ((w_i, q_i), e, (w_j, q_j)) \in \tau_S, & \\ \pi_e(w_i, q_i)(e) \cdot T(w_i, w_j) \cdot (1 - g(w_j, e)) & \\ \qquad\qquad\qquad\qquad\qquad\qquad \textit{if } ((w_i, q_i), e, (w_j, q_j)) \in \tau_U & \\ \qquad\qquad\qquad\qquad\qquad\qquad \textit{and } \ g(w_j, e) > 0, & \\ \pi_e(w_i, q_i)(e) \cdot T(w_i, w_j) \qquad \textit{if } ((w_i, q_i), e, (w_j, q_j)) \in \tau_U & \\ \qquad\qquad\qquad\qquad\qquad\qquad \textit{and } \ g(w_j, e) = 0, & \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad \textit{otherwise.} & \end{cases}$$

Given a state $(w_j, q_j) \in V_{\pi_e}$, we call the state *successful* if there is at least one state $(w_i, q_i) \in V_{\pi_e}$, and $e \in \mathbb{E}$, such that $((w_i, q_i), e, (w_j, q_j)) \in \tau_S$.

A crucial decision at this point is to decide where to put the style planning decision. One option would be to utilize only the event model and decide styles based on the world state and the event, $\pi_s : W \times \mathbb{E} \times \Gamma^{k-1} \to \Gamma$. However, this is a very coarse approach to solve the problem. The stochasticity involved in the transition prevents us from predicting which event is going to be captured next. Another option would be associate styles to the edge of the TEG. That is to assign a style to each event that the robot attempts to capture. However, this is too fine an approach to this problem. If we focus our attention on the unsuccessful transition edges, we notice that this policy is assigning style to events that are not going to be captured. The proper approach would then be to just focus on the successful transitions of the TEG. Thus, to correctly associate style to

events, we need to find the probability of successfully capturing a new event, given an event has been successfully captured.

Figs. 3.2, 3.3 and 3.5 provide an example construction. Examining Fig. 3.5, one sees that it is structured: each block corresponds to a state in the story automaton (compare to Fig. 3.3). The unsuccessful transitions revisit states within the same block, while successful transitions shift from one block to another.

To find the probability of successfully capturing an event, we compute the probability of all possible sequences of unsuccessful transitions which then lead to a successful transition and capture. Using this approach, we reduce the TEG to a new graph containing only successful transitions, along with the probability of successfully capturing events. The graph thus created encodes the joint progress of both the event model and the story automata together, so we call it the *World Story Joint Progress Graph (JPG).*

Definition 7 (World Story Joint Progress Graph)**.** *For TEG $\mathcal{G}_{\pi_e} = (V_{\pi_e}, v_0, \tau_S, \tau_U, \omega_{\pi_e})$, we construct the JPG $\mathcal{G}_{J,\pi_e} = (V_J, j_0, \mathbb{E}, \tau_J, \omega_J)$, where*

- *$V_J \subseteq V_{\pi_e}$, are the vertices;*
- *$j_0 \in V_J$ is the initial vertex;*
- *$\mathbb{E}$ is the set of all possible events;*
- *$\tau_J \subseteq V_J \times \mathbb{E} \times V_J$ are the successful transitions such that $((w_i, q_i), e, (w_j, q_j)) \in \tau_J$ iff $g(w_j, e) > 0$ and $\delta(q_i, e) = q_j$;*
- *$\omega_J : \tau_J \to [0, 1]$ is the probability of successful capture, such that for each transition $((w_i, q_i), e, (w_j, \delta(q_i, e))) \in \tau_J$, $\omega_J((w_i, q_i), e, (w_j, \delta(q_i, e)))$ is the probability that when, at some time, the robot is at (w_i, q_i) the next event it successfully captures is e and the world arrives in state w_j.*

The probability of capture, ω_J, is calculated via another Markov chain that is constructed by augmenting the TEG with V'_{π_e}, a set of additional absorbing states, shown in dashed blue in Fig. 3.5.

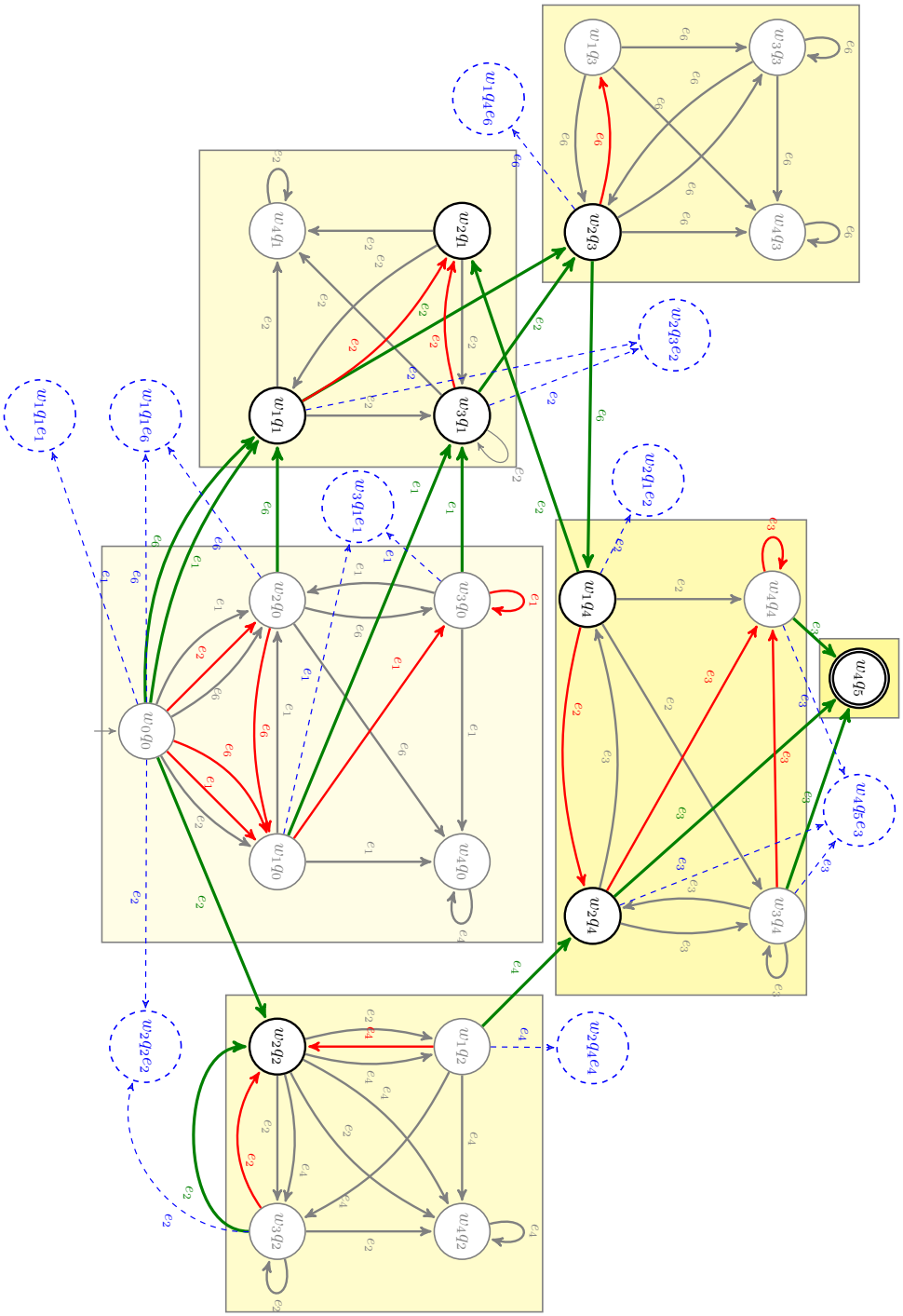


Figure 3.5: Example of Time/Event Graph for Event model in Figure 3.2 and Story Automaton in Figure 3.3. The green arrows denote the successful transitions. Grey arrows denote unsuccessful transitions with $g(w, e) = 0$, while red arrows denote the unsuccessful transitions with $g(w, e) > 0$. Edge weights have been omitted to improve clarity. (The dashed blue elements that have been superimposed are not part of the TEG, but are an augmentation used to compute ω_J values for the JPG.) (Reprinted with permission from [2].)

An absorbing state is added for each successful state, along with its incoming event. For example, in the figure, for the successful state (w_1, q_1) we add two absorbing states (w_1, q_1, e_1) and (w_1, q_1, e_6) to V'_{π_e}. The edge weight function $\hat{\omega}_{\pi_e} : V_{\pi_e} \times (V_{\pi_e} \cup V'_{\pi_e}) \to [0, 1]$ is defined as:

– For a successful transition

$$((w, q), e, (w', q')) \in \tau_S, \hat{\omega}_{\pi_e}((w, q), (w', q', e)) = \omega_{\pi_e}((w, q), e, (w', q')).$$

– For unsuccessful transitions

$$((w, q), e, (w', q')) \in \tau_U, \hat{\omega}_{\pi_e}((w, q), (w', q')) = \sum_{e \in \mathbb{E}} \omega_{\pi_e}((w, q), e, (w', q')).$$

Markov chain $\mathcal{G}'_{\pi_e} = (V_{\pi_e} \cup V'_{\pi_e}, \hat{\omega}_{\pi_e})$, with states $V_{\pi_e} \cup V'_{\pi_e}$ and transitional probabilities $\hat{\omega}_{\pi_e}$, enable calculation of absorbing probabilities (see [87, Appendix A]). These values define ω_J.

3.2.2.2 Planning for style

To solve SVC with event and style decoupled, given some π_e, we need a way to produce a π_s. To do this we construct a Markov Decision Process, called the *Style Planning Automaton (SPA)*, as follows:

Definition 8 (Style Planning Automaton)**.** *Given JPG $\mathcal{G}_{J,\pi_e} = (V_J, j_0, \mathbb{E}, \tau_J, \omega_J)$, available styles Γ, and style graph $\mathcal{G}_{\mathcal{S}} = (V_{\mathcal{S}}, \tau_{\mathcal{S}}, \omega_{\mathcal{S}})$, construct the style planning automaton $\mathbb{M}_{J\times\mathcal{S}} = \mathcal{G}_{J,\pi_e} \times \mathcal{G}_{\mathcal{S}}$, as a tuple $\mathbb{M}_{J\times\mathcal{S}} = (X_{J\times\mathcal{S}}, y_0, \mathcal{A}_\Gamma, P_{J\times\mathcal{S}}, R_{J\times\mathcal{S}})$,*

- *$X_{J\times\mathcal{S}} \subseteq V_J \times V_{\mathcal{S}}$ is a finite set of states, encapsulating a state in the joint graph, and styles of recent captures;*
- *An initial state $y_0 = (w_0, q_0, \sigma^{k-1})$;*
- *The set of actions $\mathcal{A}_\Gamma$ contains elements, each indicating the next style to be chosen. Specifically, each $a \in \mathcal{A}_\Gamma$ is a function prescribing which style to employ, given the attempted*

capture of some particular event and the current world state is observed; So a is a function that maps $w_k \in W$ and $e_\ell \in \mathbb{E}$ to a style permissible for e_ℓ:

$$W \times \mathbb{E} \ni (w_k, e_\ell) \xmapsto{a} s \in \kappa(e_\ell).$$

- *$P_{J\times\mathcal{S}} : X_{J\times\mathcal{S}} \times \mathbb{E} \times X_{J\times\mathcal{S}} \to \mathbb{R}$ is the transition probability function such that for (w_i, q_i, ζ_i), $(w_j, q_j, \zeta_j) \in X_{J\times\mathcal{S}}$ and e, we have,*

$$P_{J\times\mathcal{S}}((w_i, q_i, \zeta_i), (e, s_k), (w_j, q_j, \zeta_j)) = \omega_J((w_i, q_i), e, (w_j, q_j))$$

  when $((w_i, q_i), e, (w_j, q_j)) \in \tau_J$, and, for all other inputs, $P_{J\times\mathcal{S}}(\cdot, \cdot, \cdot)$ is 0.

- *$R_{J\times\mathcal{S}} : X_{J\times\mathcal{S}} \times \Gamma \times X_{J\times\mathcal{S}} \to \mathbb{R}$ is the reward function such that for $(w_i, q_i, \zeta_i), (w_j, q_j, \zeta_j) \in X_{J\times\mathcal{S}}$, and $s_k \in \Gamma$, where $\zeta_i = (s_1 s_2 \dots s_{k-1})$, and $\zeta_j = (s_2 \dots s_{k-1} s_k)$,*

$$R_{J\times\mathcal{S}}((w_i, q_i, \zeta_i), s_k, (w_j, q_j, \zeta_j)) = \log \omega_{\mathcal{S}}(\zeta_i, s_k),$$

  and for other inputs takes some constant value r_- with $r_- < \min_{\zeta,s} \log \omega_{\mathcal{S}}(\zeta, s)$.

For this MDP with horizon N, a policy can be obtained using standard finite-horizon solution techniques. An optimal deterministic policy $\pi_s^* : X_{J\times\mathcal{S}} \to \mathcal{A}_\Gamma$ serves as a style policy.

3.2.3 Observability

In this section, we examine the different problem parameters that are observable to the agent. For both methods, at any point in time, the agent observes the current state of the event model and the story automaton, along with the styles used for recording the previous events. The monolithic approach uses these observations to prescribe the event and style together without any other information. On the other hand, the decoupled solution utilizes additional information by first prescribing the event that the agent should attempt to capture and then, based on the observation of where the event is realized — specifying the style.

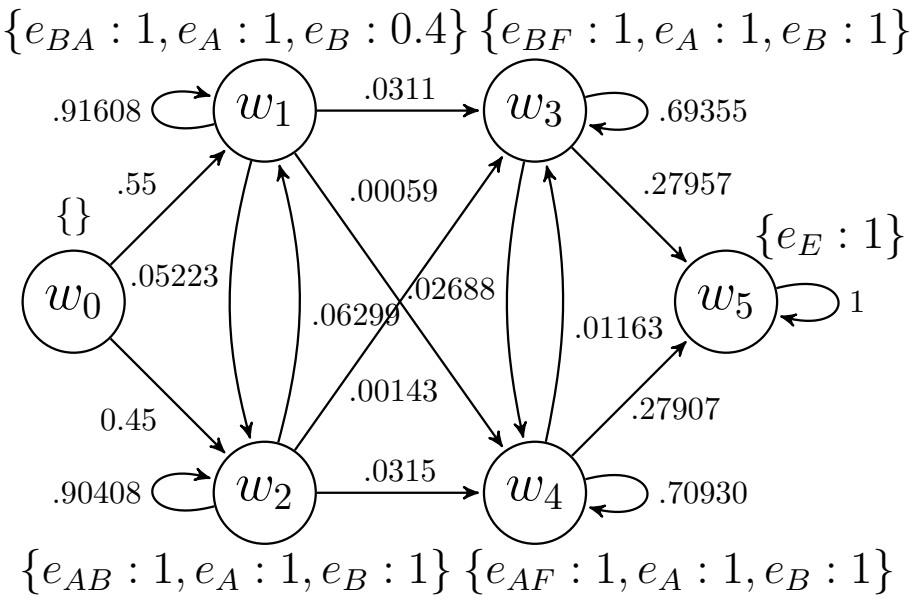


Figure 3.6: Event model used for the simulations. (Reprinted with permission from [2].)

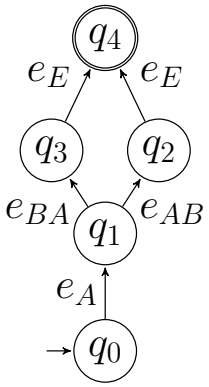
Is the loss of precision incurred by planning for events (in the absence of style considerations), and then prescribing styles afterward, offset by the extra flexibility obtained by delaying the style selection? And, given that the output of the decoupled problem is more complex than the joint one, is the cost to compute favourable compared to the monolithic solution? Next, we examine these questions empirically.

3.3 Results

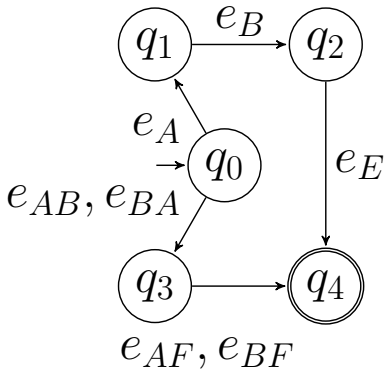
In this section, we present results of our Python implementation of the algorithms, which we executed on an Ubuntu 16.04 computer with a 3.6GHz CPU.

3.3.1 Decoupled vs traditional solution

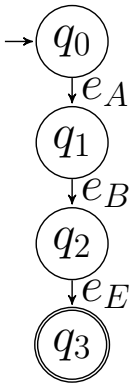
Consider an athletic race between runners A and B. The events of interest are: e_A, A is running; e_B, B is running; e_{AB}, A is overtaking B; e_{BA}, B is overtaking A; e_{AF}, A is crossing the finish line; e_{BF}, B is crossing the finish line; e_E, the race is ended. We want to capture those events from five relative poses, namely: front, rear, left, right, and front-left side, to represent which we respectively use styles s_f, s_b, s_l, s_r, and s_{fl}. We model the contest with the event model in Fig. 3.6.



(a) Story automation for the first variation.



(b) Story automation for the second variation.



(c) Story automation for the third variation.

Figure 3.7: Story automation used for the different variations of the case study. (Reprinted with permission from [2].)

e_A	s_f, s_b, s_l, s_r
e_B	s_f, s_b, s_l, s_r
e_{AB}	s_l
e_{BA}	s_r
e_{AF}	s_l, s_r
e_{BF}	s_l, s_r
e_E	s_{fl}

Table 3.1: Style catalogue used for the simulations. (Reprinted with permission from [2].)

$\hat{\omega}$	s_f	s_b	s_l	s_r	s_{fl}
σ	0.4	0.4	0.07	0.07	0.06
s_f	0.15	0.03	0.25	0.5	0.07
s_b	0.03	0.15	0.5	0.25	0.07
s_l	0.25	0.07	0.15	0.03	0.5
s_r	0.03	0.25	0.07	0.15	0.5
s_{fl}	0.2	0.2	0.2	0.2	0.2

Table 3.2: Style-gram ($k = 2$) used for the simulations. (Reprinted with permission from [2].)

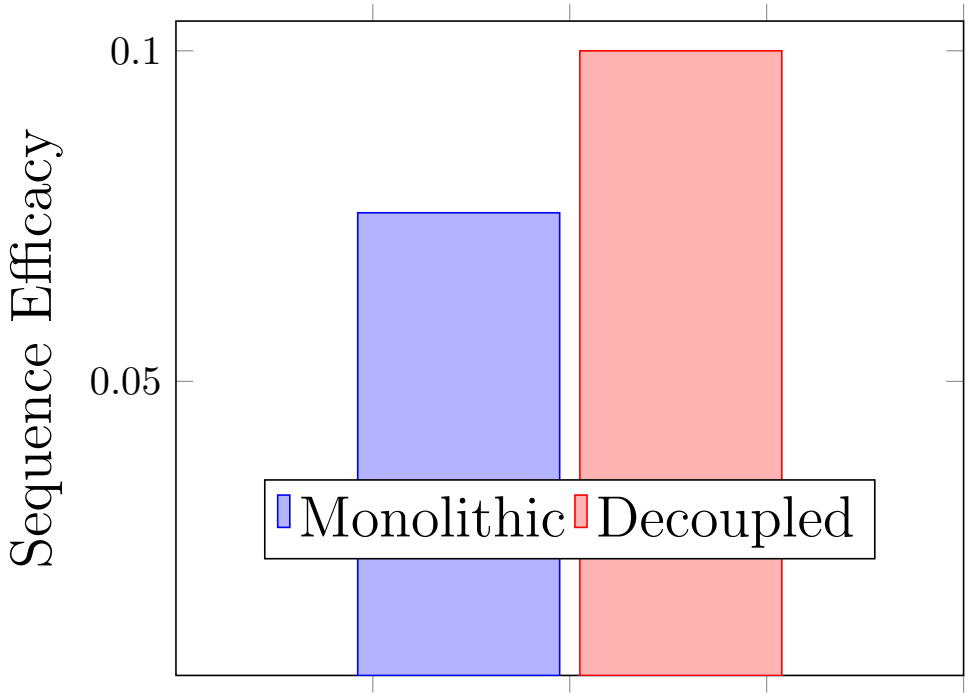


Figure 3.8: Plot comparing the accepted sequence efficacies of the decoupled and the monolithic approach for 100 simulations corresponding to the story automation in Figure 3.7a. (Reprinted with permission from [2].)

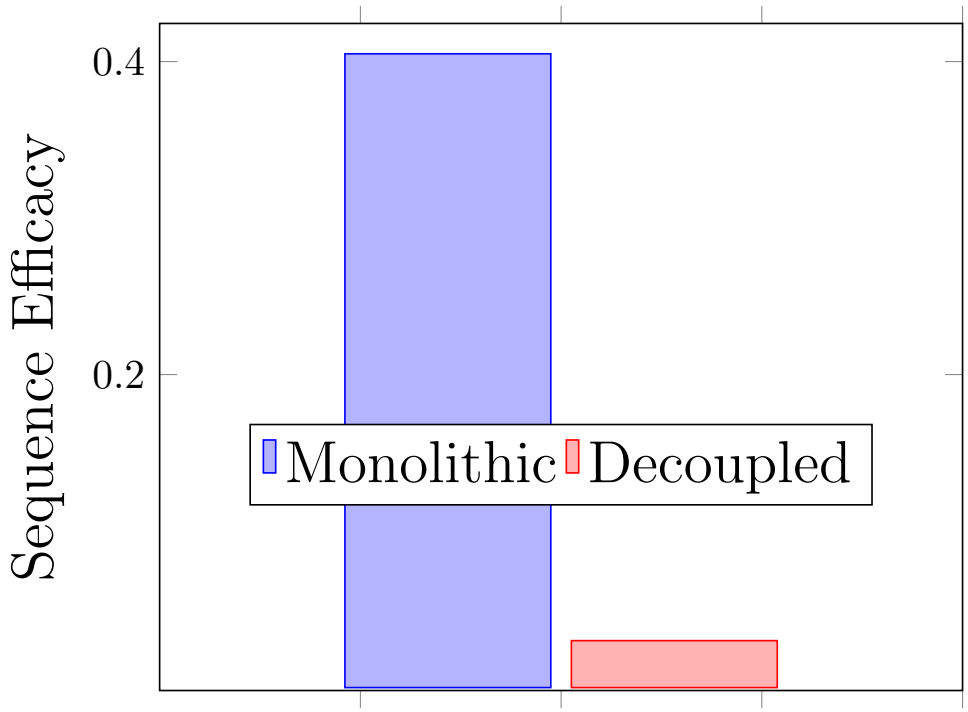


Figure 3.9: Plot comparing the accepted sequence efficacies of the decoupled and the monolithic approach for 100 simulations corresponding to the story automation in Figure 3.7b. (Reprinted with permission from [2].)

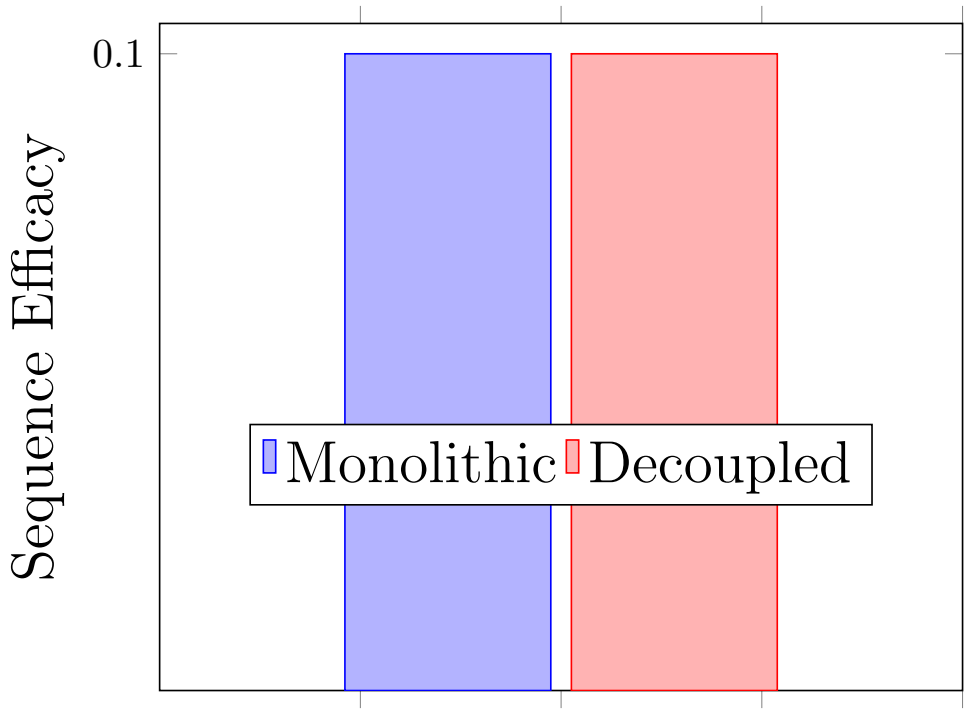


Figure 3.10: Plot comparing the accepted sequence efficacies of the decoupled and the monolithic approach for 100 simulations corresponding to the story automation in Figure 3.7c. (Reprinted with permission from [2].)

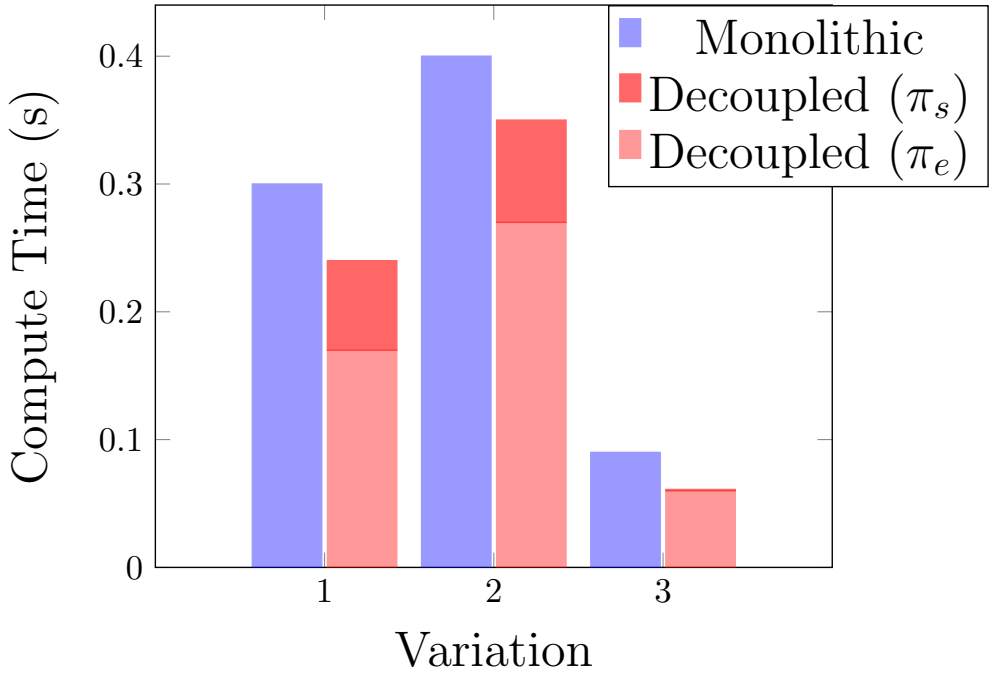


Figure 3.11: Plot showing the computation time for both monolith and decoupled solution corresponding to the three variations of the story automatons in Figure 3.7. (Reprinted with permission from [2].)

The style catalogue and style-gram for the model appear in Table 3.1 and Table 3.2, respectively. Note that $k = 2$. Each row of the style-gram gives a preference over the styles reflecting choices for possible positions to which it might move. For example, based on the second row, if the robot is currently capturing an event from the front, then the right (rear) has the highest (lowest) desirability for next capturing an event.

We now consider three variations on this scenario. In the first variation, the desired story is specified by the story automaton in Fig. 3.7a. Accordingly, we are interested in only two stories, $e_A e_{AB} e_E$ and $e_A e_{BA} e_E$. We computed optimal policies using both the monolithic and the decoupled approaches, and we used each computed policy in 100 simulations. Using the decoupled approach, the robot was able to capture $e_A e_{BA} e_E$ with style sequence $s_f s_r s_{fl}$ and $e_A e_{AB} e_E$ with style sequence $s_b s_l s_{fl}$, each one in 55 and 45 simulations, respectively. Both these two style sequences have an efficacy value of 0.1, which is optimal. Using the monolithic approach, the robot captured the style sequence $s_f s_l s_{fl}$ for story $e_A e_{AB} e_E$ and $s_f s_r s_{fl}$ for $e_A e_{BA} e_E$, which have efficacy values 0.05 and 0.1 respectively. The average style sequence efficacy value for those 100 simulations was 0.0755. This means that the decoupled approach did better than the monolithic approach in capturing stories with better qualities in terms of style sequence efficacy.

This result follows from the fact that in the decoupled approach, the robot has the freedom to choose the style after it observes that the event it predicted is occurring (it observes additional information about the state where the event is realized) while in the monolithic approach, the robot does not have such freedom and, in fact, it chooses an event and a style together.

For the second variation, the robot is tasked to capture a story specified by the story automaton in Fig. 3.7b. For 100 simulations using the decoupled approach: the robot captured either $e_{AB} e_{AF}$ with style sequence $s_l s_l$ or $e_{BA} e_{BF}$ with style sequence $s_r s_l$ in 43 simulations, and captured $e_{BA} e_{BF}$ with style sequence $s_r s_l$ in 57 simulations. The efficacy of these two sequences was 0.0105 and 0.0049, respectively. The monolithic approach captured $e_A e_B e_E$ with style $s_f s_r s_{fl}$, the accepted sequence efficacy of which is 0.1. In this scenario, the monolithic approach yielded a better style sequence.

This result is justified by the fact that, while the monolithic approach chooses a tuple of event-style that maximizes the excepted value of accepted sequence efficacy, the decoupled approach chooses styles to capture in the next time step, but for events that were chosen (independent of style considerations) to minimize the expected number of steps. Thus the monolithic solution chooses the longer event sequence, which provides better sequence efficacy, while the sequential approach captures the shorter event sequence, which can be captured in a shorter time.

In the third variation, the robot captures the sequence in the story automaton in Fig. 3.7c, Since the event sequence to be captured is fixed $e_A e_B e_E$, both approaches yielded identical style sequences ($s_f s_r s_{fl}$), having an efficacy of 0.1.

Figure 3.11 shows the times to compute the optimal policy for the monolithic and the decoupled approaches. Note that the state space of the monolithic MDP is $W \times S_A \times S_S$. In contrast, the decoupled solution has states $W \times S_A$ for computing the event policy and $S_J \times S_S$ for the style policy. These respective state spaces mean that the decoupled version scales better than the monolithic solution. The decoupled approach was consistently faster than the monolithic approach, with the difference becoming significant when the story automaton and the event model are large, leading to a prohibitively large product automaton.

3.3.2 Comparing decoupled solution to state-of-the-art RL techniques

In this section, we provide a comparison of the solution approaches mentioned in the chapter with state-of-the-art neural network approximate solution methods. To be specific, we provide a comparison with the A2C model [23] and the PPO model [24]. The comparison was made by first modeling the first variation of the model (story automaton in Figure 3.7a), discussed in the previous section, as an "OpenAI gym" [88] environment and using the default implementations of the A2C model (learning rate $= 0.0007$, discount $= 0.99$) and the PPO model (learning rate $= 0.0003$, discount $= 0.99$) from the "stable-baseline3" [89] package. The action choice for the agent for each step consisted of a discrete value, each value corresponding to an event-style pair. And at each step, the observation received by the agent consisted of the following:

1. current world model state;

2. current story automaton state;

3. previous style used (since the problem only uses a style-gram with $k = 2$); and

4. list of events that process the current story automaton state.

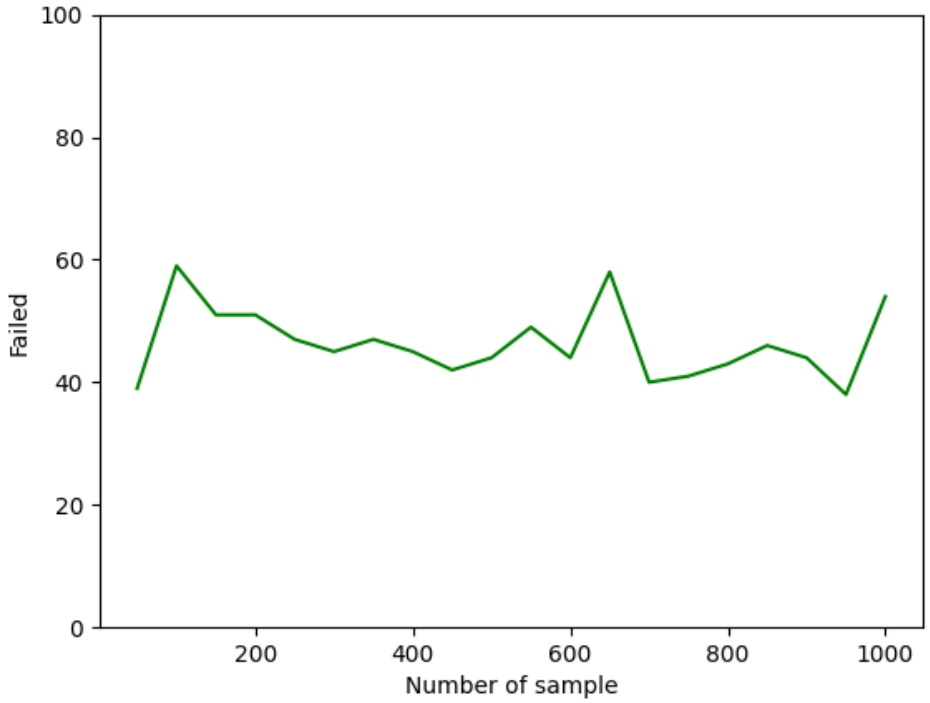


Figure 3.12: Plot showing the number of times, of the 100 simulations, the A2C model failed to capture the narrative with respect to the sample size used for training the model.

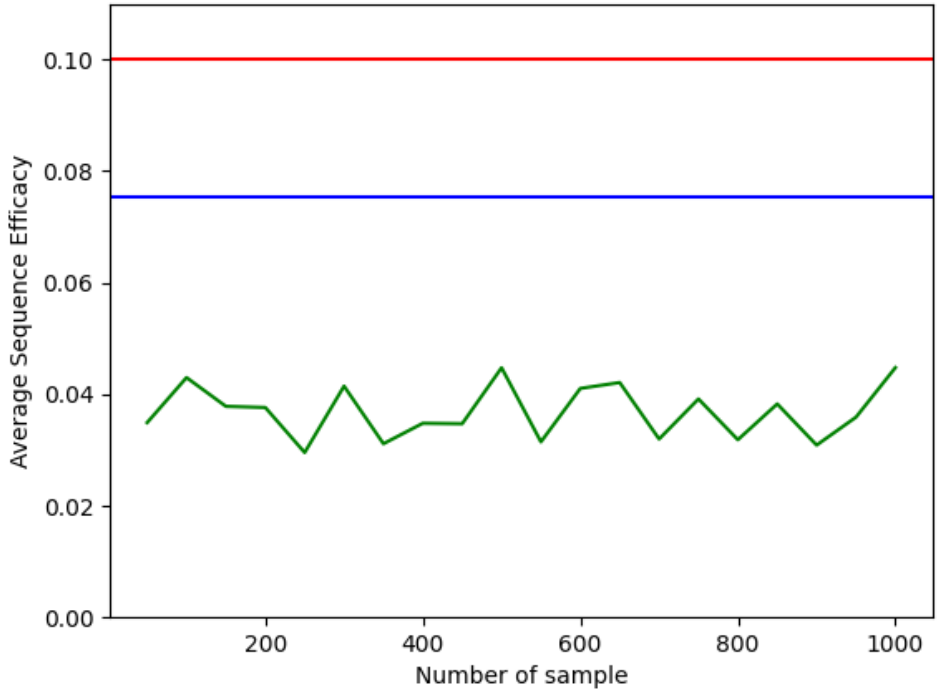


Figure 3.13: Plot showing the average efficacy of the successfully captured stylized narrative according to the policy generated by the A2C model with respect to the sample size used for training the model. Red line denotes the average efficacy for the decoupled solution. Blue line denotes the average efficacy for the monolithic solution.

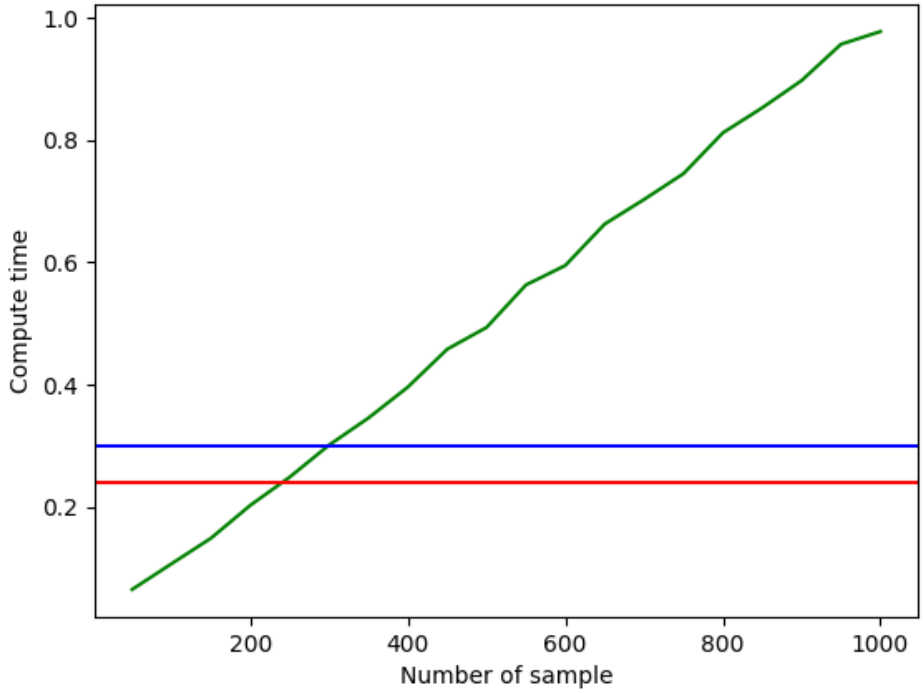


Figure 3.14: Plot showing computation time (seconds) taken by the A2C model to be trained with respect to the sample size used for training the model. Red line denotes the computation time for the decoupled solution. Blue line denotes the computation time for the monolithic solution.

The models were trained using sample size ranging from 50 to 1000 with 50 increments between each. Then each trained model was tested by running 100 simulations. The model is considered to have failed to produce a story if the narrative is not captured in 100 steps. For each simulation, the A2C model failed to capture the narrative multiple times (shown in Figure 3.12). The average efficacy value for the ones successfully captured is shown in Figure 3.13. The compute times are shown in Figure 3.14. For the PPO model, the number of failures, the average efficacy of the successfully captured narratives, and the compute time are shown in Figure 3.15, Figure 3.16, and Figure 3.17 respectively. From the results, we can clearly see that the solution approaches presented here perform superior as compared to both the models.

RL algorithms learn by exploring the state space investing more time in areas where high

precision in action space is needed. Since for these problems the actions have no influence over parts of the problem, a plausible explanation for the poor performance of the RL solution methods may be due to the non-causality of the problem formulation.

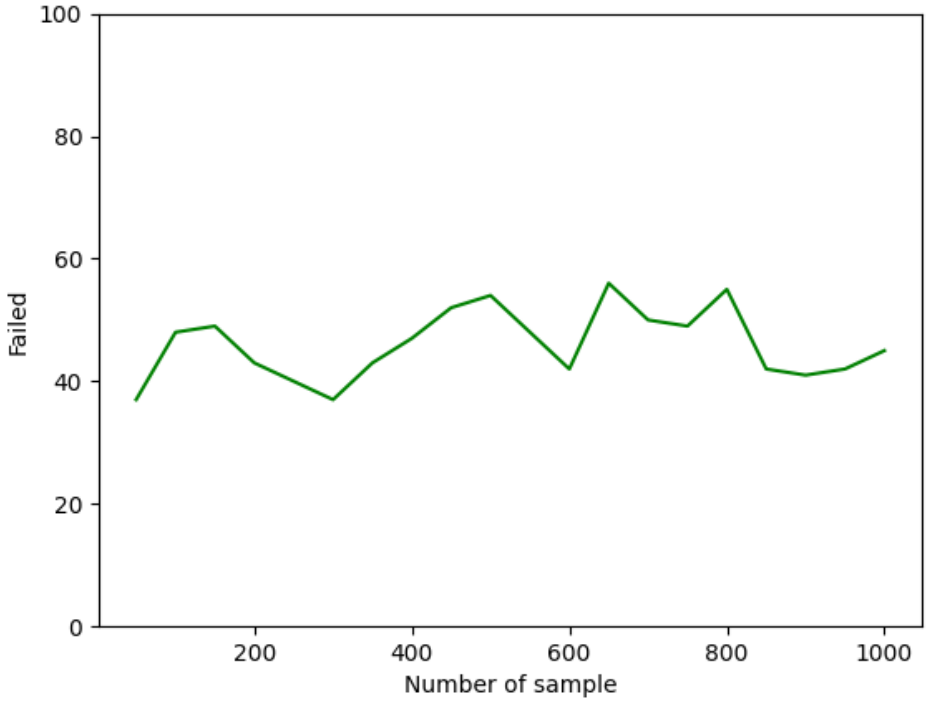


Figure 3.15: Plot showing the number of times, of the 100 simulations, the PPO model failed to capture the narrative with respect to the sample size used for training the model.

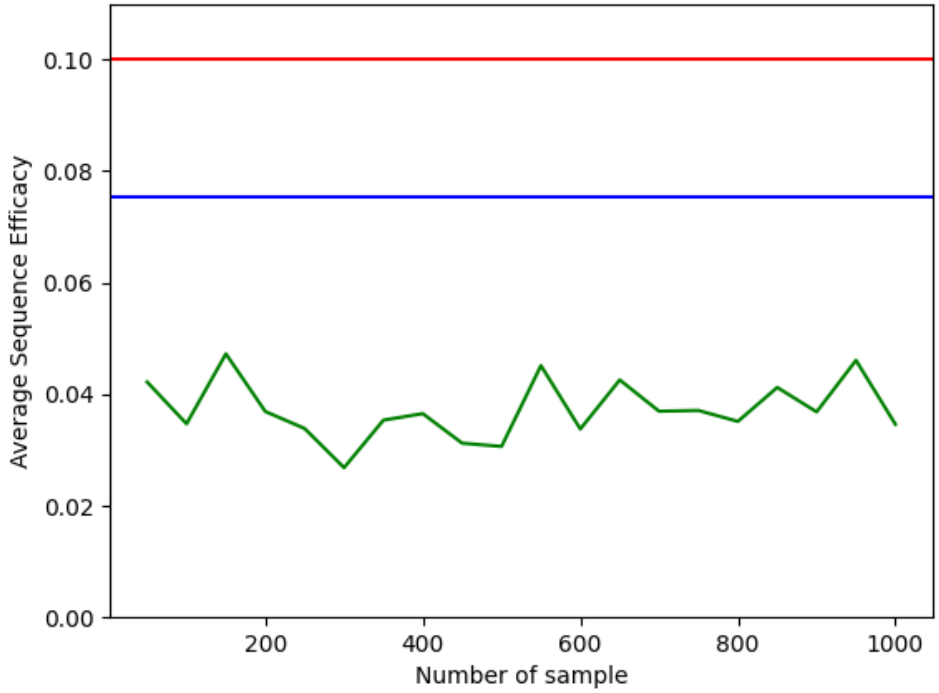


Figure 3.16: Plot (green) showing the average efficacy of the successfully captured stylized narrative according to the policy generated by the PPO model with respect to the sample size used for training the model. Red line denotes the average efficacy for the decoupled solution. Blue line denotes the average efficacy for the monolithic solution.

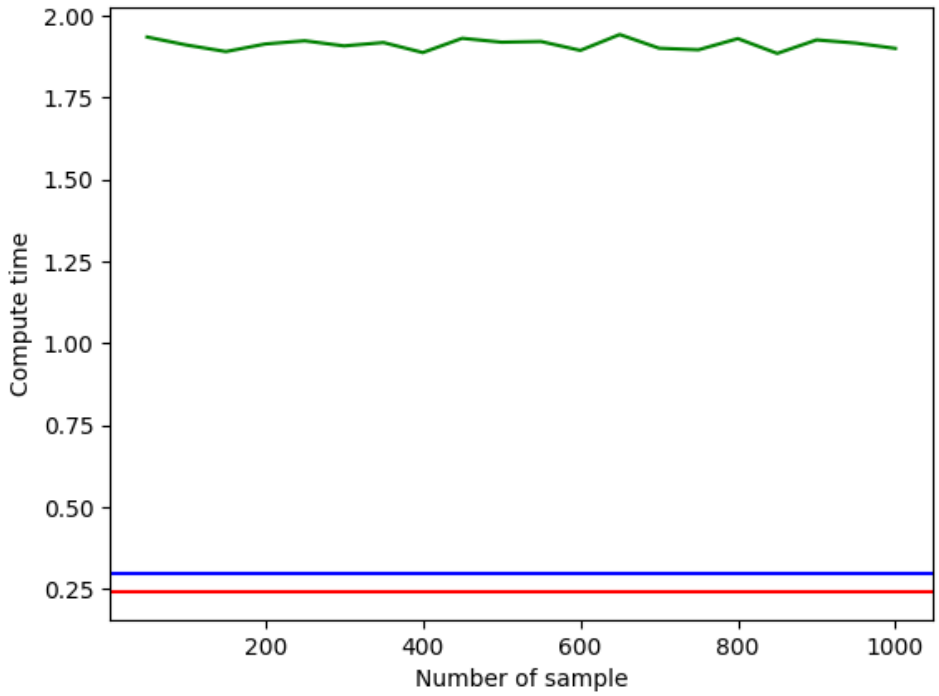


Figure 3.17: Plot (green) showing computation time (seconds) taken by the PPO model to be trained with respect to the sample size used for training the model. Red line denotes the computation time for the decoupled solution. Blue line denotes the computation time for the monolithic solution.

3.4 Extension: serendipity

In this section, we study an extension of the narrative problem, where an agent, while attempting to capture an event, can capture other events in the process. Such a scenario is not improbable but rather something that can happen often. To understand how and where such a situation might arise, consider the following from the Boston marathon example: while the agents attempt to capture an event required for the narrative, say Alice running past Boston College, the framing also includes another event of Bob tripping and falling. Thus, in this example, the robot manages to capture two events while attempting to capture one. Situations might also arise when attempting to capture an event: the agent captures some alternate events altogether. Here in this section, we extend the formulation to incorporate such situations.

Note that in such scenarios, one might ask the question as to what happens when the agent unintentionally captures some events that prevent the sequence from being in $\mathcal{L}(\mathcal{D})$. As mentioned previously, such edits, like ignoring unintentional event sub-sequences, can be encoded by mutating the story automaton to produce a new one [7]. Therefore, without any loss of generality, we assume that whatever post-production steps are required have already been expressed in $\mathcal{D}$.

Next, we provide the formulation necessary to incorporate the extension in Element-generating Markov chain (event model). We model this by substituting the occurrence labeling function in Definition 1 with the following *multi-occurrence function* $\mathbf{g} : W \times \mathbb{E} \to \Delta(2^{\mathbb{E}})$, such that, for $w \in W$, $\mathbf{e} \in 2^{\mathbb{E}}$ and $e \in \mathbb{E}$, $\mathbf{g}(w, e)(\mathbf{e})$ is the probability that all events in $\mathbf{e}$ will be captured and none of the events $\mathbb{E} \setminus \mathbf{e}$, when attempting to capture event e.

To demonstrate the solution approach that can be taken to solve problems involving the multi-occurrence function we introduce the following optimization problem:

Optimization Problem: Serendipitous Capture

Given: Event model with multi-occurrence function $\mathcal{M} = (W, \mathbb{E}, T, w_0, \mathbf{g})$ over event set $\mathbb{E}$, and a DFA $\mathcal{D} = (Q, \mathbb{E}, \delta, q_0, F)$,

Output: A policy $\pi_e : W \times Q \to \mathbb{E}$ that minimizes the expected number of steps k until the event sequence $\xi_k \in \mathcal{L}(\mathcal{D})$.

3.4.1 Solution

A conceptually straightforward solution approach is to search over an action space ($\mathbb{E}$), yielding the Markov Decision Process [5] described below:

Definition 9. *Given event model with multi-occurrence function $\mathcal{M} = (W, \mathbb{E}, T, w_0, \mathbf{g})$, and story automaton $\mathcal{D} = (Q, \mathbb{E}, \delta, q_0, F)$, construct MDP $\mathbb{M} = (X, x_0, \mathbb{E}, P, \mathcal{F}, J)$, where*

- $X \subseteq W \times 2^Q$ *is the finite state space;*
- $x_0 = (w_0, q_0)$ *is the initial state;*
- $\mathbb{E}$ *is the action space;*

- $P : X \times \mathbb{E} \times X \to [0, 1]$ *is the transition probability function, such that, for* $(w_i, \mathbf{q}_i), (w_j, \mathbf{q}_j) \in X$, $e \in \mathbb{E}$,

$$P((w_i, \mathbf{q}_i), e, (w_j, \mathbf{q}_j)) = T(w_i, w_j) \cdot \sum_{\mathbf{e} \in 2^{\mathbb{E}}} \mathbf{1}_{\mathbf{q}_j}(\mathbf{q}_i, \mathbf{e}) \mathbf{g}(w_j, e)(\mathbf{e})$$

where $\mathbf{1}_{\mathbf{q}_j}(\mathbf{q}_i, \mathbf{e}) = 1$*, if states in* $\mathbf{q}_j$ *can be reached from* $\mathbf{q}_i$ *using some or all events of the set* $\mathbf{e}$*, otherwise 0;*

- $\mathcal{F} = W \times \mathbf{F} \subseteq X$ *is the set of goal states, where* $\mathbf{q} \in \mathbf{F}$ *if* $\exists q \in \mathbf{q}$*. such that,* $q \in F$*; and*

- $J : X \times \mathbb{E} \to \mathbb{R}_{\geq 0}$ *is the cost function such that for* $(w_i, \mathbf{q}) \in X$, $e \in \mathbb{E}$, $J((w_i, \mathbf{q}), e) = 1$ *if* $(w_i, \mathbf{q}) \notin \mathcal{F}$*, otherwise 0.*

An optimal policy $\pi_e : X \to \mathbb{E}$ for this MDP provides the event policy.

The main difference between the MDP used to solve for serendipitous capture and the MDP for a non-serendipitous capture lies in the transition probability function. The non-serendipitous transition probability considers only two possibilities, either the event is captured or not, and based on that progresses the story automaton by a single state. On the other hand, the serendipitous capture scans over all the permutations of the events captured and may progress the story multiple state at a time. We use this property of the serendipitous capture to develop an approximate solution for the multi-agent version of the structured narrative problem (see Section 4.2.2).

3.5 Summary

In this chapter, we formulate the problem of autonomously capturing a stylistically sound and narratively coherent sequence of events in an uncertain environment by introducing the concept of an Element-generating Markov chain, Story Automaton, and quantifying the qualitative nature of cinematic techniques (styles) via a structure called the style-gram. Two solution approaches were presented, each with its respective merits. A traditional approach that solves for the event-style pair jointly. And a decoupled approach that solves for events first, then plan for styles. The second step of the decoupled approach grants extra flexibility by using freshly revealed informa-

tion of the event model state where the event was captured. Case studies demonstrate the merits of each approach and also show the performance of each as compared to state-of-the-art neural network approximation solutions, A2C and PPO. The chapter also introduces an extension to the problem for scenarios where attempting to capture an event can lead to serendipitous capture of other events. The next chapter extends the formulation to incorporate multiple agents, each with action constraints, working cooperatively to capture a narratively coherent sequence of events in an uncertain environment.

4. DECOUPLING ACROSS MULTIPLE AGENTS*

In Chapter 3, we analyzed planning for a single non-causal agent. In this chapter, we extend the formulation for multiple agents working in a non-causal environment. Here we formulate the problem of deploying multiple agents to overserve and record evolution of a stochastic process as seen in the previous chapter (event model) in order to output a sequence of observations that fit some given specifications (story automaton), We start by motivating the problem via the following scenario.

Imagine a nature documentary. Muffled, but in his signature rasping hush, David Attenborough intones: *"We see now the baby gazelle, utterly unaware of danger lurking close, as she edges toward the water's edge. Nearby, Mother gazelle is distracted, only for a moment, but..."* and the wild drama ensues—tooth, claw, and all. Later, as the credits roll by, it turns out that the rare footage making up this documentary was captured not by expert human camera operators, but by a team of autonomous videographer robots. These robots, aided by traditional tags for tracking animals, have only a coarse sense of the locations of certain animals and are only able to make imperfect predictions for what activities the creatures will engage in. But they are also given a description of the sorts of events that are worth capturing, events to help describe Nature's unfolding story. Multiple robots ought to be engaged to ensure good coverage of the district, especially as there may be events of interest occurring simultaneously at different locations, such as when the flamingos take wing all at once, while a lone cheetah breaks cover from a thicket of trees elsewhere (See Figure 4.1). The robots plan their movements and capture events strategically so that, ultimately, the captured events form a depiction that is an engaging record of activity within the wildlife reserve. For example, one robot captures a majestic predator silhouetted against the moon, other robots capture (many) scenes of frolicking young. Comparatively little footage represents animals lolling about during the heat of the day.

*Reprinted with permission from "Tractable Planning for Coordinated Story Capture: Sequential Stochastic Decoupling" by Diptanil Chaudhuri, Hazhar Rahmani, Dylan Shell and Jason M. O'Kane. *2021 Proceedings of International Symposium on Distributed Autonomous Robotic Systems*. Copyright 2022 by *Springer*.

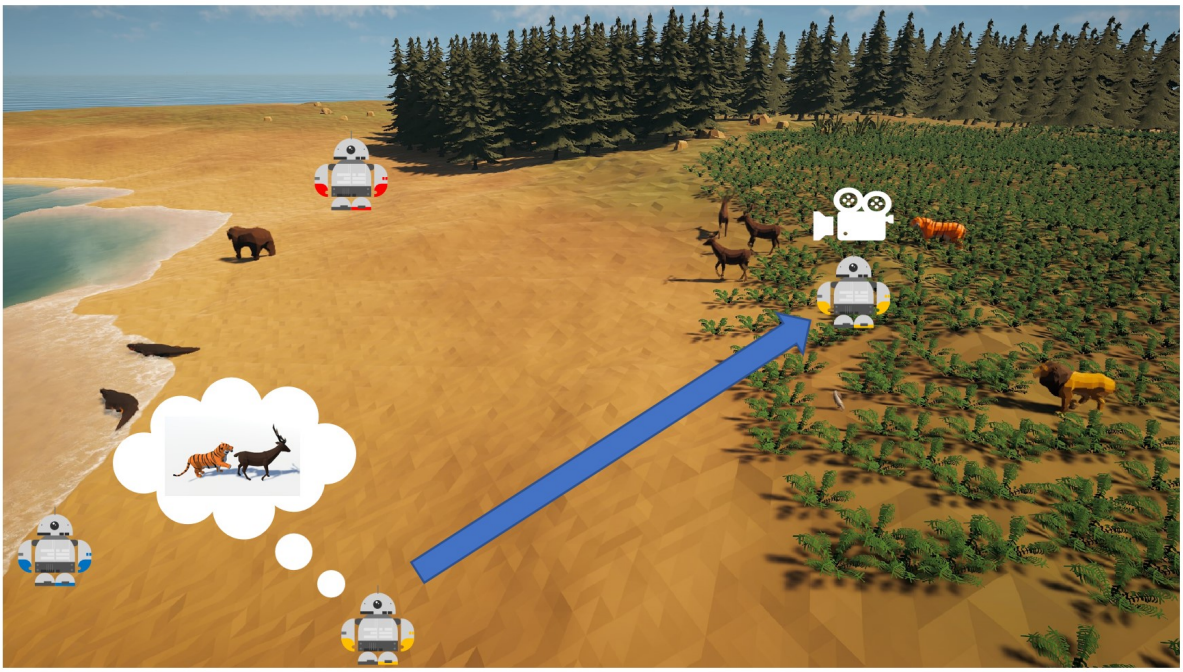


Figure 4.1: An example of a wildlife sanctuary with a group of three robots attempting to record a documentary.

The chapter formalizes settings like the preceding scenario and focuses on the question of how to compute effective multi-robot plans. We present two different approaches to solve the optimization problem. A joint product treatment, involving the optimal selection of joint choices, i.e., choosing the next elements to attempt to capture by all robots, is formulated where costs are minimized in an expected sense. Since such plans are prohibitive to compute, variants based on an approximation scheme based on solving a sequence of individual planning problems are then introduced. The approximate solution decouples the planning for each agent based on the plans for the agents that were solved before it. This scheme sacrifices some solution quality but requires far less computational expense; we show this permits one to scale to greater numbers of robots.

4.1 Problem definition

For this problem, we assume the world model (event model) and the narrative to be formulated as defined in 3.1.2, $\mathcal{M} = (W, \mathbb{E}, T, w_0, g)$ and $\mathcal{D} = (Q, \mathbb{E}, \delta, q_0, F)$, with $\mathbb{E}$ being the set of all possible events. In this chapter, we extend the formulation to incorporate multiple agents along

with their constraints. The following section provides the formal definition of the agent model.

4.1.1 Agent model

The current state of the event model is assumed to be observable to all the agents, i.e., at each time t; the agents know the current state of the event model (or the world) w_t; however, they do not know what the next state, w_{t+1}, will be.

To cooperatively capture a sequence of events each of the n agents chooses an action to execute from $\mathcal{A}$, the set of all possible actions. Each action is associated with the event they aim to capture via the *recording function*, $r : \mathcal{A} \to \mathbb{E} \cup \{\epsilon\}$. Since some actions may not involve any recording, the symbol ϵ is included, indicating that the associated action will never capture any event. At every time step, each agent executes an action from $\mathcal{A}$, if that action aims to capture an event and that event occurs during the execution of the action; the agent will succeed in capturing that event. We assume every event can be recorded by some action, i.e., for every $e \in \mathbb{E}$, there is some action $a_e \in \mathcal{A}$ such that $r(a_e) = e$. Further, the set of actions includes a *no-action* choice, $\perp \in \mathcal{A}$, that does nothing and records no event, $r(\perp) = \epsilon$. Occasionally we will apply $r(\cdot)$ to a tuple in a point-wise fashion.

Owing to constraints, present either in the world or in the way the agents interact with the world, not all actions can be executed at all times. Hence, an action $a \in \mathcal{A}$ with $r(a) = \epsilon$ may still be useful because, though it won't capture an event itself, it may alter what can be captured subsequently. Think, for instance, of an agent using the time step's duration to shift location, or to deploy a stalking horse. The following structure expresses such constraints and also associates costs to each action.

Definition 10 (Valid-action Automaton (VA))**.** *For agent $i \in \{1, \ldots, n\}$, we define its* valid-action automaton *as a 5-tuple, $\mathcal{D}_V^{(i)} = (Q_V^{(i)}, v_0^{(i)}, \mathcal{A}, \delta_V^{(i)}, J_V^{(i)})$,*

- *$Q_V^{(i)}$ is the set of vertices;*
- *$v_0^{(i)} \in Q_V^{(i)}$ is the initial vertex;*
- *$\mathcal{A}$, its alphabet, a set of all possible agent's actions;*

- $\delta_V^{(i)} : Q_V^{(i)} \times \mathcal{A} \hookrightarrow Q_V^{(i)}$*, which could be partial, is the transition function; and*
- $J_V^{(i)} : Q_V^{(i)} \times \mathcal{A} \to \mathbb{R}_{>0} \cup \{+\infty\}$ *is the cost function, such that for each* $(v^{(i)}, a) \in Q_V^{(i)} \times \mathcal{A}$*,* $J_V^{(i)}(v^{(i)}, a)$ *is the cost of taking action* a *at vertex* $v^{(i)}$*. We assume that* $J_V^{(i)}(v^{(i)}, a) = +\infty$ *for any* $(v^{(i)}, a)$ *such that* $\delta_V^{(i)}(v^{(i)}, a)$ *is not defined.*

Each agent $i \in \{1, 2, \ldots, n\}$ keeps track of the current state of its own valid-action automaton, denoted $v_t^{(i)}$. Actions are performed as follows. Agent i makes a choice, from among those actions a for which $\delta_V^{(i)}(v_t^{(i)}, a)$ is defined, to enact at time $t + 1$. We denote the action $a_t^{(i)}$, because it is chosen at time t. The world evolves from w_t to w_{t+1}, and agent i pays cost $J_V^{(i)}(v_t^{(i)}, a_t^{(i)})$ executing $a_t^{(i)}$ to change its circumstances, with aspects relevant for subsequent actions being represented in $v_{t+1}^{(i)}$. Finally, if $r(a_t^{(i)}) \neq \epsilon$, the agent attempts to record event $r(a_t^{(i)}) \in \mathbb{E}$, which succeeds with probability $g(w_{t+1}, r(a_t^{(i)}))$.

For each time step $t \geq 1$, we define $\xi_t \subseteq \mathbb{E}^{\leq n}$ to be the set of all event sequences, in any order, formed from all the events that were captured by the agents at time step t. For example, if at time step t_0, e_1 was captured by agent 1, e_2 was captured by agent 2, and ϵ (nothing) was captured by agent 3, then $\xi_{t_0} = \{e_1e_2, e_2e_1\}$. We also let $\Xi_t = \prod_{i=1}^{t} \xi_i$ be the set of all event sequences obtained by concatenating the event sequences made for the time steps $1, \ldots, t$. As an example, if $\xi_1 = \{e_3, e_4e_2\}$ and $\xi_2 = \{e_1e_2, e_2e_1\}$, then $\Xi_2 = \{e_3e_1e_2, e_3e_2e_1, e_4e_2e_1e_2, e_4e_2e_2e_1\}$. The agents check at each time t, if there exists an event sequence $\xi \in \Xi_t$ such that $\xi \in \mathcal{L}(\mathcal{D})$ or not. If yes, then it means that the agents have successfully collected events to make a desired story, namely ξ, and they terminate. Note that Ξ_t is the set of all event sequences the agents can make by concatenating all the events they have captured until time step t with the constraint that for times t_1 and t_2 for which $t_1 < t_2$, no event captured at t_2 precedes an event captured at t_1.

4.1.2 Policies and problem statement

The agents' choice of actions is governed by a policy $\pi(\cdot, \cdot, \cdot)$. The policy, at time t, is based on the current state of the event model, states in the story automaton, and current states in the agents' valid-action automata. It produces an n-tuple of actions, termed a joint action, telling each agent

what action to execute.

Given valid-action automata $\mathcal{D}_V^{(i)} = (Q_V^{(i)}, v_0^{(i)}, \mathcal{A}, \delta_V^{(i)}, J_V^{(i)})$, $i \in \{1, \ldots, n\}$, we will write $\mathbf{Q}_V = Q_V^{(1)} \times \cdots \times Q_V^{(n)}$. Similarly, for joint actions, we have $\mathbf{A} = \mathcal{A} \times \cdots \times \mathcal{A} = \mathcal{A}^n$. (To lighten the notation, we assume that $\mathcal{A}$ is identical for every agent; no generality is lost because, should agent i be unable to execute some $a \in \mathcal{A}$, then a simply does not appear in $Q_V^{(i)}$.)

Given $Q_V^{(i)}$ for $i \in \{1, \ldots, n\}$, we define $\mathbf{J}_V : \mathbf{Q}_V \times \mathbf{A} \to \mathbb{R}_{>0}$, the *aggregate cost function*, by $\mathbf{J}_V((v^{(1)}, \ldots, v^{(n)}), (a_1, \ldots, a_n)) = \sum_{i=1}^{n} J_V^{(i)}(v^{(i)}, a_i)$. Then the total cost incurred up until time t is $\mathbf{J}_{\text{tot}}^t = \sum_{i=0}^{t-1} \mathbf{J}_V((v_t^{(1)}, \ldots, v_t^{(n)}), (a_t^{(1)}, \ldots, a_t^{(n)}))$. Let T be the first time such that $\mathbf{J}_{\text{tot}}^T \cap \mathcal{L}(\mathcal{D}) \neq \emptyset$, then we say the story has been captured at time T and we write the cost of capturing the story as $\mathbf{J}_{\text{tot}}^T$.

We now define the problem we study in this chapter.

Problem: Multi-Robot Recording Cost Minimization (MRRCM)

Given: An event set $\mathbb{E}$; an event model $\mathcal{M} = (W, \mathbb{E}, T, w_0, g)$; the story automaton $\mathcal{D} = (Q, \mathbb{E}, \delta, q_0, F)$; the number of agents, n; a set of n valid-action automata $\mathcal{D}_V^{(i)} = (Q_V^{(i)}, v_0^{(i)}, \mathcal{A}, \delta_V^{(i)}, J_V^{(i)}), \forall i \in \{1, \cdots n\}$.

Output: A policy, $\pi^* : W \times 2^Q \times \mathbf{Q}_V \to \mathbf{A}$, that minimizes the expected cost, $\mathbf{J}_{\text{tot}}$, to capture a story in $\mathcal{L}(\mathcal{D})$.

4.2 Solution approaches

In the initial step, the algorithm makes use of the story automaton and n, the number of agents, to construct a *footage automaton* as follows:

Definition 11 (Footage Automaton)**.** *Let $\mathcal{D} = (Q, \mathbb{E}, \delta, q_0, F)$ be the story automaton and n be the number of agents. We construct the* footage automaton *as a nondeterministic finite automaton (NFA) $\mathbf{N} = (Q, q_0, \mathbf{E}, \delta_{\mathbf{N}}, F)$, where*

- *Q is the state space;*
- *q_0 is the initial state*

- $\mathbf{E} = \{(e^{(1)}, \ldots, e^{(n)}) \mid e^{(i)} \in \mathbb{E} \cup \{\epsilon\}, \forall i \in \{1, \ldots, n\}\}$ *are its alphabet;*

- $\delta_{\mathbf{N}} : Q \times \mathbf{E} \hookrightarrow 2^{Q}$*, is the transition function, such that for* $q \in Q$ *and* $(e^{(1)}, \ldots, e^{(n)}) \in \mathbf{E}$*,* $\delta_{\mathbf{N}}(q, (e^{(1)}, \ldots, e^{(n)})) = \{q_i | q_i \in Q$*, where, in* $\mathcal{D}$*,* q_i *is reachable from any state in* q *using some permutation of the tuple* $(e^{(1)}, \ldots, e^{(n)})\}$*; and*

- F *is the set of final states.*

Each transition starting from a state in the footage automaton corresponds to at most n consecutive transitions starting from that state in the story automaton. The idea is that the footage automaton tracks the story automaton states which can be reached using the events captured by all of the agents. The next step converts the footage automaton $\mathbf{N}$ into a *deterministic footage automaton* $\mathbf{D} = (\mathbf{Q}, \mathbf{E}, \delta_{\mathbf{D}}, q_0, \mathbf{F})$, which is, in fact, a deterministic finite automaton, using the well-known technique of NFA to DFA conversion [9]. The number of edges in the constructed $\mathbf{N}$, of the output $\mathbf{D}$, and the work needed in this conversion step, can be reduced by fixing a canonical representative, equivalent up to permutation, for the n-tuples comprising $\mathbf{E}$. Since the transition function accepts any combination of the events captured by all agents (ordering does not matter), sorting the tuples works.

We define a function $h : \mathbf{A} \times 2^{\mathbb{E}} \to (\mathbb{E} \cup \{\epsilon\})^n$ such that for each joint action $\mathbf{a} \in \mathbf{A}$, and a set of events $B \subseteq \mathbb{E}$, $h(\mathbf{a}, B) = (d^{(1)}, \ldots, d^{(n)})$ in which for each $j \in \{1, \ldots, n\}$, $d^{(j)} = r(a^{(j)})$ if $r(a^{(j)}) \in B$, otherwise $d^{(j)} = \epsilon$ (being consistent with that above, we use $a^{(j)}$ to denote the j^{th} element of $\mathbf{a}$). Intuitively, given that only the events in B happen, the function h outputs an n-tuple of events which are captured by the action $\mathbf{a}$. We then define $o : \mathbf{A} \to 2^{(\mathbb{E} \cup \{\epsilon\})^n}$ such that for each $\mathbf{a} \in \mathbf{A}$, $o(\mathbf{a}) = \bigcup_{B \subseteq \mathbb{E}} \{h(\mathbf{a}, B)\}$. This function produces any tuple of events that could be captured by a joint action.

Two additional functions will be needed. Let $\varrho : \mathbf{A} \times w \times (\mathbb{E} \cup \{\epsilon\})^n \hookrightarrow \mathbb{R}_{\geq 0}$ be a partial function such that for each $\mathbf{a} \in \mathbf{A}$, $w \in W$, and $\mathbf{b} \in o(\mathbf{a})$,

$$\varrho(\mathbf{a}, w, \mathbf{b}) = \left(\prod_{e_0 \in \mathbf{b}} g(w, e_0) \right) \cdot \left(\prod_{\substack{e_1 \in r(\mathbf{a}) \\ e_1 \notin \mathbf{b}}} (1 - g(w, e_1)) \right).$$

The interpretation is: assuming that at time t the agents execute joint action $\mathbf{a}$ and, at $t+1$, the event model transitions to w, then $\varrho(w, \mathbf{a}, \mathbf{b})$ gives the probability that $\mathbf{b}$ is realized by w. Or, in other words, among those events attempted to be captured by $\mathbf{a}$, only those within $\mathbf{b}$ happened in state w.

Next, using $\bar{\mathbf{1}}_I(\cdot)$ for set I's indicator function, let $\lambda : \mathbf{Q} \times \mathbf{A} \times W \times \mathbf{Q}$ be

$$\lambda(\mathbf{q}, \mathbf{a}, w, \mathbf{q}') = \sum_{\mathbf{b} \in o(\mathbf{a})} \bar{\mathbf{1}}_{\{\mathbf{q}'\}}(\delta_{\mathbf{D}}(\mathbf{q}, \mathbf{b})) \cdot \varrho(\mathbf{a}, w, \mathbf{q}).$$

At time t, if the footage automaton is in state q and the agents execute $\mathbf{a}$, and thereupon the event model transitions next to state w, then $\lambda(\mathbf{q}, \mathbf{a}, w, \mathbf{q}')$ is the probability that the footage automaton transitions to $\mathbf{q}'$ at $t+1$.

With these definitions, we now present our algorithms for solving MRRCM.

4.2.1 Joint solution

The first step of the algorithm makes from valid-action automata of the agents, an automaton defined as follows:

Definition 12 (Joint Action Automaton (JA))**.** *Given the valid-action automatons $\mathcal{D}_V^{(i)}$ for $i \in \{1, 2, \ldots n\}$, and the aggregate cost function $\mathbf{J}_V : \mathbf{Q}_V \times \mathbf{A} \to \mathbb{R}_{>0}$, their* joint action automaton *is $\mathcal{D}_{\mathbf{V}} = (\mathbf{Q}_V, \mathbf{j}_0, \mathbf{E}, \delta_{\mathbf{V}}, \mathbf{J}_V)$, where:*

- $\mathbf{Q}_V = Q_V^{(1)} \times Q_V^{(2)} \times \cdots \times Q_V^{(n)}$ *is the set of all the vertices;*
- $\mathbf{j}_0 = (v_0^{(1)}, v_0^{(2)}, \ldots, v_0^{(n)})$ *is the initial vertex;*
- $\mathbf{E} = \mathcal{A}^n$ *is the set of all actions;*
- $\delta_{\mathbf{V}} : \mathbf{Q}_V \times \mathbf{E} \hookrightarrow \mathbf{Q}_V$ *is the valid transitions function, such that for each* $(v^{(1)}, v^{(2)}, \ldots, v^{(n)})$, $(w^{(1)}, w^{(2)}, \ldots, w^{(n)}) \in \mathbf{Q}_V$ *and* $(a^{(1)}, a^{(2)}, \ldots, a^{(n)}) \in \mathbf{A}$, $\delta_{\mathbf{V}}((v^{(1)}, v^{(2)}, \ldots, v^{(n)}), (a^{(1)}, a^{(2)}, \ldots, a^{(n)})) = (w^{(1)}, w^{(2)}, \ldots, w^{(n)})$ *where for each* $i \in \{1, \ldots, n\}$, $\delta_V^{(i)}(v^{(i)}, a^{(i)}) = w^{(i)}$;

- $\mathbf{J}_V : \mathbf{Q}_V \times \mathbf{E} \to \mathbb{R}_{>0}$ *is the aggregate cost function.*

Now, to solve the MRRCM problem jointly for all the agents, we search over all a joint action space. To do so we construct an MDP, called the *joint MDP*.

Definition 13 (Joint MDP)**.** *For event set* $\mathbb{E}$ *and model* $\mathcal{M} = (W, \mathbb{E}, T, w_0, g)$*, joint action automaton* $\mathcal{D}_{\mathbf{V}} = (\mathbf{Q}_V, \mathbf{j}_0, \mathbf{E}, \delta_{\mathbf{V}}, \mathbf{J}_V)$*, and the deterministic footage automaton* $\mathbf{D} = (\mathbf{Q}, \mathbf{E}, \delta_{\mathbf{D}}, q_0, \mathbf{F})$*, construct* $\mathbb{M}_{\mathcal{M},\mathbf{D},\mathcal{D}_{\mathbf{V}}} = (X_{\mathcal{M},\mathbf{D},\mathcal{D}_{\mathbf{V}}}, x_0, \mathbb{A}, P_{\mathbf{J}}, J_{\mathbf{J}}, \mathcal{F}_{\mathbf{J}})$*, where*

- $X_{\mathcal{M},\mathbf{D},\mathcal{D}_{\mathbf{V}}} \subseteq W \times \mathbf{Q} \times \mathbf{Q}_V$*, is the set of states;*
- $x_0 = (w_0, q_0, \mathbf{j}_0) \in X_{\mathcal{M},\mathbf{D},\mathcal{D}_{\mathbf{V}}}$ *is the initial state;*
- $\mathbb{A} : X_{\mathcal{M},\mathbf{D},\mathcal{D}_{\mathbf{V}}} \to 2^{\mathbf{A}}$ *is the action function, such that for each* $x = (w, \mathbf{q}, \mathbf{j}) \in X_{\mathcal{M},\mathbf{D},\mathcal{D}_{\mathbf{V}}}$*,* $\mathbb{A}(x) = \{\mathbf{a} \in \mathbf{A} \mid \delta_{\mathbf{D}}(\mathbf{q}, r(\mathbf{a}))$ *and* $\delta_{\mathbf{V}}(\mathbf{j}, \mathbf{a})$ *are defined*$\}$*;*
- $P_{\mathbf{J}} : X_{\mathcal{M},\mathbf{D},\mathcal{D}_{\mathbf{V}}} \times \mathbf{A} \times X_{\mathcal{M},\mathbf{D},\mathcal{D}_{\mathbf{V}}} \to [0, 1]$*, is the probability function,*

$$P_{\mathbf{J}}\left((w, \mathbf{q}, \mathbf{j}), \mathbf{a}, (w', \mathbf{q}', \mathbf{j}')\right) = \bar{\mathbf{1}}_{\{\mathbf{j}'\}}(\delta_{\mathbf{V}}(\mathbf{j}, \mathbf{a})) \cdot T(w, w') \cdot \lambda(\mathbf{q}, \mathbf{a}, w', \mathbf{q}');$$

- $J_{\mathbf{J}} : X_{\mathcal{M},\mathbf{D},\mathcal{D}_{\mathbf{V}}} \times \mathbf{A} \to \mathbb{R}$*, is the cost function,* $J_{\mathbf{J}}\left((w, \mathbf{q}, \mathbf{j}), \mathbf{a}\right) = \bar{\mathbf{1}}_{\mathbf{F}}(\mathbf{q}) \cdot \mathbf{J}_V(\mathbf{j}, \mathbf{a})$*;*
- $\mathcal{F}_{\mathbf{J}} = W \times \mathbf{F} \times \mathbf{Q}_V$ *is the set of goal states.*

Note that this construction is a goal MDP, meaning it is an MDP supplemented with a set of goal states. The MRRCM problem then is reduced to finding for this MDP, a policy that minimizes the expected cost of reaching goal states. Such a policy, denoted $\pi^*_{\mathbb{M}}$, is a function $\pi^*_{\mathbb{M}} : X_{\mathcal{M},\mathbf{D},\mathcal{D}_{\mathbf{V}}} \to \mathbf{A}$ which gives the optimal action for each $x \in X_{\mathcal{M},\mathbf{D},\mathcal{D}_{\mathbf{V}}} \setminus \mathcal{F}_{\mathbf{J}}$ using the Bellman equation, which may be computed by using dynamic programming methods such as value iteration, or policy iteration [87].

Note that for all $x \in \mathcal{F}_{\mathbf{J}}$, $V^*_{\mathbb{M}}(x) = 0$. As each state $x \in X_{\mathcal{M},\mathbf{D},\mathcal{D}_{\mathbf{V}}}$ is a triple $(w, \mathbf{q}, \mathbf{j}) \in W \times \mathbf{Q} \times \mathbf{Q}_V$, and each state $\mathbf{q}$ of the deterministic footage automaton corresponds to a set of states of the story automaton, the computed policy $\pi^*_{\mathbb{M}}$ is a solution to MRRCM.

This MDP has a state space of size $\Theta(|W||\mathbf{Q}||Q_V^{(1)}||Q_V^{(2)}|\cdots|Q_V^{(n)}|)$ and an action space of size $|\mathcal{A}|^n$. As the number of agents increases, both the state space and the action space of the MDP grows exponentially. Because of this, the next section pursues a solution to the MRRCM problem with less expense.

4.2.2 Sequential solution

The overall idea of our second algorithm, following the classical approach in multi-agent planning [90], is to solve a sequence of MDPs, each being considerably smaller than the full joint MDP. Each MDP is constructed for a single agent in the team, the structure of each conditioned on the optimal policies of those preceding it in the sequence. Each is a goal MDP and, hence, an optimal policy can be computed via Bellman recurrences. Suppose that the n agents are ordered: $b_1 b_2 \ldots b_n$, where $b_k \in \{1, \ldots, n\}$. If $b_1, \ldots, b_{k-1}$ have determined how they will act, robot b_k can solve an MDP with stochastic transitions incorporating the events that the other $k-1$ might record, along with the associated probabilities of the events actually occurring, as gratis contributions. Once agent b_k solves this to obtain a policy, we have policies for the first k agents, and could proceed onward to agent b_{k+1}. And so on, until b_n.

The difficulty is that, even if the $k-1$ agents do have policies, those policies involve states within $\mathcal{D}_V^{(b_1)}, \mathcal{D}_V^{(b_2)}, \ldots, \mathcal{D}_V^{(b_{k-1})}$, which is information that agent b_k is not privy to, so policies will not give a determination of the actions of the first $k-1$ agents. Even if the agent had that information —obtained, say, by copious broadcast communication— this would still yield a policy for b_k as a function over $W \times \mathbf{Q} \times \mathcal{D}_V^{(b_1)} \times \cdots \mathcal{D}_V^{(b_k)}$, which grows exponentially in n in the worse case.

We pursue the following alternative, with more attractive scaling properties. For agent b_k, we compute a policy over state space $W \times \mathbf{Q} \times \mathcal{D}_V^{(b_k)}$. The construction of the b_k's MDP is as follows:

Definition 14. *For agent b_k, event model $\mathcal{M} = (W, \mathbb{E}, T, w_0, g)$, deterministic footage automaton $\mathbf{D} = (\mathbf{Q}, \mathbf{E}, \delta_{\mathbf{D}}, q_0, \mathbf{F})$, valid-action automaton for the robot $\mathcal{D}_V^{(b_k)} = (Q_V^{(b_k)}, v_0^{(b_k)}, \mathcal{A}, \delta_V^{(b_k)}, J_V^{(b_k)})$, policies π_{b_m} for $m \in \{1, \ldots, k-1\}$, and a distribution over the valid-action automata states for the $k-1$ agents, $\Delta^{(b_m)} : Q_V^{(b_m)} \to [0,1]$ for $m \in \{1, \ldots, k-1\}$, we construct the sequential MDP $\mathbb{M}^{(b_k)} = (X^{(b_k)}, x_0^{(b_k)}, \mathcal{A}, P^{(b_k)}, J^{(b_k)})$, where*

- $X^{(b_k)} \subseteq W \times \mathbf{Q} \times Q_V^{(b_k)}$ *is the state space;*

- $x_0^{(b_k)} = (w_0, q_0, v_0^{(b_k)}) \in X^{(b_k)}$ *is the initial state;*

- $\mathcal{A}$ *is the action space;*

- $P^{(b_k)} : X^{(b_k)} \times \mathcal{A} \times X^{(b_k)} \to [0, 1]$ *is the probability function such that*

$$P^{(b_k)}(x, a, x') = \bar{\mathbf{1}}_{\{v'\}}(\delta_V^{(b_k)}(v, a)) \sum_{\boldsymbol{\alpha} \in \mathbf{A}_a^k} T(w, w') \mu_{w,\mathbf{q}}(\boldsymbol{\alpha}) \lambda(\mathbf{q}, \boldsymbol{\alpha}, w', \mathbf{q}')$$

  with $x = (w, \mathbf{q}, v)$, $x' = (w', \mathbf{q}', v')$, *and where* $\mu_{w,\mathbf{q}} : \mathbf{A}_a^k \to [0, 1]$ *is*

$$\mu_{w,\mathbf{q}}(\boldsymbol{\alpha}) = \sum_{\substack{v^{(1)} \in Q_V^{(b_1)} \\ \vdots \\ v^{(k-1)} \in Q_V^{(b_{k-1})}}} \prod_{m=1}^{k-1} \left(\bar{\mathbf{1}}_{\{a^{(m)}\}}(\pi_{b_m}(w, \mathbf{q}, v^{(m)})) \Delta^{(b_m)}(v^{(m)}) \right)$$

  and $\mathbf{A}_a^k$ *consists of joint actions* $(a^{(1)}, a^{(2)}, \ldots, a^{(k-1)}, a, \overbrace{\perp, \perp, \cdots, \perp}^{n-k})$;

- $J^{(b_k)} : X^{(b_k)} \times \mathcal{A} \to \mathbb{R}$ *is the cost function, such that for* $x = (w, \mathbf{q}, v) \in X^{(b_k)}$ *and* $a \in \mathcal{A}$, *we have* $J^{(b_k)}(x, a) = (1 - \bar{\mathbf{1}}_{\mathbf{F}}(\mathbf{q}))\, J_V^{(b_k)}(v, a)$.

The intuition here is that, in lieu of actual information on the state of each $\mathcal{D}_V^{(b_k)}$, estimates (in the form of distribution $\Delta^{(b_k)}$) are used as an approximation. In what follows, we make a maximum entropy assumption over the states of the valid-action automata, i.e.,$\forall v \in Q_V^{(i)}$, $\Delta^{(i)}(v) = \frac{1}{|Q_V^{(i)}|}$, though cleverer choices may exist.

Based on this treatment, one expects that the ordering of the agents would affect the overall solution quality. Though a random order will work, it may fail to give a good policy so we employ the following greedy heuristic to choose a favorable ordering. First, we calculate the policies for all n agents tentatively assuming each would be operating alone. Then we select the agent whose individual policy gives the least expected cost to capture the story, and use it as the agent for the first spot. Having determined b_1, we compute policies for the remaining $n - 1$ agents, given b_1 and

its policy π_{b_1}. The agent with the least expected cost becomes b_2, and the process is repeated but now with $\{b_1, b_2\}$ determined. This is repeated until all n have been ordered.

4.2.3 Observability

Next, for the solution approaches, let us examine the problem parameters that are observable to the agents. For this problem, all the solution approaches assume that the agent can observe the current event model state, the current story automaton state, and all the events captured by all the agents till that time. Since the joint solution solves for all the agents simultaneously, it observes the current valid-action automaton state of all the agents jointly. In contrast, the sequential solution observes the valid-action automaton state for a single agent whose policy is being calculated. Both the solution approaches use the observations to assign events to each agent they need to capture. Unlike the decoupled solution presented in the previous chapter, the sequential (decoupled) solution here does not utilize any additional information the joint solution is unaware of.

In the following section, we provide simulated case studies, comparing the three solution approaches (joint solution, sequential solution with greedy ordering, and sequential solution with random ordering). We postulate that greedy ordering performs better compared to random ordering. The data presented below shows that this is indeed the case.

4.3 Results

In this section, we present results of our Python implementation of the algorithms, which we executed on an Ubuntu 16.04 computer with a 3.6GHz CPU.

4.3.1 Sequential vs joint solution

We revisit the shooting of documentary in a wildlife reserve as used as motivation initially, and outlined in Figure 4.2. A system-level event model is constructed from event models for the animals. Figure 4.3 shows, for each type of animal, the Markov Chain associated with its event model, in which each edge is the probability that the animal(s) go from the current location l_1, to l_2 in the next hour. Based on these event models, we see the flamingos are only interested in the lake, while the crocodile is interested in both the river and the lake. The others roam more widely. The

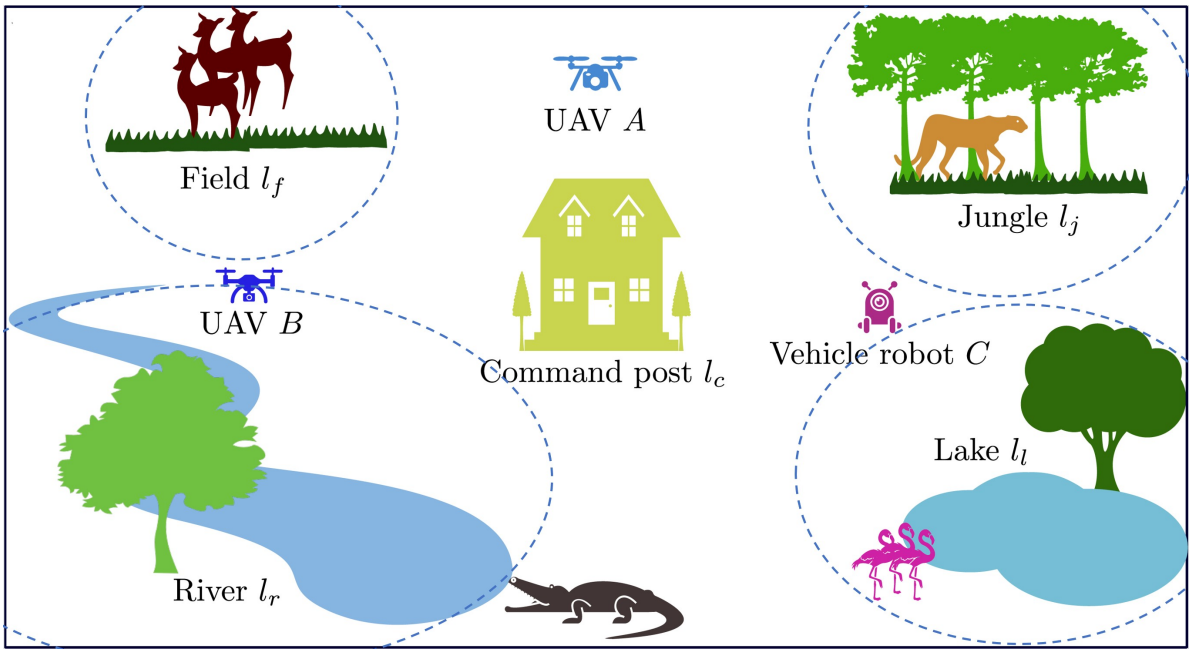
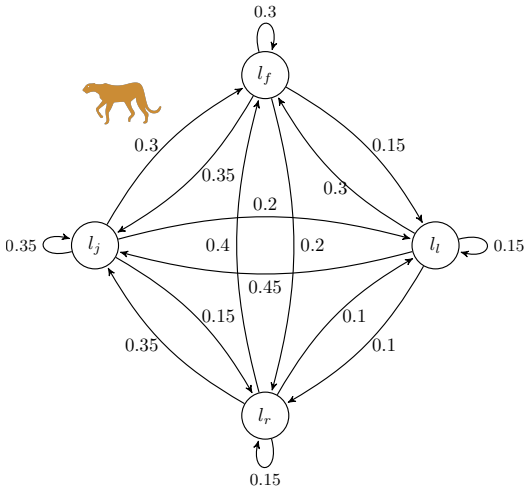


Figure 4.2: A overall view of the wildlife reserve, with 5 main regions, a grass field l_f, a jungle l_j, a command post l_c, a riverside l_r, and a lakeside l_l. Notable fauna includes a cheetah, a crocodile, a herd of gazelle, and a flamboyance of flamingos; the latter two, being gregarious, remain as a group. (Reprinted with permission from [3].)

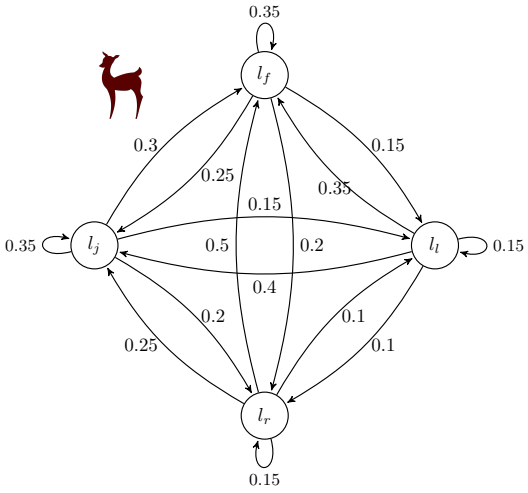
event model for the whole system is obtained via a Cartesian product of these individual models.

The events of interest are gazelle grazing, g_e; cheetah eating gazelle, c_g; crocodile eating gazelle, k_g; flamingos mating, f_m; and crocodile eating flamingo, k_f. Event g_e happens with probability 1 for both cases, whether the gazelle are in the field or the jungle. The probabilities that the cheetah can successfully capture and kill a gazelle in the field and the jungle are, respectively, 0.2 and 0.25. The probabilities that the crocodile can successfully capture and kill a gazelle by the river and by the lake are, respectively, 0.2 and 0.25. The probability that the crocodile can capture and kill a flamingo is 0.15. The probability that a flamingos mating event happens in an hour is 0.4.

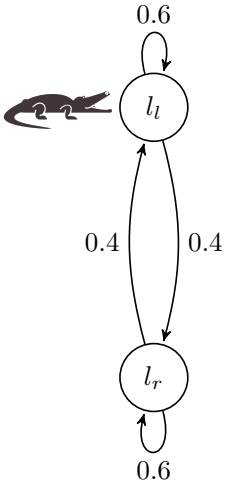
The robots cannot track or follow animal(s) continually, and video clips are only recorded at locations l_f, l_j, l_r, and l_l. At each time step, each robot observes the current state of the event model by receiving a message from the command post, which reports the rough locations of the animals based on GPS trackers connected to tags on the animals. The robot does not know which



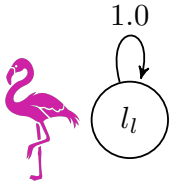
(a) Circadian rhythm for cheetahs.



(b) Circadian rhythm for gazelles.



(c) Circadian rhythm for alligators.



(d) Circadian rhythm for flamingos.

Figure 4.3: Circadian rhythm for the animals in the wildlife park as Markov Chains.

location will be the creatures' destination in the next hour. Also, at every step t, each robot relays to the other robots whether it was successful in recording the event it attempted. Accordingly, all the robots can compute $\mathbf{X}_t$ and $\mathbf{Q}_t$. At each time step, a robot must guess the destination of an animal of interest so that it can go to that destination and prepare to capture a desired event.

Three kinds of videographer robots A, B, and C are available. The robots are camouflaged, so their presence does not change the animals' behavior, i.e., we make a non-causality assumption. Robots of type A are UAVs capable of traveling between any of those locations in a time step. Its actions are unconstrained, so it has a trivial (single state) valid-action automaton. Robots of type B are short-endurance UAVs with a simple constraint on their actions: if at time t, such a robot chooses event k_g to capture at time $t+1$, then it is forbidden from repeating it (regardless of whether it successfully captured k_g at $t+1$ or not). This constraint can be a preference assigned to the robot by the user, perhaps to prevent the robot to stay near the river or the lake for a long time. Robots of type C are ground vehicles. When starting out from the river or lake, these vehicles must take a detour, visiting the command post (for one time step), to be outfitted with equipment required to travel through the jungle or field. Likewise, when in the jungle/field, travel to riverside/lakeside, requires a sojourn at the command post first. Accordingly, if a robot of type C moves to l_c, then in the next time step no event will be captured by that robot.

We set the cost function in a way that the objective in the MRRCM problem becomes minimizing the expected number of hours to capture a desired story. For this purpose, for each robot i and action a, for each vertex v of the valid-action automaton of robot i, we set $c^{(i)}(v, a) = 1$. Because the joint cost of a group of robots is the sum of costs for individual robots, in reporting the results we have divided by the number of robots, each of the expected and the average costs obtained for a joint plan so that these figures represent, respectively, the excepted number of hours and the average number of hours to record a story. For the sequential plan, no division is needed and we use the expected and average costs obtained for the last MDP in the sequential plan directly.

In this case study we are interested in capturing a sequence that is a supersequence for both the sequences $g_e f_m c_g k_f$ and $g_e f_m k_g k_f$, each of which chronicles both a gazelle's life and a flamingo

life. Once a desired sequence was captured, we post-process it to make two videos $g_e f_m c_g k_f$ and $g_e f_m k_g k_f$ from it, each for a TV channel. Note that the language of the specification DFA in this case is infinite. We considered several scenarios in which different numbers of each type of robot are tasked to capture a desired story. For our implementations, we use the value-iteration method to solve the underlying MDPs. For each scenario, we solved the MRRCM problem using the joint approach and the sequential approach with the random and greedy strategies. Also for each scenario, we generated 1000 simulations of executing the event model and for each simulation the robot(s) use the computed policy to capture a story. For each case, we computed the average cost of capturing a desired story over the 1000 simulations. Figure 4.5 shows the expected number of hours and the average number of hours for those simulations. While Figure 4.4 shows the time to compute the policies for the different robot teams. As was expected, a robot of type A (one with the least constraints) outperformed the other two robot types B and C, yielding a smaller expected cost when using a single robot to capture events. For teams composed of two or higher robots, we see that the joint always performs slightly better than the sequential approach. Also, among the sequential approach, we see that the greedy ordering approach outperforms the random ordering. This result is expected as no extra information is being utilized by the sequential solution that the joint approach is unaware of. Also, for each experiment, the expected cost was very close to the average cost for 1000 simulations. We were able to generate a joint plan only for up to three robots; it took approximately 14 hours to generate a joint plan for three robots, while generating a sequential plan for three robots with each of the random and the greedy strategies took approximately 43 minutes and 85 minutes respectively.

4.3.2 Comparing sequential solution to state-of-the-art RL techniques

Next, we provide the comparison of the joint and sequential approach (greedy and random) provided in the chapter with the neural network approximate solution methods, the A2C model [23] and the PPO model [24]. For the comparison we model the scenario mentioned in the previous as an "OpenAI gym"'[88] environment using the team of videographer robots consisting of robot A and robot B. We use the default implementation of the A2C model (learning rate $= 0.0007$,

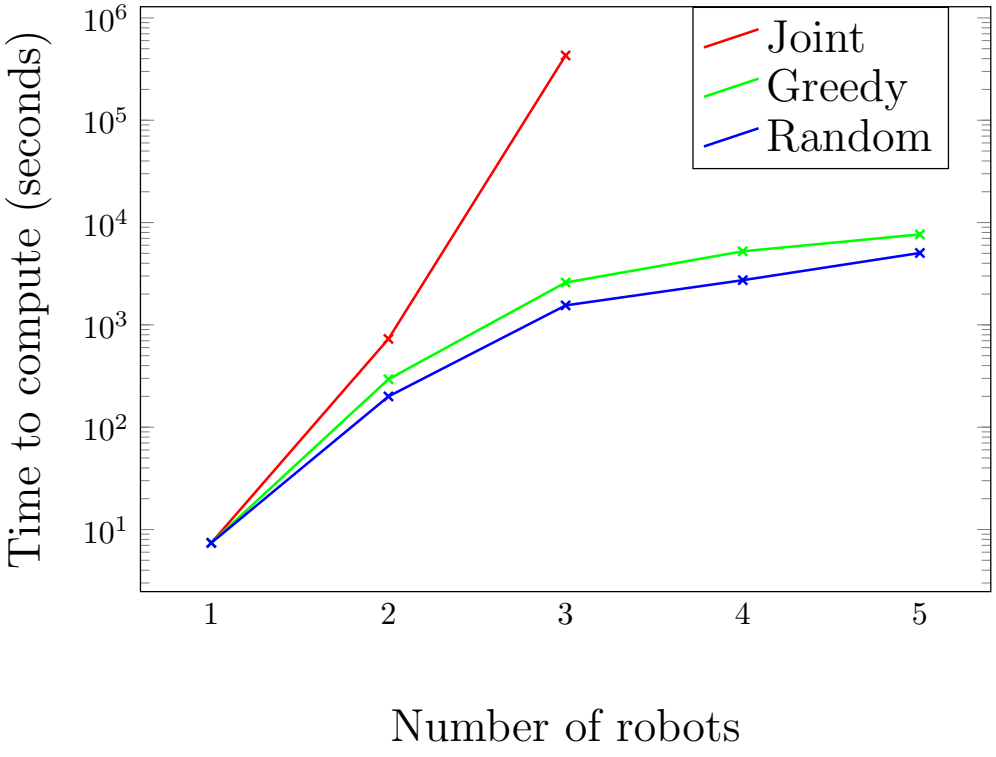


Figure 4.4: Time to compute the policies. (The vertical axis is in the logarithmic scale.) (Reprinted with permission from [3].)

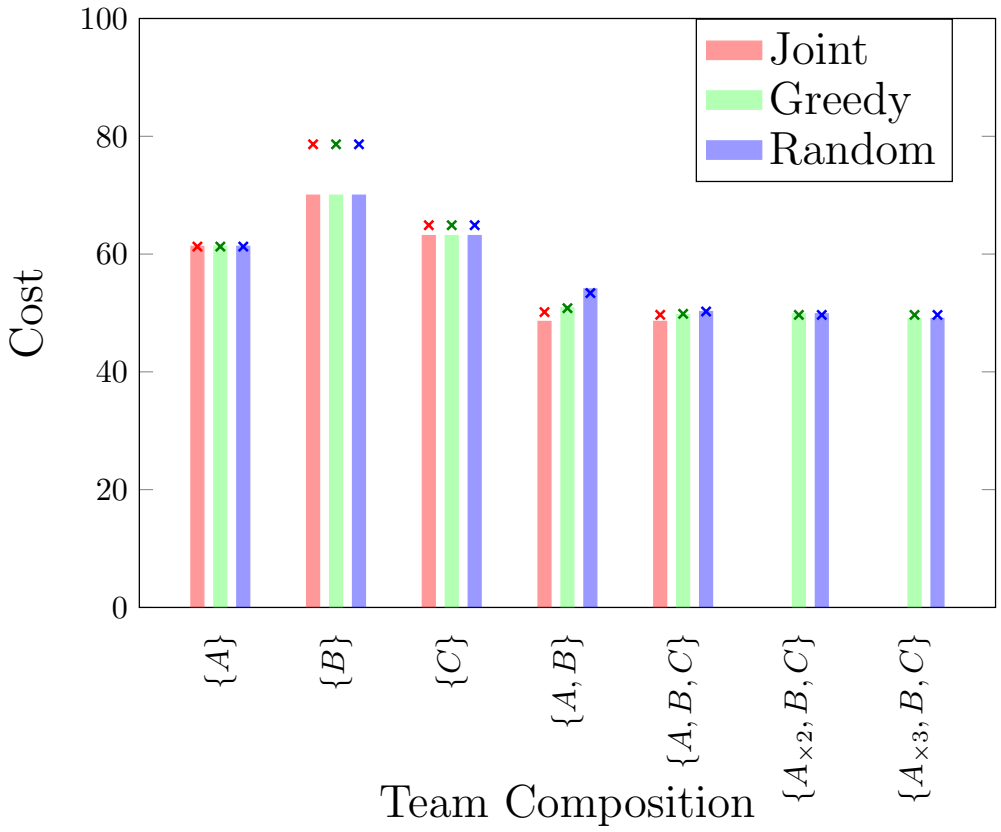


Figure 4.5: Cost to capture the story. The theoretical prediction (expected cost for policy) is shown via $\times$ marks. The average cost for 1000 simulations is shown via bars. (Reprinted with permission from [3].)

discount $= 0.99$) and the PPO model (learning rate $= 0.0003$, discount $= 0.99$) from the "stable-baseline3" [89] package. The action space consisted of an array of two discrete values, where the values corresponds to the event that the robots should attempt to capture, with the first value being correspondent to robot A, and the second being correspondent to robot B. And at each step, the observation received by the agent consisted of the following:

1. current world model state;
2. current story automaton state;
3. current valid-action automaton state for robot A;
4. current valid-action automaton state for robot B;
5. valid actions for robot A;
6. valid actions for robot B;
7. events required for next story update; and
8. events required for next to next story update.

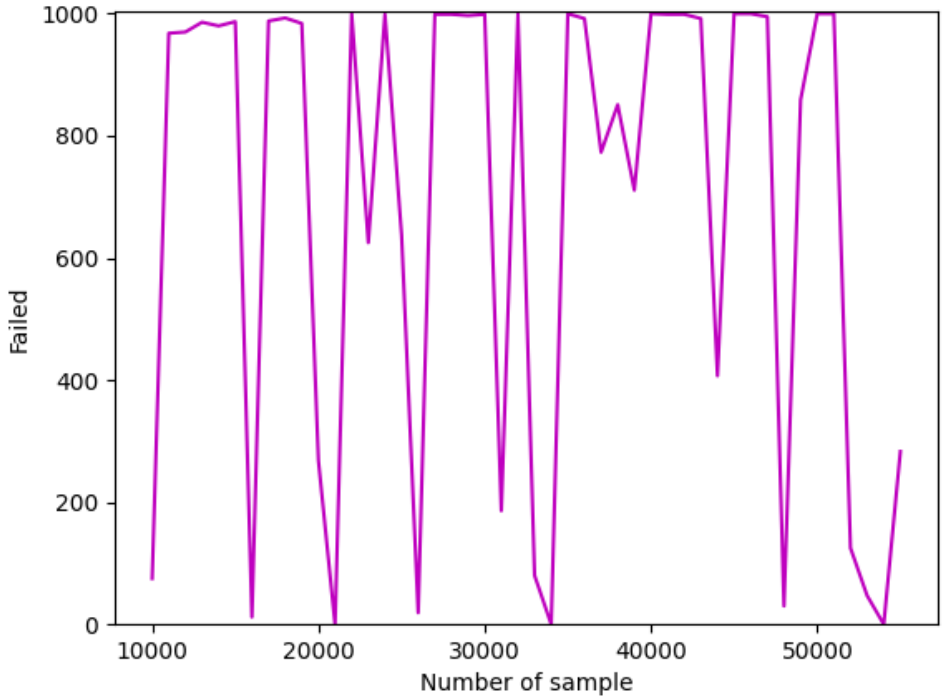


Figure 4.6: Plot showing the number of times the A2C model failed to capture the narrative with respect to the sample size used for training the model.

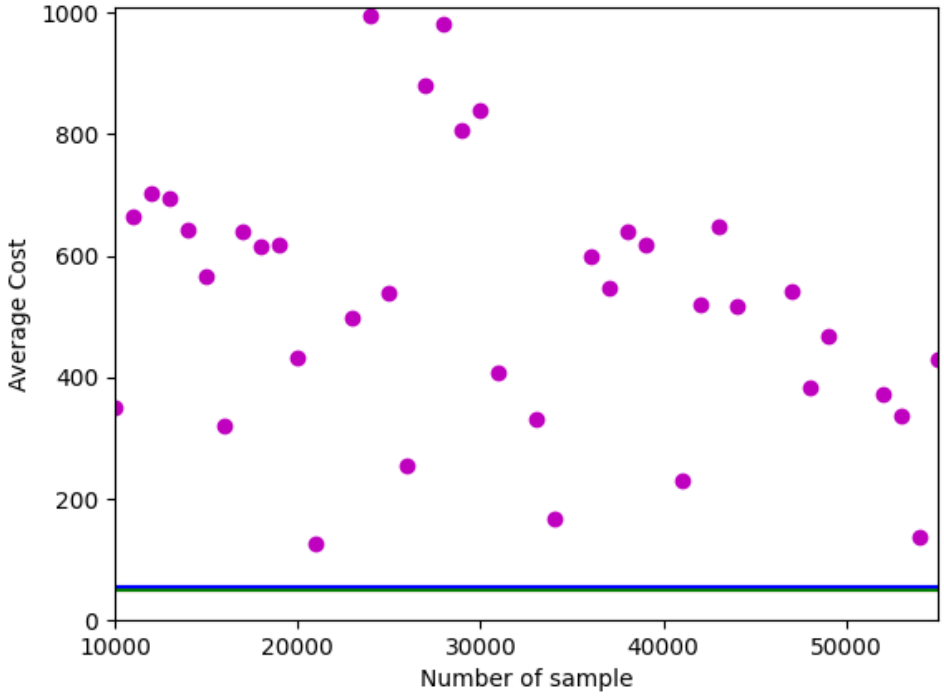


Figure 4.7: Scatter plot (magenta) showing the average cost of the successfully captured narrative according to the policy generated by the A2C model with respect to the sample size used for training the model. Red line denotes the average cost for the joint solution. Blue and green line denotes the average cost for the sequential solution with random and greedy heuristic, respectively.

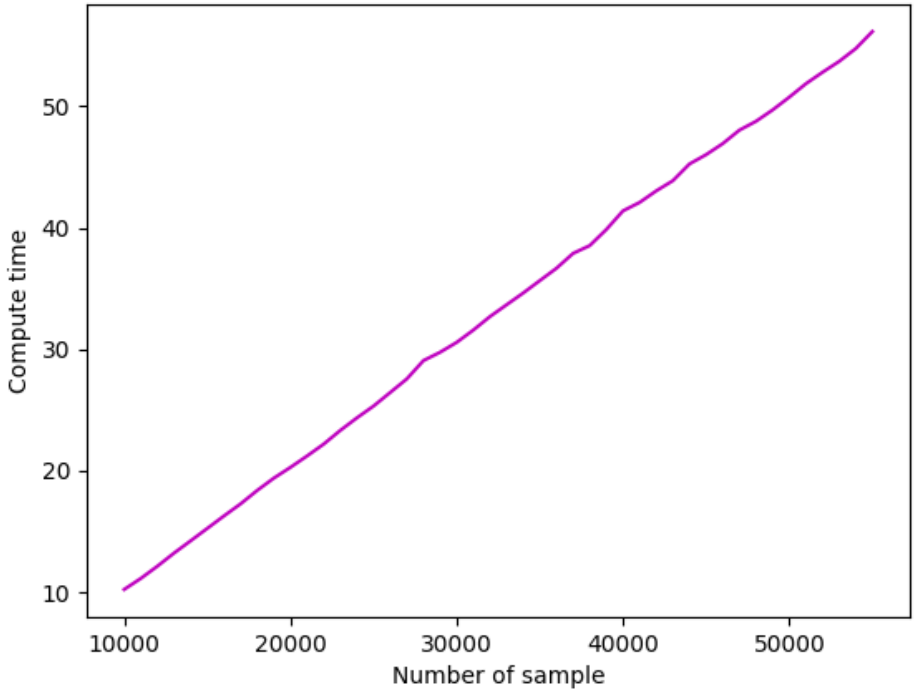


Figure 4.8: Plot showing computation time (seconds) taken by the A2C model to be trained with respect to the sample size used for training the model.

The models were trained using sample size ranging from 10000 to 55000 with 1000 increments between each. Then each trained model was tested by running 1000 simulations. The model is considered to have failed to produce a story if the narrative is not captured in 1000 steps. The A2C model failed to capture the narrative multiple times as shown in Figure 4.6 times with the average cost for the ones successfully captured shown in Figure 4.7. The compute times for the A2C models are shown in Figure 4.8. On the other hand, the PPO model failed to generate any decent policy and failed to capture the narrative for all the simulations. As evident from the results, though the A2C model was trained relatively faster, the policy it produced is subpar to the policy generated by the solution approaches mentioned in this chapter. Similar to the previous chapter, a plausible explanation for the poor performance of the RL solution methods may be due to the non-causality of the problem formulation.

4.4 Stylized video narrative with multiple agents

As mentioned in Chapter 3, merely concatenating a sequence of clips, one for each event, does not produce an aesthetically pleasing video. In this section, we extend the multi-agent structured narrative by introducing style formulation.

4.4.1 Problem definition

We extend the problem in section 4.1 with the style formulation presented in section 3.1.3. Let's assume that Γ be the set of all available style choices. We associate each agent i with a style catalogue $\kappa^{(i)}$ that maps an event to all the available styles that the agent can use to record the event. A single style-gram quantifies the qualitative nature of the styles to keep the overall time-extended style transitions to be consistent.

The agents' choice of the actions, in this case event-style pair for each agent, is governed by a policy π. At any given time the policy is based on the following observable problem parameters: (a) the current state of the event model, (b) states of the story automaton, (c) the valid-action automaton for each agent, and (d) the previous $k-1$ styles used to capture the previous $k-1$ events.

Next, we define the optimization problem as:

Problem: Multi-Robot Styled Video Capture (MRSVC)

Given: An event set $\mathbb{E}$; an event model $\mathcal{M} = (W, \mathbb{E}, T, w_0, g)$; the story automaton $\mathcal{D} = (Q, \mathbb{E}, \delta, q_0, F)$; the number of robots, n; a set of n valid-action automata $\mathcal{D}_V^{(i)} = (Q_V^{(i)}, v_0^{(i)}, \mathcal{A}, \delta_V^{(i)}, J_V^{(i)}), \forall i \in \{1, \cdots n\}$; a set of available styles Γ; a style-graph $\mathcal{G}_\mathcal{S} = (V_\mathcal{S}, \tau_\mathcal{S}, \omega_\mathcal{S})$; and a set of n style catalogues $\kappa^{(i)} : \mathbb{E} \to 2^\Gamma, \forall i \in \{1, \cdots n\}$.

Output: Some prescription for events and styles to capture so the expected accepted sequence efficacy $\mathbf{E}[\nu_{\mathcal{G}_\mathcal{S}}(\xi, \zeta)]$ is maximal, expectation being taken over sequences ξ and ζ

4.4.2 Solution

The direct solution to the problem of choosing which events the agents should capture and how to capture them would form a product automaton from the inputs of the problem. This can be treated as a Markov Decision Problem (MDP) and solved for an optimal policy prescribing the joint action for the agents that specifies for each agent which event it should attempt to capture and in what style it should use to capture it. Unfortunately, though the direct solution would be exact, it is infeasible in practice because both the state and action spaces of the product MDP grow exponentially with the number of agents.

Instead, our algorithm uses the approximate solution similar to the *sequential planning* approach taken in Section 4.2.2 to solve for the events that the agent should attempt to capture. This approach treats the choice of event to capture separately from the choice of style. Once the event policy has been determined, a modified version of the decoupled solution approach taken in Section 3.2.2 is used to generate the style policy. This decoupled solution solves the style policy conditioned on the event policies for each agent.

4.4.3 Robot implementation and field test

We, in collaboration with teams from the University of Houston* and the University of South Carolina†, conducted a field test with two runners running on a race track, each carrying a GPS tracker module in their pockets and two ground robots acting as videographers (see Figure 4.10). Among the two ground robots used, one is faster than the runners, and the other whose top speed is slower than the runners.

Fast ground robot The fast ground robot uses a Conquest XLR 10SC Remote Controlled (RC) car as a base because of its excellent suspension system and speed; the car's speed can reach 6.7 m/s. The robot is equipped with a Raspberry Pi camera and a GoPro camera. Both cameras are attached to the pan unit so they capture the same view with different resolution.

*Dr. Aaron T. Becker, Rhema Ike, and Omar Romero
†Dr. Jason M O'Kane and Dr. Hazhar Rahmani

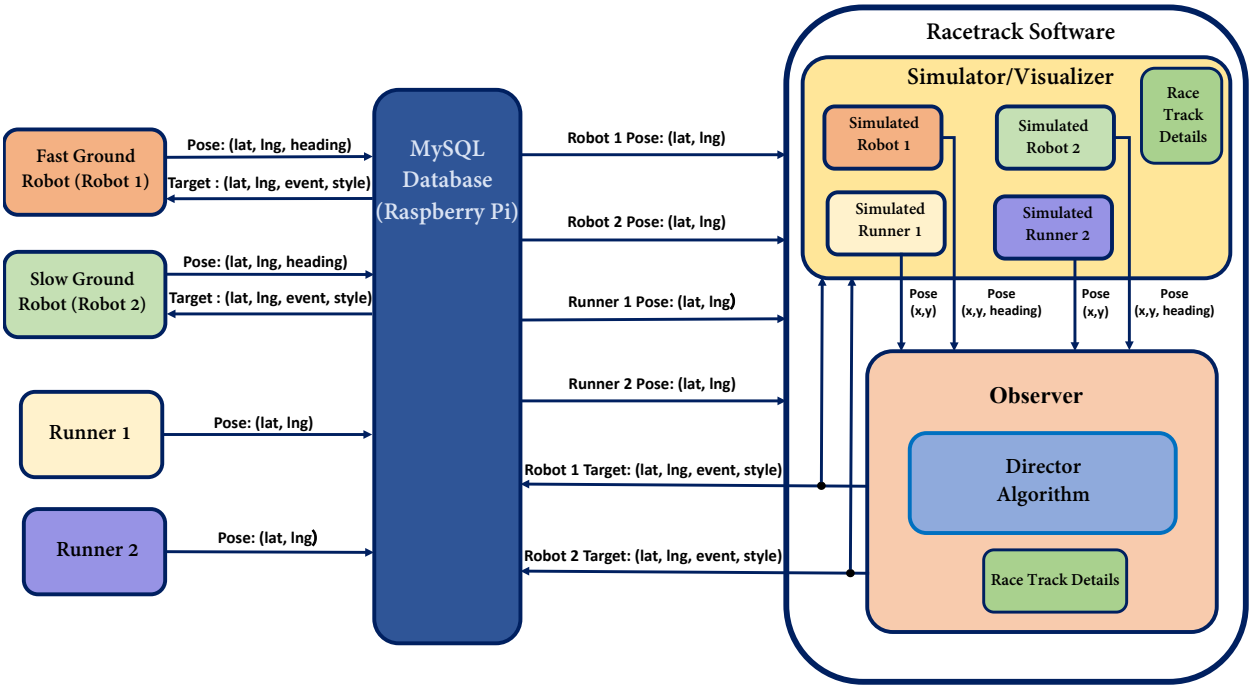


Figure 4.9: System overview, divided into six components. Each component runs on a separate computer, communicating with each other through a local network. We run the SIMULATOR software and the OBSERVER on a laptop as one process with two separate threads. (Figure provided by Dr. Aaron T. Becker, Rhema Ike, and Omar Romero of the University of Houston.)

The pan unit uses a 270° servo to pan. The input from the Raspberry Pi camera is used to track runners, and the GoPro video is used for the final video.

Slow ground robot The slow ground robot uses the Traxxas XL-5 RC car as its base. The maximum speed of this car was restricted to 2 m/s to prevent the car from overtaking the runners. This also restricted the robot's shot types to back shots.

The system developed to capture the compelling story narratives is divided into six components that run on separate computer systems and interface through a local area network using a router. The components are shown schematically in Fig. 4.9. The main components of the system are the hardware robots, the human runners, a local MySQL database running on a Raspberry Pi 3, and Racetrack software. Each runner carries a GPS tracker (Raspberry Pi Zero, mobile hotspot, and GPS module) in their backpack. These upload locations at 1 Hz to a central MySQL database.

Figure 4.10: The two robots used for the field test. (Picture provided by Dr. Aaron T. Becker, Rhema Ike, and Omar Romero of the University of Houston.)

The robot videographers uploads its location at the same rate. The Racetrack software is run on a personal computer. The software comprises of two major components running on separate threads, The SIMULATOR and the OBSERVER. The SIMULATOR encapsulates the simulated runners and robots when running a simulation; however, for the field test, it is used to display the coordinates of the real robots and human runners. The OBSERVER observes the state of the event model by monitoring the locations of the runners and the robots, interacts with a "director" algorithm, and oversees each robot to evaluate if the filming is executed properly. The director algorithm sends styles and events to capture to the database based on if the previous event was captured and the current world state. Each robot queries the database, via the observer, for events to capture and

corresponding styles and sends back a Boolean of the event being successfully captured in the desired style. The observer also calculates the likelihood of an event being captured and reports that back to the director after a scheduled 15-second delay. If robots do not report back their information within the 15-second window or communication is lost, the observer marks the events as not captured.

The story the robots were set to capture was was Runner A and Runner B running, Runner A overtaking Runner B, Runner B overtaking Runner A, and Runner A winning the race. Robot 2 captured Runner A running with a back shot, and Robot 2 captured Runner B running with a side shot. The robots did not capture any overtaking events because they did not occur in the race. Finally, Robot 1 captured Runner A winning the race with a side shot. Not all the predicted events occurred, but the robots were able to film the entire race with a variety of shot types.

4.5 Summary

This chapter formulates the multi-agent extension of the problem studied in Chapter 3. Specifically, the chapter addresses the problem of computing a policy for a team of agents working in coordination to record a sequence of events happening in an uncertain environment at a minimal cost. The first solution provided in this chapter produces a joint MDP, and thus solving for all the agents simultaneously. However, this solution gets quite large in terms of state space and action space. To overcome the computational complexity of solving the joint MDP, the chapter provides an approximate solution by decoupling the solution for each agent via a sequence of smaller MDP, one for each agent, calculated in order either greedily or at random. We provide results showcasing the solution's effectiveness via simulations based on a wildlife scenario and provide a comparison with state-of-the-art neural network techniques, A2C and PPO. Lastly, we provide a multi-robot stylized video narrative implementation showing the real-world feasibility of the solution methods in real-world scenarios.

5. DECOUPLING ENVIRONMENT MODELS THAT CAN BE FACTORED*

In this chapter, we present a planning problem involving an environment with more than one component uninfluenced by the agent's actions, each of which can be formulated via EGMCs. We present it as a multi-commodity logistic planning problem, where the demand for commodities are modeled via EGMCs.

Consider a scenario where an autonomous operations agent oversees the supply chain logistics of a wholesale company. The network consists of storehouses where items could be stored and retail units where the items could be stored or sold. An item can only be sold if there exists a demand for the item at the retail unit and the item is available. The agent's objective is to sell all available items quickly. The chapter formulates the planning problem for the operations agent responsible for routing multiple commodities within a logistic network. We use the generic term 'commodity' to refer to any item, where multiple such items are seen as equivalent to one another in the sense of being exchangeable (e.g., wholesale company selling rice). We study the problem of routing $m \in \mathbb{N}_+$ commodities within a logistic network, with one unit of any commodity being the smallest atomic quantity being considered for storage, transportation, or use within the network. Each of the m commodities be indexed by $\mathbb{E} = \{1, 2, \ldots, m\}$, called the commodity set. The problem assumes an initial quantity of each commodity within the network as given, and the objective of the problem is to devise a plan for the operations agent to route the commodities to the appropriate retail units so that they can be consumed, i.e., driving the quantity of each remaining commodity to zero, in the shortest time.

The planning problem presented here is composed of modular components. The logistic network, being influenced by the agent's decisions, and the consumption behavior (therefore, demand) being unaffected by it. This chapter explores dynamic demands via stateful models, as these can help express some valuable time-extended and structural aspects of the process involved. However,

*Reprinted with permission from "A Causal Decoupling Approach to Efficient Planning for Logistics Problems with Stateful Stochastic Demand" by Diptanil Chaudhuri and Dylan Shell. *2023 Proceedings of IEEE International Conference on Robotics and Automation*. Copyright 2023 by *IEEE*.

the fundamental issue with the statefulness of the demand models is that they increase the size of the planning problem multiplicatively, increasing the computational complexity quickly. Furthermore, logistic problems get still more involved when there are multiple goods. In this capter, we exploit three postulates to subdue the growth in complexity via 'decoupling.' The first involves causality: state transitions within the demand model reflect aspects of the stochastic process which describe uncertainty. Oftentimes, these are driven entirely by external factors with dynamics being uninfluenced or only weakly affected by happenings within the network. Secondly, when there are multiple sites and different influences on demand at these sites, they can be factored by splitting into separate site-specific models. Thirdly, though one may not know future demand, one can usually determine current demand through suitable instrumentation (say, via market analysis). That is, we assume the current state(s) of the demand model(s) can be ascertained. The observation process is, thus, decoupled and distinct from the other aspects involved.

5.1 Problem definition

We build up to the precise problem formalization, first identifying and defining the basic objects involved.

5.1.1 Logistic network

The logistic network is modeled as a graph where the vertices of the graph either represent a storage vertex or retail vertex (viz., locations where the commodities can be sold/consumed); edges of the network represent transportation routes along with their bandwidth. Each vertex of the graph is associated with a utilization model (see Section. 5.1.2), that defines the dynamics of the utilization of commodities within that vertex. Storage vertices are associated with trivial zero utilization model. Hence:

Definition 15. *A* logistic network *is a 4-tuple, $\mathcal{G}_L = (V_L, \tau_L, \omega_S, \omega_U)$, where:*

- $V_L = \{1, 2, \ldots, n\}$ *is the non-empty finite set of vertices;*
- $\tau_L \subseteq V_L \times V_L$ *be the directed edges of the network. We assume the relation to be symmetric, that is, for $v, v' \in V_L$ the edge $\langle v, v' \rangle$ is the same edge as $\langle v, v' \rangle$;*

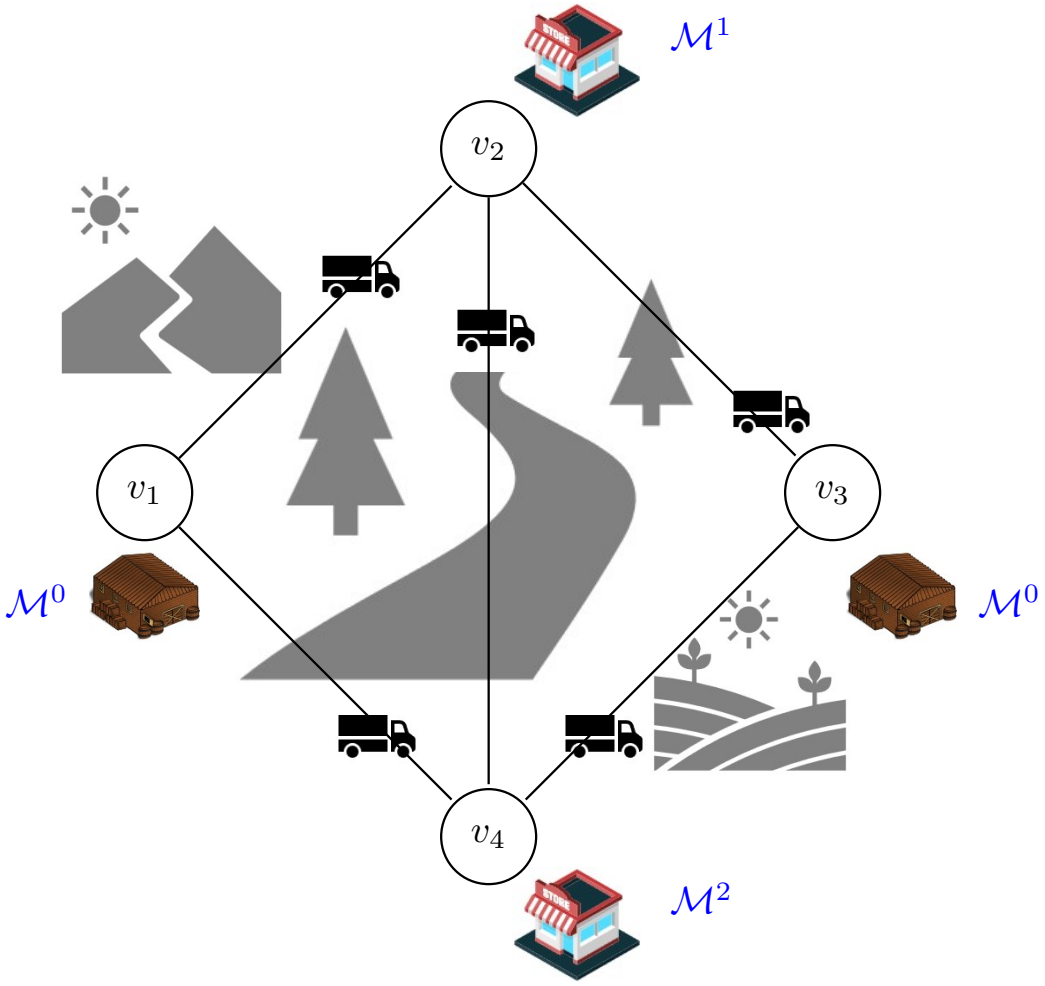


Figure 5.1: Logistic network example consisting of two storehouses and two retail units. The edge bandwidth for all the edges are considered as 1, and the maximum storage capacity for both the storehouses and the retail units are 5. Symbols in blue show the mapping of the vertex to its corresponding demand model (vertex-demand mapping). (Reprinted with permission from [4].)

- $\omega_S : V_L \to \mathbb{N}_+$ *is the vertex storage capacity function, where* $\omega_S(v)$ *is the maximal quantity of all commodities that can be stored at the vertex* v *at any time;*

- $\omega_U : \tau_L \to \mathbb{N}_+$ *is the edge bandwidth function, where* $\omega_U(e)$ *is the maximum quantity of commodities that can be transported via the edge at any time.*

As an example, consider the network for the wholesale example be given by Figure 5.1. The execution of the problem begins with a quantity of each commodity within each vertex, and the objective is to route the commodities until all the commodities are consumed. Therefore, an important variable that provides a snapshot of the network is the quantity of each commodity available in each vertex at any given time. We define $\mathcal{S}$ to be a $n \times m$ storage matrix, where s_{ij} is the quantity of commodity j at the i^{th} vertex. Let the set of possible storage matrices be $\mathbb{S} = \{\mathcal{S} | s_{ij} \geq 0 \text{ and } \sum_{e \in \mathbb{E}} s_{ie} \leq \omega_S(i)\}$, for the logistic network $\mathcal{G}_L$. By $\theta_{\mathcal{S}} \in \mathbb{S}$ denote the zero storage matrix.

5.1.2 The consumer: demand model

The dynamics of the consumption/utilization of commodities within each vertex of the logistic network is modeled in this paper via a stateful, discrete-time stochastic process we call the *demand model*. Formally, a demand model is an EGMC (see Section 3.1.1), $\mathcal{M} = (W, \mathbb{E}, T, w_0, g)$, with $\mathbb{E}$ being the set of all commodities. At any time step, it is possible that there might be a demand for more than a single type of commodity depending upon the realization of the occurrence function $g(\cdot, \cdot)$; however, we assume each commodity to have a demand for 1 unit only. When demand for a unit of some commodity arises within a vertex, if one or more units of commodity are present at that vertex, 1 unit is consumed; otherwise, the opportunity is lost and commodity is not consumed.

Let the set of all demand models be denoted by $\mathbb{M}$. Define the special, trivial demand model where there are never demands for any commodity to be $\mathcal{M}^0 = (W^0, \mathbb{E}^0, T^0, w_0^0, g^0) \in \mathbb{M}$, with (a) $W^0 = \{w_0^0\}$; (b) $T^0(w_0^0, w_0^0) = 1$; and (c) $g^0(w_0^0, e) = 0$. The association between the logistic network's vertices and demand models is via a *vertex-demand mapping* function $\mathcal{F} : V_L \to \mathbb{M}$. The consumer model for storage vertices can be represented by $\mathcal{M}^0$.

5.1.3 Routing policies and problem statement

To define the problem, we first need to identify the problem parameters that are observable to the operational agent. The agent is responsible for routing commodities between the vertices of the network, therefore, we assume that the quantity of each commodity within each vertex of the logistic network to be observable to the agent. For the demand model, the exact demand (realization of the δ function) cannot be observed by the agent; however, the demand model states are observable. This is reasonable when suitable instrumentation (e.g., marketplace analytics, consumer surveys, etc.) is employed. Based on the current observable demand model states, the agent needs to anticipate where commodities might be utilized next and has to take preemptive routing actions for the commodities. Therefore, for a given vertex-demand mapping function $\mathcal{F}$, the states of each demand model corresponding to each vertex of the logistic network is a variable of which the agent must keep track. Let $\mathbb{W}^{\mathcal{F}} = \{\mathbf{w} | \mathbf{w} \in W^1 \times \cdots \times W^n, W^i = \mathcal{F}(i)^{(1)}\}$ be the set of all possible demand state configurations for the function $\mathcal{F}$.

At any time t, the agent's choice is governed by a policy $\pi(\cdot, \cdot)$ based on the states of the demand models corresponding to each vertex of the logistic network ($\mathbf{w}^t \in \mathbb{W}^{\mathcal{F}}$) and the quantity of each commodity available for each vertex ($\mathcal{S}^t \in \mathbb{S}$). The agent's policy governs the quantity of each commodity that is routed through each edge of the network. The action space for the agent is denoted in this paper as $\mathcal{A}$, where each action, $a \in \mathcal{A}$, is a function, $a : \tau_L \times \mathbb{E} \rightarrow \mathbb{Z}$, such that for $\langle v_s, v_d \rangle \in \tau_L$ and $e \in \mathbb{E}$, $a(\langle v_s, v_d \rangle, e) = q$ moves quantity q of commodity e from v_s to v_d if $q \geq 0$; otherwise, if $q < 0$ quantity q of commodity e is moved from v_d to v_s. An action, a, is said to satisfy the edge bandwidth if for all $\langle v_s, v_d \rangle \in \tau_L$, the quantity of all the commodities being routed through this edge is less than or equal to the edge bandwidth function value of that edge, that is, $\sum_{e \in \mathbb{E}} |a(\langle v_s, v_d \rangle, e)| \leq \omega_U(\langle v_s, v_d \rangle)$.

Let the storage matrix at time t be denoted as $\mathcal{S}^t$. At time t, action $a \in \mathcal{A}$ is said to be *valid* for $\mathcal{S}$ if, for all $\langle u, v \rangle \in \tau_L$ and $e \in \mathbb{E}$, we have:

1. the action satisfies the edge bandwidth;

2. the action does not move quantities of commodities more than available from vertex u, that is, $\forall e \in \mathbb{E}$, $s^t_{u,e} - a(\langle u, v\rangle, e) \geq 0$;

3. the action satisfies the vertex storage capacity of the vertex u, that is, $\sum_{e\in\mathbb{E}} s^t_{u,e} - a(\langle u, v\rangle, e) \leq \omega_S(u)$;

4. the action does not move quantities of commodities more than available from vertex v, that is, $\forall e \in \mathbb{E}$, $s^t_{v,e} + a(\langle u, v\rangle, e) \geq 0$;

5. the action satisfies the vertex storage capacity of the vertex v, $\sum_{e\in\mathbb{E}} s^t_{v,e} + a(\langle u, v\rangle, e) \leq \omega_S(v)$.

Let $\hat{\mathbf{1}}_a(\cdot)$ be an indicator function, such that, for $a \in \mathcal{A}$ and $\mathcal{S} \in \mathbb{S}$, $\hat{\mathbf{1}}_a(\mathcal{S}) = 1$ if action $\mathbf{a}$ is valid for $\mathcal{S}$; otherwise, 0.

The goal is to have all the commodities consumed as quickly as possible. We state the problem formally:

Optimization Problem: Logistics with Demand (LWD)

Given: Commodity set $\mathbb{E}$, a logistic network $\mathcal{G}_L = (V_L, \tau_L, \omega_S, \omega_U)$, set of demand models $\mathbb{M}$, a vertex-demand mapping function $\mathcal{F}$, and an initial storage matrix $\mathcal{S}^0 \in \mathbb{S}$.

Output: A policy $\pi^* : \mathbb{W}^{\mathcal{F}} \times \mathbb{S} \to \mathcal{A}$ of valid actions that minimizes the expected time for all commodities to be consumed.

5.2 Solution approaches

To solve the optimization problem, we construct a specific Markov Decision Problem, called the **LWD** MDP.

However, before we formally define the **LWD** MDP, we need to define some preliminary functions. We start by define the *vertex-demand transition* function, $\mathcal{T}^{\mathcal{F}} : \mathbb{W}^{\mathcal{F}} \times \mathbb{W}^{\mathcal{F}} \to [0, 1]$, such that, for $\mathbf{w} = (w^1, \ldots, w^n)$, $\mathbf{y} = (y^1, \ldots, y^n) \in \mathbb{W}^{\mathcal{F}}$, we have $\mathcal{T}^{\mathcal{F}}(\mathbf{w}, \mathbf{y}) = \prod_{v\in V_L} \tau^v_L(w^v, y^v)$, where $\tau^v_L = \mathcal{F}(v)^{(4)}$, specifying the transition probability from one sequence of demand states in the logistic network to another sequence. Next we define the *transport* partial function $\Delta^{\mathcal{F}}_T : \mathbb{S} \times \mathcal{A} \hookrightarrow \mathbb{S}$,

such that, for $\mathcal{S}, \mathcal{S}^- \in \mathbb{S}$ and $a \in \mathcal{A}$, if a is valid: $\Delta_T^{\mathcal{F}}(\mathcal{S}, a) = \mathcal{S}^-$ if $\forall s_{v,e} \in \mathcal{S}$ and $\forall s'_{v,e} \in \mathcal{S}^-$, we have $s'_{v,e} = s_{v,e} - \sum_{v' \in V_L} a(\langle v, v' \rangle, e)$. That is, this function returns the storage matrix that results from the routing actions. Lastly, we define the *consumption* function, $\Delta_C^{\mathcal{F}} : \mathbb{S} \times \mathbb{W}^{\mathcal{F}} \times \mathbb{S} \to [0, 1]$, to be a function of $\mathcal{S}^-, \mathcal{S}^+ \in \mathbb{S}$ and $\mathbf{w} = (w^1, \ldots, w^n) \in \mathbb{W}^{\mathcal{F}}$, such that $\Delta_C^{\mathcal{F}}(\mathcal{S}^-, \mathbf{w}, \mathcal{S}^+) = \prod_{(v,e) \in V_L \times \mathbb{E}} \varphi(s^-_{v,e}, w^v, s^+_{v,e})$ where $\varphi(s^-_{v,e}, w^v, s^+_{v,e}) = g(w^v, e)$ if $s^+_{v,e} - s^-_{v,e} = 1$; $1 - g(w^v, e)$, if $s^+_{v,e} - s^-_{v,e} = 0$; and 0 otherwise. That is, this function determines the probability that some commodities are consumed from one storage matrix and result in the other.

With the requisite functions given, we are now ready:

Definition 16 (**LWD** MDP)**.** *Given an initial storage matrix $\mathcal{S}^0 \in \mathbb{S}$, the logistic network $\mathcal{G}_L = (V_L, \tau_L, \omega_S, \omega_U)$, a vertex-demand mapping function $\mathcal{F}_D$, the set of demand models $\mathbf{M} = \{\mathcal{M} | \mathcal{M} \in \mathbb{M}$ and $\exists v \in V_L, \mathcal{F}_D(v) = \mathcal{M}\}$, we construct the* **LWD** *MDP, $\mathbb{M}_{\mathcal{S}^0, \mathcal{F}_D, \mathcal{G}_L} = (X, x_0, \mathcal{A}, P_L, \mathcal{F}_L, J_L)$, where*

1. *$X \subseteq \mathbb{W}^{\mathcal{F}} \times \mathbb{S}$ is the set of states;*
2. *$x_0 = (\mathbf{w}_0, \mathcal{S}^0) \in X$, such that $\mathbf{w}_0 = (w_0^1, w_0^2, \ldots, w_0^n)$, where $w_0^i \in \mathcal{F}(i)$, is the initial state;*
3. *$\mathcal{A}$ is the action space;*
4. *$P_L : X \times \mathcal{A} \times X \to [0, 1]$ is the transition probability function, such that, for $(\mathbf{w}, \mathcal{S})$, $(\mathbf{w}', \mathcal{S}') \in X$ and $a \in \mathcal{A}$, $P_L((\mathbf{w}, \mathcal{S}), a, (\mathbf{w}', \mathcal{S}')) = \hat{\mathbf{1}}_a(\mathcal{S}) \mathcal{T}^{\mathcal{F}}(\mathbf{w}, \mathbf{w}') \Delta_C^{\mathcal{F}}(\Delta_T^{\mathcal{F}}(\mathcal{S}, a), \mathbf{w}', \mathcal{S}')$;*
5. *$\mathcal{F}_L = \mathbb{W}^{\mathcal{F}} \times \{\theta_{\mathcal{S}}\} \subseteq X$ is the set of goal states;*
6. *$J_L : X \times \mathcal{A} \to \mathbb{R}_{\geq 0}$ is the cost function, so, for $x \in X$ and $a \in \mathcal{A}$, $J_L(x, a) = 1$ if $x \notin \mathcal{F}_L$; otherwise 0.*

An optimal policy for product MDP, $\pi^* : X \to \mathcal{A}$ provides the routing policy. The full product would directly construct **LWD** MDP $\mathbb{M}_{\mathcal{S}^0, \mathcal{F}_D, \mathcal{G}_L}$. Then, the policy can be obtained by using standard solution techniques (e.g. value iteration [5]).

5.2.1 Solution via approximation

As just formulated, the planning problem comprises of individual modular components, consisting of demand models and a logistic network. The number of states in each demand model increases the size of the planning problem multiplicatively via the product in Definition 16. Rather than use the vertex-demand mapping function $\mathcal{F}_D$ directly, as the full solution does, the property of the demand models being uninfluenced by the agent's action provides an opportunity to analyze and simplify the demand models independently. As the demand is not contingent on the actions, the dynamics of the demand do not influence the iterative policy update and thus can be analyzed beforehand. We do this by collapsing the states of the demand model. The analysis leads to constructing a new demand model with fewer states that essentially attempts to emulate the original demand model. Since the LWD MDP is constructed by taking the product of the demand models and the logistic network, the analysis essentially constructs a new MDP with fewer states than the original. An important distinction is the difference between the collapse and the state abstraction techniques used to reduce MDP state space [91]. While the latter attempts to reduce the state of the original MDP by merging states with similar transitions and reward function, the method proposed here creates a new MDP by taking advantage of the non-causality of the demand models to reduce them. Two such approaches follow next.

5.2.1.1 Fundamental matrix analysis

Our second approach to the problem, which we call the *FMA approach*, takes inspiration from [79], and uses matrix analysis on each demand model to collapse all the states into a single state. This collapse of states destroys the temporal structure of the original demand models and reduces the dynamics of the consumer demand for every commodity into a Bernoulli random variable.

To solve the problem using this approach we need to first define two matrices. First, for demand model $\mathcal{M} = (W, \mathbb{E}, T, w_0, g) \in \mathbb{M}$, and commodity $e \in \mathbb{E}$, we solve for the $|W| \times 1$ matrix, $\psi^{(\mathcal{M},e)}$, where $\psi_i^{(\mathcal{M},e)}$, the i^{th} element, is the expected number of steps before demand for commodity e will

occur in the demand model if starting at the i^{th} state of the demand model. We can calculate the matrix $\psi^{(\mathcal{M},e)}$, by following the procedure described next.

Given a commodity $e \in \mathbb{E}$ and a demand model $\mathcal{M} = (W, \mathbb{E}, T, w_0, g)$, we construct a new absorbing Markov chain $(\mathcal{M}', T', w_0)$. To form this, first, we define a new set of states $W' = W \cup \{w^e_{\text{ABS}}\}$. For all $w, w' \in W$, such that, $T(w, w') > 0$ if $g(w', e) > 0$, we add the transitions $T'(w, w') = T(w, w')(1 - g(w', e))$ and $T'(w, w^e_{\text{ABS}}) = T(w, w')g(w', e)$ to the new absorbing Markov chain and if $g(w', e) = 0$, we add the transition $T'(w, w') = T(w, w')$. Performing fundamental matrix analysis [92] on the newly generated absorbing Markov chain yields matrix $\psi^{(\mathcal{M},e)}$.

The second matrix needed for this approach, $\phi^{\mathcal{M}}$, a $|W| \times 1$ matrix, is the stationary distribution. For a demand model $\mathcal{M} = (W, \mathbb{E}, T, w_0, g)$, if the Markov chain (W, w_0, T) is non-absorbing the stationary distribution matrix can be calculated by solving the equation $\phi^{\mathcal{M}} = \phi^{\mathcal{M}} T$. Otherwise, if (W, w_0, T) is absorbing the stationary distribution matrix $\phi^{\mathcal{M}} = [q, q, \ldots, q]^T$, where $q = (|W|)^{-1}$.

The FMA approach replaces the original demand model with one having only a single state. Thus, define new set $\mathbf{M}^1 = \left\{\mathcal{M}^1 = (W^1, \mathbb{E}, w^1_0, T^1, g^1) \in \mathbb{M} \,|\, W^1 = \{w^1_0\} \text{ and } T^1(w^1_0, w^1_0) = 1\right\}$. Notice, $\mathcal{M}^0 \in \mathbf{M}^1$.

By $\mathcal{F}_1 : \mathbb{M} \to \mathbf{M}^1$ denote the *fundamental matrix reduction function*, where, for $\mathcal{M} = (W, \mathbb{E}, T, w_0, g) \in \mathbb{M}$, we have $\mathcal{F}_1(\mathcal{M}) = (W^1, \mathbb{E}, w^1_0, T^1, g^1)$, and for all $e \in \mathbb{E}$, $g^1(w^1_0, e) = \left(\sum_{w \in W} \phi^{\mathcal{M}}_w \psi^{\mathcal{M},e}_w\right)^{-1}$ Note, $\mathcal{F}_1(\mathcal{M}^0) = \mathcal{M}^0$.

The solution to **LWD** using this approach can be generated by constructing the **LWD** MDP $\mathbb{M}_{\mathcal{S}^0, \mathcal{F}_{1 \circ D}, \mathcal{G}_L}$, where for $v \in V_L$, $\mathcal{F}_{1 \circ D} = \mathcal{F}_1(\mathcal{F}_D(v))$. And then solving that reduced MDP using some standard technique.

5.2.1.2 Model reduction via collapsing state pairs

The two approaches just seen can be considered as two extremes: the first without any reductions, while the second reducing the whole demand model to a single state. In this section, we will devise a reduction function that gives a spectrum of approximations in-between, as it can be applied to the original demand models iteratively to reduce the number of states one at a time.

For a given demand model $\mathcal{M} = (W, \mathbb{E}, T, w_0, g) \in \mathbb{M}$, an intuitive approach to reduce the number of states is to merge the two states within W that are most similar to each other. Each state w of the demand model is associated with two distributions:

- distribution over the states of the demand model, given by the transition function $T(w, \cdot)$, and
- joint probability distribution over every commodity derived from the occurrence function $g(w, \cdot)$.

Therefore to quantify the similarity between two states, we would need to quantify the similarity between their distributions for both (a) and (b). We introduce a modified formulation of the Hellinger distance [93], with a parameter α, to quantify the similarity between two states of the same demand model. The parameter α acts as weights for the Hellinger distance of the two distributions, assigning preference of one over the other. For any two states $w, w' \in W$, the modified Hellinger distance, $\mathbf{H}_\alpha(w, w') = \alpha \mathbf{H}_T(w, w') + (1 - \alpha)\mathbf{H}_g(w, w')$, where $\mathbf{H}_T(w, w')$ is the Hellinger distance between the two distributions $T(w, \cdot)$ and $T(w', \cdot)$, and $\mathbf{H}_g(w, w')$ is the Hellinger distance between the two joint distribution derived from $g(w, \cdot)$ and $g(w', \cdot)$. Since the Hellinger distance is symmetric, we have, $\mathbf{H}_\alpha(w, w') = \mathbf{H}_\alpha(w', w)$. The two most similar states in terms of Hellinger distance are given as $W_\alpha^{\mathcal{M}} = \arg\min_{w,w' \in W} \mathbf{H}_\alpha(w, w')$.

Using this, we can give the *Hellinger reduction* function $\mathcal{F}_H : \mathbb{M} \times [0, 1] \to \mathbb{M}$, so that, for $\mathcal{M} = (W, \mathbb{E}, T, w_0, g) \in \mathbb{M}$ and $\alpha \in [0, 1]$, $\mathcal{F}_H(\mathcal{M}, \alpha) = \mathcal{M}$, if $\mathcal{M} \in \mathbf{M}^1$, otherwise, $\mathcal{F}_H(\mathcal{M}, \alpha) = \mathcal{M}' = (W', \mathbb{E}, w_0', T', g')$, where:

1. $W' = \{w_{\text{new}}\} \cup W \setminus W_\alpha^{\mathcal{M}}$ is the non-empty finite state space;
2. $\mathbb{E}$ is the set of all commodities;
3. $w_0' \in W'$ is the initial state, such that, if $w_0 \in W_\alpha^{\mathcal{M}}$, $w_0' = w_{\text{new}}$, otherwise take $w_0' = w_0$;
4. $T' : W' \times W' \to [0, 1]$ is the transition probability function, such that for $w, w', w_{\text{new}} \in W'$, and $\{w_a, w_b\} \in W_\alpha^{\mathcal{M}}$,

- if $w' \neq w \neq w_{\text{new}}$, $T'(w, w') = \eta_w T(w, w')$,
- if $w = w_{\text{new}}$ and $w' \neq w_{\text{new}}$, $T'(w, w') = \eta_w \left(T(w_a, w') + T(w_b, w')\right)$,
- if $w \neq w_{\text{new}}$ and $w' = w_{\text{new}}$, $T'(w, w') = \eta_w \left(T(w, w_a) + T(w, w_b)\right)$, and
- if $w = w' = w_{\text{new}}$, $T'(w, w') = \eta_w \left(T(w_a, w_a) + T(w_a, w_b) + T(w_b, w_a) + T(w_b, w_b)\right)$,

where η_w is a normalizing factor such that $\sum_{w' \in W'} T'(w, w') = 1$;

5. $g' : W' \times \mathbb{E} \to [0, 1]$ is a occurrence function, such that, for $w \in W'$ $g'(w, e) = g(w, e)$ if $w \neq w_{\text{new}}$; otherwise $g'(w, e) = \frac{1}{\phi^{\mathcal{M}}_{w_a} + \phi^{\mathcal{M}}_{w_b}} \left(\phi^{\mathcal{M}}_{w_a} g(w_a, e) + \phi^{\mathcal{M}}_{w_b} g(w_b, e)\right)$, where $\{w_a, w_b\} \in W^{\mathcal{M}}_{\alpha}$ and $\phi^{\mathcal{M}}$ is the $|W| \times 1$ stationary distribution matrix.

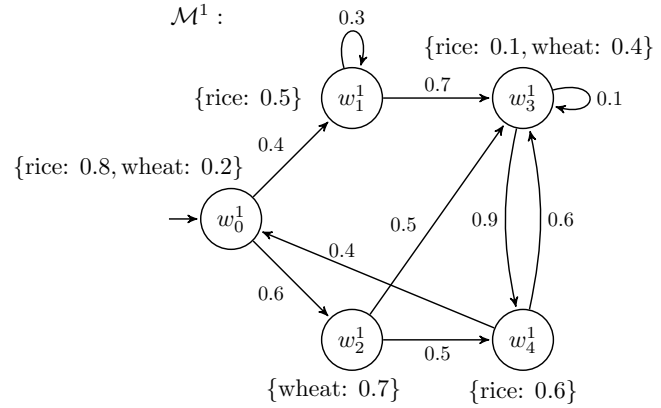
For $\rho = (\rho_1, \ldots, \rho_n) \in \mathbb{N}^n_+$, and $v \in V_L$, let us write $\mathcal{F}_{\rho \circ D} = \mathcal{F}^{\rho_v}_H(\mathcal{F}_D(v))$, where $\mathcal{F}^{\rho_v}_H(\cdot)$ signifies application of the Hellinger distance function ρ_v times iteratively.

Then solution to **LWD**, via this approach, is obtained by reducing the original demand model by ρ, and constructing the **LWD** MDP $\mathbb{M}_{\mathcal{S}^0, \mathcal{F}_{\rho \circ D}, \mathcal{G}_L}$. This MDP is then solved.

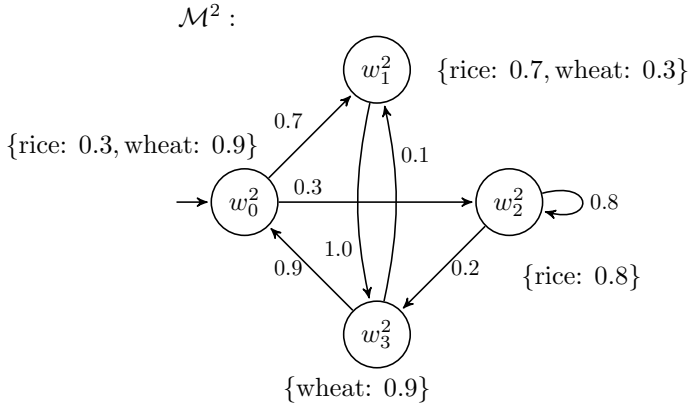
One thing to note here is that these are a few of the many approaches that can be used to pre-analyze the demand model in order to generate efficient approximate solutions. Other approaches could consider different measures for the similarity between the distributions, for example Bhattacharyya distance. These measures of similarity are myopic and does not consider the influence of the merging on the **LWD** MDP, other heuristic based approaches may provide better efficiency and traceability to the approximate solutions.

5.2.2 Observability

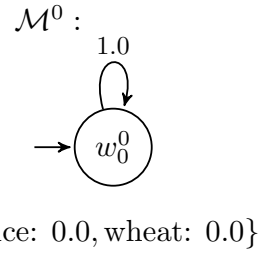
In this section, we elaborate on the different problem parameters that are observable to the agent. For all the solution approaches, the logistic network parameters (edge bandwidth and maximum storage capacity) are observable to the agent along with the number of each commodity in each node. The difference in observability for the different approaches comes from the collapsing of the demand model states. While the full product solution observes the current state of the



(a) Demand model $\mathcal{M}^1$.



(b) Demand model $\mathcal{M}^2$.



(c) Demand model $\mathcal{M}^0$.

Figure 5.2: Demand models associated with the logistic network presented in Figure 5.1. (Reprinted with permission from [4].)

demand model for each node, the approximate solutions observe the equivalent state within the reduced demand model, emulating the original.

5.3 Results

Here, we present simulation results using a Python implementation of the algorithms, executed on a Windows 11 computer with a 2.90GHz CPU.

5.3.1 Traditional solution vs approximate solution

We turn now to examine two example scenarios.

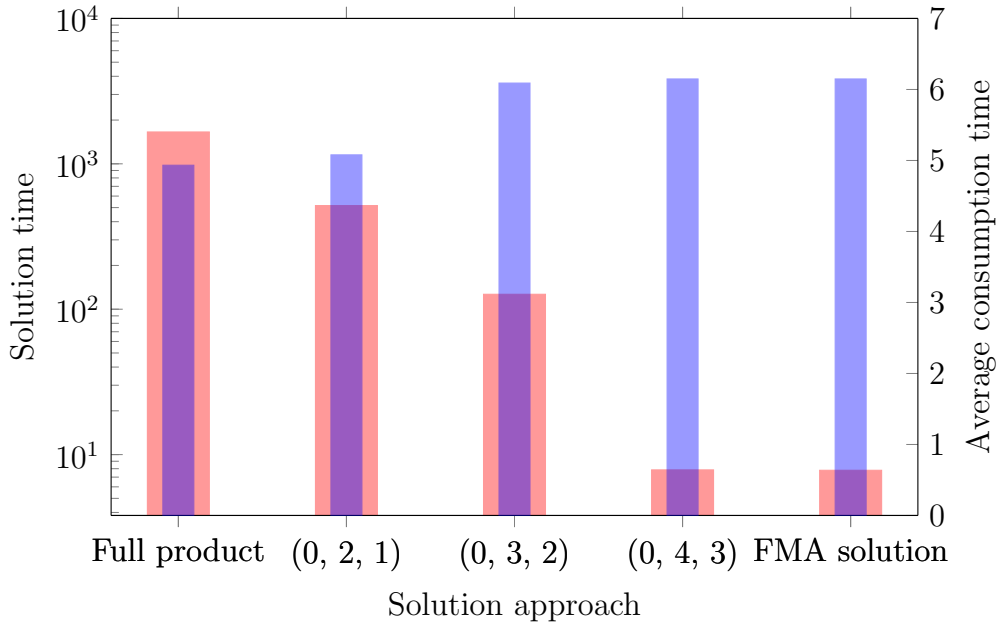


Figure 5.3: Bar plot showing the solution time (red) on the left axis (logarithmic scale) and the average time for the commodities to be consumed (blue) on the right for the different solution approaches corresponding to the logistic network shown in Figure 5.1 and demand models in Figure 5.2. (Reprinted with permission from [4].)

5.3.1.1 Routing grain: rice and wheat

For the first scenario, we revisit the wholesale company example with two commodities: rice, and wheat. The logistic network for this problem is shown in Figure 5.1 and their associated demand models are shown in Figure 5.2.

We solve the problem using approaches presented, each generating its own policy. First, we generate the policy by solving for the MDP considering the full demand models without any reduction. The second, third, and fourth solutions generate the MDP by first using the Hellinger distance reduction technique to collapse the number of states in the given demand models with $\rho = (0, 2, 1), (0, 3, 2), (0, 4, 3)$. Notice $\rho = (0, 4, 3)$ converts the demand model into a demand model with only a single state. Last, we use FMA to reduce the given demand models into a single state approximation, then used to generate the approximate policy. To keep the problem small enough for the full solution we study the problem with initial storage of 2 units of each commodity in the storage vertex v_1.

To verify the correctness of the different solution approaches, we simulated the execution of the policies 1000 times. The results are shown in Figure 5.3. The graph presents the data in a way that allows comparison of performance of the solution method on two axes. The first is the time that it takes to generate a policy (including the time taken for reduction and constructing the **LWD** MDP), shown with respect to the left-hand (log-scale) axis. The second is the quality of the resulting policy, which is measured as the average time taken by a policy to go from initial storage to $\theta_{\mathcal{S}}$ (zero storage) in the 1000 simulations, shown with respect to the right-hand axis.

As expected, the full solution (without any reduction applied to the demand models) provides the best policy selling all the commodities in less time, on average, than the others. However, it takes the longest to provide this solution. As is evident from Figure 5.3, as more reductions are applied, the time to generate the solutions decreases, while time to sell increases. These approach the demand model of a single state with both FMA and recursive Hellinger distance finally collapsing. Both 1-state models take significantly less time to generate the solution compared to the other solutions.

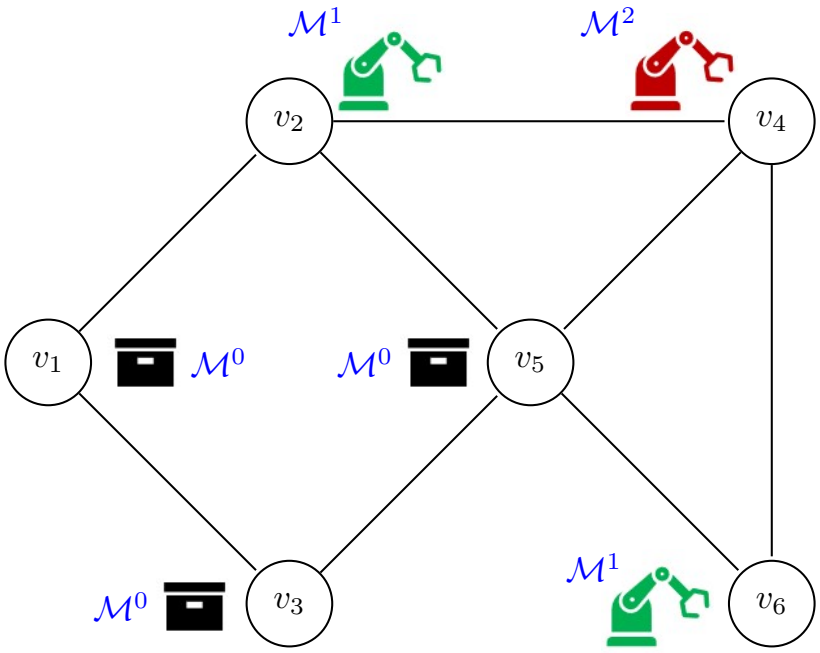
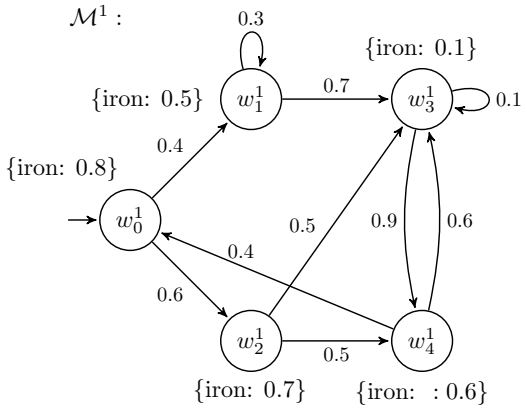


Figure 5.4: Logistic network within a factory floor consisting of three storage vertices, two nail manufacturing machines (green), and one screw manufacturing machine (red). Symbols in blue show the mapping of the vertex to its corresponding demand model (vertex-demand mapping). (Reprinted with permission from [4].)

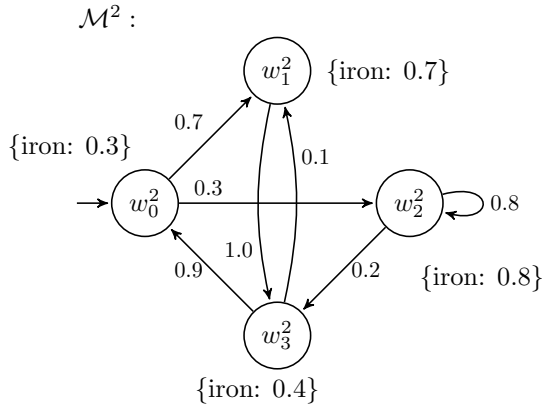
5.3.1.2 Lean manufacturing

Next, we consider a scenario where an agent in a lean manufacturing factory floor producing nails and screws must transport the raw materials (iron bars) to the different machines on the floor (Figure 5.4). Assume that the demand for nails and screws is directly reflected in demand for iron bars for each machine (Figure 5.5). Thus, when there arises a demand for nails, one of the nail manufacturing machines produces a demand for iron bars, which are used to manufacture the nails. The agent's objective is to route the iron bars among the storage areas and the machines effectively, so as to reduce the overall time needed for some initial quantity of iron bars to be used completely. For this example, assume that the demand for iron bars arising from a machine lasts for one time step, i.e., if there are no available iron bars at that point of time, the machine stops (does not consume the iron bars).

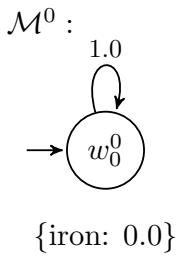
To verify the correctness of the solutions, we take an approach similar to the other example by



(a) Demand model $\mathcal{M}^1$.



(b) Demand model $\mathcal{M}^2$.



(c) Demand model $\mathcal{M}^0$.

Figure 5.5: Demand models associated with the logistic network presented in Figure 5.4. (Reprinted with permission from [4].)

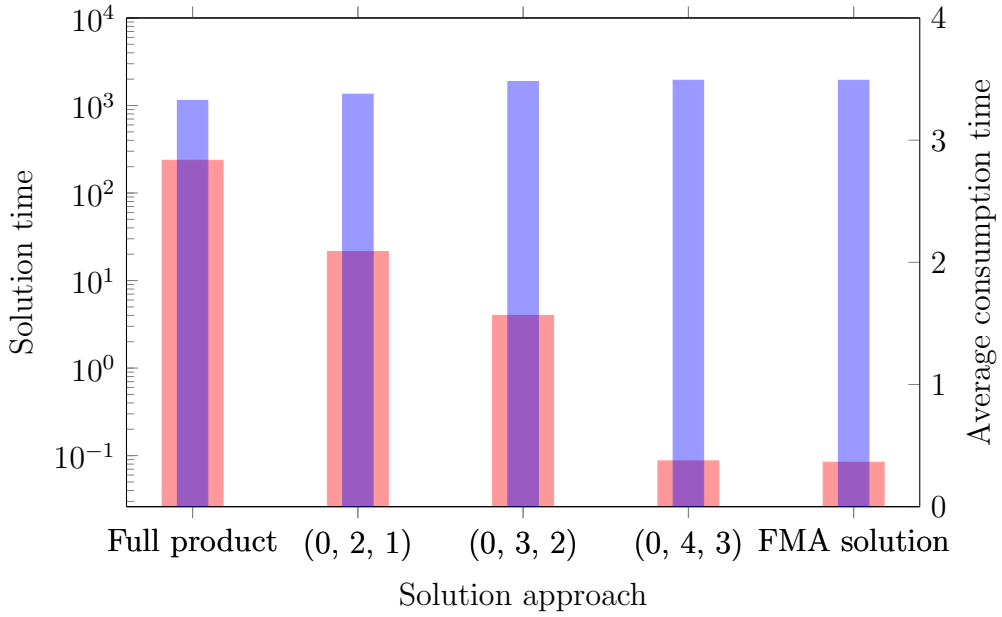


Figure 5.6: Bar plot showing the solution time (red) on the left axis (logarithmic scale) and the average time for the commodities to be consumed (blue) on the right for the different solution approaches corresponding to the logistic network shown in Figure 5.4 and demand models in Figure 5.5. (Reprinted with permission from [4].)

executing the policies a 1000 times. The results are shown in Figure 5.6 (Bottom).

Similar to the other case study, the full solution provides the best solution but takes the longest time to generate the solution, while the fully collapsed (single state) demand models provide significantly faster solutions with slight decrease in quality. All the other reduction solutions lie between the full and 1-state solutions, with the solution quality and time decreasing as more reductions are applied.

5.3.2 Comparing approximate solution to state-of-the-art RL techniques

As with the previous two chapters, here we provide a comparison of the solution approaches mentioned in the chapter with state-of-the-art neural network approximate solution methods, the A2C model [23] and the PPO model [24]. The comparison was made by first modeling the the case study in Section 5.3.1.1, as an "OpenAI gym" [88] environment and using the default implementations of the A2C model (learning rate $= 0.0007$, discount $= 0.99$) and the PPO model (learning

rate $= 0.0003$, discount $= 0.99$) from the "stable-baseline3" [89] package. The action choice for the agent for each step consisted of the number of rice and wheat to be transported over the edges of the network. And at each step, the observation received by the agent consisted of the following:

1. current demand model states;
2. amount of rice stored at each vertex;
3. amount of wheat stored at each vertex;
4. the maximum storage capacity of each vertex; and
5. the edge bandwidth for all the edges.

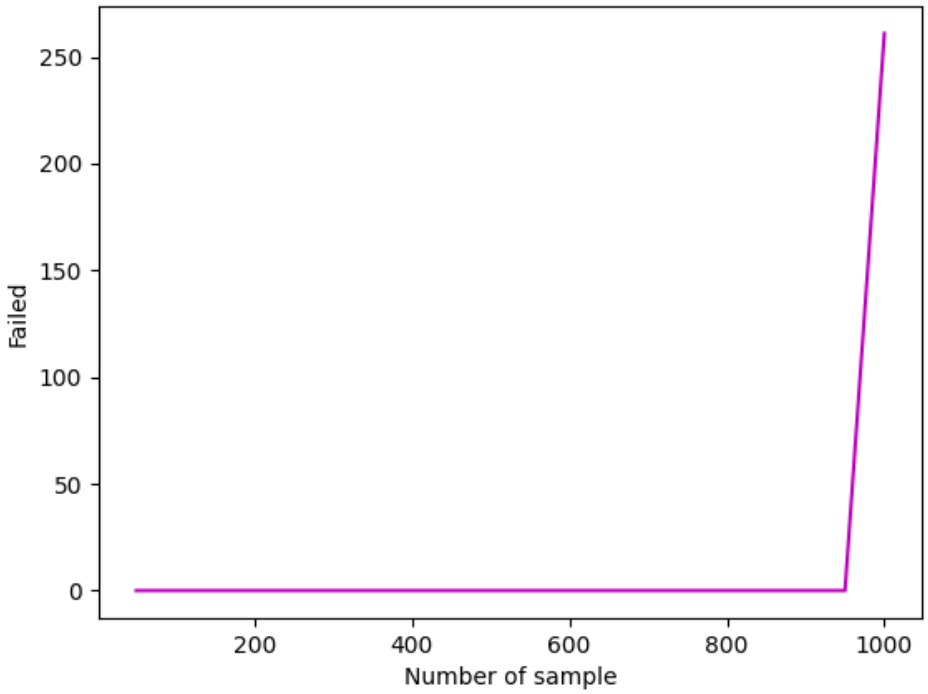


Figure 5.7: Plot showing the number of times the A2C model failed to have all the commodities consumed with respect to the sample size used for training the model.

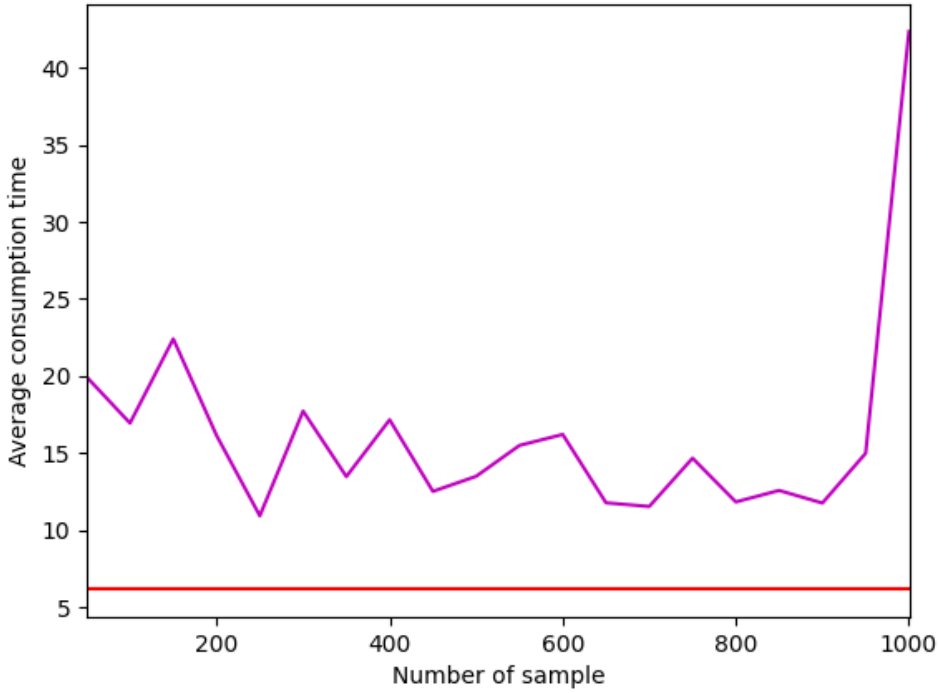


Figure 5.8: Plot showing the average time for all the commodities to be consumed according to the policy generated by the A2C model with respect to the sample size used for training the model. The red line denotes the average consumption time for the FMA solution (highest consumption time among the proposed solution approaches).

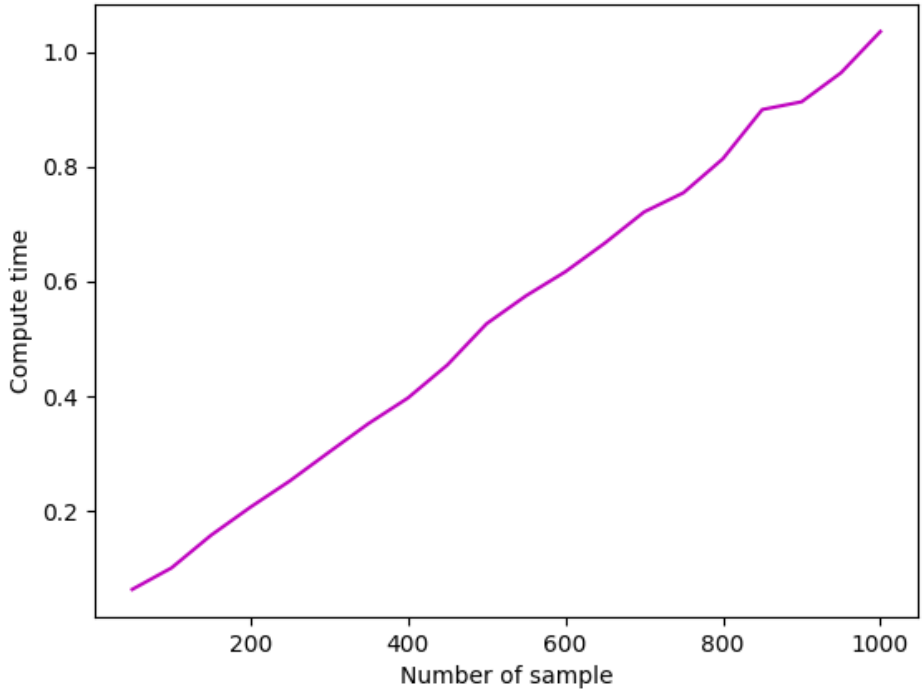


Figure 5.9: Plot showing computation time (seconds) taken by the A2C model to be trained with respect to the sample size used for training the model.

The models were trained using sample size ranging from 50 to 1000 with 50 increments between each. Then each trained model was tested by running 1000 simulations. The model is considered to have failed if all the commodities are not consumed in 100 steps. For each simulation, the A2C model failed to have all the commodities consumed multiple times (shown in Figure 5.7). For each sample size the average number of time steps needed for all the commodities to be consumed (only for ones whose commodities were consumed in 100 steps) is shown in Figure 5.8. The compute times are shown in Figure 5.9. The PPO model succeeded in having the commodities consumed in every simulation. The average time taken for the commodities to be consumed, and the compute time are shown in Figure 5.10, and Figure 5.11 respectively. Though both the models took less time to compute the policy than the solution approaches presented here (except for the ones where the demand model was reduced to a single state), the quality of the policy generated by

them was inferior to the policies generated by our solution approach. A plausible explanation for the poor performance of the RL solution methods may be due to the non-causality of the problem formulation.

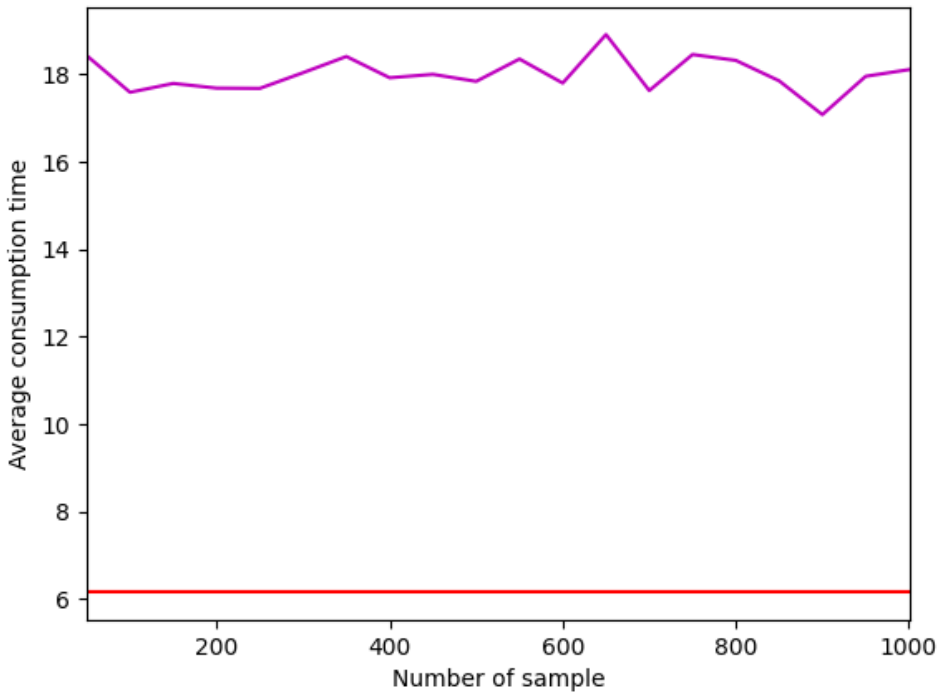


Figure 5.10: Plot showing the average time for all the commodities to be consumed according to the policy generated by the PPO model with respect to the sample size used for training the model. The red line denotes the average consumption time for the FMA solution (the highest consumption time among the proposed solution approaches).

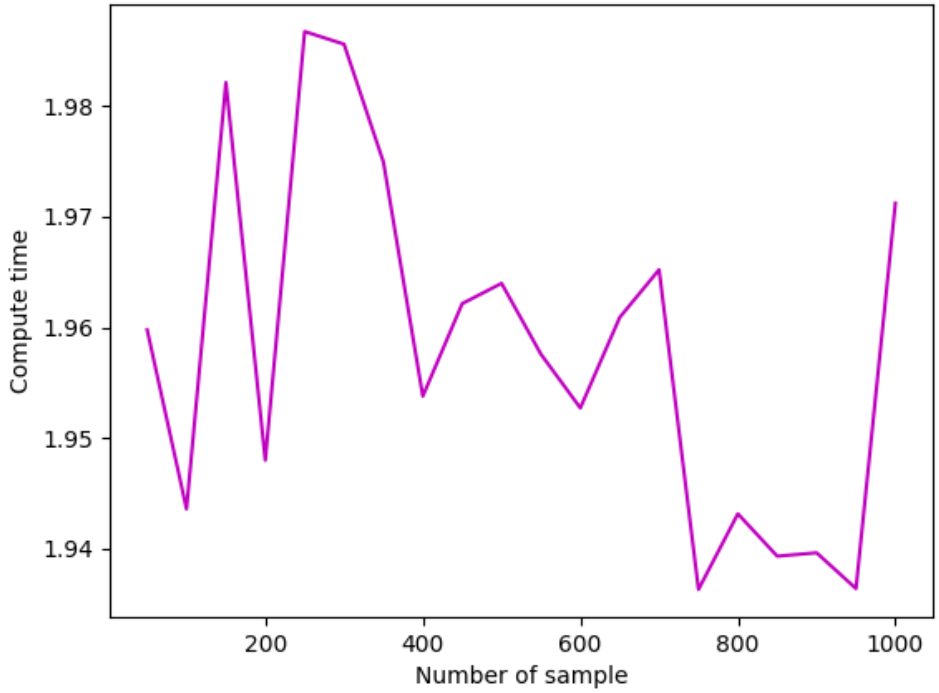


Figure 5.11: Plot showing computation time (seconds) taken by the PPO model to be trained with respect to the sample size used for training the model.

5.4 Limitations

Previous sections formulated the problem, gave methods for compression of demand models, and examined simulations showing how such compression improves solution time significantly, with minor decreases in solution quality. Now, we give a constructed example where the reduction based on Hellinger distance is detrimental: giving a poor policy and requiring a longer time to solve.

Consider Figure 5.12 and Figure 5.13. The problem consists of two demand models that are deterministic regarding their state transitions. The problem is solved considering six different approximations, which are

1. the full product MDP;

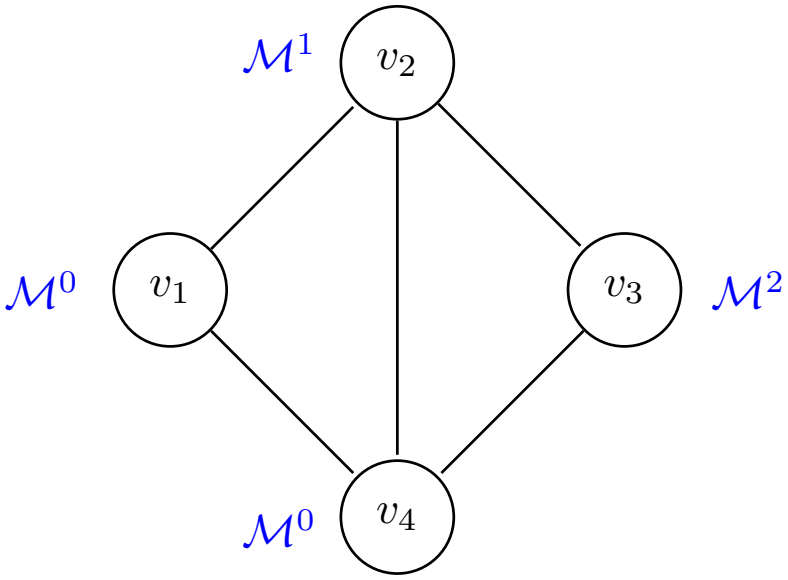


Figure 5.12: Logistic network consisting of two storage vertices, and two two consumer vertices. Symbols in blue show the mapping of the vertex to its corresponding demand model (vertex-demand mapping). (Reprinted with permission from [4].)

2. a reduced MDP solution by reducing both the demand models into 7 states each via Hellinger distance approach ($\rho = (0, 2, 2)$);

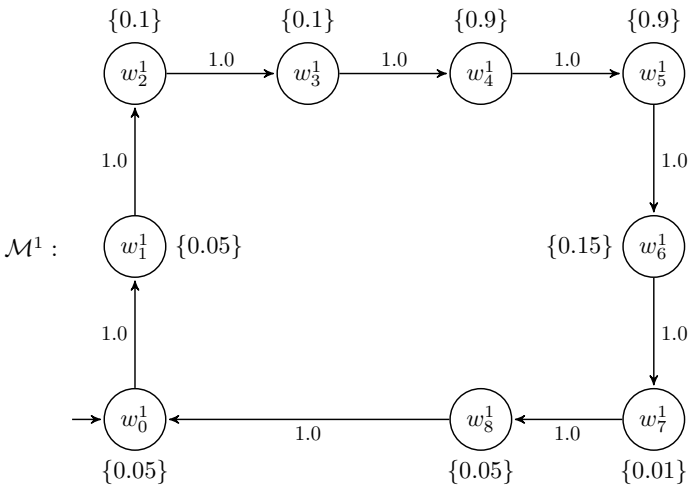
3. a reduced MDP solution with $\rho = (0, 7, 7)$ (reduced to 2 states);

4. a reduced MDP solution with $\rho = (0, 8, 8)$ (reduced to 1 state);

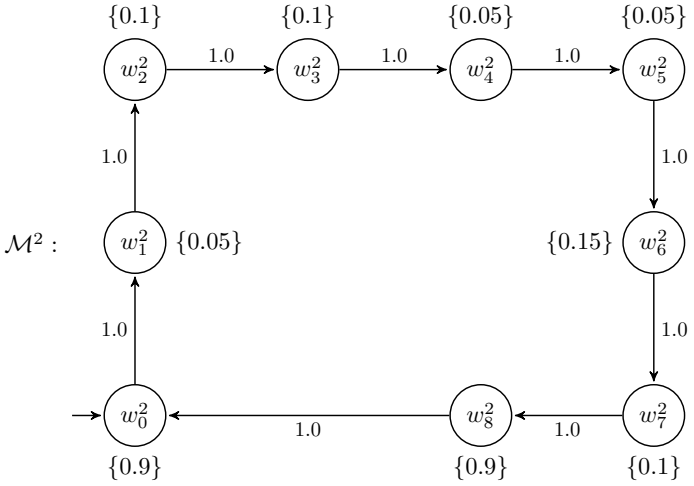
5. the FMA approach; and

6. a reduced solution MDP produced by reducing the demand models into 2 states each manually.

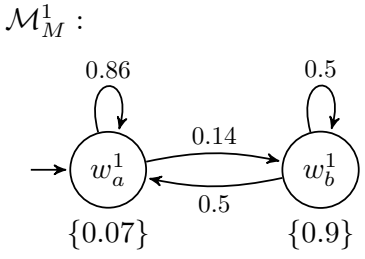
Quantitative results appear in the plot in Figure 5.14 .We can see from the results that even though the quality of the solution is similar for both approaches (1) and (2), the reduction solution here takes longer to generate the solution than the full-product MDP solution approach. The solution quality for the manually reduced solution approach (6) is better than the 2-state reduction via Hellinger distance (3). Not only that, the solution's quality is better compared to (4) or (5). The



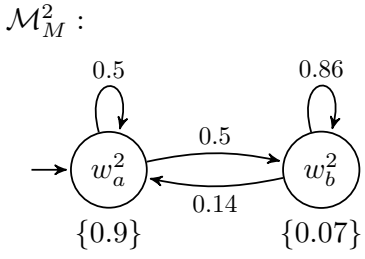
(a) Demand model $\mathcal{M}^1$.



(b) Demand model $\mathcal{M}^2$.



(c) Manual reduction of the demand model $\mathcal{M}^1$.



(d) Manual reduction of the demand model $\mathcal{M}^2$.

Figure 5.13: Demand models and their manual reductions associated with the logistic network presented in Figure 5.12. (Reprinted with permission from [4].)

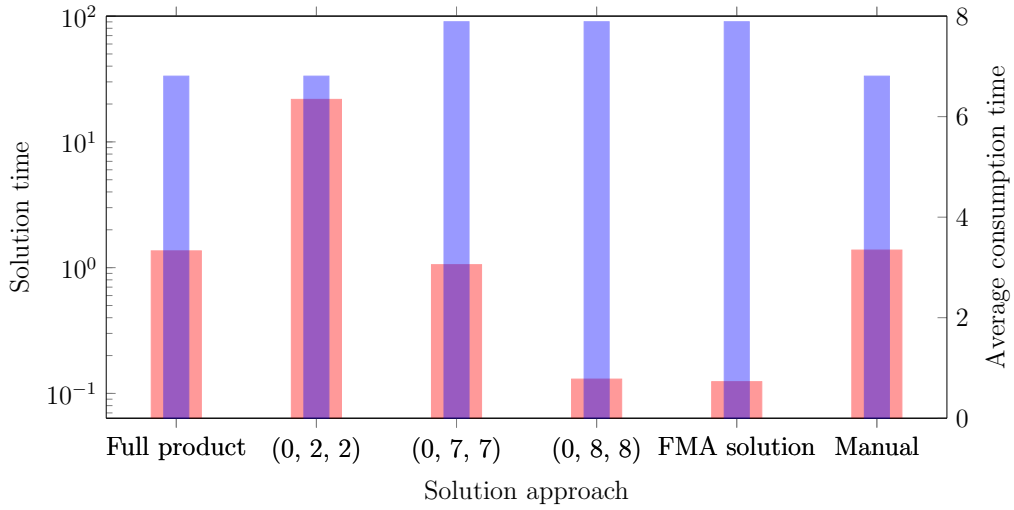


Figure 5.14: Bar plot showing the solution time (red) on the left axis (logarithmic scale) and the average time for the commodities to be consumed (blue) on the right for the different solution approaches corresponding to the logistic network shown in Figure 5.12 and demand models in Figure 5.13. (Reprinted with permission from [4].)

demand model here, being totally deterministic, was contrived to show that compression which operates by merging pairs of states incrementally can be too myopic. The variation in solution times arises from the fact that by reducing the deterministic demand model from 9-states, we make it stochastic, which in turn increases the state space of the product MDP. For example, the product of the two deterministic demand models, $\mathcal{M}^1$ and $\mathcal{M}^2$, results in a structure with 9 states; on the other hand, if one of them, say $\mathcal{M}^1$, is reduced to 8 (making the demand model stochastic) states while keeping the other, $\mathcal{M}^2$, at 9 states (deterministic), the resulting structure has more than 9 states.

The purpose of (6) is to show that idiosyncratic models of demand may require techniques other than the Hellinger reduction one we propose, but that the casual decoupling ideas which underlie the planning approach remains effective, no matter the source of the reduced demand model. The ideas we have explored are, thus, modular.

5.5 Summary

This chapter considers an environment where there exist multiple components that are beyond the influence of an agent's action and presents the formulation in the guise of a scenario from the domain of multi-commodity logistics. The planning problems involve a single autonomous operations agent responsible for routing multiple commodities within a logistic network comprising of nodes that act as either storage units or retail units. We formulate the problem in a modular form consisting of multiple demand models (stochastic process modeling the demand on each retail node) and a graph structure (modeling the logistic network). The chapter provides three approaches to solving the problem by decoupling the analysis of parallel aspects of consumption and presents case studies depicting each approach's effect on solution time and quality. To showcase the effectiveness of the approximate solution approaches presented here, we also provide comparisons with state-of-the-art neural network techniques, A2C and PPO, where we see that even though the neural network techniques were faster in computing the solution, the resulting policy was sub-par to the policies generated via the approximate solutions presented in this chapter.

6. CONCLUSION AND FUTURE WORK

Planning problems involve developing strategies that can be used by autonomous agents to perform tasks in order to achieve some desired goal. In general, planning problems consider autonomous agents performing a series of actions that directly influence the environment they operate in. The research presented in this dissertation deviates from this traditional approach and studies problems where aspects of the problem formulation exist that are not under the influence of the agent's action. Though this assumption of non-influence might appear to be antithetical, these situations often occur in real-world scenarios, often in contexts such as automated narrative generation, event narration, situation depiction, active perception, and logistics, to name a few. This dissertation studies the problem by considering example scenarios involving the generation of structured narrative and multi-commodity logistics and provides various decoupling approaches to solve the problems approximately.

Three problem scenarios are presented in the previous three chapters. The first chapter studies decoupling within the agent's objective by considering the structured video narrative generation problem, where an autonomous videographer robot is tasked with observing its environment and later selectively summarizing what it saw as a vivid structured narrative. The next chapter extends the first for multiple agents and studies the decoupling of each agent's plan from the other agents. These problems are studied in the context of anticipating the evolution of a single stochastic process in order to anticipate events so that the agent can observe those to make progress toward its goal. Finally, we study decoupling by analyzing parallel aspects of the environment by considering a multi-commodity logistics problem. It formulates an autonomous operations agent responsible for routing multiple commodities within a logistic network comprised of retail units and storage units. The context of the last problem involves anticipating multiple stochastic processes that evolve in parallel in order to anticipate future demands.

The decoupling approach taken for all the problems is quite useful as it allows for various efficient and tractable approximate solutions. For the stylized structured narrative generation prob-

lem, the decoupling allows separation of the substance "what to capture?" from the style "how to capture?" for a just-in-time decoupled solution. The multi-agent variation of the structured narrative generation problem decouples the decisions of the agents to be sequentialized. And lastly, for the multi-commodity logistic scenario decoupling allows us to take advantage of the Fundamental Matrix of the Markov chain associated with the EGMC for pre-analyzing the customer demands. All the approaches help reduce the state space, compared to a traditional solution, and therefore provide a computationally effective way of solving the problem.

The next section summarizes the contributions.

6.1 Contributions

The main contributions of this dissertation lie in identifying and decomposing problems as modular components and using decoupling techniques to solve them efficiently and tractably. In terms of modular components, this dissertation provides two contributions. The stochastic process that models the aspects of the formulation uninfluenced by the agent's actions is modeled via a special stateful Markovian structure that traces the temporal correlation within the stochastic process. Secondly, taking inspiration from the field of Natural Language Processing (specifically n-grams), the dissertation quantifies the qualitative nature of the art of videography via a structure called style-gram, allowing the model to be induced from representative corpora and permitting direct incorporation into planning considerations.

The modular formulation allows different opportunities for decoupling that this dissertation takes advantage of to provide efficient approximate solutions. For the structured video narrative problem, decoupling of events from styles stems from the observation that capturing events entails a large-scale activity, needing time to execute — e.g., moving to location — while style choices are more local allowing for the just-in-time solution. The decoupling for the multi-agent structured narrative generation problem leads to a sequential approach solving for each agent by considering stochastic transitions incorporating the events that the other agents (calculated before it) might record, along with the associated probabilities of the events actually occurring, as gratis contributions. And lastly, for the multi-commodity logistic scenario, we observe that when there are

multiple sites in the logistic network and different influences on demand at these sites, they can be factored by splitting into separate site-specific models and analyzed separately. This allows for separate analysis and compression of the stochastic demand models, giving a reduced planning problem that is easier to solve.

To demonstrate the real-world feasibility of the solution approaches presented in this dissertation, a hardware implementation of the stylized structured narrative problem has been presented where multiple robots work cooperatively to produce a video narrative of a race. This implementation was done in collaboration with the University of Houston*, and the University of South Carolina †.

In conclusion, this dissertation studies planning problems where the agents' actions have minimal or no influence over the environment it is operating by providing example scenarios from the domain of automated videography and logistic networks. The dissertation provides a modular approach to formulating the problems and provides decoupled approximate solution approaches to solve the problems efficiently and tractably. And it also provides details of a real-world hardware implementation that shows the real-world feasibility of the approaches.

6.2 Future work

In this section, we focus on the potential immediate extensions of the work and present long-term goals that can be explored.

6.2.1 Immediate extension of the research

An immediate extension of this dissertation can formulate the occurrence of alphabet in the Element-generating Markov chain under various probabilistic generative models, as compared to the Bernoulli generative model used here. Future work could also consider the extension of the EGMC involving an infinite alphabet set as compared to the finite alphabet set structure studied here.

As an example, one could consider the occurrence function to model a Gaussian generative pro-

*Dr. Aaron T. Becker, Rhema Ike, and Omar Romero
†Dr. Jason M. O'Kane, and Dr. Hazhar Rahmani

cess involving an infinite alphabet set. This extension would allow the formulation to be applied to domains such as trajectory control and tracking problems by considering the error in the actuators to be a stateful process being modeled as the Gaussian Element-generating Markov chain. This would allow the study of stateful actuator error instead of the i.i.d. assumption of error mostly considered in research today. The agent in such systems could then utilize the formulation to predict errors in a more refined fashion and take appropriate actions in advance to keep error bounds constrained to some parameter.

6.2.2 Long-term research

For long-term goals, studies can aim to understand the underlying properties that would allow generic MDPs, those whose formulations are not predefined into the environment components (uninfluenced by agents' action) and an objective structure, to be decomposed into such, either precisely or approximately. This would allow the application of the various solution approaches mentioned in the dissertation to be applied to generic MDPs so as to improve the computational efficiency of solving them.

REFERENCES

[1] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 ed., 2010.

[2] D. Chaudhuri, R. Ike, H. Rahmani, D. A. Shell, A. T. Becker, and J. M. O'Kane, "Conditioning style on substance: Plans for narrative observation," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, (Xi'an, China/Online), May 2021.

[3] D. Chaudhuri, H. Rahmani, D. A. Shell, and J. M. O'Kane, "Tractable planning for coordinated story capture: Sequential stochastic decoupling," in *Proceedings of International Symposium on Distributed Autonomous Robotic Systems (DARS)*, (Kyoto, Japan/Online), June 2021.

[4] D. Chaudhuri and D. A. Shell, "A causal decoupling approach to efficient planning for logistics problems with stateful stochastic demand," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, (London, UK), May 2023.

[5] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena scientific, 4 ed., 1995.

[6] D. A. Shell, L. Huang, A. T. Becker, and J. M. O'Kane, "Planning coordinated event observation for structured narratives," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, (Montreal, Canada), May 2019.

[7] H. Rahmani, D. A. Shell, and J. M. O'Kane, "Planning to chronicle," in *Workshop on the Algorithmic Foundations of Robotics (WAFR XIV)*, (Oulu, Finland/Online), June 2020.

[8] W. Feller, *An introduction to probability theory and its applications*. John Wiley & Sons, Inc, 3 ed., 1971.

[9] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2006.

[10] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable markov processes over a finite horizon," *Operations Research*, vol. 21, pp. 1071–1088, September 1973.

[11] M. Araya-López, V. Thomas, O. Buffet, and F. Charpillet, "A closer look at MOMDPs," in *Proceedings of IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, (Arras, France), October 2010.

[12] F. A. Oliehoek and C. Amato, "Dec-POMDPs as non-observable MDPs," ias technical report, Intelligent Systems Lab, University of Amsterdam, Amsterdam, The Netherlands, October 2014.

[13] S. Bradtke and M. Duff, "Reinforcement learning methods for continuous-time markov decision problems," *Advances in neural information processing systems*, vol. 7, pp. 393–400, 1994.

[14] S. Mahadevan, N. Marchalleck, T. K. Das, and A. Gosavi, "Self-improving factory simulation using continuous-time average-reward reinforcement learning," in *Proceedings of International Conference on Machine Learning (ICML)*, (San Francisco, CA, United States), July 1997.

[15] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, pp. 181–211, August 1999.

[16] C. Boutilier, T. Dean, and S. Hanks, "Decision-theoretic planning: Structural assumptions and computational leverage," *Journal of Artificial Intelligence Research (JAIR)*, vol. 11, pp. 1–94, July 1999.

[17] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman, "Efficient solution algorithms for factored MDPs," *Journal of Artificial Intelligence Research (JAIR)*, vol. 19, pp. 399–468, July 2003.

[18] C. Guestrin, M. Hauskrecht, and B. Kveton, "Solving factored MDPs with continuous and discrete variables," in *Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI)*, (Banff, Canada), July 2004.

[19] M. Kearns and D. Koller, "Efficient reinforcement learning in factored MDPs," in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, (Stockholm, Sweden), July 1999.

[20] C. Guestrin, R. Patrascu, and D. Schuurmans, "Algorithm-directed exploration for model-based reinforcement learning in factored MDPs," in *Proceedings of International Conference on Machine Learning (ICML)*, (San Francisco, CA, USA), July 2002.

[21] A. L. Strehl, C. Diuk, and M. L. Littman, "Efficient structure learning in factored-state mdps," in *Proceedings of National Conference on Artificial Intelligence (AAAI)*, (Vancouver, British Columbia, Canada), July 2007.

[22] D. P. De Farias and B. Van Roy, "The linear programming approach to approximate dynamic programming," *Operations Research*, vol. 51, pp. 850–865, December 2003.

[23] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning (ICML)*, (New York, USA), June 2016.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, July 2017.

[25] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, vol. 6, pp. 503–556, April 2005.

[26] I. Mademlis, V. Mygdalis, N. Nikolaidis, M. Montagnuolo, F. Negro, A. Messina, and I. Pitas, "Highlevel multiple-UAV cinematography tools for covering outdoor events," *IEEE Transactions on Broadcasting*, vol. 65, pp. 627–635, September 2019.

[27] I. Mademlis, N. Nikolaidis, A. Tefas, I. Pitas, T. Wagner, and A. Messina, "Autonomous UAV cinematography: A tutorial and a formalized shot-type taxonomy," *ACM Computing Surveys (CSUR)*, vol. 52, pp. 1–33, September 2020.

[28] B. Sabetghadam, A. Alcántara, J. Capitan, R. Cunha, A. Ollero, and A. Pascoal, "Optimal trajectory planning for autonomous drone cinematography," in *Proceedings of European Conference on Mobile Robots (ECMR)*, (Prague, Czech Republic), September 2019.

[29] A. Alcántara, J. Capitan, R. Cunha, and A. Ollero, "Optimal trajectory planning for cinematography with multiple unmanned aerial vehicles," *Robotics and Autonomous Systems (RAS)*, vol. 140, p. 103778, June 2021.

[30] H.-Y. Wu, Q. Galvane, C. Lino, and M. Christie, "Analyzing elements of style in annotated film clips," in *Proceedings of the Eurographics Workshop on Intelligent Cinematography and Editing (WICED)*, (Lyon, France), April 2017.

[31] C. Connolly, "The determination of next best views," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, (Louis, MO, USA), March 1985.

[32] N. Palomeras, N. Hurtós, E. Vidal, and M. Carreras, "Autonomous exploration of complex underwater environments using a probabilistic Next-Best-View planner," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, pp. 1619–1625, April 2019.

[33] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon "next-best-view" planner for 3D exploration," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, (Stockholm, Sweden), June 2016.

[34] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting active perception," *Autonomous Robots*, vol. 42, pp. 177–196, February 2018.

[35] Y. Aloimonos, *Active perception*. Lawrence Erlbaum Associates, Inc., 1993.

[36] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, pp. 966–1005, August 1988.

[37] Y. Jiang, H. Yedidsion, S. Zhang, G. Sharon, and P. Stone, "Multi-robot planning with conflicts and synergies," *Autonomous Robots*, vol. 43, pp. 2011–2032, March 2019.

[38] C. Boutilier and R. Brafman, "Partial-order planning with concurrent interacting actions," *Journal of Artificial Intelligence Research (JAIR)*, vol. 14, pp. 105–136, June 2001.

[39] M. Turpin, N. Michael, and V. Kumar, "Capt: Concurrent assignment and planning of trajectories for multiple robots," *The International Journal of Robotics Research*, vol. 33, pp. 98–112, June 2014.

[40] M. Corah and N. Michael, "Scalable distributed planning for multi-robot, multi-target tracking," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, (Xi'an, China/Online), May 2021.

[41] M. Corah and N. Michael, "Volumetric objectives for multi-robot exploration of three-dimensional environments," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, (Xi'an, China/Online), May 2021.

[42] J. Yu and S. M. LaValle, "Cyber detectives: Determining when robots or people misbehave," in *Workshop on the Algorithmic Foundations of Robotics (WAFR IX)*, (Singapore), December 2010.

[43] J. Yu and S. M. LaValle, "Story validation and approximate path inference with a sparse network of heterogeneous sensors," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), August 2011.

[44] Y. J. Lee, J. Ghosh, and K. Grauman, "Discovering important people and objects for egocentric video summarization," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Providence, RI, USA), June 2012.

[45] M. Gygli, H. Grabner, H. Riemenschneider, and L. V. Gool, "Creating summaries from user videos," in *Proceedings of European Conference on Computer Vision (ECCV)*, (Zurich, Switzerland), September 2014.

[46] B. Gong, W.-L. Chao, K. Grauman, and F. Sha, "Diverse sequential subset selection for supervised video summarization," in *Advances in Neural Information Processing Systems (NIPS)*, (Montreal, Canada), December 2014.

[47] Z. Lu and K. Grauman, "Story-driven summarization for egocentric video," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Portland, OR, USA), June 2013.

[48] Y. Girdhar and G. Dudek, "Efficient on-line data summarization using extremum summaries," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, (St. Paul, MN, USA), May 2012.

[49] Y. Girdhar, P. Giguere, and G. Dudek, "Autonomous adaptive exploration using realtime online spatiotemporal topic modeling," *International Journal of Robotics Research (IJRR)*, vol. 33, pp. 645–657, April 2014.

[50] C.-W. Ngo, Y.-F. Ma, and H.-J. Zhang, "Automatic video summarization by graph modeling," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, (Nice, France), October 2003.

[51] R. Hong, J. Tang, H.-K. Tan, C.-W. Ngo, S. Yan, and T.-S. Chua, "Beyond search: Event-driven summarization for web videos," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 7, pp. 1–18, November 2011.

[52] D. Potapov, M. Douze, Z. Harchaoui, and C. Schmid, "Category-specific video summarization," in *Proceedings of European Conference on Computer Vision (ECCV)*, (Zurich, Switzerland), September 2014.

[53] L. Feng, Z. Li, Z. Kuang, and W. Zhang, "Extractive video summarizer with memory augmented neural networks," in *Proceedings of International Conference on Multimedia (MM)*, (Seoul, Korea), October 2018.

[54] P. Chang, M. Han, and Y. Gong, "Extract highlights from baseball game video with hidden markov models," in *Proceedings of International Conference on Image Processing (ICIP)*, (Rochester, NY, USA), September 2002.

[55] M. H. Kolekar and S. Sengupta, "Event-importance based customized and automatic cricket highlight generation," in *Proceedings of International Conference on Multimedia and Expo. (ICME)*, (Toronto, ON, Canada), July 2006.

[56] H. Hajishirzi, J. Hockenmaier, E. T. Mueller, and E. Amir, "Reasoning in robocup soccer narratives," in *Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI)*, (Barcelona, Spain), July 2011.

[57] S. Rosenthal, S. P. Selvaraj, and M. Veloso, "Verbalization: Narration of autonomous robot experience," in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, (New York City, NY, USA), July 2016.

[58] M. O. Riedl and R. M. Young, "Narrative planning: Balancing plot and character," *Journal of Artificial Intelligence Research (JAIR)*, vol. 39, pp. 217–268, September 2010.

[59] N. D. Allen, J. R. Templon, P. S. McNally, L. Birnbaum, and K. J. Hammond, "StatsMonkey: A data-driven sports narrative writer," in *AAAI Fall Symposium: Computational Models of Narrative*, November 2010.

[60] A. Jhala and R. M. Young, "Cinematic visual discourse: Representation, generation, and evaluation," *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games (T-CIAIG)*, vol. 2, pp. 69–81, June 2010.

[61] S. Chen, A. M. Smith, A. Jhala, N. Wardrip-Fruin, and M. Mateas, "RoleModel: towards a formal model of dramatic roles for story generation," in *Proceedings of Intelligent Narrative Technologies III Workshop (INT)*, June 2010.

[62] N. Szilas, M. Axelrad, and U. Richle, "Propositions for innovative forms of digital interactive storytelling based on narrative theories and practices," *Transactions on Edutainment VII*, pp. 161–179, 2012.

[63] H. Yu and M. O. Riedl, "Personalized interactive narratives via sequential recommendation of plot points," *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games (T-CIAIG)*, vol. 6, pp. 174–187, June 2014.

[64] A. Amos-Binks, C. Potts, and R. Young, "Planning graphs for efficient generation of desirable narrative trajectories," in *Proceedings of AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, (Little Cottonwood Canyon, Utah USA), October 2017.

[65] J. Robertson and R. M. Young, "Narrative mediation as probabilistic planning," in *Proceedings of AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, (Little Cottonwood Canyon, Utah USA), October 2017.

[66] C. Barot, M. Branon, R. E. Cardona-Rivera, M. Eger, M. Glatz, N. Green, J. Mattice, C. M. Potts, J. Robertson, M. Shukonobe, L. Tateosian, B. R. Thorne, and R. M. Young, "Bardic: Generating multimedia narrative reports for game logs," in *Proceedings of AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, (Little Cottonwood Canyon, Utah USA), October 2017.

[67] R. Zhang, F. Rossi, and M. Pavone, "Model predictive control of autonomous mobility-on-demand systems," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, (Stockholm, Sweden), May 2016.

[68] F. Rossi, R. Zhang, Y. Hindy, and M. Pavone, "Routing autonomous vehicles in congested transportation networks: structural properties and coordination algorithms," *Autonomous Robots*, vol. 42, pp. 1427–1442, February 2018.

[69] M. Schaefer, M. Čáp, J. Mrkos, and J. Vokřínek, "Routing a fleet of automated vehicles in a capacitated transportation network," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, (Stockholm, Sweden), May 2016.

[70] D. R. Fulkerson and L. R. Ford, *Flows in Networks*. Princeton University Press, 1962.

[71] T. C. Hu, "Multi-commodity network flows," *Operations Research*, vol. 11, pp. 344–360, May 1963.

[72] K. Salimifard and S. Bigharaz, "The multicommodity network flow problem: state of the art classification, applications, and solution methods," *Operational Research*, vol. 22, pp. 1–47, April 2020.

[73] Q. Jin, S. Feng, M. Li-xin, and T. Gui-jun, "Optimal model and algorithm for multicommodity logistics network design considering stochastic demand and inventory control," *Systems Engineering — Theory and Practice*, vol. 29, pp. 176–183, April 2009.

[74] G. A. Tanonkou, L. Benyoucef, and X. Xie, "Design of multi-commodity distribution network with random demands and supply lead-times," in *Proceedings of IEEE International Conference on Automation Science and Engineering (CASE)*, (Scottsdale, AZ, USA), September 2007.

[75] X. Gao and G. M. Lee, "A stochastic programming model for multicommodity redistribution planning in disaster response," in *Proceedings of International Conference on Advances in Production Management Systems (APMS)*, (Seoul, Korea), August 2018.

[76] I. Mejri, S. B. Layeb, M. Haouari, and F. Z. Mansour, "A simulation-optimization approach for the stochastic discrete cost multicommodity flow problem," *Engineering Optimization*, vol. 52, pp. 507–526, May 2019.

[77] Q. Liu, Q. Zhao, and W. Zang, "Study on multi-objective optimization of flow allocation in a multicommodity stochastic-flow network with unreliable nodes," *Journal of Applied Mathematics and Computing*, vol. 28, pp. 185–198, June 2008.

[78] S. Ding, "Uncertain minimum cost multicommodity flow problem," *Soft Computing*, vol. 21, pp. 223–231, September 2017.

[79] H. Topaloglu and W. B. Powell, "Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems," *INFORMS Journal on Computing*, vol. 18, pp. 31–42, February 2006.

[80] Y.-K. Lin, "Study on the multicommodity reliability of a capacitated-flow network," *Computers and Mathematics with Applications*, vol. 42, pp. 255–264, July 2001.

[81] G. Levitin, I. Gertsbakh, and Y. Shpungin, "Evaluating the damage associated with intentional supply deprivation in multicommodity network," *Reliability Engineering & System Safety*, vol. 119, pp. 11–17, November 2013.

[82] R. Givan, T. Dean, and M. Greig, "Equivalence notions and model minimization in markov decision processes," *Artificial Intelligence*, vol. 147, pp. 163–223, July 2003.

[83] B. Ravindran and A. G. Barto, "SMDP homomorphisms: An algebraic approach to abstraction in semi-markov decision processes," in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, (Acapulco, Mexico), August 2003.

[84] A. K. Mccallum, *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, The University of Rochester, NY, 1996.

[85] N. K. Jong and P. Stone, "State abstraction discovery from irrelevant state variables," in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, (Edinburgh, Scotland), July 2005.

[86] D. Jurafsky and J. H. Martin, *Speech and Language Processing*. Prentice Hall, 2nd ed., 2008.

[87] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

[88] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, June 2016.

[89] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, pp. 1–8, January 2021.

[90] E. Ephrati and J. S. Rosenschein, "Divide and conquer in multi-agent planning," in *Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence (AAAI-94)*, August 1994.

[91] L. Li, T. J. Walsh, and M. L. Littman, "Towards a unified theory of state abstraction for mdps," in *International Symposium on Artificial Intelligence and Mathematics (AI&M)*, (Fort Lauderdale, Florida, USA), January 2006.

[92] C. M. Grinstead and J. L. Snell, *Introduction to probability*. American Mathematical Soc., 1997.

[93] E. Hellinger, "Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen.," *Journal für die reine und angewandte Mathematik*, vol. 1909, no. 136, pp. 210–271, 1909.