8-31-2023

Human-AI complex task planning

Sepideh Nikookar
New Jersey Institute of Technology, sn627@njit.edu

ABSTRACT

HUMAN-AI COMPLEX TASK PLANNING

by
Sepideh Nikookar

The process of complex task planning is ubiquitous and arises in a variety of compelling applications. A few leading examples include designing a personalized course plan or trip plan, designing music playlists/work sessions in web applications, or even planning routes of naval assets to collaboratively discover an unknown destination. For all of these aforementioned applications, creating a plan requires satisfying a basic construct, i.e., composing a sequence of sub-tasks (or items) that optimizes several criteria and satisfies constraints. For instance, in course planning, sub-tasks or items are core and elective courses, and degree requirements capture their complex dependencies as constraints. In trip planning, sub-tasks are points of interest (POIs) and constraints represent time and monetary budget, or user-specified requirements. Needless to say, task plans are to be individualized and designed considering uncertainty. When done manually, the process is human-intensive and tedious, and unlikely to scale. The goal of this dissertation is to present computational frameworks that synthesize the capabilities of human and AI algorithms to enable task planning at scale while satisfying multiple objectives and complex constraints.

This dissertation makes significant contributions in four main areas, (i) proposing novel models, (ii) designing principled scalable algorithms, (iii) conducting rigorous experimental analysis, and (iv) deploying designed solutions in the real-world. A suite of constrained and multi-objective optimization problems has been formalized, with a focus on their applicability across diverse domains. From an algorithmic perspective, the dissertation proposes principled algorithms with theoretical guarantees adapted from discrete optimization techniques, as well as Reinforcement Learning based solutions. The memory and computational efficiency

of these algorithms have been studied, and optimization opportunities have been proposed. The designed solutions are extensively evaluated on various large-scale real-world and synthetic datasets and compared against multiple baseline solutions after appropriate adaptation. This dissertation also presents user study results involving human subjects to validate the effectiveness of the proposed models. Lastly, a notable outcome of this dissertation is the deployment of one of the developed solutions at the Naval Postgraduate School. This deployment enables simultaneous route planning for multiple assets that are robust to uncertainty under multiple contexts.

HUMAN-AI COMPLEX TASK PLANNING

by
Sepideh Nikookar

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology and
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Sciences, NJIT

August 2023

APPROVAL PAGE

HUMAN-AI COMPLEX TASK PLANNING

Sepideh Nikookar

Senjuti Basu Roy, Dissertation Advisor — Date
Associate Professor of Computer Science, NJIT

Guiling Wang, Committee Member — Date
Distinguished Professor of Computer Science, NJIT

Yiannis Koutis, Committee Member — Date
Associate Professor of Computer Science, NJIT

David Shirokoff, Committee Member — Date
Associate Professor of Mathematical Science, NJIT

Krishna R. Pattipati, Committee Member — Date
Distinguished Professor of Electrical and Computer Engineering,
University of Connecticut, Storrs, CT

BIOGRAPHICAL SKETCH

Author: Sepideh Nikookar

Degree: Doctor of Philosophy

Date: August 2023

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2023
- Master of Science in Applied Mathematics,
  K.N. Toosi University of Technology, Tehran, Iran, 2016
- Bachelor of Science in Applied Mathematics,
  K.N. Toosi University of Technology, Tehran, Iran, 2014

Major: Computer Science

Publications:

S. Nikookar. "Human-AI Complex Task Planning," *39th IEEE International Conference on Data Engineering Ph.D. Symposium, 2023.*

S. Nikookar, S. Somasunder, P. Sakharkar, S. Basu Roy, A. Bienkowski, M. Macesker, K. Pattipati, D. Sidoti. "Cooperative route planning framework for multiple distributed assets in maritime applications," *Proceedings of the 2022 International Conference on Management of Data.*

S. Nikookar, P. Sakharkar, B. Smagh, S. Amer-Yahia, S. Basu Roy. "Guided task planning under complex constraints," *2022 IEEE 38th International Conference on Data Engineering.*

S. Nikookar, M. R. Esfandiari, R. M. Borromeo, P. Sakharkar, S. Amer-Yahia, S. Basu Roy. "Diversifying recommendations on sequences of sets," *The VLDB Journal, 2022.*

M. R. Esfandiari, R. M. Borromeo, **S. Nikookar**, P. Sakharkar, S. Amer-Yahia, S. Basu Roy. "Multi-session diversity to improve user satisfaction in web applications," *Proceedings of the Web Conference 2021.*

Presentations:

S. Nikookar."Human-AI Complex Task Planning," *39th IEEE International Conference on Data Engineering Ph.D. Symposium*, Anaheim, CA, April 2023.

S. Nikookar."Introduction to Reinforcement Learning," *NJIT Department of Mathematical Sciences Optimization and Machine Learning Workshops*, Invited Talk, December 2022.

S. Nikookar."Human-AI Complex Task Planning," *31st ACM International Conference on Information and Knowledge Management Workshop on Human-in-the-loop Data Curation*, Atlanta, GA, October 2022.

S. Nikookar, S. Somasunder, P. Sakharkar, S. Basu Roy, A. Bienkowski, M. Macesker, K. Pattipati, D. Sidoti. "Cooperative Route Planning Framework for Multiple Distributed Assets in Maritime Applications," *SIGMOD/PODS '22: Proceedings of the 2022 International Conference on Management of Data*, Philadelphia, PA, June 2022.

S. Nikookar, P. Sakharkar, B. Smagh, S. Amer-Yahia, S. Basu Roy. "Guided Task Planning Under Complex Constraints," *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, Virtual, May 2022.

S. Nikookar."Human-AI Complex Task Planning," *NJIT Department of Mathematical Sciences Optimization and Machine Learning Talks*, Invited Talk, April 2022.

M. R. Esfandiari, R. M. Borromeo, **S. Nikookar**, P. Sakharkar, S. Amer-Yahia, S. Basu Roy. "Multi-session diversity to improve user satisfaction in web applications," *WWW '21: Proceedings of the Web Conference 2021*, Virtual, April 2021.

Posters:

S. Nikookar. "Human-in-the-loop (HIL) AI," *NAI-NJIT Workshop on Sustainable Societies: Data Revolution*, October 2022, Newark, NJ.

S. Nikookar. "Human-in-the-loop (HIL) AI," *NJIT Research Institutes, Centers and Laboratories Showcase and President's Forum*, 2019, Newark, NJ.

@Blacksuitdragon

To the guiding light that has illuminated my path throughout my life and infused my life with purpose and strength.

In loving memory of my father, whose indomitable spirit and wisdom continue to resonate within me.

To my incredible mother for her boundless love and sacrifices.

To my beloved husband for his endless support and encouragement.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor Dr. Senjuti Basu Roy for her invaluable guidance and unwavering support throughout my Ph.D. Her expertise, patience, and encouragement were instrumental in the successful completion of this dissertation. I am also grateful for the opportunities she provided me to develop my skills and knowledge. I will always be grateful for her mentorship and the lessons learned from her.

I extend my sincere appreciation to the members of my dissertation committee, Drs. Guiling Wang, Yiannis Koutis, David Shirokoff, and Krishna R. Pattipati for their valuable insights, feedback, and guidance throughout the process of completing this dissertation. Their expertise and constructive criticism have been invaluable in shaping my research and enhancing the quality of my work.

I would also like to thank Dr. Sihem Amer-Yahia for her invaluable support and guidance throughout my research. I am grateful for the time and effort that she dedicated to our collaboration, and for her willingness to share her expertise with me.

My thanks to the Department of Computer Science for providing me with the opportunity to pursue a degree at such a prestigious university. I am grateful for the financial support that I received through the National Science Foundation (CAREER Award #1942913, IIS #2007935, IIS #1814595, PPoSS: Planning #2118458) and the Office of Naval Research (Grants No, N000141812838, N000142112966) that has enabled me to focus on my studies and research.

Additionally, thanks to my fellow students: Dong Wei, Mahsa Asadi, Md. Mouinul Islam, Sohrab Namazi Nia, and Paras Sakharkar for their support and many insightful conversations we have had about our research topics, which have broadened my understanding of different fields and provided me with new ideas. Their support

has made this journey much more enjoyable, and I am proud to have been part of such an inspiring group.

Special thanks go to my best friends: Akram Madahian, Shima Ghavimi, Christeen Bisnath, Hamed Azimi, and Pejman Sanaei, for their enduring friendship, which has been a constant source of joy and comfort in my life. Thank you for being my rock and my cheerleader.

I would like to take a moment to remember my father, Hossein Nikookar, and grandmother, Masomeh Zandi, who unfortunately passed away before I was able to complete this dissertation. My father and grandmother were an important part of my life, and I am grateful for the love and support that they provided me with throughout the years. Although they are no longer with me, I am sure that they would be proud of my accomplishments and the hard work that I have put into this research. I would like to dedicate this dissertation to their memory and honor their legacy. Their love, support, and guidance will always be remembered, and their absence is felt deeply. I wish they could be here with me to share this moment, but I know that they are watching over me from above. I will always be grateful for the lessons that they taught me. Thank you for everything, and rest in peace.

I also need to thank my mom, Fahimeh Johari, who raised my brother and me single-handedly after the passing of our father. She made countless sacrifices to ensure that we had everything we needed to succeed in life, often at the expense of her own needs and desires. Her selflessness and dedication have been instrumental in shaping me into the person I am today. I am grateful for her constant belief in me, her words of encouragement, and her unending support. Mom, your love and sacrifice have been the driving force behind my success, and for that, I am forever grateful. This dissertation is a testament to your unwavering love and the immeasurable impact you have had on my life. Thank you for everything.

I like to show my appreciation to my brother, Tooraj, his wife, Sadaf, and my nieces, Nila and Neli for their unconditional love and support throughout my academic journey. While I was studying for my Ph.D. in another country, they took care of my mother and provided her with the care and support she needed. Their selflessness and dedication allowed me to focus on my studies and achieve my academic goals. I am grateful for their understanding and their practical support, which enabled me to pursue my passion while knowing that my mother was in good hands. They are my family and my support system, and I am blessed to have them in my life.

I would also like to thank my brother-in-law, Sepehr Saghafi, for his support, encouragement, and friendship throughout my dissertation journey. His constant presence and belief in my abilities have been invaluable.

Last but not least, I would like to express my heartfelt gratitude to my husband, Soheil Saghafi, for his support and dedication throughout my academic journey in a foreign country. As my only support system in a foreign land, his love, care, and emotional support have been invaluable to me during a time of immense personal and academic growth. He has been my confidante, my friend, and my partner in every sense of the word. His love and encouragement have kept me going during moments of doubt and uncertainty. I could not have done this without his love and support, and for that, I am forever grateful. He has been a true partner in every sense of the word.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

Table **Page**

LIST OF TABLES
(Continued)

Table **Page**

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

The process of complex task planning is ubiquitous and arises in a variety of compelling applications. A few leading examples include designing a personalized course plan [21, 23, 41, 44, 54, 57, 58, 82, 103, 129] or trip plan [10, 19, 39, 43, 62, 68], designing music playlists/work sessions in web applications [17, 91, 122], or even planning routes of naval assets [16, 16, 32, 98, 121] to collaboratively discover an unknown destination. Additionally, planning routes for multiple agents, such as ships submarines, and unmanned aerial/surface/underwater vehicles (UAVs, USVs, UUVs), considering multiple objectives, such as fuel, battery usage, time taken, and progress towards a goal, is a complex, but highly-relevant, problem for search and rescue, reconnaissance, and interdiction missions in maritime applications. These problems involve trade-offs among different objectives and require a coordinated search for an object in a very high-dimensional space by multiple geographically distributed searchers (herein referred to as assets or agents).

Not only that, complex task planning finds applications in fields like manufacturing [80,92], project management [132], healthcare [90], and resource allocation [93]. Whether it's optimizing production schedules, coordinating tasks in a construction project, allocating healthcare resources efficiently, or managing complex supply chains, studying and developing effective planning techniques can lead to improved productivity, cost savings, and enhanced decision-making in these domains. The motivation for studying complex task planning lies in the need to address challenges and optimize decision-making processes in various real-world applications. By understanding and developing effective planning algorithms, we can unlock potential advancements and improvements in autonomous systems, manufacturing, logistics,

healthcare, and many other fields where complex task execution is a critical component.

For all of these aforementioned applications, creating a plan requires satisfying a basic construct, i.e., composing a sequence of sub-tasks (or items) that optimizes several criteria and satisfies constraints. For instance, in course planning, sub-tasks or items are core and elective courses, and degree requirements capture their complex dependencies as constraints. In trip planning, sub-tasks are points of interests (POIs) and constraints represent time and monetary budget, or user-specified requirements. Needless to say, task plans are to be individualized and designed considering uncertainty. Moreover, the dynamic nature of real-world environments introduces additional complexities. Unforeseen events, changes in task requirements, or disruptions necessitate the ability to adapt plans on-the-fly. This requires planning algorithms that can handle uncertainty, reason under incomplete information, and generate flexible plans that can be adjusted as the situation evolves. When done manually, the process is human-intensive and tedious, and unlikely to scale. On the other extreme, embracing these challenges present opportunities to develop fully automated computational processes that leverage techniques such as machine learning, optimization, and knowledge representation to generate efficient and adaptable plans. Additionally, advancements in collaborative planning and multi-agent systems offer opportunities for decentralized planning and coordination among multiple agents, enabling the execution of complex tasks in distributed environments. However, fully automated techniques [6, 43, 44, 62, 103] require significant training data to be effective, produce plans that are less transparent or opaque to human comprehension. A middle ground is to judiciously leverage human inputs and design-guided computational frameworks that satisfy multiple objectives and complex constraints, are generalizable across multiple applications, and can scale. The goal of this dissertation is to present computational frameworks that synthesize

the capabilities of human and AI algorithms to enable task planning at scale while satisfying multiple objectives and complex constraints.

1.1 Contributions

The limitations of the state-of-the-art approaches in complex task planning can vary depending on the specific techniques and methodologies employed. However, there are several common limitations that researchers often encounter, including 1. Scalability, 2. Uncertainty and Dynamic Environment, 3. Large State and Action Spaces, and 4. Real-Time Constraints. To fill these gaps in the related works, we intend to build on prior expertise in modeling several high-fidelity data science problems and studying efficiency opportunities. Therefore, we present a set of computational frameworks for automated task planning as a sequence generation problem that requires minimal inputs from the end users and produces personalized task plans in an uncertain environment while satisfying multiple objectives and complex constraints. At the core, we propose a set of multi-objective optimization problems with constraints, solving which will generate task plans as a sequence of sub-tasks that are highly dependent and optimize the underlying problems. From the algorithmic standpoint, this dissertation designs novel algorithms by adapting Reinforcement Learning and discrete optimization-based techniques with theoretical guarantees. This dissertation also studies data engineering and data management opportunities to design scalable algorithms. Finally, the work presents large-scale synthetic and real-world experiments, as well as deployment challenges in the real-world environment.

In Chapter 3, we study the Task Planning Problem (TPP) where items are considered as sub-tasks, with the goal of generating a sequence of items that optimizes multiple objectives while satisfying complex constraints. TPP is modeled as a Constrained Markov Decision Process, and we adapt weighted reinforcement

learning to learn a policy that satisfies complex dependencies between sub-tasks, user requirements, and satisfaction. We present a computational framework RL-Planner for TPP. RL-Planner produces personalized plans satisfying all constraints while requiring minimal input from domain experts. We run extensive experiments on datasets from university programs and from travel agencies. We compare our solutions with plans drafted by human experts and with fully automated approaches. Our experiments corroborate that existing automated solutions are not suitable to solve TPP and that our plans are highly comparable to expensive handcrafted ones. Unlike existing works, our approach is highly amenable to scenarios where historical data is not available. That is often the case with new education programs or new travel plans.

Diversifying recommendations on a sequence of sets (or sessions) of items captures a variety of applications. Notable examples of such applications include recommending online music playlists, where a session is a channel and multiple channels are listed in sequence, or recommending tasks in crowdsourcing, where a session is a sub-task and multiple sub-tasks are completed in sequence. Item diversity can be defined in more than one way, e.g., as an artist or genre diversity for music, or as a function of task description or reward in crowdsourcing. A user who engages in multiple sessions may intend to experience diversity within and/or across sessions. *Intra-session diversity* is set-based, whereas, *inter-session diversity* is naturally sequence-based. This novel formulation gives rise to four bi-objective problems with the goal of optimizing inter and intra diversities in Chapter 4. We prove hardness and develop efficient algorithms with theoretical guarantees. Our experiments with human subjects on two real datasets, music, and crowdsourcing, show our diversity formulations do serve different user needs and yield high user satisfaction. Our large data experiments on real and synthetic data empirically

demonstrate that our solutions satisfy the theoretical bounds and are highly scalable, compared to baselines.

Planning routes for multiple assets while considering multiple objectives is a complex, but highly relevant, problem. These challenges require a coordinated search for an item in a very high-dimensional space by multiple searchers. We consider the case where multiple assets have a single destination, whose location is unknown until one of the assets comes within the sensing range. This means that the assets need to first search for the destination or target, and then all assets are required to find a route to the destination. The Route Planning Problem (**RPP**) is an attempt to address the problem of navigating a set of distributed assets across a discrete grid. The aim is to determine a sequence of actions/sub-tasks for each asset (route plan) that minimizes overall fuel consumption and optimizes time to attain the mission goal while avoiding collisions among themselves. In Chapter 5, RPP is formalized as a Team Discrete Markov Decision Process (TDMDP) and we propose a Multi-agent Multi-objective Reinforcement Learning (MaMoRL) framework for solving it. We investigate challenges in deploying the solution in real-world settings and study approximation opportunities. We also continue to investigate further efficiency issues in order to overcome the scalability challenges by looking into how to use the Rollout policy in a distributed environment to design a multi-objective decision-making framework with multiple agents. We experimentally demonstrate MaMoRL's effectiveness on multiple real-world and synthetic grids, as well as for transfer learning. MaMoRL is deployed for use by the Naval Postgraduate School.

1.2 Ongoing Work

To summarize, this dissertation presents a set of computational frameworks for automated task planning with minimal end-user input despite the existing approaches that mainly rely on logs or training data to model user preference. Our algorithms

are effective for transfer learning, and we've conducted comprehensive experiments to demonstrate their superiority over multiple baselines.

We continue to explore several related challenges in this space. Classical RL frameworks, such as SARSA, and Q-learning [127], suffer from prohibitively expensive memory and CPU requirements. These processes require storing the maximum expected future rewards (Q values) for each action-state pair which needs terabytes of space, hence bringing them into the main memory is impossible. In particular, there are three significant challenges that we continue to study.

- **Challenge 1: Exponential state/action space.** One of the limitations of RL frameworks is their struggle in dealing with problems characterized by an exponential number of states and actions. RL algorithms typically rely on exploration and exploitation of the state/action space to learn an optimal policy. However, when the complexity of the problem grows, the number of potential states and actions increases exponentially, making it challenging for RL algorithms to explore and evaluate all the potential combinations. Consequently, as the size of the state/action space expands, the performance of RL algorithms can degrade rapidly.
- **Challenge 2: Large memory requirement to store the Q values.** Classical RL algorithms often employ Q-values to estimate the expected rewards associated with different actions in specific states. However, in scenarios characterized by large state and action spaces, the number of Q-values that must be stored and updated can be overwhelmingly large. Managing and retaining these values in memory can become a bottleneck, as it necessitates substantial memory resources. As the complexity of the task increases, the memory requirements can grow to an impractical extent, limiting the applicability of RL algorithms to domains with extensive state and action spaces.
- **Challenge 3: Significant learning time to be effective.** Another limitation of classical RL frameworks is the substantial learning time required to achieve effective performance. RL algorithms typically learn through interaction with the environment, gradually updating their policies based on observed rewards. In complex domains, the learning process can be time-consuming due to the need for extensive exploration and exploitation to discover optimal policy. Moreover, when reward signals are scarce or delayed, the learning time can be further prolonged as RL agents struggle to identify the most rewarding actions. This long learning time can be impractical in scenarios where quick decision-making or adaptability is required, hindering the widespread application of RL to real-world problems.

We have been investigating the computational and memory bottlenecks of RL from two distinct perspectives as follows:

1. **Design an Online Policy:** We investigate how to design an online policy using techniques such as Rollout policy [29]. The Rollout policy is a one-time policy iteration approach that can be utilized in a distributed environment. By employing this technique, a multi-objective decision-making framework can be created, involving multiple agents. This framework enables efficient decision-making by considering various objectives and optimizing the actions of multiple agents simultaneously. By leveraging the Rollout policy and incorporating it into a distributed setting, the proposed solution tackles Challenge 1 as well as Challenge 3.

2. **Employ Data Management Techniques:** One of the critical challenges in RL arises when a substantial amount of training is required for effective learning, which is commonly referred to as Challenge 3. In order to overcome this challenge, we propose the implementation of various data management techniques. By utilizing these techniques, we can successfully address the issue of extensive training requirements in RL. Moreover, our proposed solution not only tackles Challenge 3 but also effectively handles Challenge 1 and Challenge 2.

CHAPTER 2

BROAD OVERVIEW OF THE RELATED WORK

My research focus lies on the context of Data Management for AI and Complex Task Planning by designing efficient solutions for AI and leveraging data management techniques.

2.1 Data Management for AI

Machine Learning (ML) methods encounter several challenges, including the need for a) Data Discovery, Access, and Versioning; b) Model Management; and c) Model Tracking and Provenance. To overcome these challenges, researchers are concentrating on leveraging various data management techniques. Notable examples include the works of Agrawal et al., 2019 [5] and Karanaso et al., 2019 [86], who proposed the development of Raven, a system that combines ML inference within the database and employs cross-optimization strategies. The core contributions of these works include integrating ML inference into traditional relational query processing and enabling the execution of ML models within relational databases for enhanced data analytics and decision-making. This integration expands the range of queries that can be performed on relational data by incorporating ML inference capabilities. Integrating ML inference into relational query processing poses scalability challenges due to the computational intensity of ML models, requiring careful resource allocation and optimization techniques. Additionally, limitations in the integration of certain ML model types or complex architectures may restrict the applicability of the proposed approach within the relational query processing framework.

In the study conducted by Teh et al., 2017 [131], data management techniques called distill and transfer learning were employed to simultaneously train multiple tasks by sharing a "distilled" policy that encompasses common behavior across the

tasks. This approach involves extracting the knowledge acquired from one task and distilling it into a shared policy, which is then transferred to other tasks. The process of policy distillation can be utilized to extract the policy of a Reinforcement Learning agent and train a new network that achieves expert-level performance, all while being significantly smaller and more efficient. This work faces limitations in terms of the diversity of behaviors it can exhibit due to constraints in incorporating multiple distilled policies or latent variables. This limitation may impact its adaptability to various scenarios and its effectiveness in complex environments. Additionally, the proposed approach does not address the challenges of sequential task learning, potentially hindering its ability to effectively adapt to new tasks while retaining knowledge from previous ones.

The utilization of the Reward Machine (RM) is an example of a data management technique explored in the research conducted by Icarte et al., 2018 [78]. RM enables the composition of different reward functions in a flexible manner. They operate by taking abstracted descriptions of the environment as input and generating corresponding reward functions as output. The underlying idea is that the agent receives rewards from different functions at different times, based on the transitions performed within the RM. Using RM for task specification and decomposition in Reinforcement Learning has potential limitations to consider. One limitation relates to the expressiveness of RM. While RM provides a structured approach to task specification, it may have limitations in accurately representing complex and nuanced objectives. This limitation could potentially hinder the learning process and the agent's ability to achieve desired outcomes. Another limitation pertains to generalization to new tasks. The approach's effectiveness in transferring and adapting learned knowledge to novel, unseen tasks beyond the initially specified ones might be challenging. Additional mechanisms or techniques may be necessary to ensure successful generalization, which is crucial for practical applications.

2.2 Complex Task Planning

Traditionally, complex task planning is highly domain dependent, hard to generalize, and requires heavy domain expertise. As an example, course planning processes are primarily manual, designed by academic advisors, and therefore time-consuming. On the other hand, generating task sessions in online job platforms, such as Amazon Mechanical Turk, are primarily ad-hoc, not personalized to individual user needs or dependency on the tasks in generating sessions, leading to poor user satisfaction and performance. Some of the existing fully automated approaches [6, 43, 44, 62, 103] heavily rely on logs or training data to model user preference, which is not the case in our work. Task Planning is also explored in [133], [22] and [111]. However, they do not account for constraints when producing a sequence of sub-tasks, which is crucial for our problems.

As an illustration of Complex Task Planning, let's consider the problem of Task Planning for Robotized Warehouses. The existing cutting-edge solutions for this problem have a fixed duration for completing tasks, making them inflexible when it comes to planning for varying arrival times of items. Another drawback of current research is the inefficient planning they offer for warehouses with a large number of robots and items. To overcome these limitations, Shi et al., 2022 [119] proposed an adaptive approach called Adaptive Task Planning (ATP). This method involves selecting a subset of racks for robots to pick up and deliver, rather than processing all racks immediately upon item arrival. Shi et al., 2022 [119] also employ data management techniques, such as memory compression, to optimize path finding and rack selection. The proposed approach, although effective in a research setting, may face challenges when it comes to implementing it practically in real-world warehouses. Factors such as integrating with existing warehouse management systems, compatibility with various types of robots, and the requirement for additional infrastructure can present limitations and obstacles.

Basu Roy et al., 2011 [19] introduce an interactive approach to itinerary planning that leverages user feedback and itinerary expected scores. The main contribution lies in formalizing this approach and addressing the challenges of constructing personalized itineraries while minimizing user effort. The system suggests POIs and recommends itineraries based on user feedback, with the user providing input at each step until they are satisfied. The paper defines the POI Feedback Model and Itinerary Scoring Semantics to capture preferences and score itineraries. It also presents efficient algorithms for real-time computation of the best-scoring itineraries and proposes a probabilistic model by reducing the problem to the rooted orienteering problem and utilizing strategies such as computing a Hamiltonian path in a hypercube and employing an efficient heap-based data structure for POI pruning that maximize expected scores. The paper's limitations include a lack of exploration of different feedback models, primarily focusing on the binary model. Multi-user scenarios are not considered, neglecting conflicting preferences and group dynamics. Additionally, the paper fails to adequately address budget constraints other than time, such as monetary constraints or physical limitations. The scoring semantics are simplified, focusing only on set semantics, without considering the effectiveness of alternative scoring semantics like chain semantics. Addressing these limitations would provide valuable insights, enhance practical applicability, and enable more personalized recommendations in itinerary planning.

The core contribution of [36] is the development of three approximation algorithms to efficiently solve the Keyword-aware Optimal Route (KOR) query problem. The KOR query aims to find the optimal route in a graph that satisfies constraints such as covering specific keywords and adhering to a budget limit while optimizing an objective score. The first algorithm, called OSScaling, scales the objective values of edges and generates routes with objective scores no worse than a certain factor of the optimal route. The second algorithm, BucketBound,

further improves efficiency while providing approximate solutions with performance guarantees. The paper also introduces a greedy approach that trades off constraints but is efficient. Cao et al., 2012 [36] make certain assumptions about the feasibility of assigning objective scores and budget scores to every edge in the graph. Additionally, they assume that the only constraints to consider are keyword coverage and budget limit. However, these assumptions may not be valid in all real-world situations, which can render the proposed algorithms inapplicable or ineffective in such cases.

The research conducted in [68] revolves around creating a framework for personalized tour recommendations in urban navigation, specifically within smart cities. Gionis, et al., 2014 [68] utilize contextual information gathered from location-based social networks to offer personalized tour suggestions. They highlight the limitations of existing recommendation systems that evaluate venues independently and propose a problem called TourRec that takes into account venue types, user constraints, and user satisfaction. The paper introduces two variations of the problem and presents efficient algorithmic solutions for each. The first variation, AdditiveTour, involves assigning a benefit value to each location and determining the optimal sequence of venues that maximizes the overall benefit. This is subject to constraints such as a distance budget and a preferred order of visiting venue types. The second variation, CoveringTour, associates each venue with a set of attractions it covers within a specified range, aiming to maximize the total number of attractions covered while considering the same constraints. The paper's limitations include a dependency on location-based social networks, which limits its applicability in regions with less popular networks. The simplified satisfaction measurement methods may not accurately capture all factors contributing to user satisfaction. Real-time constraints, such as dynamically changing conditions and time-specific events, are not explicitly considered, which affects the relevance and timeliness of the recommendations.

Exploratory Data Analysis (EDA) shares similarities with complex task planning aimed at assisting users in refining their requirements and discovering relevant items within extensive collections of structured data. Within the realm of EDA, there exists a specific case known as Text-based Item Exploration (TIE), which presents unique challenges. These challenges include the abstraction and querying of textual data and the need to combine queries involving both structured and unstructured content. To tackle these challenges, Omidvar-Tehrani et al., 2022 [102] have developed GUIDE, a framework that encompasses text dimensions such as sentiment and topics. GUIDE introduces novel text-based operators that seamlessly integrate with conventional EDA operators, enabling effective exploration of text-based data alongside structured information. A limitation of this work is the absence of user feedback incorporation within the reward mechanism, which could enhance exploration by offering a more personalized and effective experience.

While guided EDA bears similarities to complex task planning, to the best of our knowledge, there is no EDA framework that accounts for the intricate constraints required in task planning.

CHAPTER 3

GUIDED TASK PLANNING UNDER COMPLEX CONSTRAINTS

3.1 Introduction

Task planning [10, 19, 23, 103] is a complex and time-consuming effort that is ubiquitous and has a wide range of applications, such as planning courses or trips.

Consider the scenario of an aspiring youngster wanting to jump-start her career as a data scientist right after her B.S. in Computer Science, or a seasoned IT analyst with years of experience in the industry wanting to join the bandwagon of data science to change her career focus. For both individuals, designing a course plan (e.g., for an M.S. degree or a certificate in data science) is a complex and intellectually demanding task with the goal of managing their education goal, and satisfying different requirements that are compatible with their experience and background. Similarly, a student wanting to discover museums in Europe, or a senior traveler seeking to enjoy Southeast Asia, will need a trip plan that combines very different requirements. In this chapter, we propose a computational framework to automate task planning that is applicable to a variety of domains.

The current best practices of task planning offer a continuous and consistent process which is mostly done under the guidance of human experts (e.g., academic advisors and travel agents). It is needless to say that such a fully manual approach is expensive and inherently not scalable. Contrarily, a fully automated approach [6, 43, 44, 62, 103] may require significant historical data or logs to learn personalized models. We advocate that task planning must be studied as a *sequence generation problem that is sensitive to the ordering and interleaving of items, personalized and captures progression in task achievement, as well as satisfies a multitude of complex constraints*. While this bears similarity to guided Exploratory Data Analysis (*EDA*),

to the best of our knowledge, there is no *EDA* framework that accounts for intricate constraints required in task planning.

We propose, RL-Planner, a mostly automated computational framework that requires minimal input from stakeholders, yet produces highly effective task plans that are personalized and relevant. Scenarios where in-person education or travel advising is rare and costly and the platforms that need to scale up the process to thousands of items (such as MOOCs [118] and vacation rentals [1]) are ideal for our problem.

Example 3.1. *Let us consider the case of a student aspiring to obtain an M.S. degree in Data Science Computational Track (DS-CT) with a B.S. degree in a STEM discipline. At the very minimum, the student must have knowledge of the core (i.e., primary) CS subjects, such as Algorithms & Data Structures, in Mathematical Science, such as Probability and Linear Algebra, in Programming Languages (such as Python and RStudio), and finally Applied Data Science topics (Databases, Data Mining, Machine Learning). Additionally, the student must satisfy the minimum credit requirement as well as the primary vs. secondary split (e.g., 5 core courses and 5 electives). The student must also take the prerequisites before the electives (e.g., take Linear Algebra before Machine Learning or DBMS before Data Mining) at least a semester before. The recommended courses should satisfy the student's broader goal of becoming a computational data scientist after completing the degree program. Additionally, the student may aspire to learn some specific topics (e.g.,* Classification, Clustering, Neural Networks, Linear Systems*), or may be interested in taking elective courses or in completing a project that is specifically designed to gain knowledge in Data Science in some application domains (such as pharmaceutical, health-care, or fintech). Such specifications must come from the student. How to design the ideal sequence of core and elective courses and interleave them (e.g., start with one or*

[1] https://www.airbnb.com/

two core courses, then take two electives, then another core course) requires domain expertise that only academic advisors are capable of providing.

Example 3.2. *Consider a first-time traveler looking for an exciting day-long trip in Paris (must be completed in 6 hours of visitation time). There are some "must-visit places" that are the primary POIs (e.g., Eiffel Tower, Louvre Museum). The remaining POIs are secondary/optional types (e.g., Pantheon, Rue des Martyrs, etc). The traveler provides some preferences and requirements, such as, she wants to visit Museum, Art Gallery, be by a River, enjoy local food (Restaurant/Cafe), and experience Architecture of historic importance. An ideal itinerary should start the day with POIs that are time-consuming and physically strenuous to visit (such as Museum/Art Gallery) (formalized as prerequisites or antecedents later on) following which one can experience some relaxation time by visiting a Restaurant/Cafe, does not contain many POIs of the same type (e.g., only one Museum/Art Gallery, one River, etc), and the itinerary is easily commutable. Only an experienced travel agent can craft one such itinerary that satisfies these multitudes of requirements, i.e., ensuring the presence of primary POIs and selecting the best from the secondary types to bring variety, ensuring appropriate interleaving, and satisfying other constraints the traveler provides.*

The aforementioned examples call out the following requirements in task planning - (1) **Satisfying Hard Constraints:** Plans must match these constraints as part of the requirement for the task (e.g., # primary vs. # secondary, as well as antecedents/prerequisite requirements). (2) **Maximizing Soft Constraints:** These are of two different kinds: (a) Designed plans must maximize the coverage of the topics/themes users wish to acquire (e.g., recommend courses on clustering and neural networks, or POIs related to library and cathedral); (b) Recommended sequences must adhere to the domain expert's provided "template" as much as possible. A "template" is a set of ideal permutations of primary and secondary

items (Sub-section 3.2.1 has further details) and the generated task plan must follow these *ideal compositions as closely as possible*.

This work makes the following contributions: **1. Formalizing Task Planning as a Decision-Making Problem:** Our first contribution is to formalize the task Planning Problem **(TPP)** as a constrained sequence generation problem. We model **TPP** as a Constrained Markov Decision Process (CMDP) [9]. where a state is an item, an action generates a transition that adds one more item, and a "reward" is associated with every transition to quantify how well the action satisfies the hard constraints, and maximizes the soft constraints. In fact, designing a reward function that captures all these nuances is a complex and intellectually demanding data science task, as we shall describe in Section 3.2. We are unaware of a generic framework that is capable of handling multiple hard and soft constraints to generate sequence-aware outputs for multiple applications. Section 3.5 contains further details.

2. Solving TPP by adapting Reinforcement Learning: Our second contribution is to present a computational framework (Sub-section 3.3.3), that is inspired by Constrained Reinforcement Learning (C-RL) [4,66], specifically Weighted RL [65], but non-trivially adapts it to handle multiple hard and soft constraints. Essentially, we propose a weighted reward function to transform the CMDP to an unconstrained MDP that captures multiple hard constraints as well as maximizes the actual value by maximizing the soft constraints. We prove that our designed reward function satisfies all hard constraints. We adapt the popular model-free on-policy algorithm SARSA [127] for updating the Q values of the states, which is known to converge faster and with fewer errors [113].

3. Experimental Evaluation: Our third contribution is an extensive evaluation (Section 3.4) using real-world datasets in the education and traveling domains. We use two datasets to plan courses for four different sought after degree programs and one dataset to plan two trips. Our results convincingly

demonstrate that: (a) Our algorithm generates task plans that are comparable in quality to handcrafted ones, and are superior to fully automated sequence-aware recommendations (e.g., *OMEGA* [133]) and to next-step recommendation in *EDA* [97]; (b.1) based on user studies involving 25 data science computational track (DS-CT) major students, our course plans achieve highly comparable satisfaction scores w.r.t. handcrafted gold standards designed by domain experts; (b.2) based on user studies involving 50 frequent travelers hired on Amazon Mechanical Turk, we find that the produced trip plans are highly satisfactory to the users and comparable to the handcrafted ones; (c.1) the policy learned by RL-Planner for the M.S. DS-CT is transferable to a different degree program in M.S. Computer Science inside the same university and vice versa; (c.2) similarly, the policy learned from a trip to NYC is transferable to a trip to Paris, and vice versa; (d) our algorithm is robust to the different parameters, takes reasonable time for learning the policy, and can therefore make interactive recommendations. We finally conclude in Section 3.6.

3.2 Task Planning Problem

We present our data model and define the Task Planning Problem. We present the used notations in Table 3.1.

3.2.1 General framework and problem definition

Let $\mathcal{I}$ and $\mathcal{T}$ represent a set of items and topics/themes, respectively. Some of the items, denoted by $\mathcal{P} \subseteq \mathcal{I}$, are also designated as antecedents meaning they must be recommended before some other items.

3.2.1.1 Item. Formally, an item m is represented as a quadruple

$$m = \langle type^m, cr^m, pre^m, \mathcal{T}^m \rangle$$

Table 3.1 Table of Notations

Symbol	Description
$\mathcal{I}$, $\mathcal{T}$,$\mathcal{P}$	Set of items, Set of topics/themes, Antecedent
$type^m$	Type of item m
cr^m	Quantifiable number toward satisfying the requirement
pre^m	Set of items that need to be recommended before item m
$\mathcal{T}^m$	Boolean vector of topics/themes that are covered by item m
$\mathcal{P}_{hard}$, $\mathcal{P}_{soft}$	Hard & Soft constraints
$\mathcal{T}^{ideal}$	Ideal topic/themes coverage
IT	Interleaving template
$\#_{primary}$, $\#_{secondary}$	# of primary items, # of secondary items
$\#cr$	Minimum hours requirement
gap	The lower bound of distance between m and its antecedents
$R(.)$	Reward function
H,$R(H)$	Trajectory of state-action pairs, Reward returned by H
$\mathcal{G} = \langle \mathcal{I}, E \rangle$	Graph with a set $\mathcal{I}$ of items as nodes and E as edges between them
S, E	State & action space
T, N	Transition function, Number of episodes
α, γ, ϵ	Learning rate, Discount factor, Topic coverage threshold
$AvgSim$	Interleaving reward
δ, β	Reward functions weights
ω_1, ω_2	Weight of primary & secondary items

An item is one of the two types: a. *primary* or b. *secondary*. An item of primary type is required for a task. Contrarily, an item of secondary type is chosen from a number of optional items according to the user's interest. cr^m designates a quantifiable number toward satisfying the requirement of a task. m may have one or more prerequisite $pre^m \subseteq \mathcal{P}$, where each prerequisite m_j is an item that needs to be recommended before m (described below with examples). If m has multiple prerequisites they are often "AND"ed meaning all antecedents are to be recommended before m. When they are "OR"ed, any one of these items must be taken before m. Both primary and secondary types could serve as prerequisites for some other items. Finally, m covers a set of topics/themes represented by a Boolean vector $\mathcal{T}^m$ of length $|\mathcal{T}|$, and the i-th bit contains 1 if m covers that topic/theme and 0 otherwise. These are typically keywords that are extracted from a course syllabus (available in training programs), or a POI description (available in Wikipedia). Given two items m and m', their vectors may or not have topics in common.

For a given end user, task planning must be personalized. That is achieved by satisfying hard constraints and maximizing soft ones.

3.2.1.2 Hard Constraints. These are provided by a domain expert as the requirement for achieving a task, typically a minimum requirement of credits hours $\#cr$ (or items if each item offers the same number of credit hours), a split of primary vs. secondary items ($\#_{primary}/\#_{secondary}$), and a specific ordering between an item and its antecedent Formally speaking, a hard constraint is a quadruple, $\mathcal{P}_{hard} = \langle \#cr, \#_{primary}, \#_{secondary}, gap \rangle$.

3.2.1.3 Soft Constraints. There are two types of soft constraints designed in consultation between a domain expert (academic advisor or travel agent) and the end user (student or traveler). Notationally, soft constraints are represented as a pair:

$$\mathcal{P}_{soft} = \langle \mathcal{T}^{ideal}, IT \rangle$$

Ideal Topics/Themes $\mathcal{T}^{ideal}$. A user needs to be recommended items that are personalized and covers topics/themes that are commensurate to her goal.

Appropriate Item Interleaving Template IT. The second soft constraint of planning is an intricate interleaving of primary and secondary items (e.g., start with a primary then follow with a secondary or two, then two more primary items, and so on) and is provided as a template IT. Formally speaking, $IT = \{I_1, I_2, ..., I_{|IT|}\}$ contains a set of permutations, where each I_j is a permutation of $\#_{primary}$ primary items and $\#_{secondary}$ secondary items.

Task Planning Problem (TPP). *Given the hard constraints $\mathcal{P}_{hard}$, and the soft constraints $\mathcal{P}_{soft}$ provided by an end user in consultation with a domain expert, the Task Planning Problem (***TPP***) is formulated as finding a personalized plan for that user. i.e., a sequence of items that satisfies $\mathcal{P}_{hard}$ and maximizes $\mathcal{P}_{soft}$.*

3.2.2 Instantiation of the framework

3.2.2.1 Course Planning. Item. For course planning, $type^m$ represents if a course is a core vs. elective. cr^m represents credit hours of a course. A prerequisite of a course is one or more courses that is/are its prerequisite. Imagine our course planning example consisting of six courses, as presented in Table 3.2. The set $\mathcal{T}$ covers 13 topics/themes: [*Algorithms, Classification, Clustering, Statistics, Regression, Data Structure, Neural Network, Probability, Data Visualization, Linear System, Matrix Decomposition, Data Management, Data Transfer*]. The last column shows the topic vectors of each course: e.g., for the Data Mining course, $\mathcal{T}^{m_2} = [0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$ covers two topics, *Classification* and *Clustering* out of the 13 topics. As an example, a data mining and a machine learning course contain overlapping topics such as clustering, classification, etc.

Hard Constraints. As an example, an M.S. DS-CT program may require a student to take at least 30 credit hours, five primary and five secondary courses. The

integer constant *gap* specifies the *distance* between m and pre^m in the recommended sequence: if a student takes three courses in each semester, $gap = 3$ enforces that the prerequisites of m must be taken at least a semester before. Thus, $\mathcal{P}_{hard} = \langle 30, 5, 5, 3 \rangle$.

Ideal Topics/Themes $\mathcal{T}^{ideal}$. For instance, in Example 3.1, *Classification, Clustering, Neural Network, Linear System* are the topics that the student wishes to learn. Therefore, for Example 3.1, $\mathcal{T}^{ideal} = [0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0]$

Appropriate Item Interleaving Template IT. A domain expert provides an IT containing three permutations of primary (core courses) and secondary items (elective courses) listed in Table 3.2:

$$\begin{aligned} IT = \{&[primary, primary, secondary, primary, secondary, secondary], \\ &[primary, secondary, secondary, secondary, primary, primary], \\ &[primary, secondary, secondary, primary, primary, secondary]\} \end{aligned}$$

Using Example 3.1, $m_1 \rightarrow m_2 \rightarrow m_4 \rightarrow m_5 \rightarrow m_6 \rightarrow m_3$ is a sequence that fully satisfies the permutation I_2 of the aforementioned IT. A recommended item sequence needs to adhere to these permutations as closely as possible.

3.2.2.2 Trip Planning. Item. A POI of primary type must be present in the planned trip (such as Eiffel Tower). cr^m designates the visitation time of POI m. An antecedent of a POI is another POI, that precedes the former POI temporally (e.g., Romanesque architecture must be visited before Gothic ones, or visit a museum before a restaurant/cafe). Let us assume that the toy trip planning dataset contains POIs such as Eiffel Tower, Louvre Museum, Pantheon, Rue des Martyrs, Musée d'Orsay, Cathédrale Notre-Dame de Paris, Palais Garnier, The River Seine, Le Cinq, etc. The set $\mathcal{T}$ covers eight topics/themes: Museum, Art Gallery, Cathedral, Palace, River, Street, Restaurant, and Architecture. The topic vector for Louvre Museum =

$[1, 1, 0, 0, 0, 0, 0, 1]$ covers three topics, *Museum*, *Art Gallery*, and *Architecture*, out of eight topics.

Hard Constraints. For the trip planning, a day-long trip in Paris to visit two primary and three secondary POIs, with a total visit time of six hours ($\#cr = 6$) and $gap = 1$ enforces that the antecedent of m must be visited before m. Thus, $\mathcal{P}_{hard} = \langle 6, 2, 3, 1 \rangle$.

Ideal Topics/Themes $\mathcal{T}^{ideal}$. Example 3.2, *Museum, Art Gallery, River, Restaurant, Architecture* are the topics/themes. These topics are captured in the ideal topic vector $\mathcal{T}^{ideal}$ and are considered as a soft constraint in the model.

Appropriate Item Interleaving Template IT.

$$\begin{aligned} IT = \{&[primary, secondary, primary, secondary, secondary], \\ &[primary, secondary, secondary, secondary, primary], \\ &[primary, secondary, secondary, primary, secondary]\} \end{aligned}$$

Using Example 3.2, Louvre Museum → Le Cinq → Eiffel Tower → Rue des Martyrs → River Seine is a sequence that fully satisfies the permutation I_1 of the aforementioned IT.

Table 3.2 Course Information

$CourseId$	$CourseName$	$type^m$	cr^m	pre^m	T^m
m_1	Data Structures and Algorithms	primary	3	[]	[1,0,0,0,0,1,0,0,0,0,0,0,0]
m_2	Data Mining	secondary	3	[]	[0,1,1,0,0,0,0,0,0,0,0,0,0]
m_3	Data Analytics	primary	3	[]	[0,0,0,1,0,0,0,1,0,0,0,0,0]
m_4	Linear Algebra	secondary	3	[]	[0,0,0,0,0,0,0,0,0,1,1,0,0]
m_5	Big Data	secondary	3	[Data Mining **OR** Data Analytics]	[1,0,0,0,0,0,0,0,0,0,0,1,1]
m_6	Machine Learning	primary	3	[Linear Algebra **AND** Data Mining]	[0,1,1,0,1,0,1,0,0,0,0,0,0]

3.3 Proposed Solution : RL-Planner

In this section, we present our proposed computational framework RL-Planner. We describe our proposed modeling first, following which we provide our solution.

3.3.1 Modeling the TPP

In general, a constrained MDP (CMDP) [9] is designed to learn a policy of an agent with the goal to

$$max_{\pi} E^{\pi}[R(H)] \text{ s.t. } E^{\pi}[D(H)] \leq c \tag{3.1}$$

where H is a trajectory of state-action pairs, $R(H)$ is the total return that can be obtained by H, and $D(H)$ is the measure of how dangerous the trajectory is.

For **TPP**, our abstraction contains a complete graph $\mathcal{G} = \langle \mathcal{I}, E \rangle$, where the nodes are items in $\mathcal{I}$ and each edge $e_{ij} \in E$ represents an interaction between two items m_i and m_j. **TPP** is a deterministic discrete CMDP [4, 65] (S, E, R):

a. The set S of states is the set of nodes $\mathcal{I}$ in $\mathcal{G}$.

b. E is a set of actions, where each action $e \in E$ is akin to adding one item to a given state. An action induces a transition between two nodes in $\mathcal{G}$ and is represented by an edge. The description T of an action is deterministic, that is, $T : S \times E \rightarrow S$, a new state is obtained by applying an action on each state.

c. $R(s_i, e_i, s_{i+1})$ is the reward of transitioning from state s_i to state s_{i+1} by taking action e_i. The reward needs to be designed to maximize $\mathcal{P}_{soft}$ and $\mathcal{P}_{hard}$ must be satisfied.

Course Planning: Trajectory H is computed considering $\#cr$ in $\mathcal{P}_{hard}$ and the cr^m of each course. As an example, if each course contributes a fixed credit of 3, a requirement of 30 credits translates to taking 10 items, thus $H = 10$. Each state s which corresponds to an item $m \in \mathcal{I}$ has a theme/topic vector $\mathcal{T}^m$ which represents the topics that course m covers.

Trip Planning: H is computed considering $\#cr$ in $\mathcal{P}_{hard}$ and the cr^m of each POI. If the user inputs six hours, the itinerary will be terminated if the total visitation time exceeds six hours. Thus $H = 6$.

3.3.2 Reward design

The process of designing the reward $R(s_i, e_i, s_{i+1})$ of taking action e_i on state s_i, is intellectually demanding and must abide by the following: (a) an action must satisfy the constraints in $\mathcal{P}_{hard}$; (b) capture how well an action increases the coverage of the ideal topic/theme vector for the soft constraints; (c) quantify "how close" it is to the pre-defined IT template; (d) weigh in primary and secondary items differently (ideally primary items should get higher weights); (e) combine these aforementioned requirements using a weighted linear function and adjust the weights empirically.

Formally speaking, adding an item m to a state s_i equates to taking action e_i that causes a transition to s_{i+1},

$$R(s_i, e_i, s_{i+1}) = \theta \times [\delta \times AvgSim(s_{i+1}, IT_{i+1}) + \beta \times weight_{type^m}] \tag{3.2}$$

$$\delta + \beta = 1, \theta = \{0, 1\}$$

$$\text{if } type^m = primary, weight_{type^m} = w_1$$

$$\text{if } type^m = secondary, weight_{type^m} = w_2, s.t. w_1 + w_2 = 1$$

We also look at the scenario where our reward function (Equation (3.2)) uses minimum similarity rather than average similarity.

We now describe the different components of this overall reward.

3.3.2.1 Reward on Topic/Theme Coverage. During an episode, we maintain a current topic/theme vector $\mathcal{T}^{current}$ that is initialized to all 0. As a state s (corresponding to an item m) is included in the episode, $\mathcal{T}^{current}$ gets updated to $\mathcal{T}^{current} = \{\mathcal{T}^{current} \bigcup \mathcal{T}^{m}\}$. Given a state s_i and an action e_i and the next state s_{i+1} (which corresponds to adding item m), the action is valid only if it increases the topic coverage of the ideal topic vector by at least a threshold ϵ, specified by the domain expert. Therefore, an action has a positive reward = 1, only when $|T^{ideal} \bigcap \{\mathcal{T}_{i+1}^{current} \setminus \mathcal{T}_{i}^{current}\}| \geq \epsilon$, 0 otherwise.

Formally speaking,

$$r_1 = \begin{cases} 1, & |T^{ideal} \bigcap \{T_{i+1}^{current} \setminus T_{i}^{current}\}| \geq \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

Since topic coverage is an important soft constraint that allows personalization, incorporating topic coverage in this fashion allows to eliminate items that are poor in topic coverage w.r.t. $\mathcal{T}^{ideal}$, even though they are good otherwise.

Course Planning: Given $\epsilon = 1$, considering Example 3.1 and $\mathcal{T}^{ideal} = [0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0]$, $s_2(m_2) \rightarrow s_4(m_4)$ has reward $r_1 = 1$, but $s_2(m_2) \rightarrow s_5(m_5)$ has $r_1 = 0$, since adding Big Data (m_5) does not increase the topic coverage of $\mathcal{T}^{current}$ w.r.t. $\mathcal{T}^{ideal}$ by at least 1.

Trip Planning: Given $\epsilon = 1$, a new POI is considered valid if it increases the topic/theme coverage at least by 1.

3.3.2.2 Reward on Antecedent/Prerequisite Gap. In state s_i, if the prerequisite(s) of m, pre^m is (are) present in the episode and $Dist(pre^m, m) \geq gap$, then an action has $r_2 = 1$, 0 otherwise. This is needed to ensure that the gap between the antecedents/prerequisites of an item and m must satisfy gap mentioned in $\mathcal{P}_{hard}$. When antecedents/prerequisites are "AND"ed, every pre^m must be present and satisfy

the *gap* when they are "OR"ed, any one of them needs to appear before m satisfying the *gap*.

Formally speaking,

$$r_2 = \begin{cases} 1, & Dist(pre^m, m) \geq gap \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

Course Planning: $r_2 = 1$, if m_2 or m_3 is taken 1 semester (1 semester enforces a gap of 3 since typically three courses are taken per semester) before m_5, 0 otherwise.

Trip Planning: If *Louvre* is recommended before *LeCinq* (restaurant), then the action gets value 1 for r_2 and 0 otherwise.

3.3.2.3 Combining Topic Coverage and Prerequisite Reward. $\theta = 1$ if the conditions on topic coverage AND antecedents are satisfied, and 0, otherwise. Therefore,

$$\theta = r_1 \times r_2 \tag{3.5}$$

3.3.2.4 Reward on Interleaving. This portion of the reward function quantifies how suitable the current sequence is based on similarity, considering the different permutations that are present in ideal composition (IT), and aggregates them as follows: $AvgSim(s_{i+1}, IT_{i+1})$. Note that the ideal composition IT is specified in $\mathcal{P}_{soft}$ and each $I \in IT$ is a permutation of length $\#_{primary} + \#_{secondary}$.

Course Planning: To quantify how close the item sequence in s_{i+1} is w.r.t. an ideal sequence I, we must use a distance function that is sequence-aware. In our implementation, we come up with a similarity notion, inspired by *Levenshtein distance* [96]. In general, given two sequences of length k ($1 \leq k \leq |I|$), one is the first k bits of an ideal composition I, and the other is a state in a session of length k, their Levenshtein distance produces a binary vector c_I of length k, where the j-th bit of the

vector represents the similarity between these two sequences considering bit j. For our problem, as an example, consider in a given session at a given state s, the so far chosen item corresponds to $\{primary, secondary, primary, primary\}$. The similarity score of this sequence and IT specified in Example 3.1 are $\{[1,0,0,1],[1,1,0,0],[1,1,0,1]\}$ where 1 means they are the same and 0 otherwise. For each permutation I, we compute $Sim(s,I)^k$ by capturing different lengths of match c and multiply that by a weight ζ, which is the maximum length of the consecutive match ($\zeta \in [0,k]$) and normalized by dividing it by k.

$$Sim(s,I)^k = \frac{\zeta \times \sum_{\forall c} length(c)}{k} \tag{3.6}$$

Finally,

$$AvgSim(s,IT)^k = \frac{\sum\limits_{\forall I \in IT} Sim(s,I)}{|IT|} \tag{3.7}$$

Using the aforementioned example, $Sim(s,I)^4 = [0.5, 1, 1.5]$ and $AvgSim(s,IT)^4 = 1$.

Theorem 3.1. *The designed reward function (Equation (3.2)) satisfies $\mathcal{P}_{hard}$ of the* ***TPP*** *problem.*

Proof. (Sketch) We note that there are actually four hard constraints present in **TPP**. (1) the total number of credits, (2) a pre-specified number of primary items, (3) a pre-specified number of secondary items (4) a pre-specified gap between items.

Without loss of generality, we present the proof using the course planning application. The trip planning could also be demonstrated in a similar manner.

(1) Total number of credit constraint: The first requirement is easily met by enforcing trajectory size H in the reward design.

Recall Sub-section 3.3.1 which describes the trajectory value H, which is computed considering $\#cr$ in $\mathcal{P}_{hard}$ and the cr^m of each recommended course. By its design choice $\sum_{\forall m \in H} cr^m = \#cr$, satisfying the first constraints.

(4) Gap between a course and its prerequisites: This constraint is satisfied through the design of r_2 in the reward function.

Clearly, $r_2 = 0$ denotes that the constrained is unsatisfied and it only gets 1, when the gap between the course and its prerequisites are met.

(2,3) Split between elective and core courses: Imagine that the reward function does not satisfy the $\#_{primary}(\#_{core})$ and $\#_{secondary}(\#_{elective})$ constraints. In a recommended sequence S, we can have two possibilities:

- **Case I:** $\#_{core} < |S_{core}| \wedge \#_{elective} > |S_{elective}|$
- **Case II:** $\#_{core} > |S_{core}| \wedge \#_{elective} < |S_{elective}|$

Case I. Is consistent, as a core course could be construed as an elective, thereby satisfying the constraint, as long as $|S|$ is H.

Case II. We prove it by contradiction. According to our reward design, $weight_{core} = \omega$ and $weight_{elective} = 1 - \omega$ s.t. $\omega > 1 - \omega$. This is akin to solving weighted RL and the appropriate ω is learned through extensive training considering the total number of core and elective courses in the dataset. By the design choice, $|\mathcal{I}_{core}| < |\mathcal{I}_{elective}|$, that is, the number of core courses is smaller than the number of electives in the dataset.

The reward function thus simplifies to $R(s_i, e_i, s_{i+1}) = \beta \times weight_{type^m}$, as long as $\theta = 1$. Therefore,

$$R(s_i, e_i, s_{i+1}) = \beta \times \omega; \qquad \forall m_j \in \mathcal{I}_{core}$$

$$R(s_i, e_i, s_{i+1}) = \beta \times (1 - \omega); \qquad \forall m_j \in \mathcal{I}_{elective}$$

Since, $\beta \times \omega > \beta \times (1 - \omega)$, therefore denotes a higher preference of the core courses over the electives. This makes $\#_{core} < |S_{core}|$ a contradiction.

Thus, the designed reward function satisfies all hard constraints in $\mathcal{P}_{hard}$.

□

Problem 3.1. (Revisiting TPP): *Based on our proposed model, given the hard constraints and soft constraints $\mathcal{P}_{soft}$, **TPP** is reformulated to find a policy π^* that maximizes the expected cumulative reward over any initial state.*

$$\pi^* = argmax_{\pi} E[\sum_{i=1}^{H} R(s_i, e_i, s_{i+1})|\pi] \tag{3.8}$$

3.3.3 Reinforcement Learning based solution

There are many well-known methods for solving MDPs, including value iteration and policy iteration, which are iterative methods and could be solved using Dynamic Programming [20], or Monte Carlo method, or the more popular Temporal Difference based approach which is a combination of Monte Carlo and Dynamic Programming [18,117,127]. Policy iteration involves two steps: policy evaluation and policy improvement, and these two are repeated iteratively until the policy converges. Contrary to that, value iteration includes: finding the optimal value function, followed by one policy extraction. There is no repeat of the two because once the value function is optimal, then the policy out of it should also be optimal (i.e. converged). While these two methods appear seemingly close, it has been proved theoretically and empirically in [104] that policy iteration is computationally more efficient and requires a smaller number of iterations to converge. So, we adapt model-free RL [18,117,127] with inputs (S, E, R, H) as a policy iteration method which fits our proposed problem remarkably well in the absence of logs.

Using model-free RL, **TPP** can be expressed as the problem of finding a policy $\pi : S \rightarrow E$ that maximizes the discounted cumulative reward. The goal is to maximize $\sum_i \gamma^i R(s_i, e_i, s_{i+1})$, where γ is the discount factor $\in [0, 1]$.

Each learning episode of RL-Planner contains H *action-values* that get periodically updated as the agent learns from the environment based on the reward function (recall Equation (3.2)). Q-values are defined for states and actions. $Q(s, e)$ is an estimation of how good is it to take action e at state s, and is iteratively updated. Based on our inputs, the size of the state action-pair is $|\mathcal{I}| \times |\mathcal{I}|$, since the agent can go to any other item (except for the ones that are chosen already) given our complete graph $\mathcal{G}$.

We use the popular SARSA algorithm [127] for learning a policy by updating the Q values. A SARSA agent interacts with the environment and updates the policy based on actions that are taken, hence this is known as an on-policy learning algorithm. The Q value for a state-action is updated by an error and adjusted by the learning rate α. Q values represent the possible reward received in the next step for taking action e in state s, plus the discounted future reward received from the next state-action observation, and expressed using Equation (3.9).

$$Q(s_i, e_i) \leftarrow Q(s_i, e_i) + \alpha[r_{i+1} + \gamma Q(s_{i+1}, e_{i+1}) - Q(s_i, e_i)] \tag{3.9}$$

where the reward r_{i+1} is computed using Equation (3.2). During the learning phase, given different hard constraints $\mathcal{P}_{hard}$ and soft constraints $\mathcal{P}_{soft}$, the agent learns Q values for a different number of episodes. For recommending item plans, it traverses the Q-table of size $|\mathcal{I}| \times |\mathcal{I}|$ with different starting states. It starts with a given initial state (corresponds to an item m), and traverses the Q table to find the next item that has the maximum Q value. This process is repeated until the sequence contains H items. The pseudo-code is presented in Algorithm 3.1.

Algorithm 3.1 Algorithm in RL-Planner

Learning Policy:

Require: $\mathcal{P}_{hard}$, $\mathcal{P}_{soft}$, $\mathcal{G} = (\mathcal{I}, E)$, number of episodes N, size of an episode H, α, γ

Ensure: a policy π satisfying $\mathcal{P}_{hard}$

for $i \leftarrow 1$ to N **do**

  $s_i \leftarrow m$

  $e_i \leftarrow argmax_{\forall md, d \in \{\mathcal{I}-m\}} R(m, md, d)$

  $r \leftarrow Equation(3.2)$

  $W \leftarrow \{m \bigcup d\}$

  $s_j \leftarrow d$

  for $j \leftarrow 2$ to H **do**

    $e_j \leftarrow argmax_{\forall s_j w, w \in \{\mathcal{I}-W\}} R(s_j, s_j w, w)$

    $W \leftarrow \{W \bigcup w\}$

    $Q(s_j, e_j) \leftarrow Q(s_j, e_j) + \alpha[r_{j+1} + \gamma Q(s_{j+1}, e_{j+1}) - Q(s_j, e_j)]$

    $s_j \leftarrow s_{j+1}$

    $e_j \leftarrow e_{j+1}$

Return Q

Recommending a plan:

Require: policy π, a starting item m

Ensure: A sequence rec of H items starting with m

$s_i \leftarrow m$

$W \leftarrow \{m\}$

$rec \leftarrow m$

for $i \leftarrow 2$ to H **do**

  $e_i \leftarrow argmax_{\forall j \in Q(s_i, j)} Q(s_i, j)$

  $rec \leftarrow [rec \rightarrow s_{i+1}]$

  $W \leftarrow \{W \bigcup s_{i+1}\}$

  $s_i \leftarrow s_{i+1}$

Sequence of items in rec

Table 3.3 RL-Planner Default Parameters Values

Parameters		N	α	γ	Threshold (ϵ)	Distance Threshold (d)	Time Threshold (t)	Starting Point (s_1)	Reward Function's Weights							
									ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	δ	β
Default Value	**Univ_1**	500	0.75	0.95	0.0025	—	—	STATS 263	0.6	0.4	—	—	—	—	0.6	0.4
	Univ_2	100	0.75	0.95	0.0025	—	—	CS 675	0.25	0.01	0.15	0.42	0.01	0.16	0.8	0.2
	NYC / Paris	500	0.95	0.75	—	5	6	—	—	—	—	—	—	—	0.6	0.4

3.4 Experimental Evaluation

We conduct various experiments to validate the effectiveness of RL-Planner and compare it with multiple baselines. All algorithms are implemented in Python 3.7 on a macOS Catalina with a 2.4 GHz Quad-Core Intel Core i5 Processor and 16 GB RAM. Our code and data are available on GitHub.[2]

3.4.1 Experimental setup

Our effort attempts to answer the following questions:

Q1. How well RL-Planner performs in comparison to baselines?

Q2. How do end users (students or travelers) compare recommendations by RL-Planner to gold standards?

Q3. How effective is RL-Planner for transfer learning?

Q4. How robust is RL-Planner w.r.t. different parameters?

Q5. How scalable is RL-Planner?

Measures. To answer Q1 and Q4, we present average scores over 10 runs. Q2 is answered through a user study and user satisfaction is measured on a scale of $1-5$. We present two representative case studies to answer Q4. The score of each recommendation is computed using Equation (3.7) for each ideal composition $I \in IT$ and the highest value is selected as the final score. Finally, we study running time to answer Q5.

[2]https://github.com/RL-Planner/RL-Planner-ICDE

3.4.1.1 Datasets. Course Planning: We consider dataset extracted from the NJIT (Univ-1) and Stanford (Univ-2) websites. The Univ-1 (NJIT) dataset contains 1216 courses comprising 126 degree programs through six professional schools and colleges. We focus on 3 M.S. degree programs: Data Science-Computational Track (DS-CT), Cybersecurity, and Computer Science (CS). The hard constraints consider the number of cores and elective courses while satisfying the gap between a course and its prerequisites. The Univ-2 (Stanford) dataset contains 3742 courses for four different departments related to data science. Each course has a title, department number, department code, course description, prerequisites, minimum and maximum number of required units. We focus on the M.S. Data Science (DS) program. The hard constraints are designed considering the number of unit constraints in the following six sub-disciplines while satisfying prerequisites gaps: a. Mathematical and Statistical Foundations; b. Experimentation; c. Scientific Computing (includes software development and large-scale computing); d. Applied Machine Learning and Data Science; e. Practical Component; f. Elective course in data science. The actual number of courses per program is 31, 30, 32, and 36 for DS-CT for NJIT, MS Cybersecurity NJIT, MS CS NJIT, and MS DS Stanford. To form topic vectors, we extract nouns from course names and removed stopwords. In Univ-1, we get 60, 61, and 100 distinct topics for the DS-CT, Cybersecurity, and for CS. We obtain 73 topics from Univ-2.

Trip Planning: Akin to a prior work of ours [19], we use publicly available Flickr data to plan trips in NYC and Paris, where the photos are tagged with corresponding POI names and the respective date/time associated with the photos define the set of possible itineraries (such as, a set of POIs visited on the same day). This dataset contains 2908 and 5494 itineraries, respectively. The number of POIs for NYC and Paris is 90, 114. The hard constraint is considered as the total time that one will allocate for visitation. We extract the themes/topics of the POIs from

Google Maps' Places API which comprises of 21 distinct themes for NYC, and 16 for the city of Paris. The gap is provided as not visiting two POIs of the same theme consecutively.

3.4.1.2 Implemented Baselines. We implement two types of baselines.

1. Fully Manual Gold Standard: Course Planning: This is a handcrafted sequence of courses designed by academic advisors for the relevant degree programs at Univ-1. For Univ-2, we obtain the gold standard from the website of the degree program. The gold standard scores (refer to Equation (3.7)) are 10 for Univ-1 and 15 for Univ-2 since the ideal course plans consist of 10 and 15 courses, respectively. **Trip Planning:** This is a handcrafted itinerary designed by a domain expert. The average of gold standard score is presented which is 5, since that is the highest popularity score of any POI in the dataset.

2. Fully Automated Solutions: We implement two types of automated solutions. One that performs sequence mining, and the other that adapts exploratory data analysis (*EDA*) based solutions. Both of these are model-free and hence can not be adapted to transfer learning.

- ***OMEGA*:** To the best of our knowledge, existing sequence recommendation algorithms leverage historical data. For the purpose of comparison, we chose a recent sequence recommendation algorithm *OMEGA* [133], that leverages the co-frequency of items (courses and POIs in our case). We non-trivially adapt it to account for topic/theme coverage and ideal compositions. *OMEGA* greedily selects edges in the graph and exploits graph-theoretic properties to determine an optimal sequence of items from a given set of edges. It first performs a topological ordering of items in the graph. At each subsequent iteration, an edge is chosen to maximize the specified utility function based on the sequence of items induced by the selected edge. *OMEGA* is NOT designed to satisfy constraints. Therefore, we adapt it into a two-step process that generates two sub-sequences and concatenates them. The first sub-sequence is generated greedily to satisfy the gap constraint. The second is recommended by *OMEGA* and is designed to optimize the soft constraint. The two sub-sequences concatenated together to satisfy the length constraint (number of primary and

secondary items). Originally, *OMEGA* uses a matrix that captures the number of times item i is consumed before item j, for each pair of items. In our implementation, we redesign it to capture the total number of topics covered by i and j.

- ***EDA*:** To the best of our knowledge, there does not exist any exploratory data analysis (*EDA*) based solution [97] that satisfies a multitude of complex constraints such as ours. We adopt the *EDA* paradigm by implementing a greedy method that chooses the action with the highest reward based on Equation (3.2) in each step. If two actions provide the same result, one will be picked at random.

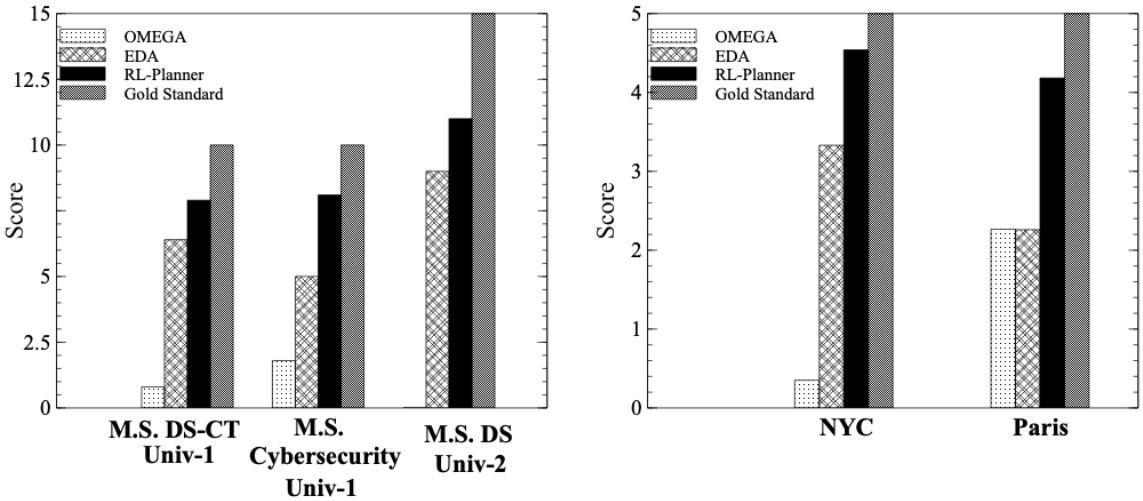
3.4.1.3 Default Parameter Settings. Table 3.3 contains default values for all parameters of our model. After consultation with students and academic advisors $|\mathcal{T}^{ideal}|$ is set to 60 for DS-CT, 61 for Cybersecurity, 100 for M.S. CS for Univ-1, and 73 for Univ-2 M.S. DS. For NYC and Paris, these are 21 and 16, respectively.

3.4.1.4 Summary of Results. Our results demonstrate that: (a) Existing fully automated approaches are not capable to adapt to sequence recommendations with a multitude of complex constraints. Both *OMEGA* [133] and *EDA* are unable to generate course plans and trip plans that satisfy the hard constraints most of the time, RL-Planner generates high-quality course plans and trip plans that are comparable to handcrafted gold standards; (b) Based on user studies involving 25 data science computational track (DS-CT) major students, RL-Planner is highly comparable w.r.t. handcrafted gold standards. RL-Planner gets a 3.39 user satisfaction score on average out of 5 compared to 3.74 for gold standards. The generated trip plans by RL-Planner are evaluated by 50 Amazon Mechanical Turk Workers and the gold standard itineraries are also rated. Itineraries that are generated using RL-Planner get an average score of 3.94 out of 5 compared to 4.15 for the gold standard; (c) RL-Planner is effective in transferring policy for both the applications, whereas, the fully automated baselines can not; (d) RL-Planner is robust to different parameters, takes reasonable time for learning the policy, and is capable to make interactive

recommendations in real-time. We conduct two sets of experiments using average similarity and minimum similarity in our reward function (Equation (3.2)), and RL-Planner outperforms all baselines in both situations. In certain cases, using minimal similarity yields a greater score than using average similarity, demonstrating that the RL-Planner works effectively regardless of the similarity metric used.

3.4.2 Comparison with baselines

We compare the plans generated by RL-Planner to the automated baselines *OMEGA*, *EDA* and to the fully manual gold standard described in Sub-section 3.4.1. Figure 3.1 presents the average scores. We observe that RL-Planner generates plans that are higher in score than the fully automated baselines for all cases while being very close to the gold standard. Contrarily, *OMEGA* fails to produce valid recommendations most of the time, leading to 0 scores. Despite our non-trivial adaptation, *OMEGA* fails to meet the stringent **TPP** requirements, and *EDA* generates a lower score compared to our proposed solutions.



(a) Time to learn policy

(b) Time to recommend course plan

Figure 3.1 RL-Planner, OMEGA, EDA, and Gold Standard comparison.

Table 3.4 Average Ratings: RL-Planner User Study

Questions:	**Course Planning**		**Trip Planning**	
	RL-Planner	**Gold Standard**	RL-Planner	**Gold Standard**
Overall Rating	3.6	4.12	4.2	4.5
Ordering of Items	3.1	3.4	3.7	4.12
Topic/Theme Coverage	3.6	3.76	3.8	3.9
Core and Elective Interleaving/Distance and Time Threshold	3.24	3.68	4.09	4.11

3.4.3 User studies

We measure user satisfaction with the sequence generated by RL-Planner against the gold standard. *OMEGA* or *EDA* are not considered in this study due to their low-quality recommendations.

The course planning study involves 25 student volunteers majoring in M.S. DS-CT at Univ-1, who are highly familiar with the courses. We validate 10 handcrafted itineraries (five for NYC and five for Paris) generated by domain experts and by RL-Planner by involving 50 unique AMT workers. Each itinerary is validated by five unique experienced travelers in NYC and Paris and their average score is presented. Each worker is paid 50 cents. The students/workers are shown two sequences of courses: by RL-Planner and the gold standard without revealing which one is which. Each volunteer is asked to provide a rating for four questions on a scale of $1-5$ with 5 being the best. Our results are summarized in Table 3.4. RL-Planner produces course/trip plans that are highly comparable to the gold standards across all four questions, demonstrating its effectiveness.

3.4.4 Case study: Transfer learning

We present the effectiveness of RL-Planner in transfer learning through a small number of case studies using the Univ-1 dataset and for NYC and Paris trip plans. Clearly, the automated baselines fail to adapt to transfer learning.

Course Planning: We learn a policy using M.S. DS-CT to recommend course plans for M.S. CS and vice versa. Table 3.5 presents the results (the full mapping to course titles is in Table 3.6). We present cases where both courses accurately meet all core and elective requirements and all other hard constraints. We also present *less effective cases*, when the learned policies produce course plans with one less core course during transfer learning.

Trip Planning: We learn a policy for NYC and apply it to Paris and vice versa. The results are shown in Table 3.7. A good sequence is one that meets the hard constraints, whereas one that does not meet these constraints is deemed to be a bad outcome. Table 3.8 shows a few results from RL-Planner, the type of POIs in each itinerary, as well as the time and distance thresholds that each one meets.

Table 3.5 RL-Planner for Course Planning: Transfer Learning between M.S. CS and M.S. DS-CT

Learnt Policy	Applied Policy	Sequence of Recommended Courses	
M.S. CS	M.S. DS-CT	**Good:**	CS 675 : core → CS 683 : elective → CS 652 : elective → CS 677 : core → CS 639 : elective → CS 645 : elective → CS 644 : core →MATH 661 : core → CS 610 : elective → CS 636 : core
		Bad:	CS 675 : core → CS 683 : elective → CS 645 : elective → CS 652 : elective → CS 636 : core → CS 644 : core → CS 639 : elective → CS 696 : elective → CS 677 : core → CS 634 : elective
M.S. DS-CT	M.S. CS	**Good:**	CS 610 : core → CS 608 : elective → CS 656 : core → CS 667 : core → CS 652 : elective → CS 634 : elective → CS 675 : elective → CS 631 : core → CS 630 : core → CS 700B : core
		Bad:	CS 610 : core → CS 608 : elective → CS 656 : core → CS 667 : core → CS 652 : elective → CS 704 : elective → CS 675: elective → CS 645 : elective → CS 636 : elective → CS 700B : core

3.4.5 Robustness of RL-Planner

We vary one parameter at a time while all other parameters are kept at the default values (see Table 3.3). The parameters are: Number of Episodes (N), Starting Point (s_1), Learning Rate (α), Discount Factor (γ), Topic/Theme Coverage Threshold (ϵ), Reward Function Weights (w_1, w_2, δ, β). The results for both average and minimum similarity are summarized in Tables 3.10, 3.9, 3.11 and Tables 3.12, 3.13, 3.14 for

Table 3.6 Course IDs and Descriptions for RL-Planner

	Course Number	Course Name
Data Science	CS 675	Machine Learning
	CS 683	Software Project Management
	CS 652	Computer Networks-Architectures, Protocols and Standards
	CS 677	Deep Learning
	CS 639	Elec. Medical Records: Med Terminologies and Comp. Imp.
	CS 645	Security and Privacy in Computer Systems
	CS 644	Introduction to Big Data
	MATH 661	Applied Statistics
	CS 610	Data Structures and Algorithms
	CS 636	Data Analytics with R Program
Computer Science	CS 610	Data Structures and Algorithms
	CS 608	Cryptography and Security
	CS 656	Internet and Higher-Layer Protocols
	CS 667	Design Techniques for Algorithms
	CS 652	Computer Networks-Architectures, Protocols and Standards
	CS 634	Data Mining
	CS 675	Machine Learning
	CS 631	Data Management System Design
	CS 630	Operating System Design
	CS 700B	Master's Project

Table 3.7 RL-Planner for Trip Planning: Transfer Learning between NYC and Paris

Learnt Policy	Applied Policy	Sequence of recommended POIs	Score
NYC	Paris	['musée du luxembourg' → 'musée des Égouts de paris' → 'Église st-sulpice']	4.3
Paris	NYC	['museum of television and radio' → 'new york university']	4.5

Table 3.8 RL-Planner for Trip Planning: Itinerary Description

Itinerary:		**Constraints:**		
		Time Threshold (t)	**Distance Threshold (d)**	**POIs' Type**
NYC	['battery park', 'brooklyn bridge', 'colonnade row']	≤ 6	≤ 4	['park', 'establishment', 'museum']
	['brooklyn bridge', 'colonnade row', 'flatiron building', 'hudson river park', 'rockefeller center']	≤ 8	≤ 5	['establishment', 'museum', 'establishment', 'park', 'establishment']
Paris	['pont neuf', 'promenade plantee', 'sainte chapelle', 'tour montparnasse', 'Eglise st-eustache']	≤ 6	≤ 5	['establishment', 'park', 'church', 'establishment', 'church']
	['pont neuf', 'promenade plantee', 'viaduc des arts', 'Église st-germain des prés']	≤ 5	≤ 5	['establishment', 'park', 'establishment', 'church']

Table 3.9 RL-Planner vs. *EDA*: Parameter Tuning Results Univ-1 M.S. DS-CT

Parameter	Topic Coverage Threshold (ϵ)					w_1 , w_2									
Value	0.0025	0.005	0.01	0.0175	0.02	0.4	0.6	0.8	0.2	0.5	0.5	0.6	0.4	0.65	0.35
RL-Planner score using Avg similarity	**7.9**	5.6	5.6	5.7	5.4	0		0		5.9		**7.9**		0	
RL-Planner score using Min similarity	**8.24**	6.48	7.6	6	7.48	0		5.44		4.8		**8.24**		7.36	
***EDA* Score**	**6.4**	3.2	**6.4**	3.2	0	—		—		—		—		—	

Univ-1 and Univ-2, respectively. Tables 3.15, 3.16 represent the results for NYC and Paris. Please note that *OMEGA* does not have those parameters - hence these experiments are not applicable to *OMEGA*. Contrarily, *EDA* is a model-free solution, hence some of the aforementioned parameters (such as N, α, γ, and s_1) can not be tuned for *EDA*. Those results are marked as "——" in the tables.

Table 3.10 RL-Planner Parameter Tuning Results Univ-1 M.S. DS-CT

Parameter	Number of Episodes (N)					Learning Rate (α)					Discount factor (γ)				
Value	100	200	300	500	1000	0.5	0.6	0.75	0.8	0.95	0.5	0.6	0.9	0.95	0.99
RL-Planner score using Avg similarity	3.2	5.6	5.6	**7.9**	4.5	3.2	4.2	**7.9**	4.8	5.8	4.8	5.6	5.6	**7.9**	5.6
RL-Planner score using Min similarity	5.68	5.72	7.92	**8.24**	6.08	5.6	6.28	**8.24**	5.36	4.08	4.16	7.48	8.08	**8.24**	7.2

Table 3.11 RL-Planner vs. *EDA*: Parameter Tuning Results Univ-1 M.S. DS-CT

Parameter	Starting Point (s_1)				δ , β									
Value	CS 644	CS 636	CS 675	MATH 661	0.4	0.6	0.45	0.55	0.5	0.5	0.55	0.45	0.6	0.4
RL-Planner score using Avg similarity	7.2	7.1	**7.9**	7.2	5.6		4.8		3.2		6.2		**7.9**	
RL-Planner score using Min similarity	**8.24**	2	6.36	6	5.6		6.72		6.16		**8.24**		**8.24**	
***EDA* Score**	——	——	——	——	4.8		4		3.2		**6.4**		**6.4**	

Table 3.12 RL-Planner vs. *EDA*: Parameter Tuning Results Univ-2 M.S. DS

Parameter	Number of Episodes (N)					Learning Rate (α)					Discount Factor (γ)					Topic Coverage Threshold (ϵ)				
Value	100	200	300	500	1000	0.5	0.6	0.75	0.8	0.9	0.7	0.75	0.8	0.9	0.95	0.0025	0.005	0.01	0.015	0.02
RL-Planner score using Avg similarity	10	10	10	**11**	10	**11**	**11**	10	9	**11**	10	**11**	**11**	10	10	**11**	10	**11**	9	10
RL-Planner score using Min similarity	**12**	11	10.4	11.2	6.2	11.2	11.4	**12**	10.8	11	10.8	11.8	11	11.2	**12**	**12**	10.6	10.6	11.6	11.4
***EDA* Score**	——	——	——	——	——	——	——	——	——	——	——	——	——	——	——	9	9	10	**11**	**11**

If the hard constraints are not satisfied, those are marked with values 0 in Tables 3.9 and 3.14.

Course Planning: For Univ-1 (Table 3.11), we notice that starting with any of the acceptable starting core courses, has minimal impact on the performance of the

Table 3.13 RL-Planner Parameter Tuning Results Univ-2 M.S. DS

Parameter	w_1, w_2, w_3, w_4, w_5, w_6																	
Value	0.2	0.01	0.16	0.4	0.01	0.22	0.21	0.01	0.15	0.41	0.02	0.2	0.25	0.01	0.15	0.4	0.01	0.18
RL-Planner score using Avg similarity	**13**						12						11					
RL-Planner score using Min similarity	11.6						11.6						**12.2**					

Table 3.14 RL-Planner vs. *EDA*: Parameter Tuning Results Univ-2 M.S. DS

Parameter	Starting Point (s_1)		δ , β											
Value	STATS 263	MS&E 237	0.2	0.8	0.3	0.7	0.4	0.6	0.6	0.4	0.7	0.3	0.8	0.2
RL-Planner score using Avg similarity	**11**	10	10		10		10		10		**11**		**11**	
RL-Planner score using Min similarity	**12**	10.4	0		0		0		0		11		**12**	
***EDA* Score**	—	—	10		9		10		9		10		**12**	

Table 3.15 RL-Planner vs. *EDA*: Parameter Tuning Results Trip Planning

Parameter	Number of Episodes (N)					Learning Rate (α)					Discount Factor (γ)					Distance Threshold (d)	
Value	100	200	300	500	1000	0.5	0.6	0.75	0.8	0.95	0.5	0.6	0.75	0.8	0.95	4	5
NYC RL-Planner score using Avg similarity	4.6	4.53	4.6	**4.63**	4.6	**4.63**	**4.63**	4.6	4.6	**4.63**	**4.63**	**4.63**	**4.63**	4.5	4.47	**4.8**	4.63
NYC RL-Planner score using Min similarity	3.76	4.56	**4.6**	**4.6**	4.55	4.33	**4.6**	**4.6**	**4.6**	**4.6**	**4.6**	**4.6**	**4.6**	**4.6**	4.59	4.59	**4.6**
NYC *EDA* Score	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	**3.65**	3.33
Paris RL-Planner score using Avg similarity	4.5	**4.58**	**4.58**	4.56	4.44	**4.63**	4.6	4.6	**4.63**	**4.63**	**4.63**	**4.63**	**4.63**	4.5	4.47	4.47	**4.56**
Paris RL-Planner score using Min similarity	**4.6**	**4.6**	4.58	4.53	4.52	**4.58**	**4.58**	4.56	4.52	4.53	4.54	**4.58**	4.53	**4.58**	4.52	**4.58**	4.53
Paris *EDA* Score	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	**3.33**	2.26

Table 3.16 RL-Planner vs. *EDA*: Parameter Tuning Results Trip Planning

Parameter	Time Threshold (t)			δ , β									
Value	5	6	8	0.4	0.6	0.45	0.55	0.5	0.5	0.55	0.45	0.6	0.4
NYC RL-Planner score using Avg similarity	4.625	4.63	**4.74**	4.6		4.6		4.6		4.45		**4.63**	
NYC RL-Planner score using Min similarity	2.33	**4.6**	**4.6**	**4.62**		4.6		4.61		4.59		**4.62**	
NYC *EDA* Score	3.55	3.33	**3.57**	3.52		3.42		3.12		**4.4**		3.33	
Paris RL-Planner score using Avg similarity	4.4	**4.56**	4.42	**4.58**		**4.58**		4.5		4.52		4.53	
Paris RL-Planner score using Min similarity	4.52	4.53	**4.58**	**4.56**		**4.56**		**4.58**		4.56		4.56	
Paris *EDA* Score	2.9	2.27	**4.1**	1.8		**3.14**		3		2.3		2.3	

model. We also note that our reward parameters are best at $\delta = 0.6, \beta = 0.4, w_1 = 0.6, w_2 = 0.4$. We also observe that reducing the threshold for accepting an action improves the score and that setting the discount factor (γ) at 0.95 and learning rate (α) at 0.75 produce the best set of results. Overall, these results demonstrate that RL-Planner is robust.

Similar results are observed for Univ-2 (Tables 3.12, 3.13, 3.14). We observe RL-Planner performs well for all parameter values and is stable with respect to the starting point, as there is not much variation in the score with a changing start point. These results corroborate the robustness of RL-Planner with different parameters.

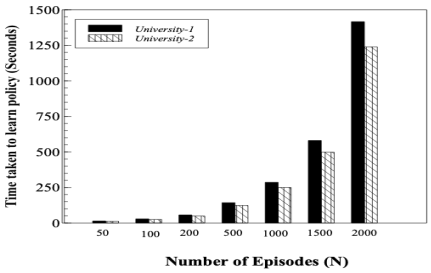
Trip Planning: For NYC and Paris, changing the learning rate (α) and the discount factor (γ) (Table 3.15) does not have a high impact on the final score and the results are stable with respect to reward's weights (δ, β).

3.4.6 Scalability evaluation

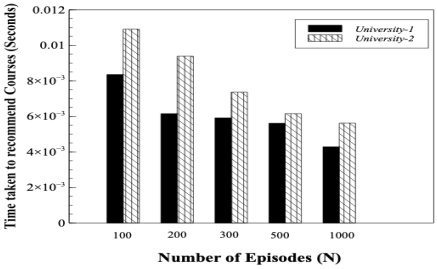
We explore the time taken to learn a policy and recommend a course plan based on the learned policy. We vary the number of episodes. All other parameters are held at the default values. In Figure 3.2 (a)(c), we plot the time taken to learn a policy against the number of episodes. We observe that the time taken to learn a policy increases linearly with the number of episodes. In Figure 3.2 (b)(d), we plot the time taken to apply a learned policy w.r.t. the number of episodes we train against. The time taken to recommend course plans is only a few seconds which means it can be used in interactive mode.

3.5 Related Work

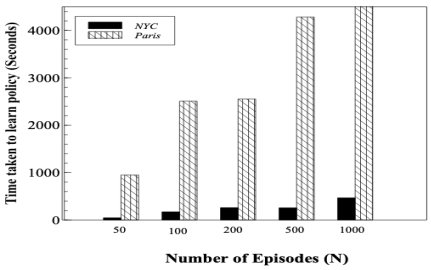
We review four types of work: (1) Sequence Recommendation, (2) Course Recommendation, (3) Trip Recommendation, and (4) Reinforcement Learning and Guided EDA.



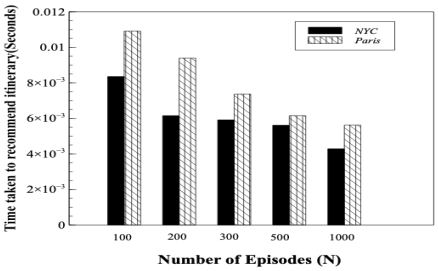
(a) Time to learn policy



(b) Time to recommend course plan



(c) Time to learn policy



(d) Time to recommend trip plan

Figure 3.2 RL-Planner scalability results.

3.5.1 Sequence recommendation

Sequence recommendation is explored in [133], [22] , [111] and [125]. In [133], a Directed Acyclic Graph is used to model sequential dependencies, and a novel class of utility functions is provided to extend the expressive power of Sub Modular functions. The OMEGA algorithm is proposed and provides a constant factor approximation guarantee when applied to a DAG, to produce an ordering which maximizes a given utility function. This algorithm, however, has a high run time and is improved upon by an edge-based algorithm [22]. However, both [133] and [22] do not account for constraints or prerequisites when producing a sequence of items, which is crucial for **TPP**.

In [81], Jiang et al., 2016 develop a framework that automatically mines user and route data, to build an optimal route that provides a personalized sequence of POIs for users visiting a city. Caser is represented in [130] that leverages Convolutional Neural Network for capturing both general preferences and sequential patterns. SASRec [85] balances Markov Chains and Recurrent Neural Network approaches to make sequence recommendations.

In [111], Quadrana et al., 2018 propose techniques to mine logs for capturing short-term user interests combined with long-term sequential patterns for making sequence recommendations. They also acknowledge that applications such as course and trip recommendations require consideration of constraints. In [125], the notion of satisfaction and disagreement is used to present the problem of sequential group recommendations. They mainly concentrated on aggregating the recommendation lists of individual group members into a group list. SDAA, SIAA, and Average+ are three new methods for aggregating group members' recommendation lists that were studied by Stratigi et al., 2021.

As shown experimentally, the above algorithms do not adapt well to constrained sequences, as is needed to solve **TPP**.

3.5.2 Course planning

The use of recommendation systems for course predictions for students has been extensively studied in [21, 23, 41, 44, 54, 57, 58, 82, 103, 129]. CourseRank [23], is a popular course planning platform developed at Stanford University. The platform provides useful course feedback on specific courses as well as course recommendations based on existing courses or preferences. Bercovitz et al., 2009 demonstrate the need for an accurate course planning tool to assist students in fulfilling their needs whilst navigating complex degree requirements. Our work achieves a similar goal using reinforcement learning techniques as well as provides a personalized learning experience that helps a student acquire expertise on a topic whilst also completing their degree. In [103], Parameswara et al., 2011 acknowledge the importance of prerequisites and ordering when recommending courses to a college student. However, adding complex prerequisites (such as AND/OR) in their context requires the use of an Integer Linear Programming algorithm which is found to be slow when recommending courses using the Extended Model described in the paper. In [41] a hybrid algorithm, using Collaborative Filtering and Sequential Pattern Mining algorithms, is used to propose a list of learning items in an e-learning setting as per a given user's interests. Grade prediction and top-n course recommendation problems are studied in [57] using collaborative filtering and popularity ranking, and a follow-up work proposes a hybrid of the Random Forest and Matrix Factorization in [129] for this. Ebner et al., 2011 [54] develop a personalized learning environment (PLE) using widgets for a university, which point to web resources to help students find relevant resources.

The approaches above heavily rely on existing logs or social networks to make recommendations, which is not the case in our work.

3.5.3 Trip planning

How to recommend trips that are personalized have been studied in many recent related works [10,39,43,62,68], including our own [19]. Unlike our proposed solutions, most of these do not account for a multitude of constraints, except for [68]. Unlike these works, we study trip planning as a guided planning problem with the goal of making an agent learn from the environment on the go and take actions based on that.

From the perspective of solution design, existing approach designs combinatorial solutions, heavily rely on logs or training data to model user preference, which is not the case in our work.

3.5.4 RL and guided EDA

RL has a broad range of applications in problems with dynamic environments where an agent needs to learn behavior through trial and error and has been studied in [63, 84,105,138]. Our guided task planning bears similarity to guiding users in performing Exploratory Data Analysis (EDA), a well-studied problem. Numerous works proposed next-step recommendations [97], by using logs of previous operations (e.g., [55]), or by relying on real-time feedback [53, 77]. Fully automated generation of EDA sessions has been examined in [18,117]. The use of Reinforcement Learning (RL) for a recommendation of user groups or sequences of EDA techniques has been documented in [117] and [18] respectively. In [117], RL is used to allow an agent to learn a policy that helps find a set of target users (such as forming a conference PC). In [18], deep RL is used to produce an ideal set of EDA strategies to explore a dataset. The objective is to provide meaningful EDA notebooks for use in data analysis. In [99], Narvekar et al., 2017 propose how to design a curriculum through transfer learning. The problem is formulated as an MDP for training the agents through a series of tasks and solved using RL.

Existing work on guided EDA (either next-step recommendation or end-to-end guidance) does not handle the intricacies of the constraints we consider in our framework. We will nevertheless adopt a next-step EDA to create a baseline for our experiments.

The closest to our work is the area of safe reinforcement learning [63], which is defined as the process of learning policies that maximize the expectation of the return in problems in which it is important to ensure reasonable system performance and/or respect safety constraints. In particular, our work borrows inspiration from constrained MDPs [65, 66], for which linear programming, weighted sum, state space extension, and recursive formulation of the constraints-based approaches are known. In [66], the reward function is designed by weighting the original value function and the risk associated with constraints. The weight parameter is adapted in order to find a feasible solution for the constrained problem that has a good performance with respect to the value function.

Our proposed solution bears similarity with the weighted approach, although traditional weighted RLs do not have to deal with multiple soft and hard constraints, such as **TPP**.

3.6 Conclusion

We formalize task planning as a constrained sequence generation problem. We present a computational framework RL-Planner for task planning, especially for scenarios where little to no data is available as logs that specify students' past preferences. RL-Planner requires minimal input from domain experts, yet produces personalized plans. We adapt reinforcement learning to learn a policy that satisfies item interleaving, requirements, and item features. We compare our solutions with item plans drafted by human experts and with fully automated ones. Our experiments corroborate that our proposed model and solution recommend high-quality plans. We also experimentally

demonstrate that RL-Planner is effective in transfer learning using several real-world data in the education and trip planning domains.

In the future, we would like to investigate an adaptive approach to task planning that leverages feedback. Feedback could come as binary values (useful item/ not useful), categorical ratings (e.g., on a scale of $1-5$), or as a probability distribution. This will allow us to create a loop that accounts for effectiveness and incorporate that in future design choices. We are also interested to investigate how to design a small number of agents that are cooperatively equipped to plan a large number of task plans.

CHAPTER 4

DIVERSIFYING RECOMMENDATIONS ON SEQUENCES OF SETS

4.1 Introduction

Diversity aims to improve user experience by addressing the problem of over-specialization, where a user receives recommendations that are often too similar to each other. To create online music playlists, users organize songs into channels and listen to a few songs within the same channel before switching to the next channels to listen to other artists in the same genre or to experience different music styles. On crowdsourcing platforms, workers complete a small set of tasks at a time (session) and sequences of sessions within a finite time (for example, half a day). Diversifying recommendations inside (Intra) and across (Inter) sessions is natural for such applications to improve user satisfaction and engagement.

Recommending playlists during a long drive may need to minimize both Intra and Inter session diversities to generate songs by the same artist within a channel and similar beats across channels. Contrarily, designing playlists for a theme party is best done by composing songs from the same period within a channel (90's, 60's, etc) and different styles across channels (thereby minimizing Intra diversity on release date within a session and maximizing Inter diversity on style across sessions). Similarly, in crowdsourcing, it may be ideal to recommend tasks requiring similar skills within a session and different completion times across sessions. Whereas, workers who have multiple expertise may be recommended tasks with different skills in a session and different rewards across sessions. More generally, applications may require minimization or maximization of Intra and Inter diversities.

These aforementioned scenarios have three things in common: first, diversity needs to be accounted for in the design of a sequence of sets of recommendations.

Second, both minimization and maximization of diversity are meaningful. Finally, the dimensions on which Intra and Inter session diversities are expressed are item features that may not be related - hence they cannot be combined. We present a framework that satisfies all three requirements focusing purely on diversity and assuming that the items consumed by the framework are always suitable (relevant) to the user.

Our goal is to develop an algorithmic framework for Inter and Intra session diversities in tandem with the goal to recommend k sessions to a user, with a small number l of relevant items in each, yielding a total of $N = k \times l$ items [1]. Intra and Inter diversities can be either minimized or maximized which gives rise to a bi-objective formalism to express four problem variants (Sub-section 4.2.2). We also study the relaxed version of our proposed framework where the sessions are of varying lengths and the total number of items recommended to the user is a subset of N items. To the best of our knowledge, our work is the first attempt to combine set and sequence diversities, two problems extensively studied individually in search and recommendation [2, 12, 42, 60, 74, 75, 100, 109, 110, 112, 134, 140, 142, 143, 146].

Our second contribution is theoretical. We first study each of the Intra and Inter diversity optimization problems individually and find that irrespective of minimization or maximization, the Inter problem is NP-hard (Sub-section 4.2.3). We also prove that the Intra minimization problem can be optimally solved in polynomial time. However, the complexity of each bi-objective problem remains NP-hard (because Inter optimization is NP-hard).

Our third contribution is algorithmic (Section 4.3). We design principled solutions with provable guarantees for Intra and Inter problems individually. Algorithm Ex-Min-Intra runs in $O(NlogN)$ time and produces an exact solution to the Min-Intra problem. For Min-Inter and Max-Inter, algorithms Ap-Min-Inter and Ap-Max-Inter

[1] A preliminary version of this work has got accepted in The Web Conference, 2021 [59].

achieve $4-2/k$- and $\frac{1}{2}$-approximation factors, respectively. We also design an efficient $\frac{1}{2-1/k}$-approximation algorithm Ap-Max-Intra to solve the Max-Intra problem.

Additionally, we investigate an alternative formulation (Sub-section 4.2.4) of all four problems to a corresponding constrained optimization problem, with the goal of obtaining one point from the Pareto front. The idea is to optimize Inter diversity, subject to constraining Intra diversity. The constraint on Intra is obtained by solving the Intra optimization first. There exists more than one benefit to this approach. First, in one of the two cases (i.e., Minimization) Intra is tractable and easier to solve, therefore, finding the optimal constraint value is computationally efficient. More importantly, the constrained optimization problem aims at finding one point in the Pareto front, which is perfectly acceptable, as the points in the Pareto front are qualitatively indistinguishable (unless further information is available). When Inter problems are optimized subject to constraining Intra, the combined solutions hold guarantees for two out of the four problems (Sub-section 4.3.4). Tables 4.2 and 4.3 summarize our theoretical and algorithmic results.

Our last contribution is experimental. We consider two real-world applications and conduct multiple experiments involving 400 human subjects, as summarized in Table 4.4, for music and task recommendation. We additionally perform large-scale experiments using real and simulation data to validate the properties of our designed algorithms. In music recommendation (Sub-section 4.4.1), our results highlight, with statistical significance, that user satisfaction is higher when playlists are recommended considering diversity and the preferred diversity scenario depends on the underlying context. In task recommendation, our results show the benefit of diversification in task sessions across different session gaps or time intervals between sessions. Our algorithms achieve higher quality and worker satisfaction with statistical significance, than a baseline with No-diversity in all the specified session gaps.

Table 4.1 Task Skill and Reward

Task	Skill	Reward	Task	Skill	Reward	Task	Skill	Reward
$\mathbf{t_1}$	0.5	0.3	$\mathbf{t_2}$	0.51	0.4	$\mathbf{t_3}$	0.54	0.49
$\mathbf{t_4}$	0.59	0.50	$\mathbf{t_5}$	0.6	0.23	$\mathbf{t_6}$	0.63	0.4
$\mathbf{t_7}$	0.69	0.1	$\mathbf{t_8}$	0.7	0.60	$\mathbf{t_9}$	0.79	0.36
$\mathbf{t_{10}}$	0.8	0.12	$\mathbf{t_{11}}$	0.89	0.55	$\mathbf{t_{12}}$	0.93	0.34

Sub-section 4.4.2 investigates approximation factors and the scalability of our algorithms against several non-trivial baselines. We observe that in most cases, our algorithms produce approximation factors that are very close to 1. For the cases where the approximation factor is slightly worse, the solution is close enough. Finally, we also observe that our approach is faster and highly scalable when varying the number of items and the number of sessions considering different data distributions. We present related works in Section 4.5 and conclude in Section 4.6.

4.2 Formalism and Problem Analysis

For the purpose of illustration, we describe a simple running example of recommending task sessions in crowdsourcing. The same example could be used for streaming music.

Example 4.1. *Consider a set of $N = 12$ tasks, which are most relevant to a specific worker. Table 4.1 shows two dimensions of these tasks. The first dimension is the skill requirement of the task as provided by the requester. The second dimension is the task reward. We want to recommend four (=k) sessions, each containing three (=l) tasks.*

4.2.1 Data model

Item. An item has a set of dimensions. t_i^d represents the d-th dimension of the i-th item. Using Example 4.1, task t_1 is represented by two dimensions, $< 0.5, 0.30 >$. In the case of a song, examples of dimensions are artist, vibe, genre, etc.

Session. A session s consists of a set of l items that can be consumed in any order.

Sequence. A sequence of sessions is an ordering of k sessions $S =< s_1, s_2, \ldots, s_k >$ where sessions are presented to a user one after another.

Intra Diversity. Given a dimension d, the diversity of a set of items in a single session s is referred to as *Intra* and defined by capturing how each item in that session deviates from the average, considering d, and taking an aggregate over l items as follows:

$$Intra^d(s) = \sum_{i=1}^{l} (t_i^d - \mu_s^d)^2 \tag{4.1}$$

where t_i^d is the value of dimension d of item t_i and μ_s^d is the average of d values of items in session s. Intra essentially captures the variance of a set of items for a dimension d. Following Example 4.1, if the session s_1 consists of $\{t_1, t_3, t_5\}$, then $Intra^{skill}(s_1) = 0.005$.

Inter Diversity. The diversity of items between two consecutive sessions is referred to as Inter and is defined for two consecutive sessions for a dimension d as follows:

$$Inter^d(s_i, s_{i+1}) = (\mu_{s_i}^d - \mu_{s_{i+1}}^d)^2 \tag{4.2}$$

which captures the difference between the average of two consecutive sessions. Given $S =< \{t_1, t_3, t_5\}, \{t_2, t_4, t_6\}, \{t_7, t_8, t_9\} >$, $Inter^{Reward}(S) = (0.34 - 0.433)^2 + (0.433 - 0.35)^2 = 0.015$ using Example 4.1 .

Changing the aggregation function from square to the exact definition of variance (i.e., divide it by the number of items in the session), taking the square root of the current definition, or changing the solutions to standard deviation will not require any changes in the solution and approximation factor, because these definitions are technically equivalent. In fact, the approximation factors remain unaltered for many popular distance functions that are part of the Minkowski family, such as L_1, L_2, and L_∞. Other set-based [2] and sequence-based [146] definitions could be considered in future work.

For the simplicity of illustration, we use one dimension at a time to model diversity. For all practical purposes, both Intra and Inter dimensions could be designed to reflect multiple attributes by combining them and allowing overlap.

We explicitly chose to handle one attribute at a time because we believe that diversity becomes more difficult to perceive users when combining several attributes. That is further exacerbated by the fact that users have to perceive Intra and Inter diversities at once. The use of a single attribute for Intra and for Inter allowed us to focus on the algorithmic and theoretical contributions. There is however a workaround to reduce any number of dimensions to one for each type of diversity by combining their values with a weighted linear function as in MMR [37].

4.2.2 Problem definitions

We formalize our problems and propose to do that in two stages: we first focus on producing Fixed Length Sessions that consume all input items, and we then relax our problem to produce Variable Length Sessions that consume only a subset of input items. This allows us to study Fixed and Variable Length Sessions in conjunction with consuming all versus some input items. The problems of Fixed Length Sessions with all input items and the problem of Variable Length Sessions with a subset of input items are omitted as they are subsumed by the ones formalized in this paper.

4.2.2.1 Fixed Length Sessions. Given N items, we are interested in finding a sequence $S =< s_1, \ldots, s_k >$ of k sessions, each consisting of l items. We consider four problem variants all of which are instances of a general problem formalized as follows:

Optimize-Intra, Optimize-Inter. Given a set of N items with two dimensions of interest d and d' on Intra and Inter respectively, we are interested in creating a sequence $S =< s_1, ..., s_k >$ of k sessions, each containing l items, s.t. $N = k \times l$ and

$$\begin{aligned} &\underset{S}{\text{optimize}} \sum_{i=1}^{k} (Intra^{d}(s_i)) \\ &\underset{S}{\text{optimize}} \sum_{i=1}^{k-1} (Inter^{d'}(s_i, s_{i+1})) \\ &\text{s.t.} \\ &\quad |S| = k,\ |s_i| = l, N = k \times l \end{aligned} \tag{4.3}$$

4.2.2.2 Variable Length Sessions. Given N items, we are interested in finding a sequence $S =< s_1, \ldots, s_k >$ of k sessions, with length $L =< l_1, \ldots, l_k >$ $s.t.\ l_i \leq l;\ \forall i = 1, \ldots, k$. We consider four problem variants all of which are instances of a general problem formalized as follows:

Optimize-Intra, Optimize-Inter. Given a set of N items with two dimensions of interest d and d' on Intra and Inter respectively, we are interested in recommending a subset of items by creating a sequence $S =< s_1, ..., s_k >$ of k sessions with length $L =< l_1, \ldots, l_k >$ $s.t.\ l_i \leq l;\ \forall i = 1, \ldots, k$, and $N = k \times l$

$$
\begin{aligned}
&\underset{S}{\text{optimize}} \sum_{i=1}^{k} (Intra^{d}(s_i)) \\
&\underset{S}{\text{optimize}} \sum_{i=1}^{k-1} (Inter^{d'}(s_i, s_{i+1})) \\
&\quad \text{s.t.} \\
&\qquad |S| = k,\ |s_i| = l_i,\ l_i \leq l,\ \sum_{\forall i} l_i < N,\ N = k \times l
\end{aligned}
\tag{4.4}
$$

We refer to Sub-section 4.4.2 for further details.

4.2.3 Analysis of the problems considering fixed length sessions

We analyze the complexity of Intra and Inter diversities. This exercise allows us to analyze the nature of these problems and sheds light on designing principled solutions.

4.2.3.1 Intra Diversity Optimization.

Theorem 4.1. *Min-Intra is Polynomial time solvable.*

Proof. Minimizing Intra diversity is akin to grouping a set of points in a line to produce the smallest aggregated variance. This requires sorting the points based on the Intra dimension d and grouping every l points to create a session. Clearly, this is polynomial-time solvable. □

Theorem 4.2. *Optimizing Max-Intra is NP-Hard.*

Proof. The proof of this theorem uses another claim that we prove later (Theorem 4.6). This latter theorem formally proves that Max-Intra happens ($\sum_{i=1}^{k}(Intra(s_i))$ is maximized) if the mean of each session is equal (or very close) to the global mean of all N items for the specific dimension d. We omit the superscript d from the proof and t_i is considered as the value of the item t_i for dimension d. Since groups have the same size l, the problem is akin to finding groups of items whose sum is equal:

$$\sum_{t_i \in s_1} t_i = \sum_{t_i \in s_2} t_i = \cdots = \sum_{t_i \in s_k} t_i \tag{4.5}$$

To prove NP-hardness we reduce an instance of the k-Equal Subset Sum of Equal Cardinality Problem (k-ESSEC) [45] to an instance of Max-Intra, as follows. Given an instance of k-ESSEC with $P = \{a_1, ..., a_N\}$ which are N positive integers and k, we set the items $t_i = a_i$ and k remains the same. A solution to the k-ESSEC with k disjoint subsets, each with equal value $sum(s_1) = sum(s_2) = \ldots = sum(s_k)$ occurs, iff a solution of the Max-Intra exists with $l = N/k$ and $\mu_{s_i} = \mu_{s_{global}} = \frac{\Sigma_{i=1}^{N} a_i}{N}$.

□

4.2.3.2 Inter Diversity Optimization. The Inter diversity problem aims to find a sequence of k sessions of length l that will optimize the aggregated Inter distance computed on a dimension d over all k sessions in that sequence.

Theorem 4.3. *Inter Problem (both Min and Max) is NP-Hard.*

Proof. (Sketch) We show the NP-hardness for the Min-Inter case, and the maximization works analogously. To prove the NP-hardness of the Min-Inter problem, we reduce an instance of the known NP-hard problem Hamiltonian Path problem [64] to an instance of the Min-Inter problem. Consider an instance of the Hamiltonian Path problem with $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each node $v_i \in V$ represents l items with the same value on the dimension of interest. Essentially, these l items form a session. For assigning the Inter diversity of two sessions, we first deal with the non-edges in G. For each edge $(v_i, v_j) \notin E$, we set the μ_{s_i} and μ_{s_j} such that $||\mu_{s_i} - \mu_{s_j}|| > X$ (where X is an integer) and for each edge $(v_i, v_j) \in E$, we create $||\mu_{s_i} - \mu_{s_j}|| \leq X$. This creates an instance of Min-Inter problem with $|V|$ (i.e., k for Min-Inter) sessions, each with l items. Clearly, this reduction can be done in polynomial time. Figure 4.1 shows such a reduction from

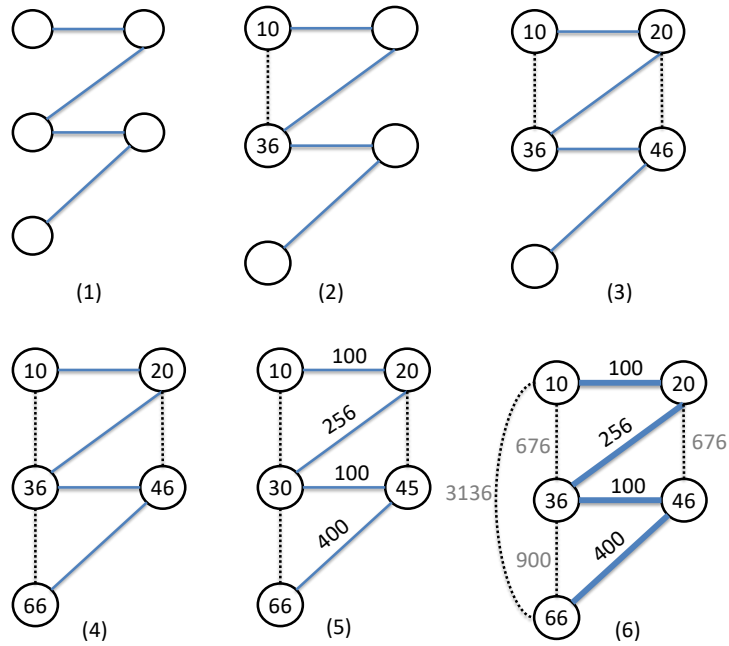


Figure 4.1 Reduction: Hamiltonian path to the Inter problem.

an example graph, where $X = 15$. Now a Hamiltonian Path exists in G, iff Min-Inter value is $< X^2 \times |V|$. □

Theorem 4.4. *The bi-objective optimization problems combining Intra and Inter diversity are all NP-Hard.*

Proof. (Sketch) We omit the formal proof for brevity - but it is easy to show that the NP-hardness remains for each of the four bi-objective problems since the individual optimization problems are NP-hard. □

4.2.4 Modified problem definitions of fixed length sessions

As proved in Theorem 4.4, each of the four bi-objective optimization problems is NP-hard. In fact, two ((*Min-Inter, Max-Intra*) and (*Max-Inter, Max-Intra*)) out of the four problems are NP-hard on both objectives. Upon careful investigation, we propose an alternative formulation of each of these bi-objective problems to a

corresponding constrained optimization problem, with the goal of obtaining one point from the Pareto front. The idea is to optimize Inter diversity, subject to the constraint of Intra diversity.

The constraint on Intra is obtained by solving the Intra optimization first. There exists more than one benefit to this approach. First, in one of the two cases (i.e., Minimization) Intra is tractable and easier to solve, therefore, coming up with the optimal value of the constraint is computationally efficient. More importantly, the constrained optimization problem aims at finding one point in the Pareto front, which is perfectly acceptable, as the points in the Pareto front are qualitatively indistinguishable (unless further information is available).

$$\begin{aligned} &\min_{S}(\max) \sum_{i=1}^{k-1} (Inter^{d_2}(s_i, s_{i+1})) \\ &\quad \text{s.t.} \\ &\qquad \sum_{i=1}^{k} (Intra(s_i))x <= OPT_{Intra^{d_1}} \\ &\qquad |S| = k,\ |s_i| = l, N = k \times l \end{aligned} \tag{4.6}$$

where OPT_{Intra} is the optimal solution of the Intra problem.

Using Example 4.1, the sequence

$$S =< \{t_5, t_6, t_7\}, \{t_1, t_2, t_3\}, \{t_9, t_{10}, t_{11}\} >$$

minimizes the $Intra^{Skill}$ score but at the same time maximizes the $Inter^{Reward}$ score whereas

$$S' =< \{t_1, t_2, t_3\}, \{t_9, t_{10}, t_{11}\}, \{t_5, t_6, t_7\} >$$

minimizes the $Intra^{Skill}$ and minimizes the $Inter^{Reward}$.

Table 4.2 Optimization Algorithms and Results for Fixed Length Sessions

Algorithm	Running Time	Approx Factor
Ex-Min-Intra	$O(NlogN)$	Exact
Ap-Max-Intra	$O(NlogN + Nl)$	$\frac{1}{2-1/k}$
Ap-Min-Inter	$O(NlogN + k^2 + logk)$	$4 - 2/k$
Ap-Max-Inter	$O(NlogN + k^2 + logk)$	$1/2$

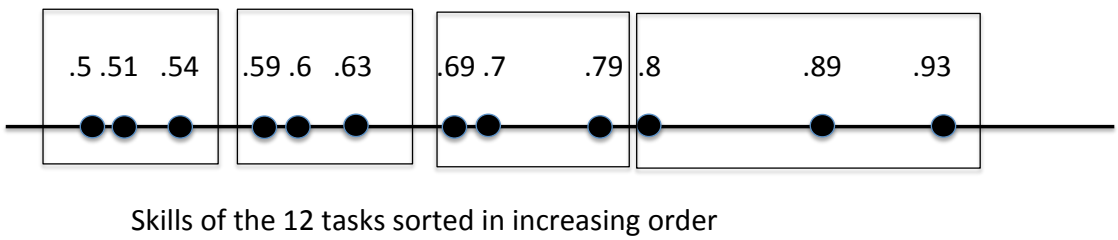


Figure 4.2 Sorted Intra diversity of skills.

4.3 Optimization Algorithms

We design optimization algorithms for the Intra and Inter problems individually, following which, we study how to solve the constrained optimization problem (Equation (4.6)). Table 4.2 summarizes our technical results.

4.3.1 Algorithm Min-Intra

4.3.1.1 Fixed Length Sessions. The objective here is to design k sessions, each of length l, such that the aggregated Intra diversity over the k sessions is minimized. Specifically, if there are l values associated with a dimension in a session, the Intra diversity is the variance of those points that is to be minimized here.

With an abstract representation, once sorted, the dimension values of N items, fall on a line, as shown in Figure 4.2. Therefore, if the aggregated variance is to be minimized, it is intuitive that the sessions need to be formed by grouping l values that are closest to each other.

Thus our proposed Ex-Min-Intra algorithm for minimizing Intra diversity first sorts the values of the dimension of interest. After that, it starts from the smallest value and finds each consecutive l points to form a session.

Theorem 4.5. *Algorithm Ex-Min-Intra is exact.*

Proof. (Sketch) Let us assume that our algorithm does not produce an exact solution. That means there exists another algorithm that produces a solution with smaller Intra diversity than that of Ex-Min-Intra. Suppose this other algorithm uses another way to create the sessions. Of course, this is different from sorting the items by increasing the value of the dimension of interests and grouping each l of them starting from the smallest one. However, that is a contradiction because then the latter algorithm will have a larger Min-Intra value, as l non-consecutive points will have higher variance than consecutive ones. Hence the proof. □

Lemma 4.1. *Algorithm Ex-Min-Intra runs in $O(NlogN)$.*

Proof. Since the only required operation is sorting, the running time of the algorithm will take $O(NlogN)$. □

4.3.1.2 Variable Length Sessions. For the Variable Length Sessions problem, we group the items depending on the specified input length after sorting them. To clarify more, in Example 4.1, if we are given $[2, 3, 2, 3]$ as the sessions' length input, we choose the first two items after sorting as the first session, then the following three items as the second session, and so on.

4.3.2 Algorithm Max-Intra

As proved in Sub-section 4.2.3, Max-Intra is NP-hard. What makes it computationally intractable is that when the objective is to maximize variance, the search space has to be combinatorially explored.

We show that Max-Intra is optimized when all sessions have the same mean, which is equal to the global mean μ_{global}. This proof is critical, as it helps us design our solution. Theorem 4.6 has the formal statement.

Theorem 4.6. $\sum_{i=1}^{k}(Intra(s_i))$ *is maximized when*

$$\mu^d_{s_1} = \mu^d_{s_2}, \ldots = \mu^d_{s_k} = \mu_{global} \tag{4.7}$$

Proof. The theorem states that the objective is maximized when the means of all sessions are equal, which in turn are equal to the global mean. It is indeed true that when $\mu^d_{s_1} = \mu^d_{s_2}, \ldots = \mu^d_{s_k}$, the global mean $\mu_{global} = \frac{1}{N}\sum_{j=1}^{N}(t^d_j) = \frac{k \times \mu^d_{s_i}}{k} = \mu^d_{s_i}$

Our intention is to prove that $\sum_{i=1}^{k}(Intra(s_i))$ is maximized when this aforementioned scenario occurs. For ease of exposition, we omit the superscript d from the proof.

We do the proof by the method of contradiction. Consider two different sets of k sessions, S and S'. For $S = s_1, s_2, \ldots, s_k$, we have $\mu_{s_1} = \frac{1}{l}\sum_{t \in s_1} t$ and similarly for other $s_i \in S$. For $S' = s'_1, s'_2, \ldots, s'_k$ where

$$\mu_{s'_1} = \mu_{s'_2} = \ldots = \mu_{s'_k} = \mu_{global} = \frac{1}{k * l}\sum t \tag{4.8}$$

We also assume, $Intra(S) > Intra(S')$.

$$\begin{aligned} Intra(S) &= \sum_{s_i} Intra(s_i) \\ &= \sum_{t \in s_1}(t - \mu_{s_1})^2 + \ldots + \sum_{t \in s_k}(t - \mu_{s_k})^2 \\ &= \sum_{i=1}^{N} t_i^2 - l(\mu^2_{s_1} + \mu^2_{s_2} + \ldots + \mu^2_{s_k}) \end{aligned} \tag{4.9}$$

$$Intra(S') = \sum_{i=1}^{N} t_i^2 - kl\mu^2_{global} \tag{4.10}$$

According to our assumption, $Intra(S) > Intra(S')$ this means that,

$$\sum_{i=1}^{N} t_i^2 - l(\mu_{s_1}^2 + \mu_{s_2}^2 + \ldots + \mu_{s_k}^2) > \sum_{i=1}^{N} t_i^2 - kl\mu_{global}^2 \tag{4.11}$$

which after considering $\mu_{opt} = \frac{\mu_{s_1}+\mu_{s_2}+\ldots+\mu_{s_k}}{k}$ we get,

$$\mu_{s_1}^2 + \mu_{s_2}^2 + \ldots + \mu_{s_k}^2 < 0 \tag{4.12}$$

which is a clear contradiction, hence the proof. □

4.3.2.1 Fixed Length Sessions. Theorem 4.6 provides a useful insight, that is, to maximize the Intra, we need to form the k sessions in such a way that the means of all the sessions are equal or very close to each other. Algorithm Ap-Max-Intra is iterative and greedy and it relies on this principle to create sessions that satisfy this property. First, it creates l bins, each has k different slots. The bins are then initialized so that each contains a subset of k items from the set of items that are sorted in ascending order. Then, in the third step, each of the l bins are scored using a scoring function described in Definition 4.1 , which captures the maximum difference between the average of all items and the ones in each bin. Finally, it merges the two bins with the highest and lowest scores greedily. The final two steps are repeated iteratively. This process is repeated for $l - 1$ times.

Definition 4.1. *(***Score of the i-th bin:***)*

$$d(b_i) = \max\{|\mu_{global} - \operatorname*{argmax}_{\forall j} b_{ij}|, |\mu_{global} - \operatorname*{argmin}_{\forall j} b_{ij}|\}$$

This scoring function captures the largest difference between items in a bin and the global average, allowing the highest and lowest-scoring bins to be merged. If we do this, as we proved in Theorem 4.6, the sessions created at the last step have an average near to μ_{global}, which maximizes the Intra value.

$$T = \{0.5,0.51,0.54,0.59,0.6,0.63,0.69\ ,0.7,0.79,0.8,0.89,0.93\}$$

Step 1: $$b = \begin{bmatrix} [] & \cdots & [] \\ \vdots & \ddots & \vdots \\ [] & \cdots & [] \end{bmatrix}$$

Step 2: $$b_{l\times k} = \begin{bmatrix} [0.5] & [0.51] & [0.54] & [0.59] \\ [0.6] & [0.63] & [0.69] & [0.7] \\ [0.79] & [0.8] & [0.89] & [0.93] \end{bmatrix}$$

Step 3: $$\begin{aligned} d(b_1) &= \max(\{|0.68 - 0.59|, |0.68 - 0.5|\}) = 0.18 \\ d(b_2) &= \max(\{|0.68 - 0.6|, |0.68 - 0.7|\}) = 0.08 \\ d(b_3) &= \max(\{|0.68 - 0.79|, |0.68 - 0.93|\}) = 0.25 \end{aligned}$$

Step 4: $$Merge(b_2, b_3) \qquad \begin{bmatrix} [0.5] & [0.51] & [0.54] & [0.59] \\ [0.6,0.93] & [0.63,0.89] & [0.69,0.8] & [0.7,0.79] \end{bmatrix}$$

Figure 4.3 Ap-Max-Intra steps on Example 4.1.

To illustrate the solution further, b_{ij} represents the j-th slot in bin i, which is kept as a placeholder for j-th session. To initialize the bins, we first sort the items in increasing order on the dimension of interests. Next, in the i-th bin $1 \leq i \leq l$, we put the sorted items $t_{(i)*k+j}$ in b_{ij}. Using Example 4.1, this amounts to creating three bins of tasks where $b_1 = \{[t_1], [t_2], [t_3], [t_4]\}$, $b_2 = \{[t_5], [t_6], [t_7], [t_8]\}$, and $b_3 = \{[t_9], [t_{10}], [t_{11}], [t_{12}]\}$. In step 3, each bin is scored, based on $d(b_i)$, as presented in Definition 4.1. Then two bins i and j are merged that have the largest and smallest score respectively. Going back to the Example 4.1, the scores are calculated as follows $d(b_1) = 0.18$, $d(b_2) = 0.08$, and $d(b_3) = 0.25$ and b_2 and b_3 are merged. Figure 4.3 details these steps.

To merge b with b', where b has the largest score and b' has the smallest score, we create a new bin b^{merge} such that, b_{ij}^{merge} contains the m-th smallest items of b and m-th largest items of b' $(1 \leq m \leq k)$. Considering Example 4.1, the new bin b^{merge} is

created by combining b_2 and b_3 , such that

$$b^{merge} = \{[t_5, t_{12}], [t_6, t_{11}], [t_7, t_{10}], [t_8, t_9]\}$$

This process is then repeated until only a single bin is left.

Algorithm 4.1 Algorithm Ap-Max-Intra

Require: N, Number of sessions k, Length of session l

1: $\mu_{global} \leftarrow$ Average of all items
2: Initialize l bins each with k slots $\leftarrow$
3: $b_i \leftarrow \{b_{i1} = [t_{il+1}], b_{i2} = [t_{il+2}, ..., b_{ik} = [t_{il+k}]]\}$
4: **while** number of bins > 1 **do**
5: pick b_i and b_j with the largest and smallest scores
6: b^{merge}=merge b_i and b_j
7: Delete b_i and b_j
8: number of bins $\leftarrow l - 1$
9: Return the final merged bin

Theorem 4.7. *Ap-Max-Intra runs in $O(NlogN + Nl)$.*

Proof. Getting the average of the items takes $O(N)$. The partitioning of items into k bins takes $O(NlogN)$ which is done by sorting items first and then putting each item in their corresponding bin by iterating over them once more. Now there are $l - 1$ iterations of the algorithm to merge the bins. Each bin merge takes at most $O(kl)$ since there are k sessions with at most l members which means for $l - 1$ iterations we will have $O(kl^2)$. Overall, the running time of the algorithm will be $O(NlogN + Nl)$ □

Theorem 4.8. *Algorithm Ap-Max-Intra has $\frac{1}{2-1/k}$ approximation factor.*

Proof. (Sketch) The detailed proof of this problem makes use of an approximation-preserving reduction. Basically, the idea of an approximation-preserving reduction is as follows: we need to show that an instance of Ap-Max-Intra is reducible to an instance of another known NP-hard problem, Balanced Number Partitioning problem [95] and by applying Algorithm BLDM, which is an approximation algorithm for the latter problem produces a solution for the problem Ap-Max-Intra. The proof sketch makes use of two arguments: the first is that an instance of Max-Intra could be reduced to an instance of the Balanced Number Partitioning problem [95] in polynomial time. Then, it can be shown that the BLDM algorithm has one-on-one correspondence with Ap-Max-Intra. Ap-Max-Intra will accept $\frac{1}{2-1/k}$ approximation factor, since BLDM holds $2 - 1/k$ approximation factor. □

4.3.2.2 Variable Length Sessions. The solution of Max-Intra for Variable Length Sessions is identical to the aforementioned one, except for the last step. If the length of each of the k bins is smaller than the length of the input variable, we merge them; otherwise, we skip that one and move on to the next one to merge. To clarify, if we want to merge one more time after step 4 in Figure 4.3, we skip the first column because the first session must have length 2, but we merge [0.51] and [0.63, 0.89] in the second column to get a session with length 3 as specified in the input length $[2, 3, 2, 3]$.

4.3.3 Algorithm Min(Max)-Inter

4.3.3.1 Fixed Length Sessions. Optimization of Inter diversity, both minimization and maximization variants, is NP-hard, and they bear remarkable similarity to each other. Given a set of N items, the Min(Max)-Inter problems will try to find an ordering of k sessions, each with l items, such that the aggregated differences between the average of two consecutive sessions is minimized (maximized). To better understand these problems, we break them into two steps. We only present these steps

for the Max-Inter problem and note that the Min-Inter version works analogously, only by inverting the optimization goals inside the algorithm. For example, for optimizing Max-Inter, our goal is to find a sequence of sessions that maximizes Equation (4.2). One intuition is that Inter diversity increases if the means of individual sessions (on the dimension of interest) are highly different from each other. Indeed, if the k sessions have the same exact mean, no matter how one orders them, Inter diversity will be zero. As we prove in Lemma 4.2, this relates to forming a set of k sessions with the goal to minimize Intra diversity. So, the first step of our algorithm is to produce a set of sessions with the smallest Intra diversity. The next step is to order these sessions, such that the resulting sequence has the Inter value maximized. This is our guiding principle in creating the algorithms to solve this problem.

Our proposed solution Ap-Max-Inter for Max-Inter works as follows: we first find k sessions obtained by running Algorithm Ap-Min-Intra. This is needed since it will generate sessions with means as different from each other as possible. After that, we create a graph of k nodes, each representing one of the k sessions. The weight of each edge (s_i, s_j) is defined as $w(s_i, s_j) = (\mu_{s_i} - \mu_{s_j})^2$. After that, the goal is to run an algorithm for the Longest path problem for Max-Inter. Since the graph is complete with positive weights on the edges, the Longest Path Problem could be solved by replacing the positive weights with negative values and running a traveling salesman problem (TSP) over it. In our implementation, we use the simple yet effective 2-approximation algorithm for TSP in metric space, described in [89,107]. The algorithm starts by finding the Minimum Spanning Tree of the input graph using Prim's algorithm. Next, it lists the nodes in Minimum Spanning Tree in a pre-order walk and adds the edge to the starting vertex to the end. This path will create an ordering of sessions by following from the starting vertex s_i to the ending vertex s_j. This algorithm runs in $O(k^2 logk)$ which is dominated by the running time of Prim's

algorithm. We further improve this running time by using Fibonacci heaps and obtain $O(k^2 + logk)$.

Inversely, Algorithm Ap-Min-Inter, designed for Min-Inter first solves the Min-Inter problem to create sessions with the largest Intra diversity. Then, we create the graph the same as we have done in Ap-Max-Inter but the edge weights do not need to be negated. Finally, we run TSP [107] to generate a sequence of sessions for minimizing Inter diversity of those sessions. For both problems, the obtained solution is a cycle and has one extra edge. We simply remove the edge with the smallest (largest) value in the solution. This produces an ordering of the sessions. Algorithm 4.2 presents the pseudo-code of Max-Inter algorithm.

Algorithm 4.2 Algorithm Ap-Max-Inter

Require: N items, Number of sessions k, Length of session l

1: $S_{init} \leftarrow Min - Intra(N, k, l)$
2: $G = (S, E) \leftarrow$ complete graph with k nodes
3: $w(s_i, s_j) = (\mu_{s_i} - \mu_{s_j})^2$
4: Run Longest path algorithm on G
5: Longest path contains the ordering of the sessions.

Theorem 4.9. *Both Ap-Max-Inter and Ap-Min-Inter run in $O(NlogN + k^2 + logk)$.*

Proof. The running time of the algorithm Ap-Max-Inter is dominated by the first step which is getting the solution of Min-Intra (for Ap-Min-Inter it is Max-Intra). The algorithm for TSP takes $O(k^2 + logk)$. This means that the overall running time will be $O(NlogN + k^2 + logk)$. □

Lemma 4.2. *Given a set of N items forming k sessions (each with l items), when defined on the same dimension of interest, Inter diversity of the k sessions is maximized (minimized), when Intra diversity of those k sessions is minimized(maximized).*

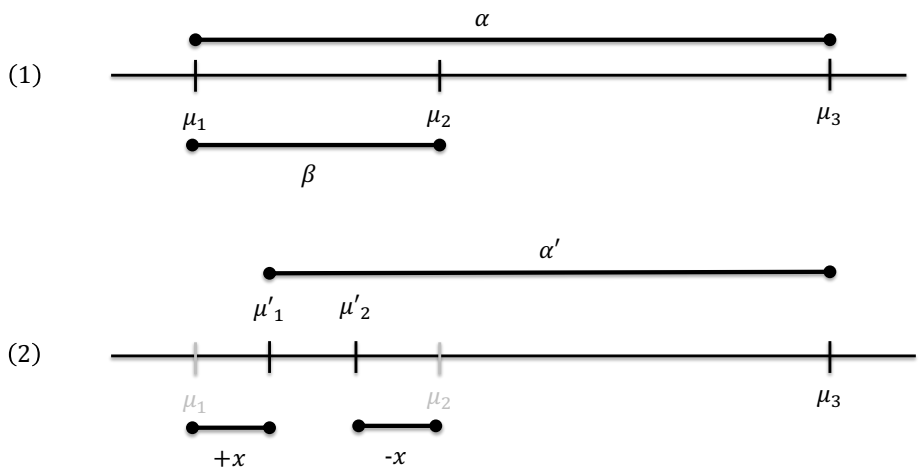


Figure 4.4 Relationship between Min-Intra and Max-Inter when defined on the same dimension (1) If $S =< s_2, s_1, s_3 >$ is the Min-Intra solution and $\mu_{s_1} \leq \mu_{s_2} \leq \mu_{s_3}$, the Inter value reaches its maximum value, which is $\alpha + \beta$; (2) If a task is swapped between sessions s_1 and s_2, the Inter value for the new sessions will be $\alpha + \beta - 3x$ which is smaller and cannot be the solution of Max-Inter.

Proof. (Sketch)

Inter Minimization Case: For the case of Max-Intra, the solution will require the averages of all groups to be the same (Recall Theorem 4.6). This results in having Min-Inter with value 0, leading to the optimal solution. Hence the proof.

Inter Maximization Case: We prove by contradiction for Min-Intra and Max-Inter, for $k = 3$. For the purpose of illustration, consider the sequence $S =< s_2, s_1, s_3 >$ where $\mu_{s_1} \leq \mu_{s_2} \leq \mu_{s_3}$. Consider s_1, s_2, s_3 are the solution of Min-Intra and $\mu_{s_3} - \mu_{s_1} = \alpha$ and $\mu_{s_2} - \mu_{s_1} = \beta$. Figure 4.4 presents one such solution. Now consider that we swap a task between s_1 and s_2. After this swap, the value of μ_{s_1} will increase by x amount, and the value of μ_{s_2} will decrease by the same x. Now it is easy to see that if the value of Inter is $\alpha + \beta$ for the solution of Min-Intra, then the value of the new solution will be $\alpha + \beta - 3x$ which is smaller. This argument extends to $k > 3$. □

Theorem 4.10. *Ap-Max-Inter produces an answer that is at least* 1/2 *of the the optimal solution.*

Proof. The approximation of Ap-Max-Inter occurs in step 2 while solving the longest path problem (Since Min-Intra has an exact solution)). Since the longest path algorithm has the 1/2 approximation factor, the overall algorithm Ap-Max-Inter has 1/2 approximation factor. □

Theorem 4.11. *Algorithm Ap-Min-Inter has* $4 - 2/k$ *approximation factor.*

Proof. Similar to the proof of Ap-Max-Inter, using Lemma 4.2, the first step of Ap-Max-Inter is finding a set of sessions that are closest to each other. Using the algorithm Ap-Max-Intra provides these sessions with $2 - 1/k$ approximation. The next step multiplies this error by a factor of 2 since the composition of the groups is not changed and we only find an ordering over the fixed groups. This yields an approximation factor of $4 - 2/k$. □

4.3.3.2 Variable Length Sessions. The Inter solution of the Variable Length Sessions is the same as Fixed Length Sessions for both minimization and maximization problems.

4.3.4 Optimizing Inter with Intra as a constraint for fixed length sessions

We now develop algorithms for the constrained optimization problems defined in Sub-section 4.2.4. When the values of the item dimension used for Intra diversity are all unique, two of these four algorithms have provable approximation factors. Table 4.3 provides the summary of our technical results.

To optimize Inter with Min-Intra as a constraint, we design two algorithms *Alg-Min-Intra, Min-Inter* and *Alg-Min-Intra, Max-Inter*. For both, we start from the solution of the Min-Intra problem using the algorithm Ex-Min-Intra. This solution is an exact algorithm for solving Min-Intra and gives a set of k sessions as the output. After that, we run Ap-Max-Inter in *Alg-Min-Intra, Min-Inter* and Ap-Min-Inter in *Alg-Min-Intra, Max-Inter*.

Table 4.3 Optimization Algorithms and Results for Fixed Length Sessions

Algorithm	Running Time	Approximation Factor
Alg-Min-Intra,Min-Inter	$O(NlogN + k^2)$	$(OPT, 4 - 2/k)$
Alg-Min-Intra, Max-Inter	$O(NlogN + k^2)$	$(OPT, 1/2)$
Alg-Max-Intra, Min-Inter	$O(NlogN + Nl + k^2)$	*heuristic*
Alg-Max-Intra, Max-Inter	$O(NLogN + Nl + k^2)$	*heuristic*

On the other hand, to optimize Inter with Max-Intra as a constraint, we start from the solution of the Max-Intra using the algorithm Ap-Max-Intra. This solution is an approximation algorithm for solving Max-Intra and returns a set of k sessions. After that, we run Ap-Max-Inter for *Max-Intra, Max-Inter* and Ap-Min-Inter for the *Max-Intra, Min-Inter*. Using Example 4.1, to solve *Alg-Min-Intra, Max-Inter* where the Intra dimension is on *Skill* and Inter dimension is on *Skill* as well, we first call the Ex-Min-Intra subroutine which sorts N items and group these items that are close to each other and obtain the following sessions, $s_1 = \{t_1, t_2, t_3\}$, $s_2 = \{t_4, t_5, t_6\}$, $s_3 = \{t_7, t_8, t_9\}$, and $s_4 = \{t_{10}, t_{11}, t_{12}\}$ where $\mu_{s_1} = 0.516$, $\mu_{s_2} = 0.6066$, $\mu_{s_3} = 0.726$, and $\mu_{s_4} = 0.873$ (see Figure 4.2). Given the solution of these 4 sessions, we then create a complete graph (Figure 4.5) by considering each session as a node. The weight of each edge in this graph is the Inter value of adjacent sessions on the *Skill* dimension that are calculated using Equation (4.2). The Ap-Max-Inter is then akin to the longest path problem. We convert these positive weights to negative weights by introducing a minus sign and then apply our proposed 2-approximation algorithm for the traveling salesman problem (TSP) on metric space that gives us the following tour $T = \{s_1 \rightarrow s_4 \rightarrow s_2 \rightarrow s_3 \rightarrow s_1\}$. We remove the edge $s_2 \rightarrow s_3$ since it has the smallest weight. The solution of Max-Inter is hence the sequence $S =< s_2, s_4, s_1, s_3 >$.

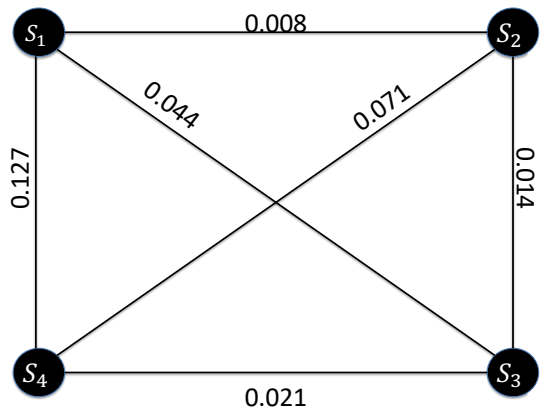


Figure 4.5 Ap-Max-Inter graph of Example 4.1.

Alg-Min-Intra, Min-Inter problem is solved in a similar manner by following the steps outlined above. The only distinction is that we don't have to convert weight to negative weight.

Algorithm 4.3 presents the generic pseudo code. These two algorithms are based on heuristics and may not have any provable bounds.

Algorithm 4.3 Algorithm for maximizing Inter with Intra as a constraint

Require: N items, Number of sessions k, Length of session l, dimensions d_1 and d_2

1: $S_{init} \leftarrow k$ sessions of l items each, obtained by running Intra optimization algorithm on d_1
2: $G = (V, E) \leftarrow$ complete graph with nodes of S_{init} and edge weights are calculated based on d_2 values between a pair of sessions
3: Call Subroutine Ap-Max-Inter or Ap-Min-Inter on G

Theorem 4.12. *Algorithm Alg-Min-Intra, Max-Inter has* $(1, 1/2)$ *approximation factor Min-Intra, Max-Inter problem and Alg-Min-Intra, Min-Inter has* $(1, 4 - 2/k)$ *approximation factor Min-Intra, Min-Inter problem, as long as items in Intra dimension have unique values.*

Proof. (Sketch) We provide the proof for *Alg-Min-Intra, Max-Inter* and the proof of Alg-Min-Intra, Min-Inter works analogously. Ex-Min-Intra is optimal. Since items have unique values on Intra diversity dimension, there exists one and only one set of k sessions that minimizes Intra diversity values. The second step of the algorithm *Alg-Min-Intra, Max-Inter* creates an ordering over these sessions. In that subset of the search space, i.e. containing only solutions that start with the sessions of Ex-Min-Intra where the Min-Intra is optimal, our Max-Inter algorithm produces a solution that is 1/2 the optimal solution. Hence the $(1, 1/2)$ approximation factor holds for *Min-Intra, Max-Inter* problem. Similarly, the $(1, 4 - 2/k)$ approximation factor holds for *Min-Intra, Min-Inter* problem. □

4.4 Experimental Evaluations

We first conduct experiments involving human subjects on music playlist recommendation and task recommendation in crowdsourcing to observe the effect of diversity on user satisfaction (in both applications) and worker performance (in crowdsourcing). Then, using large-scale real data and synthetic data, we examine the quality of our algorithms in comparison to baselines and evaluate the scalability of our approach.

Except for Sub-section 4.4.2, which is related to the Variable Length Sessions, the rest of the section focuses on the Fixed Length Sessions.

4.4.1 Experiments with human subjects

We validate our framework in two applications: music recommendation, where we generate music playlists, and task recommendation in crowdsourcing, where we generate task sessions. These experiments are summarized in Table 4.4.

4.4.1.1 Music Recommendation. We generate music playlists for users and consider different contexts namely music for long drives, theme party, Sunday morning, and learning a new music style, to observe how diversity affects user

Table 4.4 Summary of Experiments With Human Subjects

Experiment	# of workers	Setup	Observed data	Findings
1. Music Recommendation	**200**	Users rate playlists. Each playlist has 5 channels. Each channel has 10 songs.	user satisfaction no. of selected songs diversity rating	User satisfaction, no. of selected songs, and diversity rating are higher in diversified playlists.
2. Task Recommendation	**200 (total)**	Workers complete and rate task sessions. Each task session has 5 task sets. Each task set has 10 tasks.	worker satisfaction quality throughput	Worker satisfaction and quality are higher in diversified task sessions.
2.1. Controlled Session Gap	102	Workers complete task sessions in session gaps of 1 minute, 5 minutes, and 10 minutes.	workers' preference (diverse vs. similar task sessions) factors affecting workers' satisfaction (diversity, relevance, others)	Worker satisfaction is higher in diversified task sessions across all session gaps and peaks at the 5-minute session gap. Quality is higher in diversified task sessions across all session gaps. Workers prefer diverse sessions diversity is the main factor in 55% of the workers' ratings.
2.2. Random Session Gap	98	Workers complete tasks anytime.		

Table 4.5 Diversity Dimensions per Context

	Long Drive	**Theme Party**	**Sunday Morning**	**Learn Music**
Intra	tempo	period	popularity	genre
Inter	popularity	genre	genre	tempo

satisfaction in different contexts. Each playlist contains five distinct channels, each with 10 songs.

Dataset. The dataset consists of 727 songs from 54 albums, 47 artists, and 10 genres. The songs are from albums that won the Grammy Best Album of the Year Award between 1961 and 2020. The list of albums and their corresponding genres are from Wikipedia while the other information such as artist, period, popularity, tempo, and duration are from Spotify.

Experiments Flow. We first collect preferred genres and artists from users to form their profiles. We then generate five music playlists for each user. Each playlist has five channels, and each channel has 10 songs. The first four playlists are generated using the algorithms in Table 4.3, with dimensions specified for each context in Table 4.5. The 5th playlist represents the baseline with No-diversity. It consists of similar songs randomly selected from the same dimension. Specifically, in this case, all songs from the period 2000's. Lastly, users evaluate the playlists by selecting songs they would actually listen to, rating how much they like diversity in the sessions, and providing an overall rating of the entire playlist. The ratings are based on a 5-pt Likert scale where 1 is the lowest and 5 is the highest. We measure user satisfaction using the overall rating provided by users. We recruit 200 workers (50 per context) from Amazon Mechanical Turk (AMT). We pay the workers $0.10 for profile collection and $1.00 for their evaluations.

Table 4.6 Average Evaluation Scores across All Contexts

	Scenario	No. of Selected Songs	Diversity Rating	User Satisfaction
1	*Min-Intra, Min-Inter*	**15.16**	3.57	3.54
2	*Min-Intra, Max-Inter*	15.05	3.66	3.66
3	*Max-Intra, Min-Inter*	14.71	3.59	**3.71**
4	*Max-Intra, Max-Inter*	14.66	**3.69**	**3.71**
5	No-diversity	*12.83*	*3.35*	*3.44*

Summary of Results. We observe in Table 4.6 that user satisfaction in diversified playlists (Scenarios $1-4$) is higher compared to the No-diversity baseline. This observation is statistically significant at $p = 0.10$ using a one-way Analysis of Variance (ANOVA) [123]. The results are consistent with other measures: workers select the smallest number of songs from the No-diversity playlist and the No-diversity playlist receives the lowest average diversity ratings. Moreover, these observations extend to different contexts, as shown in Tables 4.7, 4.8, and 4.9. These results are summarized in Table 4.4 - Experiment 1. The sample size of 200 workers from the estimated $200,000$ workers in AMT [50] gives our results a 99% confidence level and a 10% error margin (based on the Central Limit Theorem [126]). In summary, our music experiment clearly shows that diversity is preferred over No-diversity. Additionally, diversity definitions depend on context, as observed in Tables 4.7, 4.8, and 4.9.

4.4.1.2 Task Recommendation. In these experiments, we recommend task sessions to workers in crowdsourcing. Each task session consists of five sets and each set consists of 10 tasks.

Table 4.7 Average Number of Selected Songs per Context

	Scenario	Long Drive	Theme Party	Sunday Morning	Learn Music
1	*Min-Intra, Min-Inter*	**16.58**	14.86	14.76	14.42
2	*Min-Intra, Max-Inter*	15.82	**15.06**	14.12	**15.20**
3	*Max-Intra, Min-Inter*	16.52	13.64	14.30	14.38
4	*Max-Intra, Max-Inter*	16.24	13.96	**15.04**	13.40
5	No-diversity	*14.10*	*11.92*	*13.62*	*11.68*

Table 4.8 Average Diversity Rating per Context

	Scenario	Long Drive	Theme Party	Sunday Morning	Learn Music
1	*Min-Intra, Min-Inter*	3.64	3.52	3.64	3.46
2	*Min-Intra, Max-Inter*	3.70	3.50	3.82	3.61
3	*Max-Intra, Min-Inter*	3.70	3.54	3.58	3.54
4	*Max-Intra, Max-Inter*	**3.84**	**3.68**	**3.58**	**3.64**
5	No-diversity	*3.34*	*3.30*	3.46	*3.30*

Table 4.9 Average User Satisfaction per Context

	Scenario	Long Drive	Theme Party	Sunday Morning	Learn Music
1	*Min-Intra, Min-Inter*	3.62	3.88	3.34	3.32
2	*Min-Intra, Max-Inter*	3.76	3.72	3.66	3.50
3	*Max-Intra, Min-Inter*	**3.86**	**3.98**	3.56	3.44
4	*Max-Intra, Max-Inter*	3.76	3.80	**3.70**	**3.58**
5	No-diversity	*3.60*	*3.42*	3.46	*3.28*

Dataset. The dataset consists of $20,000$ tasks from Figure Eight's open data library [1]. Each task belongs to one of 10 types such as tweet classification, image transcription, and sentiment analysis. Each task type is represented as a set of keywords that best describe the required skills. Additionally, each task has a creation date, an expected completion time (less than a minute), and a reward that varies between \$0.01 - \$0.05.

Experiments flow. We collect a total of 200 user profiles where workers indicate (from 1 to 5) their interest in completing tasks, which are described by a given set of keywords. For each user profile, we generate task sessions using the algorithms in Table 4.3 and a combination of the following dimensions: skill, reward, duration, and creation date. We also generate a No-diversity baseline session. In this session, we randomly pick a task type and randomly select similar tasks belonging to that type. Next, workers complete and rate the recommended sessions. We measure worker satisfaction, quality of the completed tasks with respect to ground truth, and task throughput.

Satisfaction refers to how satisfied workers are with the task sessions (a rating from 1 to 5 provided by each worker). Quality refers to the percentage of correct

Table 4.10 Task Recommendation Sessions

	Session Type	Worker Satisfaction	Quality(%)	Throughput
1	*Min-Intra(creation date), Min-Inter(skill)*	4.26	0.67	7.72
2	*Min-Intra(skill), Max-Inter(reward)*	**4.30**	**0.68**	7.85
3	*Max-Intra(skill), Min-Inter (reward)*	4.29	0.66	7.60
4	*Max-Intra(duration), Max-Inter (skill)*	4.28	**0.68**	7.71
5	No-diversity	4.01	0.62	**7.92**

answers from the tasks completed by a worker. To measure quality, we use the answers obtained from the dataset as the ground truth. We use a naïve script that relies on basic equality to evaluate answer correctness. Throughput refers to the average number of tasks completed per minute.

We also investigate the impact of the session gap, the time interval between completing sessions, in our proposed algorithms. We conduct Experiment 2.1 in Table 4.4 where we assign 102 of the 200 workers to complete the sessions in three fixed time intervals of 1, 5, and 10 minutes between sessions (34 workers for each fixed time interval). We also ask these workers their preference between diversified and non-diversified sessions, and to indicate which factor mainly affects their satisfaction.

In total, we recruit 200 workers, pay each $0.03 for profile collection and at least $0.35 for task completion.

Summary of Results. Table 4.10 presents the average worker satisfaction, quality, and throughput in the task recommendation experiments. We observe that

Table 4.11 Throughput, Quality, and Worker Satisfaction of Diversified and Non-diversified Sessions with Different Session Gaps

		Session Gap (minutes)		
	Session Type	**1**	**5**	**10**
Worker Satisfaction	Diversified	3.76	4.30	4.21
	Non-diversified	3.48	4.09	3.91
Quality	Diversified	0.68	0.64	0.65
	Non-diversified	0.62	0.55	0.59
Throughput	Diversified	8.50	8.82	8.01
	Non-diversified	8.57	10.67	8.18

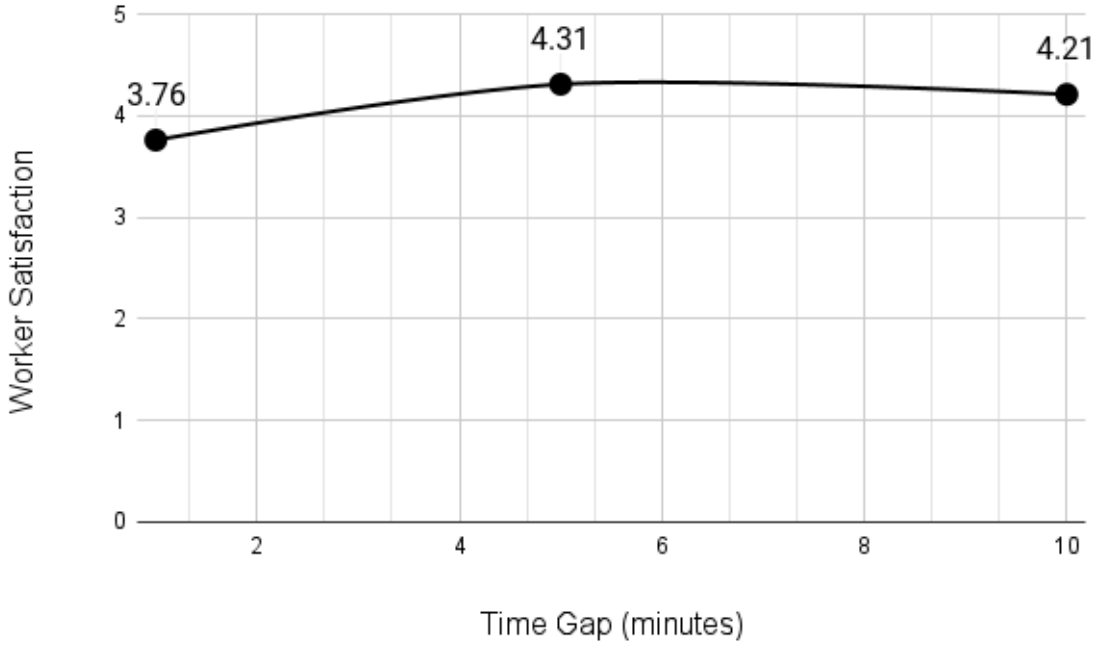


Figure 4.6 Worker satisfaction in varying session gaps.

worker satisfaction and quality in diversified sessions are higher compared to the No-diversity baseline. This observation is statistically significant at $p = 0.01$ using a one-way Analysis of Variance (ANOVA) [123]. On the other hand, throughput is marginally higher for the No-diversity case. This observation confirms previous studies where workers get more proficient in completing similar (and hence not diverse) tasks, allowing them to become faster at task completion [51].

In Table 4.11, we present the average worker satisfaction, quality, and throughput grouped by session gap. The values are from the 102 workers in Experiment 2.1 in Table 4.4 where we control the session gap. For each session gap (1, 5, 10 minutes), 34 workers complete the 5 session types listed in Table 4.10. The Diversified rows in Table 4.11 show the aggregated values obtained in the 4 task sessions generated by our algorithms while the Non-diversified rows show the values obtained in the baseline or No-diversity session.

Our findings show that quality and worker satisfaction are better in the diversified tasks sessions generated by our proposed algorithms across all session gaps. These observations are significant at $p = 0.10$ using a t-test [46]. Moreover, it is interesting to note that worker satisfaction peaks at the 5-minute session gap as seen in Figure 4.6. For the task sessions, 5 minutes may be the ideal break time workers need to alleviate fatigue or boredom [116]. We also note that our findings are consistent with the 102 workers' responses where 72.5% of the workers prefer diversified sessions over non-diversified sessions; 55% of the workers consider diversity in their ratings, 23.3% consider relevance, and 21.7% consider other factors.

In summary, our task recommendation experiments clearly show the benefit of diversity in the workers' satisfaction and the quality of crowdsourced tasks.

4.4.2 Large data experiments

The goal here is to evaluate our algorithms with appropriate baselines (including exact solutions) and compare them qualitatively (approximation factors, objective function value) and scalability-wise (running time). All algorithms are implemented in Python 3.6 on a 64-bit Windows server machine, with Intel Xeon Processor, and 16 GB of RAM. All numbers are presented as the average of five runs. For brevity, we present a subset of results that are representative.

4.4.2.1 Data Sets. a. 1-Million Song: We use the Million Songs Dataset [24] that has 1 million songs (please note the Spotify dataset used in Sub-section 4.4.1 is small in scale). We have normalized the data to be between $[0, 1]$. This dataset also includes artist popularity and hotness, genre, release date and etc. The presented results are representative and consider tempo and loudness as dimensions.
b. Synthetic dataset: The presented results on this are the ones that vary distributions of the underlying dimensions. We sample from three distributions: Normal, Uniform, and Zipfian. For Normal distribution, data is sampled with mean and standard deviation, $\mu = 250$, $\sigma = 10$. For Uniform, the dataset is sampled from Uniform distribution between [0,500], and for Zipfian distribution default exponent is set to $\alpha = 1.01$. We produce a pool of 2^{30} items for each of our three distributions.

4.4.2.2 Implemented Baselines. In addition to Random where we generate random sequences, we implement different baselines and compared the performance of our algorithms.

Optimal. The optimal baseline is based on an Integer Programming (IP) algorithm that solves the problem optimally on small instances. The rationale behind implementing IP is to verify the theoretical approximation factors of our algorithms against the optimal solution. We used Gurobi as the solver[2].

[2]https://www.gurobi.com/resource/switching-from-open-source/

Baseline-MMR. This baseline is inspired by the MMR algorithm [37] used in diversifying web search results. MMR takes a search query and returns relevant and diverse results. Hence, our mapping to MMR optimizes Intra session diversity only. At each iteration, Baseline-MMR considers an item to be included or not in the result and calculates two scores: the Intra score of adding a new item to a session and the *max* (resp., *min* Inter) score of a new session to all other sessions in the case of Max-Inter (resp., Min-Inter). It then picks the highest or the lowest weighted sum of these two scores based on the Intra part of the problem. The item with that score is chosen to be added to the session. This process is repeated until there is no item left.

Baseline-Constrained-KMean. This is a clustering technique similar to the one proposed in [33], which uses the K-Mean Clustering approach to produce a set of k clusters, each containing exactly l items. Following that, these clusters are sorted by increasing mean to generate a sequence of sets as the final result.

4.4.2.3 Summary of Results. Overall, for our problems, where both Intra and Inter diversity are to be optimized, our algorithms are the unanimous choice considering both quality and scalability. When the Intra and Inter diversity is studied individually, our algorithms outperform all the baselines and empirically produce approximation factors close to 1, across varying k, N, and different distributions. The only exception to this latter observation is Baseline-MMR, which performs better in maximizing Inter diversity (while performing very poorly for Intra optimization), which is due to its focus on optimizing Inter diversity only. Moreover, Baseline-Constrained-KMean performs poorly for the maximization problems, our algorithm convincingly outperforms it in both Intra and Inter minimization and maximization. This baseline also exhibits poor scalability. Contrarily, our algorithms are highly scalable and are much more efficient compared to the baselines.

Table 4.12 Approximation Factors on 1-Million Song Dataset

Our Scenarios	N=8192 , k=16		N=1024 , k=128	
	Intra	Inter	Intra	Inter
Min-Intra , Min-Inter	1	1.05	1	1
Min-Intra , Max-Inter	1	0.35	1	0.49
Max-Intra , Min-Inter	0.99	1.06	0.98	1.04
Max-Intra , Max-Inter	0.99	0.58	0.95	0.69

4.4.2.4 Quality Evaluation. We vary k (the number of sessions), N (the number of items), and the data distribution. The default values are N=2^{13} and k=2^7 with a uniform distribution.

Comparison against Optimal. Table 4.12 shows the approximation factors for our algorithms for two default settings: ($N = 2^{13}$, $k = 2^4$) and ($N = 2^{10}$, $k = 2^7$) using 1-Million Song dataset. We can see that our algorithms produce an approximation factor equal to 1 when Intra diversity is minimized and a factor very close to 1 when Intra diversity is maximized.

When Inter diversity is minimized, the resulting approximation factors are close to 1. However, when Inter diversity is maximized, the approximation factors are slightly low as our algorithm solves the Intra part of the problem before ordering the sessions to maximize Inter diversity. It is hence bound by the constraints of the solution to Intra. Nevertheless, the solution formulated by our algorithm for *Min-Intra, Max-Inter* and *Min-Intra, Min-Inter* is able to produce a point on the Pareto Front in the optimal solution region which meets both the Intra and Inter objectives. The synthetic dataset mimics this trend as well.

Based on the approximation factor results and the above analysis, we conclude that our algorithms produce good and in some cases the best possible solution for the four problems we attempt to optimize.

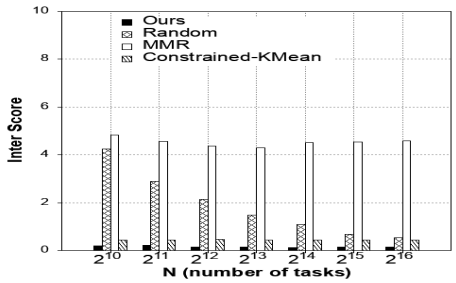
Varying N. Figures 4.7 and 4.8 show how Inter scores change as we vary N from 2^{10} to 2^{16} for Baseline-MMR, Baseline-Constrained-KMean, Random and our algorithms for 1-Million Song and synthetic dataset respectively.

Figures 4.7(a)(c) and 4.8(a)(c) confirm that our algorithm performs best when Inter diversity is minimized. The objective function improves with increasing N. On the other hand, as seen in Figures 4.7(b)(d) and 4.8(b)(d), when Inter diversity is maximized, Baseline-MMR outperforms our algorithm with increasing N. This is because our algorithm is subject to the constraints imposed by optimizing Intra diversity first and then maximizing the Inter diversity, while Baseline-MMR focuses on the Inter dimension only.

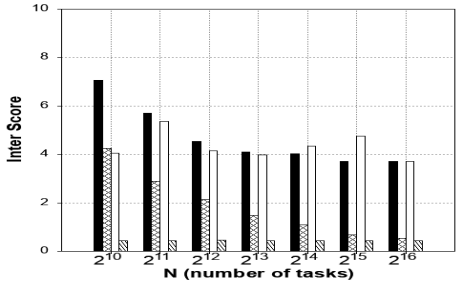
We also compare Intra scores whilst varying N. Tables 4.13 and 4.14 showcase the approximation factors of our algorithm's Intra considering Optimal for $N \leq 2^{13}$ and $N > 2^{13}$. A ratio of 1 means that the algorithm produces the best or optimal solution. These results showcase that our solutions achieve even better bounds empirically compared to the theoretical bounds. Tables 4.13 and 4.14 also show that although Baseline-MMR performs better in Max-Inter problem, it performs poorly for both Min-Intra and Max-Intra problems.

Interestingly, Random often times produces an approximation factor close to 1 for $N > 2^{13}$ when maximizing Intra. This is due to the fact that Intra is maximized when the variance of the sessions is maximized which is one of the side effects of Random algorithm. However, Baseline-MMR and Random produce very poor approximation factors when minimizing Intra. Contrarily, our solutions stay close to the 1 approximation factor for both minimization and maximization of Intra diversity. As N increases, the Intra scores do not see any drastic change in approximation factors and always stay close to 1.

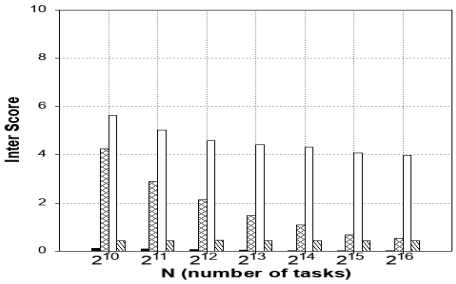
Varying k. Figures 4.9 and 4.10 present how Inter scores evolve as we vary k between 2^4 and 2^{11} for different baselines compared to our algorithm. We keep



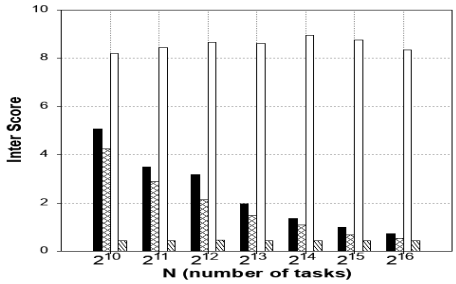
(a) Ap-Min-Inter (Min-Intra)



(b) Ap-Max-Inter (Min-Intra)

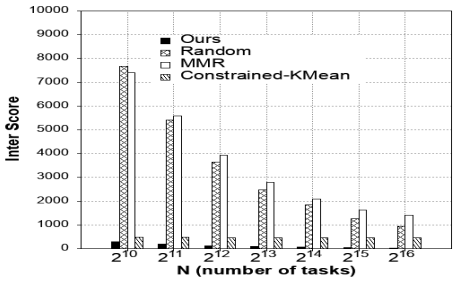


(c) Ap-Min-Inter (Max-Intra)

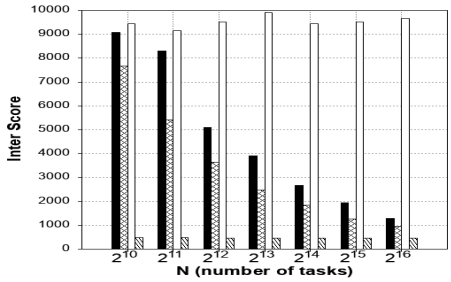


(d) Ap-Max-Inter (Max-Intra)

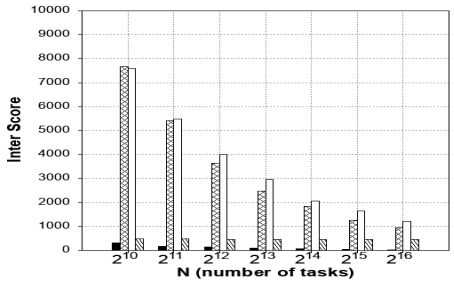
Figure 4.7 Inter scores with varying N for 1-Million Song dataset.



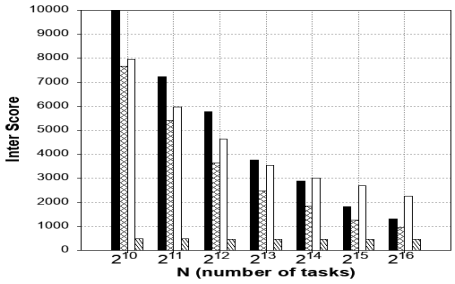
(a) Ap-Min-Inter (Min-Intra)



(b) Ap-Max-Inter (Min-Intra)

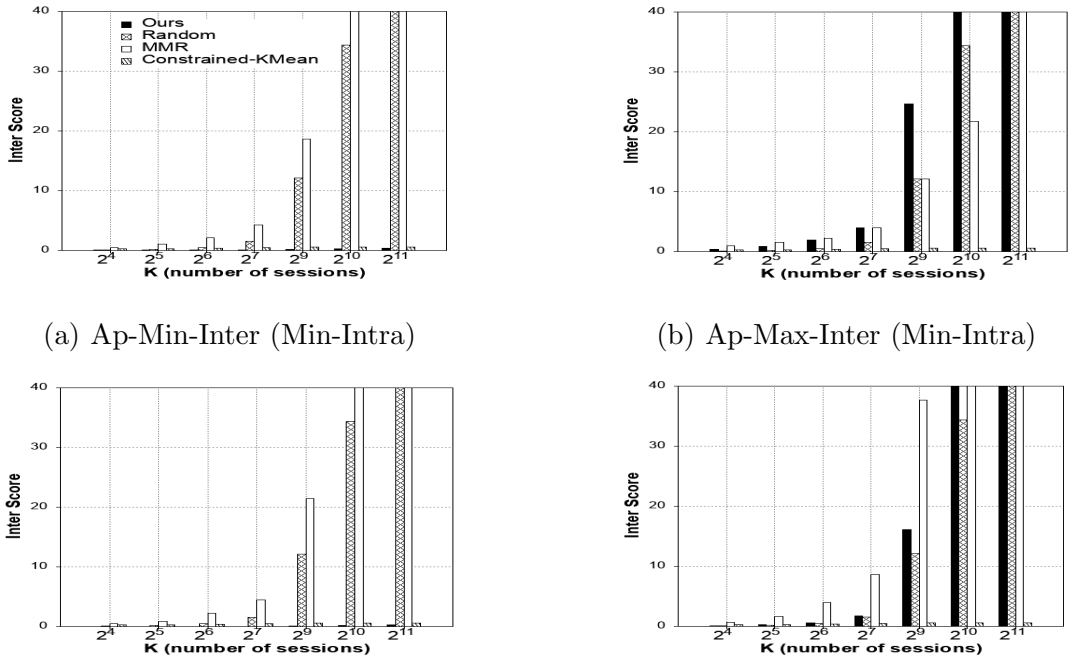


(c) Ap-Min-Inter (Max-Intra)



(d) Ap-Max-Inter (Max-Intra)

Figure 4.8 Inter scores with varying N for synthetic dataset.



(a) Ap-Min-Inter (Min-Intra)

(b) Ap-Max-Inter (Min-Intra)

(c) Ap-Min-Inter (Max-Intra)

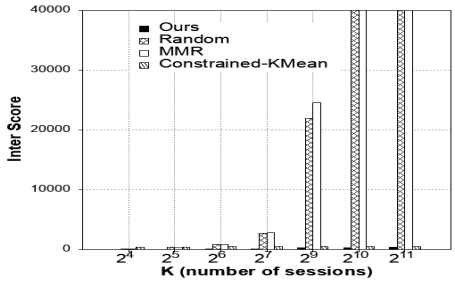
(d) Ap-Max-Inter (Max-Intra)

Figure 4.9 Inter scores with varying k for 1-Million Song dataset.

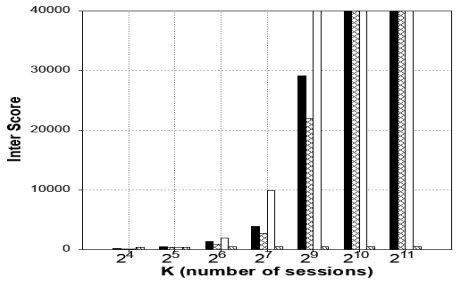
N constant at 2^{13}. We observe Figures 4.9(a)(c) and 4.10(a)(c) that our algorithm performs significantly better than other baselines in minimizing Inter diversity. For Figures 4.9(b)(d) and 4.10(b)(d), our observation is similar to the case of varying N, Baseline-MMR performs slightly better. Overall, Inter diversity increases for all four scenarios as k increases. This is because of the fact that when more sessions are present, it allows for more diversity between each session.

We present approximation factors of Intra in Tables 4.15 and 4.16 and observe a similar trend as to when we vary N. Also, similar to varying N for Intra scores, the approximation factors here also stay close to 1 for our algorithm.

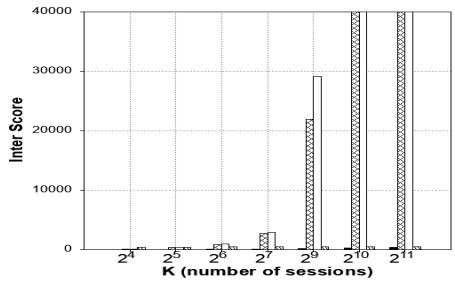
Varying distribution. Figures 4.11 and 4.12 present how our algorithm and other baselines perform as we vary data distributions. We set N to 2^{13} and k to 2^7.



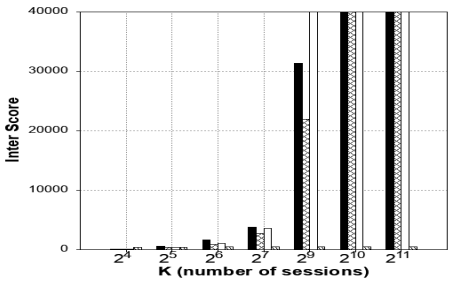
(a) Ap-Min-Inter (Min-Intra)



(b) Ap-Max-Inter (Min-Intra)

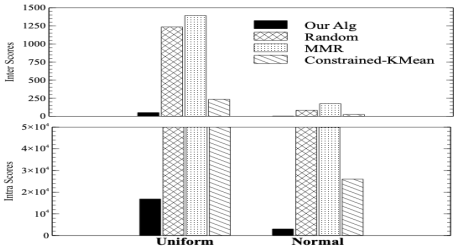


(c) Ap-Min-Inter (Max-Intra)

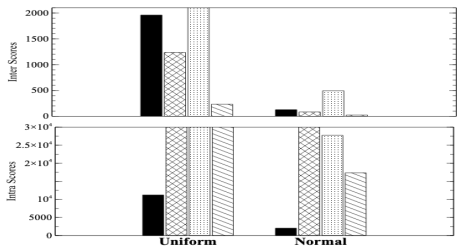


(d) Ap-Max-Inter (Max-Intra)

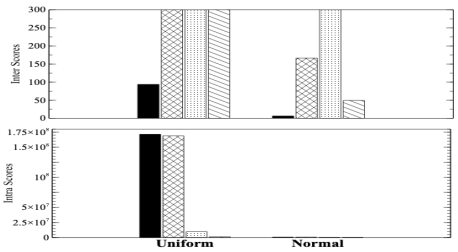
Figure 4.10 Inter scores with varying k for the synthetic dataset.



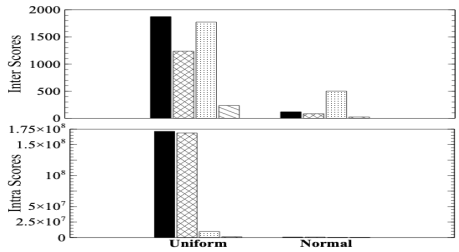
(a) Min-Intra, Min-Inter



(b) Min-Intra, Max-Inter

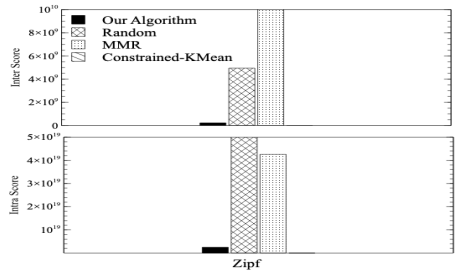


(c) Max-Intra, Min-Inter

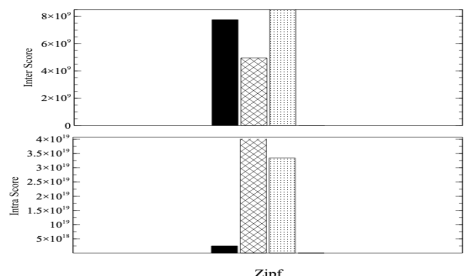


(d) Max-Intra, Max-Inter

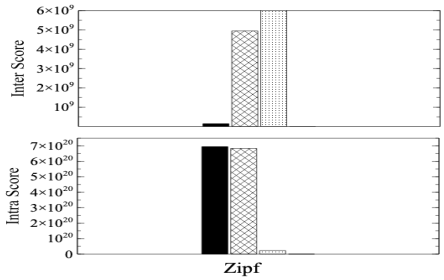
Figure 4.11 Synthetic Data: Inter and Intra scores varying distributions.



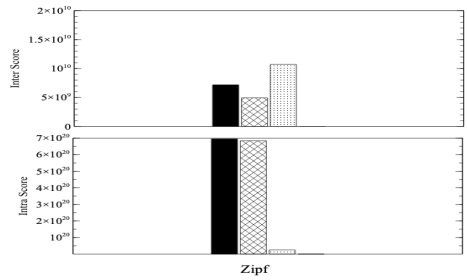
(a) Min-Intra, Min-Inter



(b) Min-Intra, Max-Inter

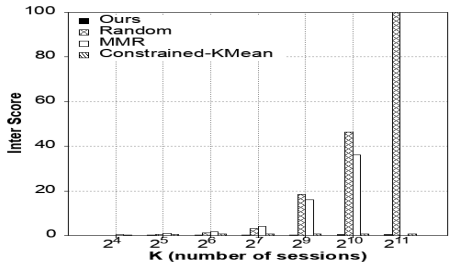


(c) Max-Intra, Min-Inter

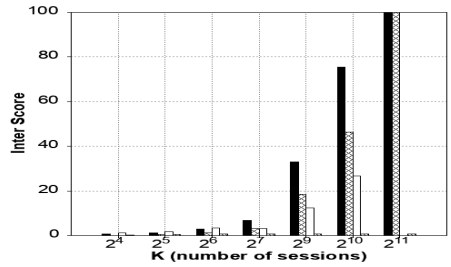


(d) Max-Intra, Max-Inter

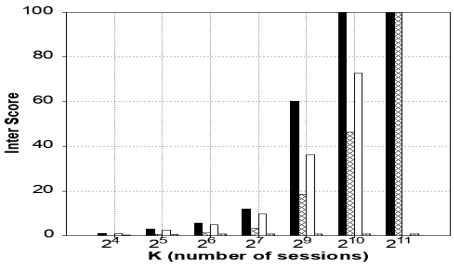
Figure 4.12 Synthetic Data: Zipf distribution.



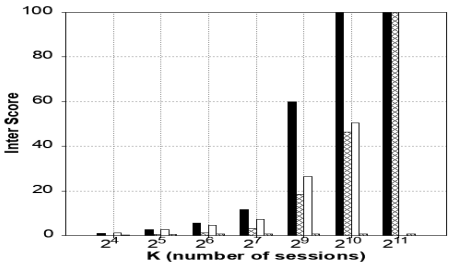
(a) Ap-Min-Inter (Min-Intra)



(b) Ap-Max-Inter (Min-Intra)



(c) Ap-Min-Inter (Max-Intra)



(d) Ap-Max-Inter (Max-Intra)

Figure 4.13 Inter scores with varying k for 1-Million Song dataset for variable length sessions.

Table 4.13 Intra Approximation Factors Varying N on 1-Million Song Dataset

	N	Algorithms			
Min-Intra		**MMR**	**Random**	**Constrained-KMean**	**Ours**
(Minimizing & Maximizing Inter)	**<= 8192**	0.008	6.41E-05	0.165	1
	>8192	0.002	5.42E-05	0.167	1
	N	**Algorithms**			
Max-Intra		**MMR**	**Random**	**Constrained-KMean**	**Ours**
(Minimizing & Maximizing Inter)	**<= 8192**	0.22	0.98	0.0173	0.99
	>8192	0.021	0.92	0.0187	0.99

Considering Intra scores, our algorithm performs the best using Uniform distribution for all four scenarios. However, Normal performs slightly worse at times with our algorithm when we attempt to maximize Intra.

When we compare Inter scores, our algorithm performs best with Uniform distribution. In Figures 4.11(b)(d), Baseline-MMR outperforms our algorithm due to the same reasons mentioned in the varying k and N sections of this paper.

Baseline-Constrained-KMean outperforms our algorithms for minimizing Intra and Inter when using the Zipf distribution.

We also observe that across all four scenarios, Zipf produces scores much larger in magnitude than Normal or Uniform distribution. This is due to the range of values in Zipf, which results in larger Intra and Inter scores. Overall, our algorithms stand out to be the best choice, with its best performance being on Uniform distribution.

4.4.2.5 Variable Length Sessions. We relaxed two limitations of the Fixed Length Sessions problem in this section: a. session lengths vary; b. just a subset of items is recommended. The complexity of the Min-Intra problem remains unchanged, while the NP-hardness of Max-Intra still holds. Finally, Inter problems' NP-hardness remains intact. We vary k between 2^4 and 2^{11} and keep N as its default value for different baselines compared to our algorithms. These experiments incorporate an

Table 4.14 Intra Approximation Factors Varying N on Synthetic Dataset

	N	Algorithms			
		MMR	**Random**	**Constrained-KMean**	**Ours**
Min-Intra (Minimizing & Maximizing Inter)	**<= 8192**	0.006	7.53E-05	0.018	1
	>8192	0.005	6.56E-05	0.0086	1
	N	**Algorithms**			
		MMR	**Random**	**Constrained-KMean**	**Ours**
Max-Intra (Minimizing & Maximizing Inter)	**<= 8192**	0.19	0.94	0.007	0.99
	>8192	0.019	0.99	0.0076	0.99

Table 4.15 Intra Approximation Factors Varying k on 1-Million Song Dataset

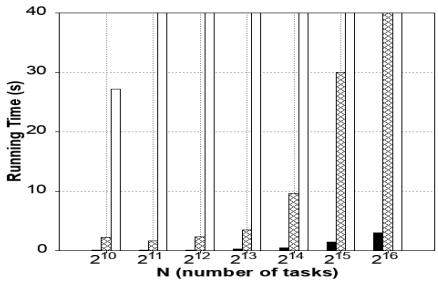
	k	Algorithms			
		MMR	**Random**	**Constrained-KMean**	**Ours**
Min-Intra (Minimizing & Maximizing Inter)	**<= 128**	0.011	0.0021	0.263	1
	>128	0.0012	4.95E-06	0.069	1
	k	**Algorithms**			
		MMR	**Random**	**Constrained-KMean**	**Ours**
Max-Intra (Minimizing & Maximizing Inter)	**<= 128**	0.035	0.92	0.05	0.99
	>128	0.29	0.85	0.0027	0.99

Table 4.16 Intra Approximation Factors Varying k on Synthetic Dataset

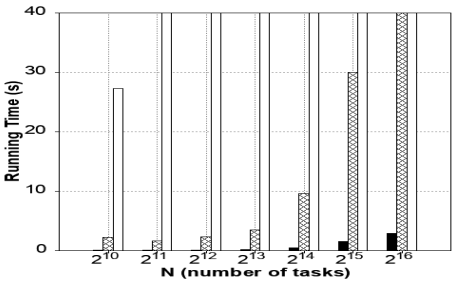
	k	Algorithms			
		MMR	**Random**	**Constrained-KMean**	**Ours**
Min-Intra (Minimizing & Maximizing Inter)	**<= 128**	0.035	0.0013	0.03	1
	>128	0.0008	5.05E-06	0.0045	1
	k	**Algorithms**			
		MMR	**Random**	**Constrained-KMean**	**Ours**
Max-Intra (Minimizing & Maximizing Inter)	**<= 128**	0.055	0.99	0.03	0.99
	>128	0.42	0.85	0.001	0.99

Table 4.17 Intra Approximation Factors Varying k on 1-Million Song Dataset for Variable Length Sessions

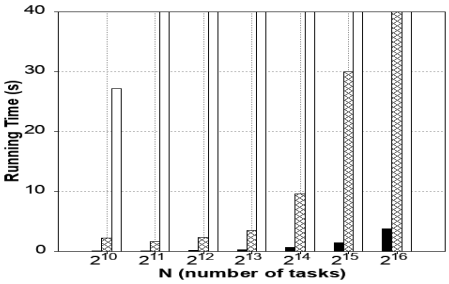
	k	Algorithms			
		MMR	**Random**	**Constrained-KMean**	**Ours**
Min-Intra (Minimizing & Maximizing Inter)	<= 128	0.215	0.0127	0.39	0.725
	>128	0.0485	0.0001	0.058	0.76
	k	**Algorithms**			
		MMR	**Random**	**Constrained-KMean**	**Ours**
Max-Intra (Minimizing & Maximizing Inter)	<= 128	0.435	0.98	0.033	0.714
	>128	0.39	0.78	0.0013	0.79



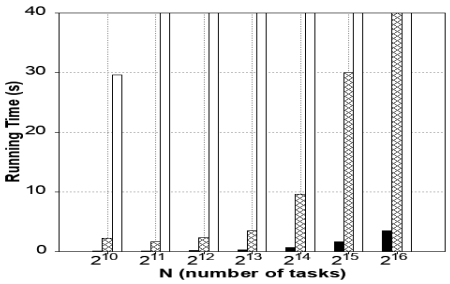
(a) Min-Intra, Min-Inter



(b) Min-Intra, Max-Inter

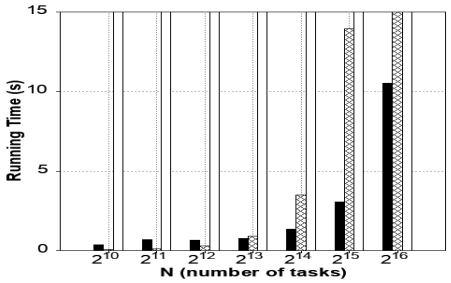


(c) Max-Intra, Min-Inter

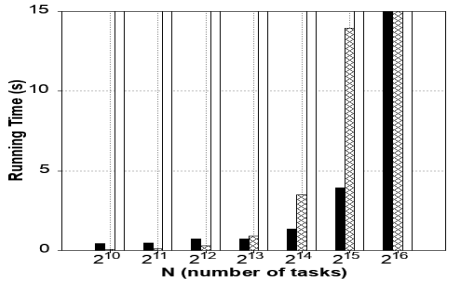


(d) Max-Intra, Max-Inter

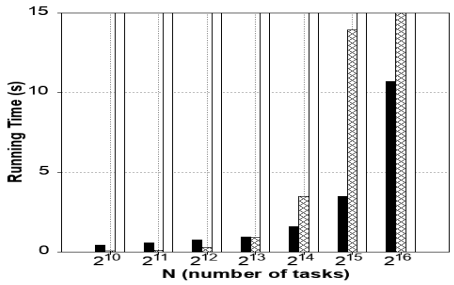
Figure 4.14 Running times varying N for 1-Million Song dataset.



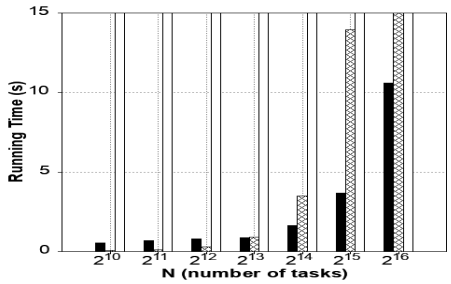
(a) Min-Intra, Min-Inter



(b) Min-Intra, Max-Inter

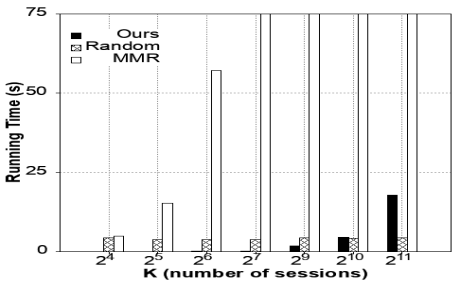


(c) Max-Intra, Min-Inter

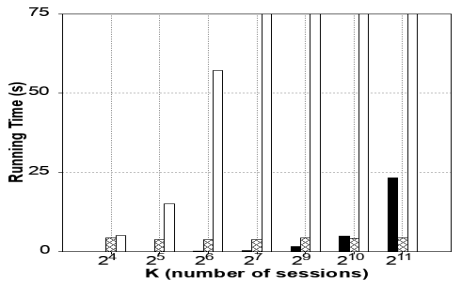


(d) Max-Intra, Max-Inter

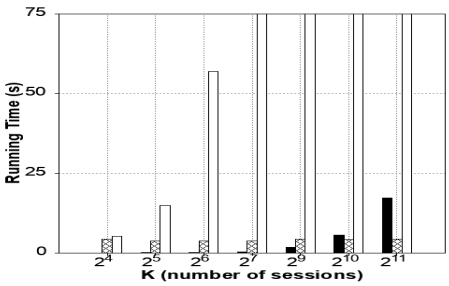
Figure 4.15 Running times varying N for synthetic dataset.



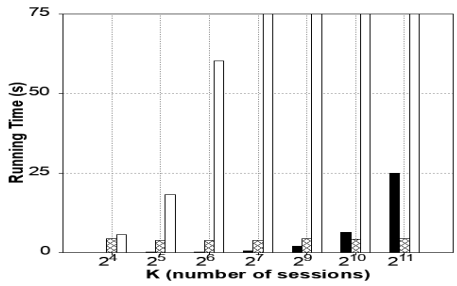
(a) Min-Intra, Min-Inter



(b) Min-Intra, Max-Inter

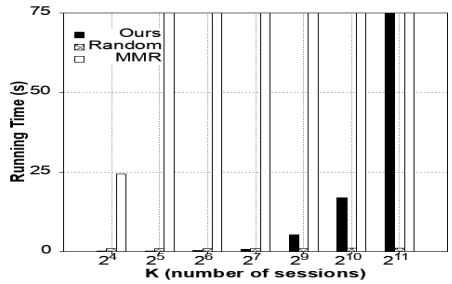


(c) Max-Intra, Min-Inter

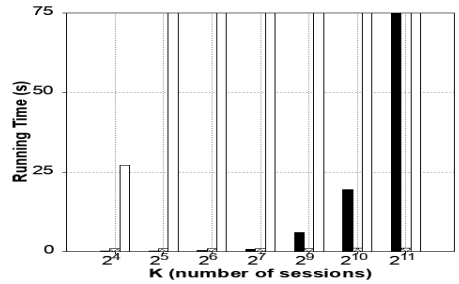


(d) Max-Intra, Max-Inter

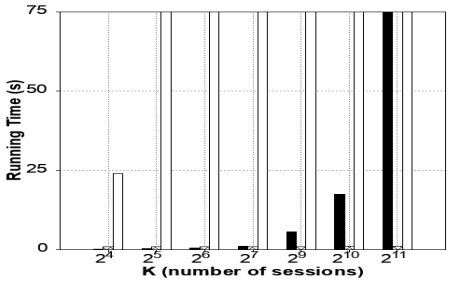
Figure 4.16 Running times varying k for 1-Million Song dataset.



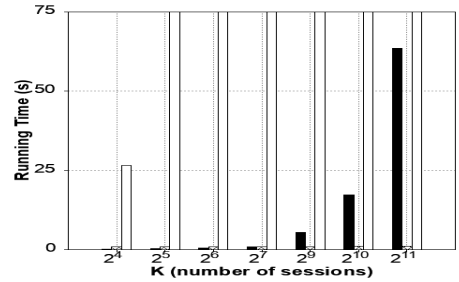
(a) Min-Intra, Min-Inter



(b) Min-Intra, Max-Inter

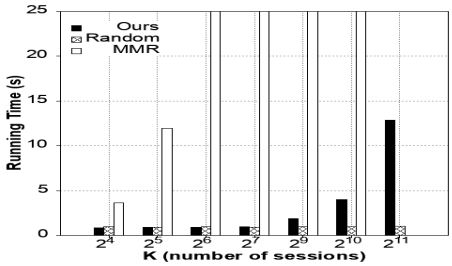


(c) Max-Intra, Min-Inter

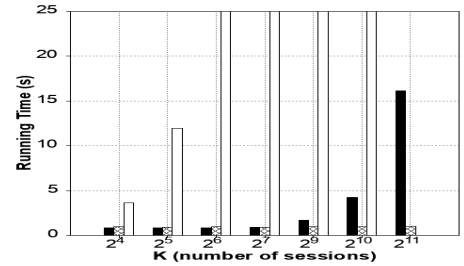


(d) Max-Intra, Max-Inter

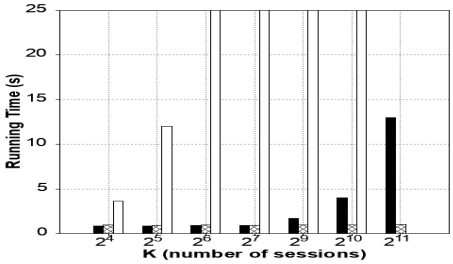
Figure 4.17 Running times varying k for synthetic dataset.



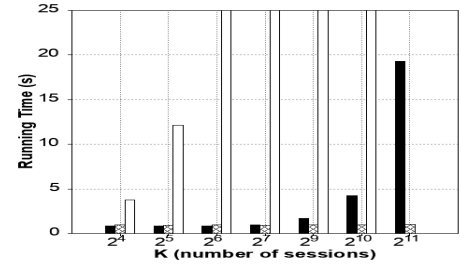
(a) Min-Intra, Min-Inter



(b) Min-Intra, Max-Inter



(c) Max-Intra, Min-Inter



(d) Max-Intra, Max-Inter

Figure 4.18 Running times varying k for 1-Million Song dataset for variable length sessions.

extra input that is generated at random and indicates the length of each session as a list of k values between 2 and l. In Example 4.1, $N = 12$ and $k = 4$, therefore $l = 3$. As a consequence, $[2, 3, 2, 3]$ is the length list containing random integers between 2 and 3, which is our l. As we have 8192 items in our experiments for each k value, averaging the items that are recommended in each scenario yields 4568 items out of 8192.

The results are presented in Figure 4.13 and Table 4.17. When maximizing Intra, Random produces an approximation factor close to 1 for $k \leq 128$. This is due to the same reason that is explained in the Varying k section that when Intra is maximized, the variance of the sessions is also maximized. Except for the *Alg-Max-Intra, Min-Inter* problem, all of the scenarios follow the same trend of Fixed Length Sessions. In comparison to the other baselines, Variable Length Sessions achieves a higher Intra approximation factor, as shown in Table 4.17.

Since Baseline-MMR could not finish in a reasonable time for the scenario when k is 2^{11}, we leave it blank in Figure 4.13.

4.4.2.6 Scalability Evaluation. Figures 4.14, 4.15, 4.16, and 4.17 compare the running time of the three algorithms for 1-Million Song and synthetic dataset. The running time of Baseline-Constrained-KMean was not included in these figures since some scenarios took many days to complete. Naturally, as N increases, the running time of our algorithms increases. We also observe that as we vary N with $k = 2^7$, our algorithms are the fastest in all diversity scenarios.

In Figures 4.16 and 4.17 , we vary k and set N to 2^{13}. We observe that our algorithms scale very well but are sometimes slightly slower than Random. This is unsurprising, as Random does not even have to do much work to generate sessions (recall that however, it performs poorly qualitatively). However, we observe that our algorithm is consistently faster with increasing values of k. The scalability evaluation

plots for relaxed problem experiments closely resemble those of the original problem for the 1-Million Song dataset, as seen in Figure 4.18. Overall, we find that our algorithms are highly scalable and produce results within a few seconds for very large values of N and k, while some of the baselines take hours to complete.

4.5 Related Work

Applications Diversity has been extensively studied in recommendation and search applications [2, 12, 42, 60, 74, 75, 100, 109, 110, 112, 134, 136, 140, 142, 143, 146], to return items that are relevant as well as cover full range of users interests. The goal is to achieve a compromise between relevance and result heterogeneity. Existing works [72, 135] have also acknowledged the need for diversity and sequence-based modeling in different recommendation applications. Recent works in crowdsourcing [61, 106] have demonstrated the importance of diversity in task recommendation. Task diversity is grounded in organization theories and has been shown to impact the motivation of the workers [38]. Amer-Yahia et al. [11] propose the notion of composite tasks (CT), a set of similar tasks that match workers' profiles, and comply with their desired reward and task arrival rate. Their experiments show that diverse CTs contribute to improving outcome quality. Recent work has studied Intra and Inter-table influence in web table matching [61] involving the crowd. Even though completing similar tasks lead to faster completion time [51], but such composition lead to fatigue and boredom, and task abandonment [48, 71, 73]. Aipe and Gadiraju [7] empirically observe that workers who perform similar tasks achieve higher accuracy and faster task completion time compared to workers who worked on diverse tasks. However, they find that these workers experience fatigue the most. Alsayasneh et al. integrate the concept of diversity in composite tasks and empirically find a positive effect of diversity in outcome quality [8]. Stratigi et al., 2020 [124] investigated a sequential group recommender that is aware of the group's previous interactions with the system by

adding the concept of satisfaction, which characterizes how relevant the recommended items are to each group member.

For all of these applications, diversity is studied set-based or sequence-based only.

These applications call for a deeper examination of diversity and a powerful framework to capture its variants, which is our focus here.

Set and Sequence Diversities Existing works on diversification could be classified as set-based only [2, 60, 100, 109, 110, 134] or sequence-based only [12, 42, 74, 88, 146]. As an example, Ziegler et al., 2005 [146] study sequence-based diversity which is defined as the diversity of any permutation of the items. Another example is [12], in which taxonomies are used to sample search results to reduce homogeneity. In [2], Abbar et al., 2013 proposed an algorithm with a provable approximation factor to find relevant and diverse news articles. In the database context, Chen and Li [42] propose to post-process structured query results, organizing them in a decision tree for easier navigation. In [14, 79] the notion of diversity is used in the results of queries to produce the closest results such that each answer is different from the rest. In recommender systems, results are typically post-processed using pair-wise item similarity to generate a list that achieves a balance between relevance and diversity. For example, in [56], recommendation diversity was formulated as a set-coverage problem. By distinguishing between item and user diversity and focusing on various definitions of each, [88] investigated a diversity-aware recommender system for a single user or a group of users.

To the best of our knowledge, existing works have focused on achieving diversity in a single set. We solve set-based and sequence-based diversities in tandem and develop algorithms with guarantees.

4.6 Conclusion

We initiate the study of a formal and algorithmic framework to address diversity for s sequence of sets that has natural recommendation applications (from song playlists to task recommendations in crowdsourcing). The combination of Intra and Inter session diversities gives rise to four bi-objective optimization problems. We propose algorithms with guarantees. Our extensive empirical evaluation, conducted using human subjects, as well as large-scale real and simulated data, shows the need for diversity to improve user satisfaction and the superiority of our algorithms against multiple baselines.

In addition to theoretical questions, this work opens up interesting directions that are of empirical interest: an immediate extension of our work is to observe users as they consume items and learn how diversity dimensions and their respective definitions could be personalized for different users. Similarly, we are empirically exploring how to choose the preferred diversity dimensions depending on the underlying context for different applications. Finally, an interesting open problem is to understand how time affects underlying contexts and fine-tune diversified recommendations based on that.

In terms of other widely used diversification functions, there exist diversity functions that consider radius (maximum/minimum distance) [69], or sum (sum of distance). One can maximize or minimize these based on the underlying optimization goal. Many of these problems relate to the Facility Allocation Problem [13] and its variants, as well as Graph Partitioning problems [114]. These problems are known to be NP-hard. Our produced greedy solutions could be adapted to solve these variants. However, whether these solutions would be just heuristics or they would accept provable approximation factors would require revisiting and analyzing each of them and that can be studied in future work.

We are also going to study the approximation factors of the proposed algorithms for the Variable Length Sessions in the future.

CHAPTER 5

COOPERATIVE ROUTE PLANNING FRAMEWORK FOR MULTIPLE DISTRIBUTED ASSETS IN MARITIME APPLICATIONS

5.1 Introduction

Planning routes for multiple agents, such as ships submarines, and unmanned aerial/surface/underwater vehicles (UAVs, USVs, UUVs), considering multiple objecti ves, such as fuel, battery usage, time taken, and progress towards a goal, is a complex, but highly-relevant, problem for search and rescue, reconnaissance, and interdiction missions in maritime applications. These problems involve trade-offs among different objectives and require a coordinated search for an object in a very high-dimensional space by multiple geographically distributed searchers (herein referred to as assets or agents). In this vein, automated tools are needed to aid human decision-makers in planning courses of action (COAs).

A canonical formulation of the problem is as follows: we are given a set of distributed assets over a discrete grid, each with a given starting location (described by its (lat, long) value), a respective speed limit, and a sensing radius. The problem is to cooperatively discover an object at an unknown location (described by a (lat, long) value). The assets communicate among themselves periodically after a fixed interval of time or communicate asynchronously by broadcasting their locations when an asset locates the destination. When two assets communicate, they get to know each other's current locations. The goal is to decide a sequence of moves, including waiting at nodes, for each asset (route plan) such that the total fuel consumption by the assets and the maximum time for reaching the mission goal over all assets (Makespan) is minimized while avoiding collisions among themselves. We formalize

the *Route Planning Problem* (**RPP**) in Section 5.2 as a team decision-making problem and describe the challenges in solving it.

Formalizing the **RPP** requires developing a principled model that captures the nuances of simultaneous movement of the assets, along with how that impacts the objectives and satisfaction of constraints. At each step, the **RPP** solution produces the direction of movement for each asset and the corresponding speed (note that a speed of 0 corresponds to the waiting option) while accounting for its sensing radius and the locations of other assets. For distributed assets, since the locations of the other assets are known intermittently, each asset has to *anticipate* the locations and moves of other assets through an appropriate "internal model".

Contribution I: A Distributed RPP Model - We model the **RPP** problem as a Team Discrete Markov Decision Problem (TDMDP) [108], where the states are the nodes in the grid and an asset's action is the decision to transit to one of the neighboring grid points at a particular speed or wait to avoid a collision. For each transition, there exist vector rewards that capture the multiple objectives of the team. Section 5.2 contains further details.

Contribution II: MaMoRL - We propose a *Multi-agent Multi-objective Reinforcement Learning* (MaMoRL) framework for cooperative route planning under constraints. We identify its data-centric challenges (Sub-section 5.2.4). The MaMoRL framework contains *a. Teammate Module* (`TMM`) that captures the probability distribution of the belief of each asset on the locations and actions of other assets at each time step; b. *Action Selection Module* (`ASM`) to enable distributed decision-making by each asset to optimize multiple objectives, while satisfying constraints at each time step; and c. *Learning Module* (`LM`) that allows each asset to incrementally learn the functional approximations to the Q-functions (current best values of state-action pairs) from the environment and incorporating that learning inside the other two modules. Section 5.3 contains further details.

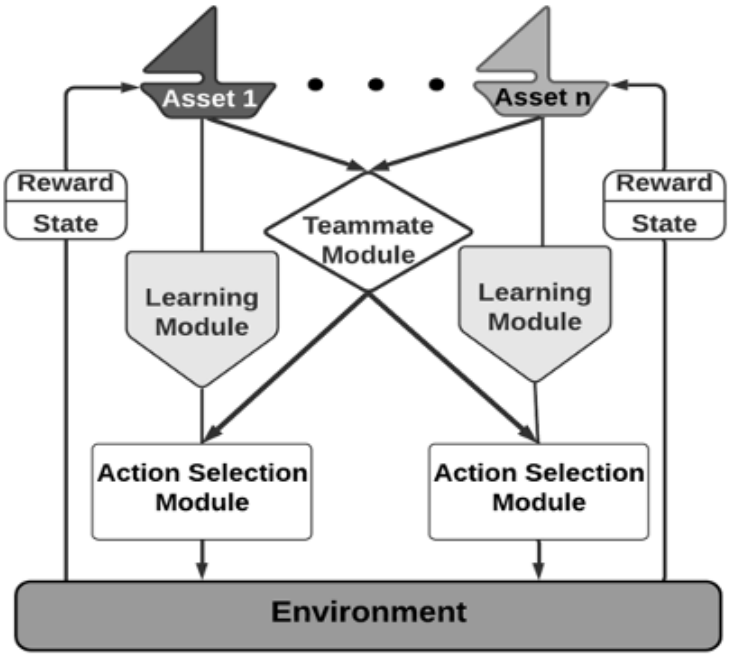


Figure 5.1 MaMoRL framework.

We also realize that solving MaMoRL exactly is infeasible in a realistic maritime setting, because the memory and CPU bottlenecks of `TMM` and `LM` increase exponentially with the increase in the number of assets, the number of nodes, and neighbors of the assets, and the cardinality of the set of speeds an asset can choose from.

We, therefore, develop two function approximation techniques - one using Linear Regression and the other using a Neural Network to effectively *approximate the Teammate Module* (`TMM`) *and the Learning Module* (`LM`) [94]. The Linear Regression based approximate solution, Approx-MaMoRL, is shown to be more effective than its Neural Network based counterpart NN-Approx-MaMoRL. It turns out to be as effective as the exact MaMoRL, while overcoming both the memory and CPU bottlenecks (refer to Section 5.4). The design of Approx-MaMoRL involves extensive feature engineering efforts.

Contribution III: Evaluation and Deployment. We perform an extensive experimental evaluation (Section 5.4) using multiple real-world and synthetic grids

and implement several baselines for comparisons. We identify the computational bottlenecks of MaMoRL (Sub-section 5.4.3) and demonstrate how Approx-MaMoRL overcomes those limitations. We design two variants of Approx-MaMoRL - one with no knowledge of the destination and the other with partial knowledge of the destination. The experimental results demonstrate the superiority of Approx-MaMoRL compared to the baseline solutions with a 95% statistical significance (Sub-section 5.4.4). Our results additionally demonstrate that the policy learned by the Approx-MaMoRL is transferable to various sized grids (Sub-section 5.4.6). Finally, in Sub-section 5.4.7, we describe how we deploy Approx-MaMoRL inside TMPLAR [121], an existing tool for Multi-objective Planning and Asset Routing used by NRL-MMD, Monterey, CA.

5.2 Preliminaries and Problem Definition

5.2.1 Preliminaries

Let N be a set of distributed assets. Each asset operates on a discrete grid $\mathcal{G} =< V, E >$. The weight of each directed edge e ($weight(e)$) between two nodes, $v_p \rightarrow v_q$, denotes the distance between v_p and v_q. The assets explore the grid cooperatively to discover the location of an object, represented as $d(x, y)$. Each asset $i \in N$ is represented as a quintuple:

$$i =< r_i, sp_i, source_i(x, y), cur_i^t(x, y), d_i(x, y) >$$

where r_i is the sensing radius of asset i, sp_i is the maximum speed/ velocity, $source_i(x, y)$ is the starting point, $cur_i^t(x, y)$ is the location of asset i at time t, and $d_i(x, y)$ is the destination from the viewpoint of asset i. All assets have the same destination, which is unknown in the beginning but gets revealed through exploration. Therefore, $d_i(x, y) = d(x, y); \quad \forall i \in N$.

5.2.2 Problem setting and definition

Each asset i is characterized as follows:

- It moves at a speed $\leq sp_i$.
- It observes the grid up to its sensing radius r_i.
- It makes a decision on its next action at each step, i.e., which neighboring node it moves to and at what speed or to wait. Decisions are made only when i is at a node in $\mathcal{G}$.
- Two assets (i, j) exchange their respective locations when they communicate periodically every k time step or when the destination $d(x, y)$ has been found.

Fuel Consumption Model: The fuel consumption of asset i to move from node $v_p \rightarrow v_q$ at speed $sp'_i \leq sp_i$ is $fuel_i(v_p \rightarrow v_q, sp'_i)$, where sp_i is the maximum allowable speed of asset i. The fuel consumption depends on the distance between v_p, v_q, and the speed sp'_i. There exist analytical models [31] for computing fuel consumption as a function of distance traveled and speed. Thus, the fuel consumption of asset i to explore $\mathcal{G}$ can be computed and we denote it by T_{Fuel_i}.

Definition 5.1. *Total fuel consumption F_{total} : The overall fuel consumption by the set of $|N|$ assets while exploring $\mathcal{G}$ to discover $d(x, y)$ is $\sum_{\forall i} T_{Fuel_i}$.*

Time Model: The time taken for asset i to move from node $v_p \rightarrow v_q$ at speed $sp'_i \leq sp_i$ is $\frac{weight(v_p \rightarrow v_q)}{sp'_i}$. Thus, the total time taken by asset i to explore $\mathcal{G}$ can be computed and is represented by T_{Time_i}.

Definition 5.2. *Overall time expended T_{total} : The overall time expended by $|N|$ assets during exploration over $\mathcal{G}$ to discover $d(x, y)$ is the maximum over T_{Time_i}, i.e., $max_{\forall i} T_{Time_i}$.*

Definition 5.3. *Collision: Two assets i, j collide if they are at the same location simultaneously, i.e., $cur_i^t(x, y) = cur_j^t(x, y)$*

Problem 5.1. ***(Route Planning Problem (*RPP*))*** *Plan routes of* $|N|$ *cooperative distributed assets (agents) with respective starting points (asset* i *with* $source_i(x, y)$*) over* $\mathcal{G}$ *to discover an initially unknown destination* $d(x, y)$ *to minimize* F_{total} *and* T_{total}*, while at no point in time during the exploration,* $cur_i^t(x, y) = cur_j^t(x, y), \forall i, j, t$.

5.2.3 Toy running example

Consider a toy example involving two assets (Table 5.1), their respective current positions, and their visited paths after two moves in Figure 5.2. Each asset can sense up to its sensing radius which is based on the distance and represented by double-circled green and blue nodes for Asset1 and Asset2, respectively.

Table 5.1 A Toy Example Using Two Assets

	r_i	sp_i	$source_i(x, y)$	$cur_i^2(x, y)$	$d_i(x, y)$
Asset1	2	3	(0,0)	(0,4)	(4,3)
Asset2	3	2	(8,7)	(5,4)	(4,3)

From starting location $source_1$, Asset1 can go to one of its two neighboring sensed nodes. It also can move at three different speeds, ($sp'_1 = 1, 2, 3 \leq sp_1$) or wait; so the number of possible actions for Asset1 is seven. Evidently, without any further information, it is beneficial to move in the direction that explores more of the not-yet-sensed nodes, as long as collisions are avoided. Asset1 takes action a_0 because more nodes will be sensed from the new position and moves at speed 2 based on Table 5.2. Its position is changed to $cur_1^1(x, y) = (0, 2)$. Similarly, Asset2 moves from $source_2$ to node $(6, 6)$ by taking action a'_0 to sense three additional nodes. Moreover, it chooses speed 2, because that minimizes the average of time and fuel consumption (see Table 5.2).

Table 5.2 Time and Fuel Consumption of the Assets

Asset1				Asset2			
Speed	**Time**	**Fuel**	**Average**	**Speed**	**Time**	**Fuel**	**Average**
1	2	3.7664	2.8832	1	2.24	4.2184	3.2292
2	1	4.2714	**2.6357**	2	1.12	4.7840	**2.9520**
3	0.66	4.7286	2.6943	3	—	—	—

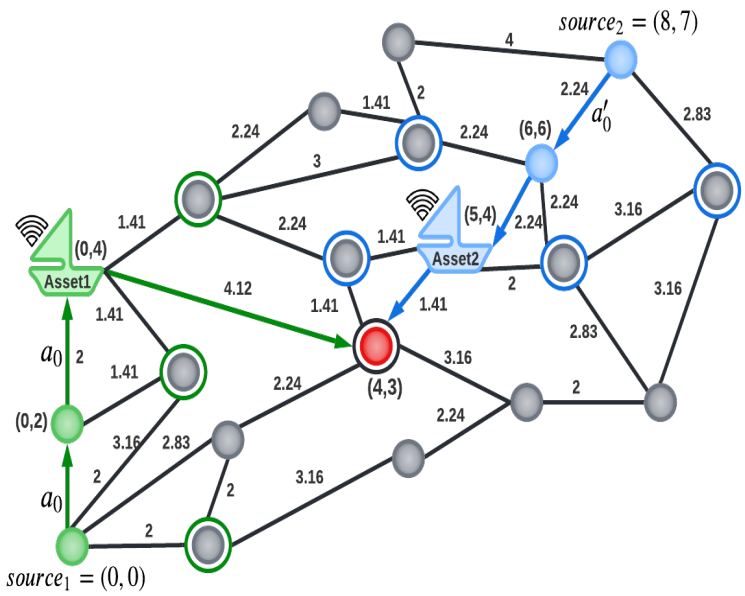


Figure 5.2 Assets' traveled paths and positions after two moves.

5.2.4 Data centric challenges

The proposed work falls under the broad category of data management/engineering for efficient AI. The major data-centric challenges to solve the **RPP** stem from two sources: (a) Model and Reward design; (b) Memory and CPU Bottlenecks.

5.2.4.1 Model and Reward Design. The problem of designing automated plans has been recently explored in several research works [18, 101, 117] that broadly fall under the umbrella of Exploratory Data Analysis (EDA). These efforts investigate

the design of RL agents to discover user groups [117], to generate exploratory sessions for data scientists [18], or to design complex recommendation tasks, such as courses or trip plans [101]. Similar to these problems, we model the **RPP** as a sequential decision-making/control problem, where the fundamental model consists of a state-action pair for which an appropriate reward function needs to be designed. Unlike existing works, a state of the **RPP** corresponds to the current locations of all $|N|$ assets and an action constitutes deciding respective actions of all assets (including waiting). Moreover, the problem requires the consideration of multiple objectives (fuel and time) and the simultaneous planning of routes by multiple distributed assets in a spatial domain with limited communication capabilities while satisfying multiple constraints (avoiding collisions and intermittent communication). Therefore, our proposed model MaMoRL, which involves extensive data engineering for the reward design process is significantly different from existing works. From a technical standpoint, MaMoRL contains a Teammate Module (Sub-section 5.3.2) that captures the belief of each asset on the locations and actions of other assets at each time step, which is unique to this distributed planning problem and has not been studied in the aforementioned prior efforts.

5.2.4.2 Memory and CPU Bottlenecks. As we shall describe in Sub-section 5.3.3, MaMoRL requires storing very large (multiple) P and Q tables in memory, and the sizes of these tables increase exponentially with the number of assets, grid size, and the number of neighbors, and the cardinality of the set of speeds an asset can choose from (refer to Lemmata 5.1, 5.2), causing a memory bottleneck (refer to Table 5.6 for experimental results). We design two function approximation methods (based on Linear Regression and a Neural Network) (Sub-section 5.3.3) that avoid pre-computing and storing these tables. The CPU bottleneck arises from the need to learn effectively in a setting with an exponential number of environmental states and

actions. To the best of our knowledge, some existing works [18, 145] propose Deep Neural Network based RL framework to overcome the large state space problem but do not comprehensively evaluate the scalability challenges. Naturally, the Neural Network based function approximation requires more training data to be effective, which is harder to obtain for the **RPP**, compared to the EDA problem studied in [18].

5.3 Proposed Solution

We present our proposed model, algorithms for `TMM`, `LM` and `ASM`, and describe approximation opportunities.

5.3.1 Modeling

We model the **RPP** problem as a Team Discrete Markov Decision Process (TDMDP) (S, A, R) [108]:

a. The state space $S = |V|^{|N|}$, where $|V|$ is the number of nodes in $\mathcal{G}$. At a given time t, state s_t is the locations of all $|N|$ assets.

$$s_t \leftarrow \{cur_1^t(x, y), cur_2^t(x, y), ..., cur_{|N|}^t(x, y)\}$$

b. A is the set of actions. Given a state s, where each asset i is at its corresponding node v_{pi}, an action a_i corresponds to making decisions to move along $v_{pi} \rightarrow v_{qi}$ at speed $sp_i' \leq sp_i$ or staying at v_{pi}, i.e., $v_{pi} \rightarrow v_{pi}$; Let $a = [a_i]$ denote the actions over all assets. The effect of an action is a state transition T_r, which is assumed to be deterministic, that is, $T_r : S \times E \rightarrow S$, a new state is obtained by taking an action at each state.

c. $R(s_t, a_t, s_{t+1})$ is the reward obtained for transitioning from state s_t to state s_{t+1} by taking action a_t.

5.3.1.1 Reward Design. The process of designing the reward must be guided by the following intuition: (1) proportional to the number of newly explored nodes by the assets, (2) inversely proportional to the time taken, and (3) inversely proportional to the fuel consumption.

1. Exploration Reward: of an action a relates to how much of the unexplored grid an asset can sense and is formulated as

$$r^{a}_{explore} = \frac{\sum_{i=1}^{|N|} Sensed(i)^{a_i}}{D_{max} \times |N|} \tag{5.1}$$

where $Sensed(i)^{a_i}$ is the number of newly sensed nodes by asset i by taking action a_i, normalized by D_{max}, which is the maximum out-degree in $\mathcal{G}$. Using example in Sub-section 5.2.3, the exploration reward of taking the first action is $r^{a_0}_{explore} = \frac{2+3}{5\times 2} = 0.5$

2. Time Reward: of an action a over all assets is the inverse of the maximum time each asset needs as part of a (where $\frac{weight(v_{pi} \rightarrow v_{qi})}{sp'_i}$ is the time asset i needs for action a_i)

$$r^{a}_{time} = \frac{1}{max_{i=1,\ldots,|N|} \frac{weight(v_{pi} \rightarrow v_{qi})}{sp'_i}} \tag{5.2}$$

For the toy example, $r^{a_0}_{time} = \frac{1}{max(1,1.2)} = 0.83$ for the first action of the example in Sub-section 5.2.3.

3. Fuel Reward: of an action a is the inverse of the sum of fuel consumed by all $|N|$ assets.

$$r^{a}_{fuel} = \frac{1}{\sum_{i=1}^{|N|} weight(v_{pi} \rightarrow v_{qi}) \times fuel_i(1, sp'_i)} \tag{5.3}$$

For our implementation, we use the following model [31] to capture fuel per unit distance.

$$fuel_i(1, sp'_i) = 0.2525 \times {sp'_i}^2 + 1.6307 \times sp'_i \tag{5.4}$$

Based on the model, the fuel reward of the initial set of actions in the toy example is $r^{a_0}_{fuel} = \frac{1}{2\times 4.2714 + 2.24\times 4.784} = 0.052$.

Evidently, at state s, an action a corresponds to two things: 1. The assets should move to the neighboring nodes to maximize the Exploration Reward (Equation (5.1)); 2. The speed of each asset is to be chosen to optimize the average of Fuel Reward (Equation (5.3)) and Time Reward (Equation (5.2)).

5.3.2 Algorithm

Our proposed solution, Multi-agent Multi-objective Reinforcement Learning or MaMoRL in short, contains three different interacting modules (Figure 5.1) and is inspired by model-free Reinforcement Learning for multiple agents [144].

5.3.2.1 Teammate Module (`TMM`). Assets are distributed and communicate among themselves intermittently. Consequently, when an asset decides on its next move, it needs to have an internal model that anticipates the current locations and moves of other assets. The teammate module is designed to represent the internal model of other team members. The purpose of this internal model is to produce the probability distribution of actions of the remaining $|N| - 1$ assets. This module is gradually updated as asset i observes the actions of asset j (a_j^*) $\forall j \in N \setminus \{i\}$ for each executable action a_j in state s, using the following formula:

$$P_{i_t}(s, a_j) = \begin{cases} P_{i_{(t-1)}}(s, a_j) + \beta^{T-t+1} \sum\limits_{a_t \in A_j(s) \setminus \{a_j\}} P_{i_{(t-1)}}(s, a_t), & a_j = a_j^* \\ (1 - \beta^{T-t+1}) P_{i_{(t-1)}}(s, a_j), & \text{otherwise} \end{cases} \tag{5.5}$$

Here $\beta \in [0, 1]$ is the learning rate for determining the effect of the previous action, T is the iteration number, A_i and A_j are the sets of possible actions for assets i and j, respectively. Clearly, these P values are to be stored and updated periodically while learning from the environment.

Lemma 5.1. `TMM` *has P tables for (exploration, time, and fuel) rewards, each with size*

$$|P| = (|V|^{|N|} \times |A| \times sp).$$

where $|V|$ is the size of the grid, $|A|$ is the number of actions, sp is the max speed and $|N|$ is the number of assets.

At the initial state, Asset1 and Asset2 take actions a_0 and a'_0, respectively, thus state s_1 is obtained. At that time, for the exploration reward, Asset1 updates the $P_1(s_0, a')$ values in the system for all Asset2's actions a' in the action set using Equation (5.5) as follows:

$$\begin{aligned} P_{1_1}(s_0, a'_j) &= (1 - 0.3^{(3-1+1)} P_{1_0}(s_0, a_0)) \\ &= (1 - 0.3^3) \times 0.2 = 0.1946, \quad \forall a'_j \in A_2(s_0) \setminus \{a'_0\} \\ P_{1_1}(s_0, a'_0) &= 0.2216 \end{aligned}$$

We consider $T = 3$ and $\beta = 0.3$ in our example. Asset2 also does the same for updating P values of Asset1's actions.

5.3.2.2 Learning Module (`LM`). After the transition

$$(s, a_1, ..., a_i, ..., a_{|N|}) \rightarrow (s', r),$$

asset i will update its Q-function table using the following equation:

$$\begin{aligned} Q(s, a_1, ..., a_i, ..., a_{|N|}) &= (1 - \alpha) Q(s, a_1, ..., a_i, ..., a_{|N|}) \\ &\quad + \alpha(r + \gamma \max_{a' \in A_i} Q(s, a'_1, ..., a'_i, ..., a'_{|N|})) \end{aligned} \tag{5.6}$$

where $a'_j = \underset{b \in A_j}{\text{argmax}}\ P_i(s', b); \quad j = 1, ..., i-1, i+1, ..., |N|$

Clearly, LM must leverage TMM, and update it periodically.

Lemma 5.2. *There exist different Q tables, one for each reward in* LM*, and each of size*

$$|Q| = (|V| \times |A| \times sp)^{|N|}.$$

In Example 5.2.3, the Q values are initialized as

$$Q(s, a, a') = \frac{1}{|A(s)||A'(s)|} = 0.0286, \quad \forall a \in A, \forall a' \in A', \forall s \in S$$

Here, we take $\alpha = 0.9$ and $\gamma = 0.8$. After Asset1 and Asset2 take actions a_0 and a'_0, respectively, the new state s_1 and exploration reward $r = \frac{2+3}{5 \times 2} = 0.5$ are obtained. Then, at $t = 1$, Asset1 will update its Q value for actions a_0 and a'_0 using Equation (5.6) as follows:

$$Q(s, a_0, a'_0) = (1 - 0.9) \times 0.0286 + 0.9(0.5 + 0.8 * 0.0286) = 0.47$$

5.3.2.3 Action Selection Module (ASM). Asset i uses a greedy policy for selecting the next action using the following equation:

$$a^*_i = \underset{a_i \in A_i}{\text{argmax}}\ V(a_i | A^*) \tag{5.7}$$

where $A^* = \{a^*_1, ..., a^*_{i-1}, a^*_{i+1}, .., a^*_{|N|}\}$ and $V(a_i|A^*)$ is the conditional expectation of an action given by:

$$V(a_i|A^*) = \begin{cases} \sum\limits_{\forall j \neq i} P(s, a_j) Q(s, a^*_1, ..., a_i, ..., a^*_{|N|}), & t \leq T \\ (\underset{\forall j \neq i}{\text{argmax}}\ P(s, a^*_j)) Q(s, a^*_1, ..., a_i, ..., a^*_{|N|}), & t > T \end{cases} \tag{5.8}$$

Here, T represents an iteration threshold. In order to see which action Asset1 is going to take in state s_1 (Figure 5.2) in Example 5.2.3, we calculate the conditional expectation for all possible actions that Asset1 can take in the current state using the first formula in Equation (5.8) considering $T = 3$ and $t = 1$:

$$V_1(a_i|\{a'_j\}) = 4 \times (0.1946 \times 0.0286) + 0.2216 \times 0.0286 = 0.0286$$

$$V_1(a_0|\{a'_j\}) = 4 \times (0.1946 \times 0.0286) + 0.2216 \times 0.47 = 0.1264$$

For $i = 1, ..., 6$ and $j = 0, ..., 4$. Thus, Asset1 chooses action a_0 for the next step, and state s_2 is obtained (see Figure 5.2).

5.3.3 Function approximation

It is infeasible to compute MaMoRL exactly in a realistic setting simply because of the exponential size of the P and Q tables (Refer to Lemmas 5.1 and 5.2). Indeed, the computational bottlenecks lie in exactly computing or traversing `TMM` *and* `LM`. Therefore, we study function approximations for `TMM` *and* `LM`. The approximate solution, Approx-MaMoRL, is designed with extensive feature engineering efforts to effectively approximate `TMM` *and* `LM` without actually building it, and the assets make decisions based on that approximation. It is computationally infeasible to produce an exact solution of MaMoRL in real-world settings.

5.3.3.1 Function approximation for `TMM`.

1. Linear Regression : As opposed to computing $P_{i_t}(s, a_j)$, the action of each teammate j at state s and time t by asset i is approximated as a linear function [35, 128] $\hat{f}_{i,a_j,s}$, given by

$$\hat{f}_{i,a_j,s} = \omega_1 degree(v_j, s) + \omega_2 \theta(v_j, s) + \omega_3 \alpha(a_j, s) + \omega_4 \beta(a_j, d, s) + \omega_5 (sp_j, s) \tag{5.9}$$

Where:

1. v_j = latest location of asset j,
2. $\theta(v_j, s) = 1$ if there is another asset within m hops, else 0,
3. $\alpha(a_j, s)$ is 1, if a_j leads to unsensed nodes; else 0,
4. $\beta(a_j, d, s) = 1$, if a_j leads to d; else 0,
5. sp_j = speed of asset j,
6. $\omega_l \in [0, 1], \quad l = 1, ..., 5$ are weights of the features.

The function approximation considers features that are useful before the goal is discovered (all features excluding β), and ones that are only useful afterward (all excluding α). β in Equation (5.9) is designed for the latter purpose.

Training: In the absence of historical data, we obtain a sample of the original ($P_{i_t}(s, a_j)$ values) coming from MaMoRL to approximate `TMM`. The features are hand-crafted and the goal here is to learn the weights ω_l that minimize the following error function:

$$Minimize \sum_{\forall i, a_j, s} [f_{i,a_j,s} - \hat{f}_{i,a_j,s}]^2 \tag{5.10}$$

Route Planning: During the actual route planning, at a given state s by asset i, an action is simply considered to be the action a_j for asset j that has the highest $P_{i_t}(s, a_j)$ value per Equation (5.9).

2. Neural Network: Using the same training instance as that of Linear Regression, a Neural Network is trained by collecting the P values and the concomitant feature values for each possible action-state pair. The trained Neural Network model is then used to predict the P values for each possible action in order to predict the action a_j with the highest $P_{i_t}(s, a_j)$ in route planning.

5.3.3.2 Function approximation of LM.

1. Linear Regression : Similarly, as opposed to storing the entire Q table, asset i learns a reward function as a linear combination of features:

$$\begin{aligned}\hat{r}_{i,a_i,s} = \omega_1 degree(v_i, s) + \omega_2 \theta(v_j, s) + \omega_3 \alpha(a_i, s) \\ + \omega_4 \beta(a_i, d, s) + \omega_5 (sp_i, s) + \omega_6 sp'_i\end{aligned} \tag{5.11}$$

Here, sp'_i is the speed of asset i that causes a collision and the remaining features are as described in Equation (5.9).

Training: Similar to the function Approximation for the TMM, we sample from the training grid to learn ω_l by minimizing the least squares objective function:

$$Minimize \sum_{\forall i, a_i, s} [r_{i,a_i,s} - \hat{r}_{i,a_i,s}]^2 \tag{5.12}$$

Route Planning: During route planing, the reward for each action a_i is calculated using Equation (5.11) and the action resulting in the highest $\hat{r}_{i,a_i,s}$ is chosen by asset i at state s.

2. Neural Network: Using the same training data as that of function approximation using Linear Regression of LM, a Neural Network is also trained.

5.4 Experimental Evaluation

Most of the experiments are run on OS Big Sur with 2.4 GHz Quad-Core Intel Core i5 Processor and 16 GB RAM. Experiments in Table 5.6 are run on a 64-bit OS server with 3.5 GHz 11th Gen Intel Core i9 and 128 GB RAM. Our code and data are available on GitHub.[1] All results are presented as an average of 10 runs.

5.4.1 Experimental setup

5.4.1.1 Datasets. I. Real World Data. A discrete grid of the entire world is produced by taking high-resolution shoreline data from the GSHHG data set and generating a mesh using Gmsh over the world's oceans [67] [139]. In this graph, each node has an out-degree of at most six. To represent the greater amount of navigational adjustments necessary near land, the mesh is generated such that regions closer to coastlines have a higher resolution than those in the middle of the ocean. This grid is then split into three datasets of increasing resolution: the Caribbean Grid, the North America Shore Grid, and the Atlantic Grid (refer to Table 5.3).

Table 5.3 Datasets Description

Datasets	**Region**	$\|V\|$	$\|E\|$
1	Caribbean Grid	710	1684
2	North America Shore Grid	3291	7811
3	Atlantic Grid	14655	35061

II. Synthetic Data. We use the NetworkX library in Python to generate grids by varying one of the three parameters at a time. (a) number of nodes; (b) number of edges; (c) max out-degree.

[1] https://github.com/RoutePlanningProblem/MaMoRL

5.4.1.2 Implemented Algorithms. We implement Approx-MaMoRL, NN-Approx-MaMoRL, MaMoRL, and compare them with several baselines. Additionally, we implement MaMoRL with partial knowledge, which works as follows:

1. **MaMoRL with partial knowledge:** We assume that under *partial knowledge*, the destination is inside a specified region (described by a bounding box of lat, long) that the assets are aware of, but the exact location (lat, long) is unknown. For that, all assets make use of Dijkstra's Shortest Path [52] algorithm to find the shortest path to the boundary of the region containing the destination. Then the assets leverage MaMoRL solutions inside the region to reach the destination.

2. **Baseline-1:** Assets plan their routes *one-by-one in a non-simulta neous round-robin fashion.* The reward functions are identical to that of the ones described in Sub-section 5.3.1. The baseline is likely to require less fuel for the assets to reach the destination (i.e., smaller F_{total}) because of their long wait at the nodes and no unnecessary moves at the expense of taking a longer time, thereby giving rise to a larger T_{total} value.

3. **Baseline-2:** In this RL-based implementation, assets use the same reward functions as before but plan their routes independently, without taking into account the actions of others. This fully distributed planning is akin to the ALOHA protocol and is prone to collisions.

4. **Random Walk-Baseline:** The implementation designs a random walk based solution, where an action in each step and the assets' speed are decided randomly from a uniform probability mass function (for actions) and density (for speed).

Experimentation Goals. Our effort attempts to answer the following questions:

Q1. How function approximation methods Approx-MaMoRL and NN-Approx-MaMoRL compare with each other?

Q2. What are the bottlenecks of implementing the proposed solutions and the baselines?

Q3. How effective Approx-MaMoRL is with or without partial knowledge w.r.t. different baselines and parameters?

Q4. The Accuracy vs. Speed Tradeoff of Approx-MaMoRL.

Table 5.4 Default Parameters Values

Parameters	#Nodes($\|V\|$)	#Edges($\|E\|$)	#Neighbors(D_{max})	#Assets($\|N\|$)	Speed of Assets(sp)	#Episodes(T_B)	Communication Frequency(k)
Default Value	400	846	9	6	5	10	3

Q5. How effective is Approx-MaMoRL in Transfer Learning?

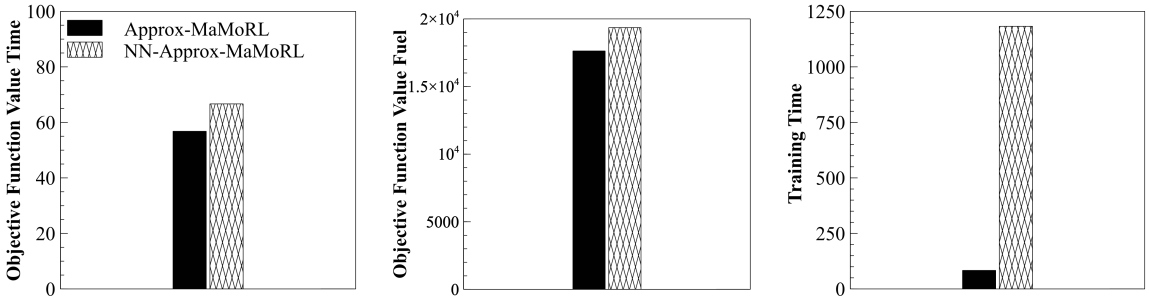
Table 5.4 contains default values for the parameters that are part of MaMoRL. We present T_{total}, F_{total}, relative improvement in objective function values (defined in Sub-section 5.4.4), memory usage, and running time of the corresponding solution for evaluation purposes. Comparison results are presented with paired t-test [76] with 95% statistical significance.

5.4.1.3 Summary of Results. Our first and foremost observation is that the Linear Regression based function approximation Approx-MaMoRL requires less training time (15× faster) and less training data and is more effective than its Neural Network counterpart NN-Approx-MaMoRL (Refer to Sub-section 5.4.2). Hence the rest of our results primarily focus on Approx-MaMoRL. Consistent with our theoretical analysis, in Sub-section 5.4.3, we demonstrate the memory and CPU bottlenecks of MaMoRL and compare that with Approx-MaMoRL along with other baselines (Table 5.6). These results demonstrate that MaMoRL is not a practical solution beyond a small grid with only two assets, whereas Approx-MaMoRL produces reasonable performance to the exact model while being an order of magnitude faster. Baseline-2 results in asset collisions for more than 97% of the runs, due to its unique design choice, making it infeasible in practice. Next, from extensive synthetic data experiments, we observe in Sub-section 5.4.4 that the proposed solution Approx-MaMoRL **with or without partial knowledge** is effective and robust to problem settings. We also observe that Approx-MaMoRL returns the Pareto Front which is better at optimizing both fuel and time when compared to the baselines. Then, in Sub-section 5.4.5, we explore the trade-off between training time and

accuracy of Approx-MaMoRL. Our results demonstrate that Approx-MaMoRL is effective as a real-world solution, making substantial improvement on T_{total} with a 95% statistical significance with a moderate compromise on F_{total}. In Sub-ssection 5.4.6, we demonstrate that our proposed model is suitable for transfer learning.

5.4.2 Function approximation methods

We compare Approx-MaMoRL with NN-Approx-MaMoRL from two standpoints - training time, and objective function values. The training data is obtained from MaMoRL on a small grid (50 nodes, 93 edges) for 2 assets. Additional information is presented in Table 5.5. These results demonstrate that, unsurprisingly, for the same amount of training data, Approx-MaMoRL is more effective than NN-Approx-MaMoRL and is 15x faster in training time (Figure 5.3). In fact, the Neural Network based approximation requires a large amount of training data to be effective which is hard to obtain for our problem since MaMoRL could only be run on small instances because of its memory and CPU bottlenecks. For the remainder of the chapter, we therefore primarily focus on Approx-MaMoRL.



(a) Objective time (b) Objective fuel (c) Training time

Figure 5.3 Approx-MaMoRL vs. NN-Approx-MaMoRL.

Table 5.5 Neural Network Parameters Setting

Parameters	# Layers	# Nodes		Activation Function		Batch Size	# Epoch
		Layer 1	Layer 2	Layer 1	Layer 2		
Default Value	2	5	1	ReLU	Linear	1000	10000

5.4.3 Bottlenecks of the implemented solutions

We compare Approx-MaMoRL with the exact model (MaMoRL) and the baselines. We present the objective function values as well as CPU time and memory usage (instances that could not be run are represented as N/A in Table 5.6). Consistent with our theoretical analysis, Table 5.6 shows that MaMoRL suffers from significant memory and CPU bottlenecks. These bottlenecks are fully averted in Approx-MaMoRL through a "lightweight" function approximation, whereas, the objective function values of Approx-MaMoRL are very close to their exact counterpart in MaMoRL. Unsurprisingly, Baseline-2 results in a majority of collisions (more than 97%). These results demonstrate that Approx-MaMoRL is indeed a suitable alternative to the intractable exact solution.

Table 5.6 Comparison Among Implemented Algorithms

Scenario	Algorithm	T_{total}	F_{total}	CPU Time	Memory Usage
$\|V\| = 704$, $\|N\| = 2$, $D_{max} = 7$	MaMoRL	**N/A**	**N/A**	**N/A**	**205 GB**
	Approx-MaMoRL	158.85	46482.9	0.8 min	1056 B
	Approx-MaMoRL with Partial Knowledge	177.94	49353.8	0.9 min	1056 B
	Baseline-1	255.5	38912.75	0.8 min	576 B
	Baseline-2	N/A	N/A	0.6 min	576 B
	Random Walk-Baseline	995.3	54391070.5	N/A	N/A
$\|V\| = 400$, $\|N\| = 3$, $D_{max} = 9$	MaMoRL	**N/A**	**N/A**	**N/A**	**17000 TB**
	Approx-MaMoRL	87.8	28891.4	1.9 min	2304 B
	Approx-MaMoRL with Partial Knowledge	69.74	13338.7	1.9 min	2304 B
	Baseline-1	128.2	19528.7	1.8 min	864 B
	Baseline-2	N/A	N/A	0.95 min	576 B
	Random Walk-Baseline	153.9	987406.5	N/A	N/A
$\|V\| = 400$, $\|N\| = 2$, $D_{max} = 6$	MaMoRL	**317.7**	**80898.4**	**208 min**	**38.5 GB**
	Approx-MaMoRL	508.1	126919.5	0.8 min	1056 B
	Approx-MaMoRL with Partial Knowledge	623.3	163528.6	0.85 min	1056 B
	Baseline-1	746.3	113666.9	0.8 min	576 B
	Baseline-2	N/A	N/A	0.6 min	576 B
	Random Walk-Baseline	2442.55	58416741.8	N/A	N/A
$\|V\| = 200$, $\|N\| = 2$, $D_{max} = 9$	MaMoRL	**36.3**	**8306.1**	**57 min**	**40 GB**
	Approx-MaMoRL	48.9	12907.5	0.8 min	1056 B
	Approx-MaMoRL with Partial Knowledge	48.95	12139.9	0.8 min	1056 B
	Baseline-1	75.4	11479.6	0.8 min	576 B
	Baseline-2	N/A	N/A	0.6 min	576 B
	Random Walk-Baseline	115.1	460182.9	N/A	N/A

5.4.4 Effectiveness of Approx-MaMoRL

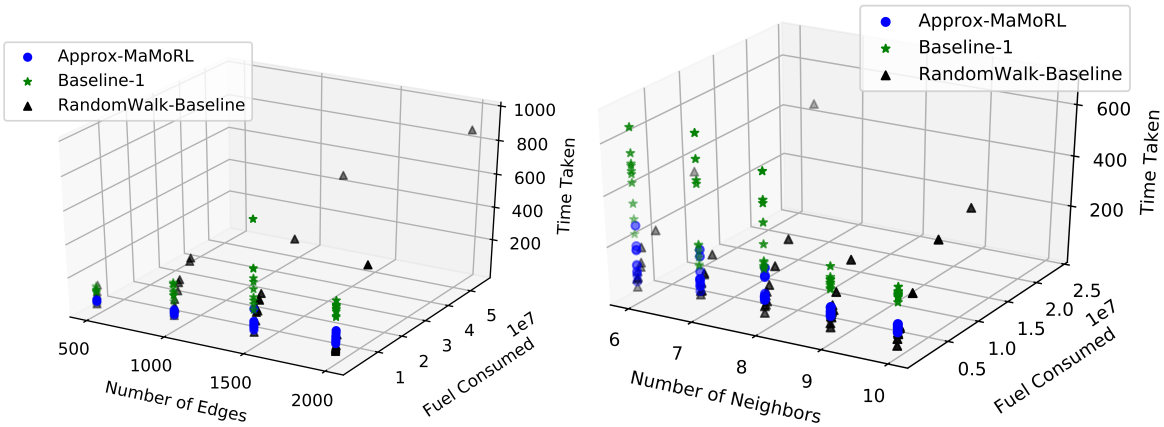
In these experiments, we vary a large number of parameters systematically that are part of our proposed solution (see Table 5.4). Figure 5.4 shows that our proposed framework is capable of returning Pareto-front based on both the objectives F_{total} and T_{total}, where Approx-MaMoRL convincingly outperforms the baselines. Additionally, we measure the percentage of relative improvement ($RI(.)$) of F_{total} and T_{total} for

Approx-MaMoRL with or without partial knowledge w.r.t other baselines using the following measure.

$$RI(Obj_{total}) = \frac{Obj_{total}^{Baseline} - Obj_{total}^{Approx-MaMoRL}}{Obj_{total}^{Baseline}} \times 100$$

As demonstrated in Figure 5.5(a) with an increase in the number of nodes, Approx-MaMoRL has around 60% relative improvement for Time w.r.t other baselines. However, it also has a negative improvement for Fuel w.r.t Baseline-1 which is understandable. A similar observation holds when the number of edges is increased in Figure 5.5(b). Figures 5.5(c) and 5.6(e) show that having more neighbors or moving assets at different speeds does not change the % relative improvement drastically. By increasing the number of assets in Figure 5.5(d), the % relative improvement for Time is positive while it is negative for Fuel. According to Figure 5.5(f), assets take better actions leading to an increase in relative improvement percentage for both objective values, when they are trained more. When assets communicate more often, the objective values improve in general, as a result, % relative improvement of Approx-MaMoRL w.r.t other baselines does not increase as shown in Figure 5.5(g). Figure 5.6 shows Approx-MaMoRL with partial knowledge behaves in a similar way.

Baseline-2 results are omitted in Figures 5.5 and 5.6, since it results in asset collisions for 97% of the runs.



(a) Varying number of edges

(b) Varying number of neighbors

Figure 5.4 Pareto Front of F_{total} and T_{total}.

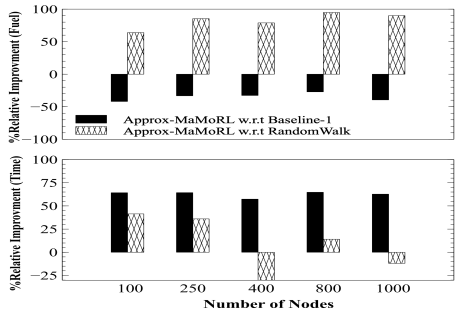
Overall, these figures demonstrate the suitability and superiority of Approx-MaMoRL as a real-world solution.

5.4.5 Accuracy and speed trade off

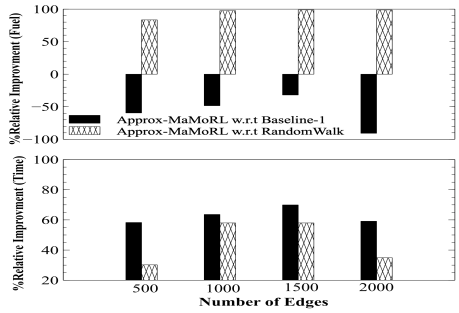
In these experiments, we measure the route planning time of our model and the baselines as seen in Figure 5.8. We do not include the Random Walk-Baseline as it fails to exhibit consistent behavior in optimizing F_{total} and T_{total}. We consistently observe that Approx-MaMoRL convincingly outperforms Baseline-1 in running time, especially for larger problem settings. These Figures 5.8(a-g) demonstrate that Approx-MaMoRL is an effective and realistic model to deploy inside TMPLAR.

5.4.6 Transfer learning

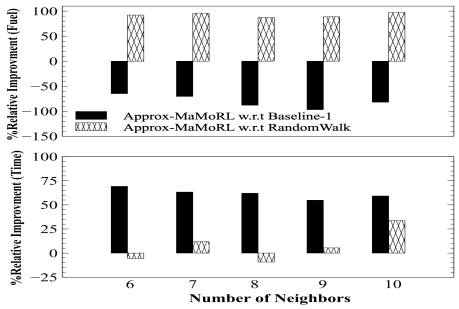
We present the effectiveness of Approx-MaMoRL in transfer learning. We learn a policy on the Caribbean Grid and use it for navigating the assets in the North America Shore Grid and vice versa. Figure 5.7 presents these results that corroborate that Approx-MaMoRL is highly effective in transfer learning, i.e., T_{total} and F_{total} of the transferred models are close to those calculated in the original grid.



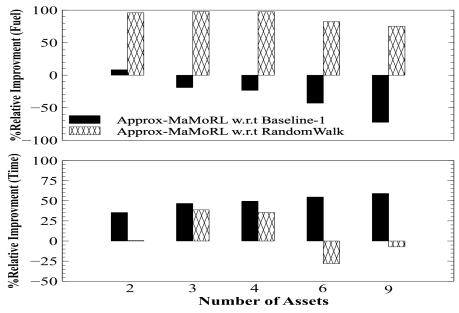
(a) Varying number of nodes



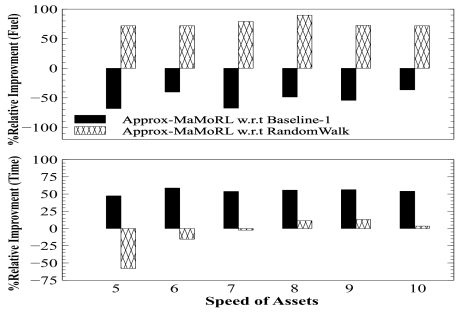
(b) Varying number of edges



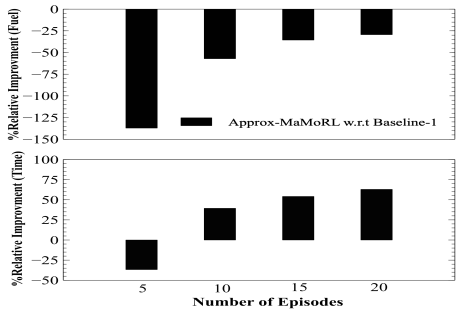
(c) Varying number of neighbors



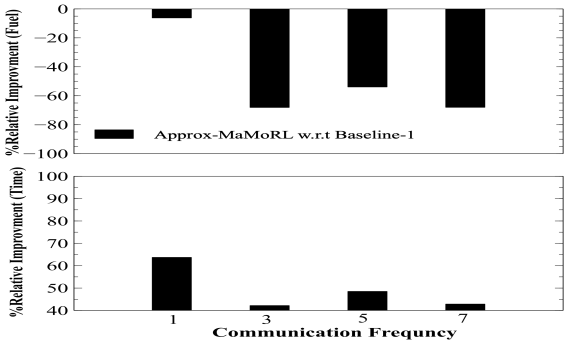
(d) Varying number of assets



(e) Varying speed of assets

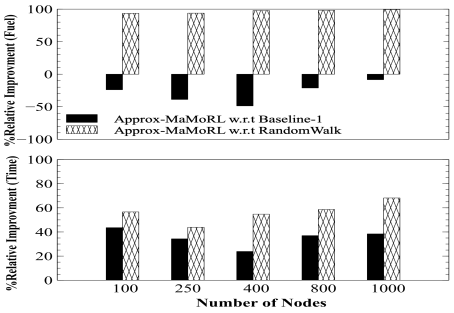


(f) Varying number of episodes

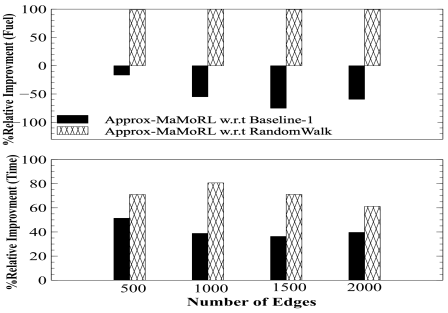


(g) Varying communication frequency

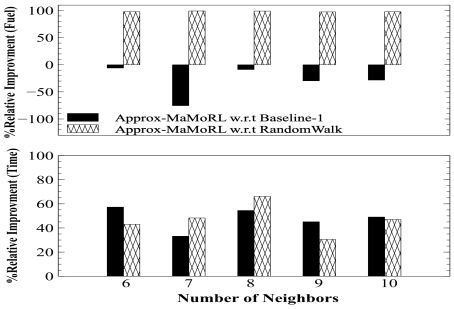
Figure 5.5 F_{total} and T_{total} varying parameters for Approx-MaMoRL.



(a) Varying number of nodes



(b) Varying number of edges



(c) Varying number of neighbors

Figure 5.6 F_{total} and T_{total} varying parameters for Approx-MaMoRL with partial knowledge.

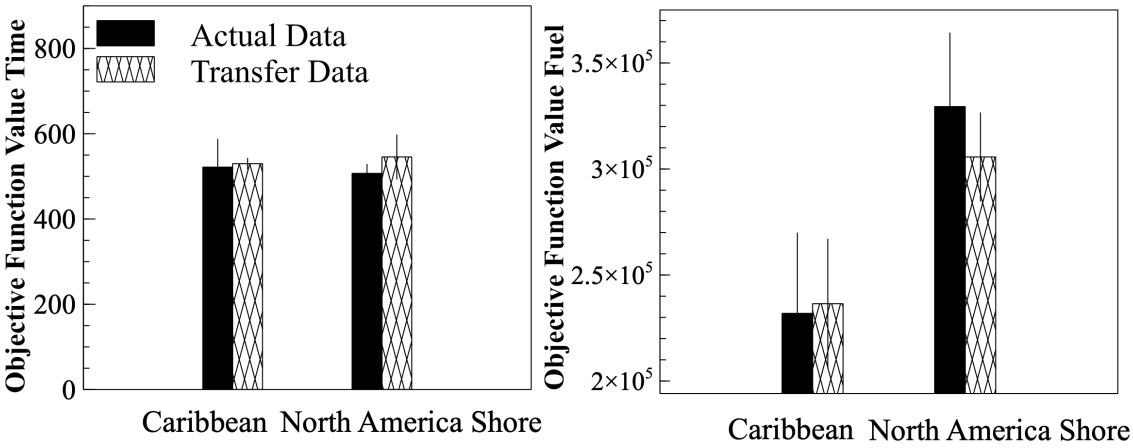
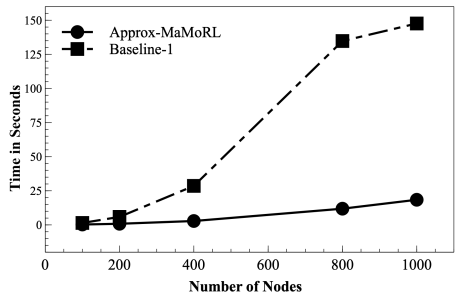
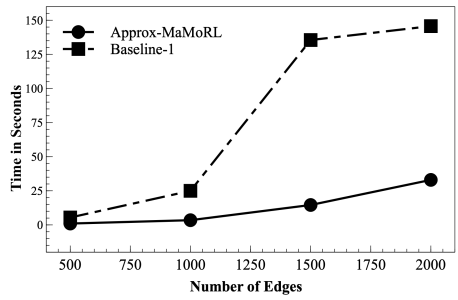


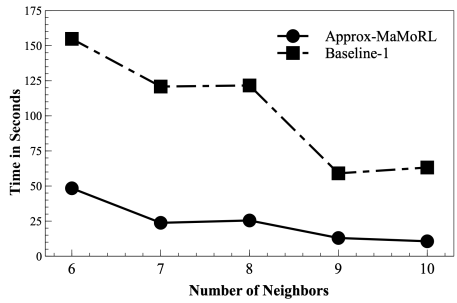
Figure 5.7 Transfer learning.



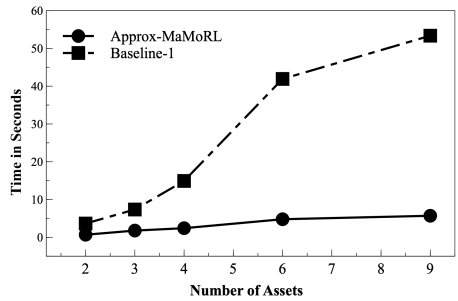
(a) Varying number of nodes



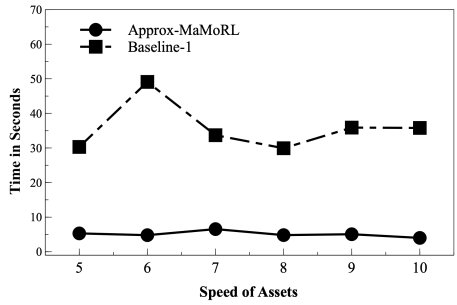
(b) Varying number of edges



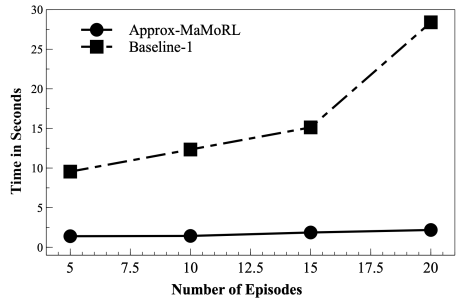
(c) Varying number of neighbors



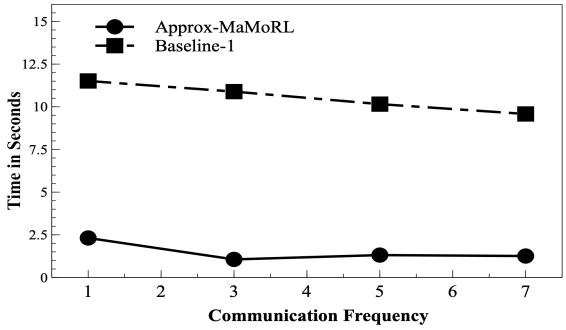
(d) Varying number of assets



(e) Varying speed of assets



(f) Varying number of episodes



(g) Varying communication frequency

Figure 5.8 Running time results.

5.4.7 Deployment inside TMPLAR

MaMoRL inside TMPLAR [121] [32] enables simultaneous route planning for *multiple assets* that are robust to uncertainty under multiple contexts (mission, environment, asset, and threat). It has two views: *a global view*, to facilitate the holistic planning of all the assets simultaneously, as well as *a local view* designed for single asset planning. The developed interface is designed through extensive user studies involving human subjects. It is fast, intuitive, logical, and designed to abstract the complex nature of the underlying solution. TMPLAR is written in Python and is intended to be used as a back-end service with JavaScript Object Notation (JSON) objects. MaMoRL is deployed as a docker container simplifying and accelerating the application workflow while giving developers the freedom to innovate with their choice of tools, application stacks, and deployment environments. The efficacy of the tool is evaluated in the Naval Postgraduate School considering different contexts, hazards, and threats and is now being analyzed by studying the click and eye tracking data of the users, to quantify and compare its ease-of-use and robustness in multi-asset decision-making.

5.5 Related Works

Reinforcement Learning. Reinforcement Learning (RL) has a broad range of applications in dynamic and uncertain environments [105] [84] [63], including traffic lights control and monitoring [138], exploratory data analysis [18], user group discovery [117], and complex task planning [101]. Kadota et al., 2006 [83] and Geibel, 2005 [65] study Markov Decision Problems (MDPs) for a single agent considering multiple objectives with constraints and propose a weighted RL. In [83], Kadota et al., 2006 introduce a Lagrange function for solving utility constrained MDPs. In [141], Yu, 2018 discusses the inefficiencies associated with sample cost in Reinforcement Learning.

Multi-Agent RL. Coordinating multiple agents in a dynamic environment is a complicated problem - some recent efforts have been made to solve this problem using RL [40] [49] [144] [34] [47]. [144] studies a cooperative Multi-Agent RL by using a teammate model and reward allotment. [40] considers multiple agents in a task-oriented environment as a Decision-Theoretic Planning (DTP) problem and adapts RL for solving it. [49] shows how to adapt Generalized Learning Automata (GLA) in multi-agent systems. [34] sheds light on the potential applications of Multi-Agent Reinforcement Learning (MARL) in a variety of different areas from robotics to different static games, whilst raising concerns about scalability. [47] points out the absence of existing work where agents reuse knowledge acquired from one another.

Path Planning. Path Planning for ships and aerial vehicles has been studied extensively in [16] [32] [121] [16] [98]. Bienkowski et al., 2018 [32] leverage Q-factor as an approximate dynamic programming method for navigating ships in uncertain environments. TMPLAR [121] is developed as a system for Multi-objective Planning and Asset Routing.

We non-trivially adapt these works to address a problem of significant interest to maritime navigation involving multiple distributed assets, objectives, and constraints, as well as study scalability challenges.

5.6 Conclusion

We study the *Route Planning Problem* (**RPP**), which is formalized as a Team Discrete Markov Decision Process and we propose a *Multi-agent Multi-objective Reinforcement Learning* (MaMoRL) framework for solving it. We demonstrate why exact MaMoRL is computationally expensive and study approximation opportunities. As an ongoing work, we are analyzing the post-deployment data collected through click and eye tracking of the behavior of the users to quantify its ease of use, flexibility, and robustness in multi-asset decision-making.

CHAPTER 6

ONGOING WORK

This chapter presents our current research efforts to address the challenges associated with classical Reinforcement Learning (RL) frameworks. In this context, our research primarily focuses on three key challenges that are currently being investigated:

1. **Exponential State/Action Space.** A major limitation of RL frameworks is their inability to effectively handle problems that involve an exponential number of states and actions. RL algorithms rely on exploring and exploiting the state-action space to learn the best policy. However, as the complexity of a problem increases, the number of potential states and actions grows exponentially, presenting a significant challenge for RL algorithms. This exponential growth makes it increasingly difficult for RL algorithms to thoroughly explore and evaluate all possible combinations of states and actions. As a consequence, the performance of RL algorithms deteriorates rapidly as the size of the state-action space expands, hindering their effectiveness in solving complex problems.

2. **Large Memory Requirement to Store Q Values.** In classical RL algorithms, Q-values are commonly used to estimate the expected rewards associated with different actions in specific states. However, when dealing with problems characterized by large state and action spaces, the number of Q-values that need to be stored and updated becomes overwhelmingly large. The storage and management of these Q-values in memory pose a significant bottleneck, requiring substantial memory resources. As the complexity of a problem increases, the memory requirements can grow to impractical extents, imposing limitations on the practical applicability of RL algorithms in domains with exponential state and action spaces. This challenge of memory management and storage becomes a critical factor in scaling RL algorithms to address real-world problems effectively.

3. **Significant Learning Time to Be Effective.** Another critical limitation of classical RL frameworks is the substantial amount of time required to achieve effective performance. RL algorithms learn by interacting with the environment and gradually updating their policies based on observed rewards. However, in complex domains, the learning process can be time-consuming due to the necessity of extensive exploration and exploitation to discover optimal policy. Furthermore, the presence of scarce or delayed reward signals further prolongs the learning time, as RL agents struggle to identify the most rewarding actions.

This extended learning time can be impractical in scenarios that demand quick decision-making or adaptability, posing a barrier to the widespread application of RL in real-world problems that require timely and dynamic responses. The challenge of significant learning time undermines the efficiency and scalability of RL algorithms in practical settings.

To address these challenges, our ongoing research explores two distinct perspectives: designing online policy and developing data management techniques. The former studies applicability of Rollout techniques [25] [26] [27] [28] [29] [30], which employs an one-time policy iteration approach in a distributed environment. Indeed, Rollout techniques has the capability of addressing Challenge 1 and Challenge 3. Additionally, we investigate data management techniques, in particular, how to store and reuse fine grained metadata of previously trained policies to overcome Challenge 3, while also tackling Challenge 1 and Challenge 2.

In the following sections, we delve into further details.

6.1 Design Online Policy

6.1.1 Introduction

In Chapter 5, we have introduced a distributed Reinforcement Learning framework that focuses on team decision-making within the context of multi-agent routing. This framework employs a teammate model to predict the actions of other agents and a reward model to learn the cost or reward associated with state-action pairs.

Building upon the Multi-Agent Multi-Objective Reinforcement Learning (MaMoRL) framework, our current exploration aims to develop a multi-objective route planning approach. This approach allows each agent to determine its own set of actions in cognizance of its teammates. Our objective is to enhance the computational efficiency of the MaMoRL framework.

In our ongoing investigations, we are specifically examining the "Rollout Algorithm", which is a one-time policy iteration technique that can be implemented in an online manner to improve policies. This algorithm trades off the complexity of the

action space for that of the state space, leading to a novel form of policy improvement known as "one-agent-at-a-time." By employing this technique, we hope to overcome the computational challenges that arise when dealing with a large number of agents in an extensive grid environment.

6.1.2 Modeling

We model our problem as Dynamic programming (DP), which involves the following components:

a. **State:** The state in our model represents the current location of l ships at stage n. It captures the spatial information of the ships at a specific time.

b. **Stage:** In the context of our formulation, a stage refers to the outcome of the transition t_n that occurs at state s_n. This transition is achieved by taking the action a_n, resulting in a new state s_{n-1} with $n-1$ stages remaining.

c. **Action:** At each stage n, the action space is denoted as $A_n^k = \bigcup_{i=1}^{l}\{A_i \cup wait_i\}$, which is the union of the sets A_i and $wait_i$ for each ship i. Here, A_i represents the set of possible actions available for ship i, while $wait_i$ corresponds to the option of staying at the same location without taking any action.

By employing this DP formulation, we can effectively model and analyze the problem, considering the state of the ships, the transitions between stages, and the available actions at each stage.

6.1.3 Proposed solution

In this section, we will introduce our proposed algorithm for solving the TPP.

6.1.3.1 Local Optimization Function.

Given the current positions $cur_i^n, cur_2^n, ..., cur_l^n$ of the ships and the number of stages remaining n, we aim to determine the optimal action a_n^* by minimizing the cost function $C_n(a_n, s_n)$.

The local optimization problem can be formulated as follows:

$$
\begin{aligned}
& a_n^* = argmin_{a_n} C_n(a_n, s_n) \\
s.t. \quad & cur_i^n \neq cur_2^n \neq ... \neq cur_l^n
\end{aligned}
\tag{6.1}
$$

Where $C_n(a_n, s_n) = \alpha \times Fuel_i(n) + \beta \times Time_i(n)$. $Fuel_i(n)$ and $Time_i(n)$ represent normalized quantities with the same unit.

By applying this local optimization function, we aim to determine the best action at each stage, considering the current ship locations and minimizing the cost function that incorporates fuel consumption and time.

6.1.3.2 Optimization Problem.

$$
\begin{aligned}
v_n(s_n) &= \min \{C_n(a_n, s_n) + C_{n-1}(a_{n-1}, s_{n-1}) + ... + C_0(a_0, s_0)\} \\
&= \min \{C_n(a_n, s_n) + v_{n-1}(s_{n-1})\}
\end{aligned}
$$

s.t.

$$
\begin{aligned}
& cur_i^m \neq cur_2^m \neq ... \neq cur_l^m, \qquad m = 0, 1, ..., n \\
& \begin{cases} s_{m-1} = s_m, & if\, a_m = wait; \qquad m = 1, ..., n \\ s_{m-1} = t_m(a_m, s_m), & \text{otherwise}; \qquad m = 1, ..., n \end{cases} \\
& a_m \in A_m; \qquad m = 0, 1, ..., n
\end{aligned}
\tag{6.2}
$$

6.1.3.3 Algorithm.

To mitigate the exponential growth in computational complexity associated with Dynamic Programming (DP) as the grid size and number of ships increase, we propose an approximation approach using the Rollout strategy. We introduce the "Multi-Objective Rollout Policy Iteration with one-step look-ahead" (MaMoRPI) framework, which is a technique for one-time policy iteration. This framework is designed to improve policies in an online fashion. The MaMoRPI framework addresses the trade-off between the complexity of the action space and the complexity of the state space,

resulting in a unique type of policy improvement known as "one-agent-at-a-time." This approach is visualized in Figure 6.1. At each time step t, MaMoRPI consists of two parts: 1) Determining which ship should take action, and 2) Deciding which action the selected ship should take. To answer these two questions, we employ two separate Rollouts.

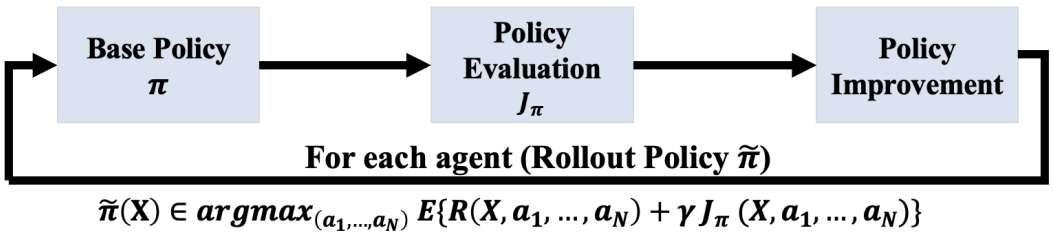


Figure 6.1 Rollout diagram.

1. In each time step t, MaMoRPI selects the ships that maximize the following equation:

$$i \in argmax_i E\{R(i,t,t') + \gamma J(i,n)\} \tag{6.3}$$

Where $R(i,t,t')$ represents the reward, which is calculated using the following equation:

$$R(i,t,t') = \frac{SF(i,t,t')}{Fuel_i} \tag{6.4}$$

The starvation factor $SF(i,t,t') = \frac{t-t'}{t}$ is incorporated into the reward design to prevent keeping a ship in the same location for an extended period. t' in starvation factor is the last time that ship i took action.

In Equation (6.3), $J(i,n)$ is the cost-to-go, which captures the number of newly unsensed nodes that will be explored by each ship. It is computed using the following equation:

$$J(i,n) = \frac{\sum_{a_i \in A_i} S_i^n(a_i)}{|A_i|} \tag{6.5}$$

Here, A_i represents the set of all possible actions for ship i, and $S_i^n(a_i)$ indicates the number of newly sensed nodes by ship i after taking action a_i.

2. After selecting the ship i based on Equation (6.3) at time step t, MaMoRPI determines the action for the chosen ship using the following equation:

$$a_i^* \in argmax_{a_i^n} E\{R'(a_i^n, v_i^n) + \gamma J'(curr_i^n, r_i)\} \tag{6.6}$$

Where the action reward is defined as $R'(a_i^n, v_i^n) = [\frac{v_i^n}{\omega(a_i^n)} + deg(a_i^n)] \times \theta(a_i^n)$.

In the above equation, $\omega(a_i^n)$ represents the distance between the current position ($curr_i$) of ship i and the next position obtained by taking action a_i, while v_i indicates its velocity. To prevent ships from getting stuck in grid corners, we consider the degree of a node as a component of the action for moving towards that node. Additionally, we include θ to avoid collisions between ships, which can be calculated using the following equation:

$$\theta(a_i^n) = \begin{cases} 0, & \exists \quad j \quad s.t. \quad curr_j^n = a_i; \qquad j = 1, ..., l \setminus \{i\} \\ 1, & \text{otherwise;} \end{cases}$$

In the above equation, $\theta(a_i^n)$ is set to 0 if there is another ship j that is one hop away from $curr_i^n$.

The cost-to-go $J'(curr_i^n, r_i)$ in Equation (6.6) captures the average shortest path between each pair of unsensed nodes, which is computed using the following equation:

$$J'(curr_i^n, r_i) = \frac{r_i}{Avg_{m_i}(Dijkstra(m_i, p))} \tag{6.7}$$

Here, r_i represents the sensing radius, $m_i \in neighbors(curr_i^n)$, and p indicates an unsensed node.

6.1.4 Generalization

We also extend our approach to select k ships from a total of l ships to simultaneously take action. Our method consists of two stages for determining which k ships should take actions and which actions they should take: 1. Before detecting the destination, and 2. After detecting the destination.

Solving the general MaMoRPI problem is NP-hard, so we reduce it from the maximum coverage problem. However, for the maximum coverage problem, we can use a greedy algorithm that iteratively selects k ships to maximize the marginal increase in coverage, providing a logarithmic approximation factor.

1. Before Detecting the Destination: For ship selection and action selection, we use the same maximization equations as in Equations (6.3) and (6.6), respectively. However, we employ different reward functions and cost-to-go:

$$R(k, i, t, t') = \frac{\underset{i=1,..,k}{min}\, SF(i, t, t')}{\sum_{i=1}^{k} Fuel_i} \tag{6.8}$$

$$J(k, i, n) = \sum_{i=1}^{k} Coverage_n^i \tag{6.9}$$

Where $Coverage_n^i = \frac{\sum_{a_i^n \in A_i} S_i^n(a_i^n)}{|A_i|}$.

After selecting k ships, we determine the action that maximizes Equation (6.6) for each of the k ships. The reward and cost-to-go for the general form can be rewritten as follows

$$R'(a_i^n, v_i^n) = [\frac{v_i^n}{\omega(a_i^n)} + deg(a_i^n)] \times \theta(a_i^n) \tag{6.10}$$

If any ship is two hops away from the current ship's position, the $\theta(a_i^n)$ in Equation (6.10) returns a value of 0; otherwise, it returns 1. This penalizes actions where two ships are 2 hops apart, as there is a risk of both ships going to the same common neighbor and colliding.

The cost-to-go for the general action selection is calculated using the following equation.

$$J'(curr_i^n, r_i) = \frac{r_i}{Avg_{m_i}(Dijkstra(m_i, p))} \tag{6.11}$$

2. After Detecting the Destination: When one of the ships detects the destination, all ships use the *Dijkstra Shortest Path algorithm* to reach the destination, and the paths for the ships are treated as hyper-edges.

- **Which Ships?** In this case, k ships that can move simultaneously are chosen to take action. It indicates that the ships' hyper-edges do not intersect, allowing them to avoid collisions.
- **Which Actions?** Ships will either follow a hyper-edge to reach their goal or remain in the same location to avoid collisions.

6.2 Reusing Pretrained Policies for Efficient Inferencing

6.2.1 Introduction

The field of Reinforcement Learning (RL) has experienced significant advancements, making it a powerful tool for addressing complex decision-making problems across

various domains [3, 80, 90, 92, 93, 105, 120, 132]. RL algorithms have demonstrated remarkable success in areas such as robotics, data exploration, and query planning. By integrating RL techniques into these applications, there is an opportunity to enhance performance, optimize resource utilization, and improve overall efficiency.

This work focuses on the utilization of RL algorithms for efficient inferencing in data-driven systems. The prominent use cases that stand out as ideal applications of RL are data exploration [18], robot movement [70, 87], and query planning [115]. By leveraging RL's ability to learn from interactions with an environment, data exploration techniques can efficiently uncover valuable insights and patterns within large datasets. Additionally, RL algorithms can enhance robot movement strategies, enabling autonomous agents to navigate complex environments effectively. Furthermore, incorporating RL into query planning processes has the potential to optimize resource allocation and improve query execution efficiency.

We present a novel approach to RL inferencing by leveraging fine-grained metadata of pre-trained policies. The main objective is to harness the reusability of pre-trained models and reduce the expensive training time typically required for RL algorithms. By focusing on inferencing, the paper explores methods that effectively utilize the knowledge captured in pre-trained policies, enabling efficient decision-making without extensive retraining. Our primary goal is to provide insights and methodologies for efficient inferencing in RL. The ability to reuse pre-existing models and reduce training time holds substantial practical value, opening avenues for improved performance and resource optimization in various domains.

The contributions of this work include proposing a computation framework that is generalizable to multiple RL models, as well as suitable to different reward functions. We present efficient inferencing algorithms with theoretical guarantees.

As part of this research, we explore three primary RL implementations: Q-Learning [137], SARSA [127], and Deep RL [15]. Q-Learning is a model-free

algorithm that learns the optimal action-value function through iterative updates. SARSA, another model-free algorithm, follows an on-policy approach by updating action-values based on the actual actions taken. Deep RL combines RL techniques with deep Neural Networks to handle high-dimensional and continuous state and action spaces.

To evaluate the performance of different approaches, several essential metrics are considered, including recall, pruning percentage, training+inference time, training time only, and energy consumption. These metrics provide a comprehensive assessment of the efficiency and effectiveness of the proposed RL techniques in the inferencing phase.

We also examine the impact of varying important parameters on inferencing performance, including inference query characteristics, discount factors for SARSA and Q-Learning algorithms, number of epochs, reward function, and the size of the state and action space. By exploring the effect of these variables, we can gain some insights into overall system performance, enabling the identification of optimal configurations.

6.2.2.1 Toy Running Example. Consider a grid environment (Figure 6.2) where an agent, represented by a robot, needs to learn two distinct tasks. The first task, Task 1, requires the robot to reach the goal state by following the shortest path. The second task, Task 2, involves the robot reaching the goal state while collecting as many powers as possible. Both tasks have the constraint that the robot must avoid bomb cells in the environment.

To train the robot for these tasks, we model the problem as an MDP, where each cell in the grid represents a state. In each state, the robot has four possible actions: Right, Down, Left, and Up. We utilize RL techniques to train the robot

separately for each task, resulting in two optimal policies that enable the robot to achieve the objectives of Task 1 and Task 2, respectively.

The challenge arises when we are asked to train the robot to perform both tasks simultaneously, where it needs to reach the goal state by following the shortest path and collecting as many powers as possible while avoiding bombs. To address this combined task, training the robot from scratch would be time-consuming and inefficient. Instead, we aim to leverage the training data from the individual tasks to obtain an optimal path for the combined task.

By using the knowledge gained from the individual task training, we can devise a strategy to adapt the robot's behavior and decision-making process to effectively solve the new combined task. This approach allows us to capitalize on the learnings and policies derived from the separate tasks and use them as a basis for training the robot to perform optimally in the combined setting.

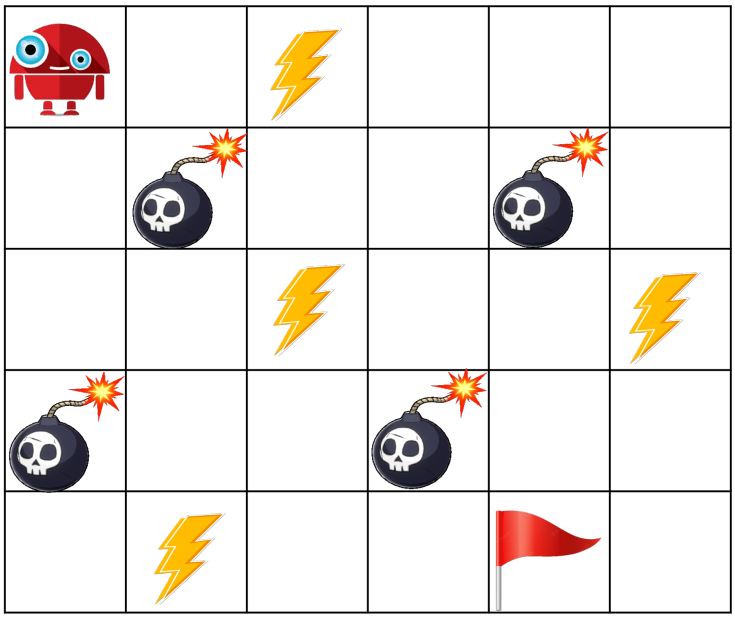


Figure 6.2 Environment of Example 6.2.1

6.2.2 Background and problem definition

The following assumptions are made:

- The reward function of the combined task is a linear function of the reward functions of the individual tasks.
- The combined task shares exactly the same states and action spaces as the individual tasks, ensuring a seamless integration of the tasks without any need for additional modifications or conversions.
- Both the state and action space are considered to be discrete, implying that there is a finite and distinct set of states and actions available for the agent to navigate through.
- The state transition function is deterministic, implying that given a particular state and action, the next state is uniquely determined, and there is no randomness involved in the transition process.
- The rewards are provided at each time step during the execution of the task, rather than being dependent on a final outcome, allowing for a continuous feedback loop for the agent's decision-making process.

Given to us are K tasks, where each is represented as a Deterministic Discrete Markov Decision Process (DDMDP) (S, A, R):

a. S is a set of states.

b. A is a set of actions.

c. $R(s_i, e_i, s_{i+1})$ is the reward of transitioning from state s_i to state s_{i+1} by taking action e_i.

6.2.2.1 Reinforcement Learning based solutions. There are various methods for solving Markov Decision Processes (MDPs), such as value iteration and policy iteration, which are iterative techniques. However, the focus of the work is on model-free Reinforcement Learning (RL). Three different RL approaches are considered:

1. **Q-Learning:** It is a popular RL method that aims to find the optimal action-selection policy in an MDP. It utilizes a Q-table to store the expected cumulative rewards for each state-action pair. Through an iterative process of exploration and exploitation, Q-Learning updates the Q-values based on observed rewards to make informed decisions and maximize long-term rewards.

2. **SARSA:** Another RL algorithm, SARSA (State-Action-Reward-State-Action), operates similarly to Q-Learning but follows an on-policy approach. It considers the current policy when determining the next action, updating Q-values based on the subsequent state and action pair.

3. **DeepRL:** DeepRL, short for Deep Reinforcement Learning, combines RL with deep neural networks to handle complex tasks and large state or action spaces, enabling learning directly from raw sensory data without manual feature engineering.

Definition 6.1. (Combined Task.) *Given K individual tasks $T_i; i = 1, ..., K$, each of which can be represented as an MDP $M_i = (S, A, R_i); i = 1, .., K$, task T is categorized as a combined task if it shares the same state and action space with the individual tasks and its reward is a linear function of a subset of these tasks. In other words, task T can be represented as an MDP $M = (S, A, R)$, where $R \subseteq R_1 \pm R_2 \pm ... \pm R_K$.*

Problem 6.1. *Given a combined task T and a set of individual tasks $T_i; i = 1, ..., K,$, a collection of fine-grained training metadata $\pi_1, ..., \pi_K$ for each, the objective is to find an optimal policy for task T maximize its cumulative reward without the need for additional training for T. The reward function R of T, is expressed as a linear function of $f(R_1, ..., R_K)$.*

6.2.3 Proposed solution

In order to solve Problem 6.1, we design two different algorithms. The first one provides an exact solution, ensuring the optimal outcome with provable guarantee. We also design a highly scalable heuristic approach. This approach is particularly useful when finding an exact solution is computationally expensive. The heuristic algorithm prioritizes computational efficiency over optimality. While it may not provide the absolute best solution, it guarantees a good approximation, as we observe in our preliminary investigation.

Exact Solution: To achieve an exact solution for the combined task, we employ Algorithm 6.1 to train K individual policies. After training, we obtain the graph representation of each policy as well as fine-grained training metadata. In these graphs, the nodes represent the states, while the edges correspond to the actions. Additionally, the Q-values associated with each state-action pair are stored as node features within the graph structure. Algorithm 6.1 follows a similar approach to the offline phase of traditional Q-Learning, with the key distinction being the storage of Q values in a graph format rather than a tabular format. This graph representation allows for efficient and organized access to Q-values during the subsequent phases of the combined task.

Next, by employing Algorithm 6.2, we iterate through all K graphs in a brute-force manner to obtain G_{union} (Lines 2-4). By leveraging the fact that the Q-value of the goal state (s_{goal}) remains unchanged throughout the training process, we utilize a backtracking approach to determine the boundaries for each (node, edge) pair. For the lower bound, we assume that each (node, edge) pair has been visited only once, while for the upper bound, we consider that each (node, edge) pair has been visited in every episode. However, it's worth noting that this approach becomes ineffective if the graph contains cycles, as the cyclic dependencies may prevent us from attaining the exact solution we seek.

Lastly, Algorithm 6.3 is employed to perform edge pruning within the graph, removing edges that cannot be included in the optimal path. In essence, an edge is pruned if its upper bound is found to be lower than the lower bound of another outgoing edge from the current node (Line 6). Once this pruning process is complete, we retrieve all the paths from the starting node to the goal node. Each path's cumulative reward is then calculated, and the path with the highest cumulative reward is selected as the optimal path for the combined task. This final step ensures that the

resulting solution maximizes the cumulative reward, providing an effective outcome for the given problem.

Heuristic Solution: In scenarios where the state and action space is exponentially large, obtaining an exact solution becomes impractical. To address this challenge, we introduce a heuristic approach that balances efficiency and approximation to achieve results close to optimality. The heuristic solution closely resembles the exact solution, with a notable modification in Algorithm 6.2. Instead of performing brute force and combining the graphs of all K individual policies, we selectively unite a subset of each graph. Specifically, we consider only the top h actions for each state, rather than evaluating all possible actions. The subsequent steps of the algorithm remain unchanged. This heuristic approach may not yield the absolute optimal path for the combined task, but it delivers a solution that closely aligns with the optimal path, differing only in a few waypoints. By striking a balance between efficiency and accuracy, this heuristic method provides a practical alternative for dealing with large state and action spaces.

Hypothesis 6.1. *In the Brute Forcing part of Algorithm 6.2, G_{union} contains all possible paths that start at the start node (s_{start}) and end at the goal node (s_{goal}).*

Hypothesis 6.2. *The number of possible paths that starts at the start node (s_{start}) and end at the goal node (s_{goal}) in $G_{scratch}$ is a subset of possible paths in G_{union}.*

Hypothesis 6.3. *At least one of retained paths produced by the Algorithm 6.3 will have the identical cumulative reward as that of the p_{opt}.*

Lemma 6.1. *In an MDP denoted by (S, A, R), where the discount factor γ is set to zero, if a specific state-action pair (s, a) has been visited j times, the Q-values after the jth visit can be calculated using the following formula:*

$$Q^j(s,a) \leftarrow (-1)^{j-1}R[(\alpha-1)^j + (-1)^{j-1}]; \qquad \forall (s,a) \in S \times A \tag{6.12}$$

Proof. Without loss of generality, let us consider Q-Learning as our chosen approach with the following Q-function:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma\, max_{a'} Q(s',a') - Q(s,a)] \tag{6.13}$$

When $\gamma = 0$, the Q-function simplifies to $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha R$.

We will proceed with the proof by induction. When (s,a) is visited for the first time, the Q-value is updated as $Q^1(s,a) \leftarrow \alpha R$, since all Q-values are initially set to 0.

Now, with $j = 1$ in Equation (6.12), we have:

$$Q^1(s,a) = (-1)^{1-1}R[(\alpha-1)^1 + (-1)^{1-1}] = R[\alpha - 1 + 1] = \alpha R$$

Next, assuming that (s,a) has been visited $(j-1)$ times, the Q-value is given by:

$$Q^{j-1}(s,a) \leftarrow (-1)^{j-2}R[(\alpha-1)^{j-1} + (-1)^{j-2}]$$

Now, we can calculate the Q-value when (s,a) has been visited j times:

$$\begin{aligned}
Q^j(s,a) &= (1-\alpha)Q^{j-1}(s,a) + \alpha R \\
&= (1-\alpha)[(-1)^{j-2}R[(\alpha-1)^{j-1} + (-1)^{j-2}]] + \alpha R \\
&= (-1)(\alpha-1)[(-1)^{j-2}R(\alpha-1)^{j-1} + (-1)^{2j-4}R] + \alpha R \\
&= (-1)^{j-1}R(\alpha-1)^j + (1-\alpha+\alpha)R \\
&= (-1)^{j-1}R[(\alpha-1)^j + (-1)^{j-1}]
\end{aligned}$$

The same steps can be applied to SARSA.

□

Theorem 6.1. *If the discount factor (γ) is set to zero in both Q-Learning and SARSA algorithms, the Q-value of the combined task for all (state, action) pair will be obtained as follows:*

$$Q_c(s,a) = Q_1(s,a) \pm Q_2(s,a) \pm ... \pm Q_K(s,a); \qquad \forall(s,a) \tag{6.14}$$

Proof. Using Lemma 6.1, the Q-value for each individual policy after j visits can be expressed as follows:

$$\begin{aligned}
Q_1^j(s,a) &= (-1)^{j-1} R_1[(\alpha-1)^j + (-1)^{j-1}] \\
Q_2^j(s,a) &= (-1)^{j-1} R_2[(\alpha-1)^j + (-1)^{j-1}] \\
&\vdots \\
Q_K^j(s,a) &= (-1)^{j-1} R_K[(\alpha-1)^j + (-1)^{j-1}]
\end{aligned}$$

For the combined task, the Q-value after the jth visit of $(s,a) is given by$:

$$Q_c^j(s,a) = (-1)^{j-1}(R_1 \pm R_2 \pm ... \pm R_K)[(\alpha-1)^j + (-1)^{j-1}]$$

Now, let's combine the Q-values of all K policies as follows:

$$\begin{aligned}
Q_1^j(s,a) \pm Q_2^j(s,a) \pm ... \pm Q_K^j(s,a) = & \\
& (-1)^{j-1} R_1[(\alpha-1)^j + (-1)^{j-1}] \\
& \pm (-1)^{j-1} R_2[(\alpha-1)^j + (-1)^{j-1}] \\
& \vdots \\
& \pm (-1)^{j-1} R_K[(\alpha-1)^j + (-1)^{j-1}] \\
& = (-1)^{j-1}(R_1 \pm R_2 \pm ... \pm R_K)[(\alpha-1)^j + (-1)^{j-1}] \\
& = Q_c^j(s,a)
\end{aligned}$$

□

Algorithm 6.1 DAG Representation Algorithm

Require: State Set S, Action Set A, Reward R, Number of Episodes N

Ensure: A Weighted Directed Graph $G = (V, E)$

```
Create a node v_i; ∀s_i ∈ S
Initialize features of node v_i as the Q value of each possible action;  F(v_i) = {a : 0, ∀a ∈ A}
for i ← 1 to N do
    while s_t ≠ s_goal do
        Observe the current state s_t
        a_t ← argmax_{∀a∈A} R(s_t, a, s_{t+1})
        Take action a_t
        r_{t+1} ← R(s_t, a_t, s_{t+1})
        e_t ← (s_t.s_{t+1})
        E ← {E ⋃ e_t}
        Q(s_t, a_t) ← Q(s_t, a_t) + α[r_{t+1} + γ max_{a'∈A} Q(s_{t+1}, a') − Q(s_t, a_t)]
        F(v_{s_t})[a_t] ← max{F(v_{s_t})[a_t], Q(s_t, a_t)}
        s_t ← s_{t+1}
```

Algorithm 6.2 Backtracking Algorithm

Require: M Individual Policies with Reward $R_1, R_2, ..., R_K$, respectively, Number of Episodes N

Ensure: A Weighted Directed Graph G_{union}

Brute Forcing:

for $i \leftarrow 1$ to K **do**

    $G_i \leftarrow$ Using Algorithm 6.1 for policy π_i

$G_{union} \leftarrow G_1 \bigcup G_2 \bigcup .... \bigcup G_M$

Bound Computation:

$node \leftarrow s_{goal}$

while $node \neq s_{start}$ **do**

    $Neighbors(node, G_{union}) \leftarrow \{n|(n, node) \in E\}$

    for $n \in Neighbors$ **do**

        $e \leftarrow (n, node)$

        $R \leftarrow R_1 \pm R_2 \pm ... \pm R_M$

        $LB(n, e) \leftarrow \alpha \times R$

        $UB(n, e) \leftarrow (-1)^{N-1} \times R \times [(1-\alpha)^N + (-1)^{N-1}] + \alpha[\gamma max\{UB(v, .)|(n, v) \in E\}]$

Algorithm 6.3 Pruning Algorithm

Require: The Weighted Directed Graph G_{union}, Start Node s_{start}, Goal Node s_{goal}

Ensure: A set of paths from s_{start} to s_{goal}

1: **Pruning:**

2: Traverse G_{union} in forward manner

3: $node \leftarrow s_{start}$

4: $Neighbors(node, G_{union}) \leftarrow \{n|(n, node) \in E\}$

5: **for** $n \in Neighbors$ **do**

6: **if** $UB(n, (node, n))) \leq LB(n', (node, n'))s.t.\, n' \in Neighbors \,\&n' \neq n$ **then**

7: Prune $(node, n)$

8: **Path Finding:**

9: **return** $P \leftarrow FindPath(G_{union}, s_{start}, s_{goal})$

CHAPTER 7

CONCLUSION

In conclusion, this dissertation addresses the challenges and opportunities in complex task planning by presenting a set of computational frameworks that synthesize the capabilities of human inputs and AI algorithms. The motivation for studying complex task planning lies in its ubiquity and relevance in various real-world applications, including personalized course planning, trip planning, music playlist design, and route planning for multiple agents. These applications demand efficient planning techniques that can handle uncertainty, optimize multiple objectives, and satisfy complex constraints.

The dissertation's contributions are centered around the development of computational frameworks for automated task planning with minimal end-user input. It presents algorithms based on Reinforcement Learning and discrete optimization techniques, along with data engineering and data management opportunities to design scalable algorithms. These frameworks are applied to tackle different aspects of complex task planning, such as task sequencing, diversifying recommendations, and multi-agent route planning.

Ongoing work in the dissertation focuses on addressing several challenges in complex task planning, including the exponential state/action space, large memory requirements, and significant learning time associated with classical RL algorithms. The research explores the use of online policy techniques like Rollout policy to create multi-objective decision-making frameworks with multiple agents. Additionally, data management techniques are proposed to reduce the training time required for effective learning, making the algorithms more scalable and applicable to real-world scenarios.

Overall, the research presented in this dissertation contributes to the advancement of complex task planning, offering solutions that combine the strengths of human inputs and AI algorithms. By developing efficient and adaptable planning algorithms, this work opens up potential advancements and improvements in autonomous systems, manufacturing, logistics, healthcare, and various other fields where complex task execution is essential. The findings and methodologies in this dissertation have the potential to pave the way for more effective and scalable task planning solutions in a wide range of practical applications.

REFERENCES

[1] Figure eight - data for everyone, 2019. https://www.figure-eight.com/data-for-everyone/.

[2] S. Abbar, S. Amer-Yahia, P. Indyk, and S. Mahabadi. Real-time recommendation of diverse related articles. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 1–12, 2013.

[3] P. Abbeel, A. Coates, M. Quigley, and A. Ng. An application of reinforcement learning to aerobatic helicopter flight. *Advances in Neural Information Processing Systems*, 19, 2006.

[4] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31. Proceedings of Machine Learning Research, 2017.

[5] A. Agrawal, R. Chatterjee, C. Curino, A. Floratou, N. Gowdal, M. Interlandi, A. Jindal, K. Karanasos, S. Krishnan, B. Kroth, J. Leeka, K. Park, H. Patel, O. Poppe, F. Psallidas, R. Ramakrishnan, A. Roy, K. Saur, R. Sen, M. Weimer, T. Wright, and Y. Zhu. Cloudy with high chance of dbms: A 10-year prediction for enterprise-grade ml. *arXiv preprint arXiv:1909.00084*, 2019.

[6] S. B. Aher and L. Lobo. Combination of machine learning algorithms for recommendation of courses in e-learning system based on historical data. *Knowledge-Based Systems*, 51:1–14, 2013.

[7] A. Aipe and U. Gadiraju. Similarhits: Revealing the role of task similarity in microtask crowdsourcing. In *Proceedings of the 29th on Hypertext and Social Media*, pages 115–122, 2018.

[8] M. Alsayasneh, S. Amer-Yahia, E. Gaussier, V. Leroy, J. Pilourdault, R. M. Borromeo, M. Toyama, and J. Renders. Personalized and diverse task composition in crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering*, 30(1):128–141, 2017.

[9] E. Altman. *Constrained Markov decision processes*, volume 7. CRC Press, New York, NY, 1999.

[10] S. Amer-Yahia, S. Elbassuoni, B. Omidvar-Tehrani, R. M. Borromeo, and M. Farokhnejad. Grouptravel: Customizing travel packages for groups. In *International Conference on Extending Database Technology*, pages 133–144, 2019.

[11] S. Amer-Yahia, E. Gaussier, V. Leroy, J. Pilourdault, R. M. Borromeo, and M. Toyama. Task composition in crowdsourcing. In *IEEE International*

Conference on Data Science and Advanced Analytics, pages 194–203. IEEE, 2016.

[12] A. Anagnostopoulos, A. Z. Broder, and D. Carmel. Sampling search-engine results. *World Wide Web*, 9(4):397–429, 2006.

[13] K. Andreev and H. Racke. Balanced graph partitioning. *Theory of Computing Systems*, 39(6):929–939, 2006.

[14] A. Angel and N. Koudas. Efficient diversity-aware search. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, pages 781–792, 2011.

[15] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[16] G. V. Avvari, D. Sidoti, L. Zhang, M. Mishra, K. Pattipati, C. R. Sampson, and J. Hansen. Robust multi-objective asset routing in a dynamic and uncertain environment. In *IEEE Aerospace Conference*, 2018.

[17] C. Baccigalupo and E. Plaza. Case-based sequential ordering of songs for playlist recommendation. In *European Conference on Case-Based Reasoning*, pages 286–300. Springer, 2006.

[18] O. Bar El, T. Milo, and A. Somech. Automatically generating data exploration sessions using deep reinforcement learning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020.

[19] S. Basu Roy, G. Das, S. Amer-Yahia, and C. Yu. Interactive itinerary planning. In *2011 IEEE 27th International Conference on Data Engineering*, pages 15–26. IEEE, 2011.

[20] R. Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[21] N. Bendakir and E. Aïmeur. Using association rules for course recommendation. In *Proceedings of the AAAI Workshop on Educational Data Mining*, volume 3, pages 1–10, 2006.

[22] I. Benouaret, S. Amer-Yahia, and S. Basu Roy. An efficient greedy algorithm for sequence recommendation. In *International Conference on Database and Expert Systems Applications*, pages 314–326. Springer, 2019.

[23] B. Bercovitz, F. Kaliszan, G. Koutrika, H. Liou, Z. Mohammadi Zadeh, and H. Garcia-Molina. Courserank: a social system for course planning. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, pages 1107–1110, 2009.

[24] T. Bertin-Mahieux, D. P.W. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval*, 2011.

[25] D. Bertsekas. Multiagent rollout algorithms and reinforcement learning. *arXiv preprint arXiv:1910.00120*, 2019.

[26] D. Bertsekas. Constrained multiagent rollout and multidimensional assignment with the auction algorithm. *arXiv preprint arXiv:2002.07407*, 2020.

[27] D. Bertsekas. Multiagent reinforcement learning: Rollout and policy iteration. *IEEE/CAA Journal of Automatica Sinica*, 8(2):249–272, 2021.

[28] D. Bertsekas. *Rollout, policy iteration, and distributed reinforcement learning*. Athena Scientific, Nashua, NH, 2021.

[29] S. Bhattacharya, S. Badyal, T. Wheeler, S. Gil, and D. Bertsekas. Reinforcement learning for pomdp: Partitioned rollout and policy iteration with application to autonomous sequential repair problems. *IEEE Robotics and Automation Letters*, 5(3):3967–3974, 2020.

[30] S. Bhattacharya, S. Kailas, S. Badyal, S.and Gil, and D. Bertsekas. Multiagent rollout and policy iteration for pomdp with application to multi-robot repair problems. *arXiv preprint arXiv:2011.04222*, 2020.

[31] N. Bialystocki and D. Konovessis. On the estimation of ship's fuel consumption and speed curve: a statistical approach. *Journal of Ocean Engineering and Science*, 1(2):157–166, 2016.

[32] A. Bienkowski, D. Sidoti, L. Zhang, K. R. Pattipati, C. R. Sampson, and J. Hansen. Path planning in an uncertain environment using approximate dynamic programming methods. In *International Conference on Information Fusion*. IEEE, 2018.

[33] P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. *Microsoft Research, Redmond*, 20(0):0, 2000.

[34] L. Buşoniu, R. Babuška, and B. De Schutter. Multi-agent reinforcement learning: An overview. *Innovations in Multi-Agent Systems and Applications-1*, 2010.

[35] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška. Approximate reinforcement learning: An overview. In *Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2011.

[36] X. Cao, L. Chen, G. Cong, and X. Xiao. Keyword-aware optimal route search. *arXiv preprint arXiv:1208.0077*, 2012.

[37] J. G. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, volume 98, pages 335–336, 1998.

[38] D. Chandler and A. Kapelner. Breaking monotony with meaning: Motivation in crowdsourcing markets. *Journal of Economic Behavior and Organization*, abs/1210.0962, 2012.

[39] H. Chang, Y. Chang, and M. Tsai. ATIPS: automatic travel itinerary planning system for domestic areas. *Computational Intelligence and Neuroscience*, 2016:1281379:1–1281379:13, 2016.

[40] G. Chen, Z. Yang, H. He, and K. M. Goh. Coordinating multiple agents via reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 10(3):273–328, 2005.

[41] W. Chen, Z. Niu, X. Zhao, and Y. Li. A hybrid recommendation algorithm adapted in e-learning environments. *World Wide Web*, 17(2):271–284, 2014.

[42] Z. Chen and T. Li. Addressing diverse user preferences in sql-query-result navigation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*, pages 641–652, 2007.

[43] M. D. Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In *Hypertext*, pages 35–44, 2010.

[44] K. Chu, M. Chang, and Y. Hsia. Designing a course recommendation system on web based on the students' course selection records. In *EdMedia+ Innovate Learning*, pages 14–21. Association for the Advancement of Computing in Education, 2003.

[45] M. Cieliebak, S. Eidenbenz, A. Pagourtzis, and K. Schlude. On the complexity of variations of equal sum subsets. *Nordic Journal of Computing.*, 14(3):151–172, 2008.

[46] N. Cressie and H. Whitford. How to use the two sample t-test. *Biometrical Journal*, 28(2):131–148, 1986.

[47] F. L. Da Silva, M. E. Taylor, and A. H. Reali Costa. Autonomously reusing knowledge in multiagent reinforcement learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 5487–5493, 2018.

[48] P. Dai, J. M. Rzeszotarski, P. Paritosh, and E. H. Chi. And now for something completely different: Improving crowdsourcing workflows with micro-diversions. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing*, pages 628–638, 2015.

[49] Y. De Hauwere, P. Vrancx, and A. Nowé. Generalized learning automata for multi-agent reinforcement learning. *AI Communications*, 23(4):311–324, 2010.

[50] D. Difallah, E. Filatova, and P. Ipeirotis. Demographics and dynamics of mechanical turk workers. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 135–143. ACM, 2018.

[51] D. E. Difallah, M. Catasta, and P. Demartini, G.and Cudré-Mauroux. Scaling-up the crowd: Micro-task pricing schemes for worker retention and latency improvement. In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

[52] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[53] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Aide: an active learning-based approach for interactive data exploration. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2842–2856, 2016.

[54] M. Ebner, S. Schön, B. Taraghi, H. Drachsler, and P. Tsang. First steps towards an integrated personal learning environment at the university level. In *International Conference on Teaching and Learning*, pages 22–36. Springer, 2011.

[55] M. Eirinaki, S. Abraham, N. Polyzotis, and N. Shaikh. Querie: Collaborative database exploration. *IEEE Transactions on Knowledge and Data Engineering*, 26(7):1778–1790, 2013.

[56] K. El-Arini, G. Veda, D. Shahaf, and C. Guestrin. Turning down the noise in the blogosphere. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 289–298, 2009.

[57] A. Elbadrawy and G. Karypis. Domain-aware grade prediction and top-n course recommendation. In *ACM Conference on Recommender Systems*, pages 183–190, 2016.

[58] G. Engin, B. Aksoyer, M. Avdagic, D. Bozanlı, U. Hanay, D. Maden, and G. Ertek. Rule-based expert systems for supporting university students. *Procedia Computer Science*, 31:22–31, 2014.

[59] M. R. Esfandiari, R. M. Borromeo, S. Nikookar, P. Sakharkar, S. Amer-Yahia, and S. Basu Roy. Multi-session diversity to improve user satisfaction in web applications. In *Proceedings of the Web Conference 2021*, pages 1928–1936, 2021.

[60] J. Fan, G. Li, B. C. Ooi, K. Tan, and J. Feng. icrowd: An adaptive crowdsourcing framework. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1015–1030, 2015.

[61] M. Fan, J. Lu, B. C. Ooi, W. Tan, and M. Zhang. A hybrid machine-crowdsourcing system for matching web tables. In *IEEE 30th International Conference on Data Engineering*, pages 976–987. IEEE, 2014.

[62] Z. Friggstad, S. Gollapudi, K. Kollias, T. Sarlos, C. Swamy, and A. Tomkins. Orienteering algorithms for generating travel itineraries. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 180–188, 2018.

[63] J. Garcıa and F. Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[64] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[65] P. Geibel. Reinforcement learning for mdps with constraints. In *European Conference on Machine Learning*, pages 646–653. Springer, 2006.

[66] P. Geibel and F. Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, 24:81–108, 2005.

[67] C. Geuzaine and J. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[68] A. Gionis, T. Lappas, K. Pelechrinis, and E. Terzi. Customized tour recommendations in urban areas. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pages 313–322, 2014.

[69] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[70] V. Gullapalli, J. A. Franklin, and H. Benbrahim. Acquiring robot skills via reinforcement learning. *IEEE Control Systems Magazine*, 14(1):13–24, 1994.

[71] L. Han, K. Roitero, U. Gadiraju, C. Sarasua, A. Checco, E. Maddalena, and G. Demartini. All those wasted hours: on task abandonment in crowdsourcing. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, Melbourne, VIC, Australia, February 11-15, 2019*, pages 321–329, 2019.

[72] N. Hariri, B. Mobasher, and R. Burke. Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the sixth ACM Conference on Recommender Systems*, pages 131–138, 2012.

[73] K. Hata, R. Krishna, F. Li, and M. S. Bernstein. A glimpse far into the future: understanding long-term crowd worker quality. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, Portland, OR, USA, February 25 - March 1, 2017*, pages 889–901, 2017.

[74] C. Ho, S. Jabbari, and J. W. Vaughan. Adaptive task assignment for crowdsourced classification. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, pages 534–542, 2013.

[75] C. Ho and J. W. Vaughan. Online task assignment in crowdsourcing markets. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[76] H. Hsu and P. A. Lachenbruch. Paired t test. *Wiley StatsRef: Statistics Reference Online*, 2014.

[77] E. Huang, L. Peng, L.and Di Palma, A. Abdelkafi, A. Liu, and Y. Diao. *Optimization for active learning-based interactive database exploration*. PhD thesis, Ecole Polytechnique; University of Massachusetts Amherst, 2018.

[78] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pages 2107–2116. Proceedings of Machine Learning Research, 2018.

[79] A. Jain, P. Sarda, and J. R. Haritsa. Providing diversity in k-nearest neighbor query results. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 404–413. Springer, 2004.

[80] C. Jiang, X. Li, J. Lin, M. Liu, and Z. Ma. Adaptive control of resource flow to optimize construction work and cash flow via online deep reinforcement learning. *Automation in Construction*, 150:104817, 2023.

[81] S. Jiang, X. Qian, T. Mei, and Y. Fu. Personalized travel sequence recommendation on multi-source big social media. *IEEE Transactions on Big Data*, 2(1):43–56, 2016.

[82] W. Jiang, Z. A. Pardos, and Q. Wei. Goal-based course recommendation. In *Proceedings of the 9th International Conference on Learning Analytics and Knowledge*, pages 36–45, 2019.

[83] Y. Kadota, M. Kurano, and M. Yasuda. Discounted markov decision processes with utility constraints. *Computers and Mathematics with Applications*, 51(2):279–284, 2006.

[84] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[85] W. Kang and J. McAuley. Self-attentive sequential recommendation. In *IEEE International Conference on Data Mining*, 2018.

[86] K. Karanasos, M. Interlandi, D. Xin, F. Psallidas, R. Sen, K. Park, I. Popivanov, S. Nakandal, S. Krishnan, M. Weimer, Y. Yu, R. Ramakrishnan, and C. Curino. Extending relational query processing with ml inference. *arXiv preprint arXiv:1911.00231*, 2019.

[87] J. Kober, E. Oztop, and J. Peters. Reinforcement learning to adjust robot movements to new situations. *Robotics: Science and Systems, MIT Press Journal*, 6:33–40, 2011.

[88] M. Kyriakidi, K. Stefanidis, and Y. Ioannidis. On achieving diversity in recommender systems. In *Proceedings of the ExploreDB'17*, pages 1–6. 2017.

[89] C. E. Leiserson, R. L. Rivest, T. H. Cormen, and C. Stein. *Introduction to algorithms*, volume 6. MIT Press, Cambridge, MA, 2001.

[90] F. Lezzar, A. Zidani, and C. Atef. A collaborative web-based application for health care tasks planning. In *Proceedings of the 4th International Conference on Web and Information Technologies*, pages 30–39. Citeseer, 2012.

[91] E. Liebman, M. Saar-Tsechansky, and P. Stone. Dj-mc: A reinforcement-learning agent for music playlist recommendation. *arXiv preprint arXiv:1401.1880*, 2014.

[92] S. Mahadevan and G. Theocharous. Optimizing production manufacturing using reinforcement learning. In *Proceedings of the Eleventh International Florida Artificial Intelligence Research Society Conference*, volume 372, page 377. Citeseer, 1998.

[93] H. Mao, M. Alizadeh, I. Menache, and S. Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pages 50–56, 2016.

[94] F. S. Melo, S. P. Meyn, and M. I. Ribeiro. An analysis of reinforcement learning with function approximation. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[95] W. Michiels, J. Korst, E. Aarts, and J. V. Leeuwen. Performance ratios for the differencing method applied to the balanced number partitioning problem. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 583–595. Springer, 2003.

[96] F. P Miller, A. F. Vandome, and J. McBrewster. *Levenshtein distance: Information theory, computer science, string (computer science), string metric, Damerau-Levenshtein distance, spell checker, Hamming distance.* Alpha Press, Troy, NY, 2009.

[97] T. Milo and A. Somech. Next-step suggestions for modern interactive data analysis platforms. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 576–585, 2018.

[98] M. Mishra, D. Sidoti, G. V. Avvari, P. Mannaru, D. F. M. Ayala, and K. R. Pattipati. Context-driven proactive decision support: Challenges and applications. In *AAAI Spring Symposium Series*, 2017.

[99] S. Narvekar, J. Sinapov, and P. Stone. Autonomous task sequencing for customized curriculum design in reinforcement learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017.

[100] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.

[101] S. Nikookar, P. Sakharkar, B. Smagh, S. Amer-Yahia, and S. Basu Roy. Guided task planning under complex constraints. In *2022 IEEE 38th International Conference on Data Engineering*, pages 833–845. IEEE, 2022.

[102] B. Omidvar-Tehrani, A. Personnaz, and S. Amer-Yahia. Guided text-based item exploration. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, pages 3410–3420, 2022.

[103] A. Parameswaran, P. Venetis, and H. Garcia-Molina. Recommendation systems with complex constraints: A course recommendation perspective. *ACM Transactions on Information Systems*, 2011.

[104] E. Pashenkova, I. Rish, and R. Dechter. Value iteration and policy iteration algorithms for markov decision problem. In *AAAI'96: Workshop on Structural Issues in Planning and Temporal Reasoning*. Citeseer, 1996.

[105] E. Pednault, N. Abe, and B. Zadrozny. Sequential cost-sensitive decision making with reinforcement learning. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 259–268, 2002.

[106] J. Pilourdault, D. Amer-Yahia, S.and Lee, and S. Basu Roy. Motivation-aware task assignment in crowdsourcing. In *International Conference on Extending Database Technology*, 2017.

[107] A. Punnen, F. Margot, and S. Kabadi. Tsp heuristics: domination analysis and complexity. *Algorithmica*, 35(2):111–127, 2003.

[108] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley and Sons, Hoboken, NJ, 2014.

[109] S. A. Puthiya Parambath, N. Usunier, and Y. Grandvalet. A coverage-based approach to recommendation diversity on similarity graph. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 15–22, 2016.

[110] L. Qin and X. Zhu. Promoting diversity in recommendation by entropy regularizer. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[111] M. Quadrana, P. Cremonesi, and D. Jannach. Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)*, 2018.

[112] H. Rahman, S. Basu Roy, S. Thirumuruganathan, S. Amer-Yahia, and G. Das. Optimized group formation for solving collaborative tasks. *The International Journal on Very Large Data Bases*, 28(1):1–23, 2019.

[113] S. Ravichandiran. *Hands-on reinforcement learning with Python: Master reinforcement and deep reinforcement learning using OpenAI Gym and TensorFlow*. Packt Publishing Ltd, Birmingham, UK, 2018.

[114] D. J. Rosenkrantz, G. K. Tayi, and S. Ravi. Facility dispersion problems under capacity and cost constraints. *Journal of Combinatorial Optimization*, 4(1):7–33, 2000.

[115] C. Rosset, D. Jose, G. Ghosh, B. Mitra, and S. Tiwary. Optimizing query evaluations using reinforcement learning for web search. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1193–1196, 2018.

[116] J. M. Rzeszotarski, E. Chi, P. Paritosh, and P. Dai. Inserting micro-breaks into crowdsourcing workflows. In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013.

[117] M. Seleznova, B. Omidvar-Tehrani, S. Amer-Yahia, and E. Simon. Guided exploration of user groups. *Proceedings of the VLDB Endowment*, 13(9):1469–1482, 2020.

[118] C. Severance. Teaching the world: Daphne koller and coursera. *Computer*, 45(8):8–9, 2012.

[119] D. Shi, Y. Tong, Z. Zhou, K. Xu, and H. Tan, W.and Li. Adaptive task planning for large-scale robotized warehouses. In *IEEE 38th International Conference on Data Engineering*, pages 3327–3339. IEEE, 2022.

[120] S. M. Shortreed, E. Laber, D. J. Lizotte, T. S. Stroup, J. Pineau, and S. A. Murphy. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine Learning*, 84:109–136, 2011.

[121] D. Sidoti, G. V. Avvari, M. Mishra, L. Zhang, B. K. Nadella, J. E. Peak, J. A. Hansen, and K. R. Pattipati. A multiobjective path-planning algorithm with time windows for asset routing in a dynamic weather-impacted environment. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2016.

[122] M. Slaney and W. White. Measuring playlist diversity for recommendation systems. In *Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia*, pages 77–82, 2006.

[123] M. R. Stoline. The status of multiple comparisons: simultaneous estimation of all pairwise comparisons in one-way anova designs. *The American Statistician*, 35(3):134–141, 1981.

[124] M. Stratigi, J. Nummenmaa, E. Pitoura, and K. Stefanidis. Fair sequential group recommendations. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pages 1443–1452, 2020.

[125] M. Stratigi, E. Pitoura, J. Nummenmaa, and K. Stefanidis. Sequential group recommendations based on satisfaction and disagreement scores. *Journal of Intelligent Information Systems*, pages 1–28, 2021.

[126] SurveyMonkey. Calculating the number of respondents you need, 1999. https://help.surveymonkey.com/articles/en_US/kb/How-many-respondents-do-I-need.

[127] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA, 2018.

[128] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 1999.

[129] M. Sweeney, H. Rangwala, J. Lester, and A. Johri. Next-term student performance prediction: A recommender systems approach. *arXiv preprint arXiv:1604.01840*, 2016.

[130] J. Tang and K. Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 565–573, 2018.

[131] Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. Distral: Robust multitask reinforcement learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[132] A. Tlili and S. Chikhi. Risks analyzing and management in software project management using fuzzy cognitive maps with reinforcement learning. *Informatica*, 45(1), 2021.

[133] S. Tschiatschek, A. Singla, and A. Krause. Selecting sequences of items via submodular maximization. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 2667–2673, 2017.

[134] S. Vargas, L. Baltrunas, A. Karatzoglou, and P. Castells. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 209–216, 2014.

[135] M. Volkovs, H. Rai, G. Cheng, Z.and Wu, Y. Lu, and S. Sanner. Two-stage model for automatic playlist continuation at scale. In *Proceedings of the ACM Recommender Systems Challenge*, pages 1–6. 2018.

[136] D. Wang, S. Deng, and G. Xu. Sequence-based context-aware music recommendation. *Information Retrieval Journal*, 21(2-3):230–252, 2018.

[137] C. JCH Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[138] H. Wei, G. Zheng, H. Yao, and Z. Li. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2496–2505, 2018.

[139] P. Wessel and W. H. F. Smith. A global, self-consistent, hierarchical, high-resolution shoreline database. *Journal of Geophysical Research*, 1996.

[140] C. Yu, L. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 368–378, 2009.

[141] Y. Yu. Towards sample efficient reinforcement learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 5739–5743, 2018.

[142] M. Zhang and N. Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 123–130, 2008.

[143] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng. QASCA: A quality-aware task assignment system for crowdsourcing applications. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1031–1046, 2015.

[144] P. Zhou and H. Shen. Multi-agent cooperation by reinforcement learning with teammate modeling and reward allotment. In *International Conference on Fuzzy Systems and Knowledge Discovery*, volume 2, pages 1316–1319. IEEE, 2011.

[145] X. Zhou, P. Wu, H. Zhang, W. Guo, and Y. Liu. Learn to navigate: cooperative path planning for unmanned surface vehicles using deep reinforcement learning. *IEEE Access*, 7:165262–165278, 2019.

[146] C. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web, Chiba, Japan, May 10-14, 2005*, pages 22–32, 2005.