Guided Task Planning Under Complex Constraints

Sepideh Nikookar, Paras Sakharkar, Baljinder Smagh
NJIT, NJ, USA
{sn627, ps863, bks46}@njit.edu

Sihem Amer-Yahia
CNRS, University Grenoble Alpes, France
sihem.amer-yahia@univ-grenoble-alpes.fr

Senjuti Basu Roy
NJIT, NJ, USA
senjutib@njit.edu

***Abstract*—Creating a plan, i.e., composing a sequence of items to achieve a task is inherently complex if done manually. This requires not only finding a sequence of relevant items but also understanding user requirements and incorporating them as constraints. For instance, in course planning, items are core and elective courses, and degree requirements capture their complex dependencies as constraints. In trip planning, items are points of interest (POIs) and constraints represent time and monetary budget, two user-specified requirements. Most importantly, a plan must comply with the ideal interleaving of items to achieve a goal such as enhancing students' skills towards the broader learning goal of an education program, or in the travel scenario, improving the overall user experience.**

We study the Task Planning Problem (TPP) 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 items, user requirements, and satisfaction. We present a computational framework `RL-Planner` for TPP. `RL-Planner` requires minimal input from domain experts (academic advisors for courses, or travel agents for trips), yet produces personalized plans satisfying all constraints. 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.

I. Introduction

Task planning [1]–[4] 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 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 paper, 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 [1], [5]–[8] 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 [9] and vacation rentals [1]) are ideal for our problem.

Example 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 on 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 Network, Linear System*), 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

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

the student. How to design the ideal sequence of core and elective courses and interleave them (e.g., start with one or two core courses, then take two electives, then another core course) requires domain expertise that only academic advisors are capable of providing.

Example 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 Architectures 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 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 multitude 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 Network, 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 (Section II-A 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) [10]. 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 III-A. 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 V** contains further details.

2. Solving TPP by adapting Reinforcement Learning: Our second contribution is to present a computational framework (Section III-C), that is inspired by Constrained Reinforcement Learning (C-RL) [11], [12], specifically Weighted RL [13], 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 [14] for updating the Q values of the states, which is known to converge faster and with fewer errors [15].

3. Experimental Evaluation: Our third contribution is an extensive evaluation (Section IV) using real-world datasets in the education and travel domains. We use two datasets to plan courses for 4 different sought after degree programs and one dataset to plan 2 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* [16]) and to next-step recommendation in *EDA* [17]; (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 VI.

II. Task Planning Problem

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

A. 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.

1) **Item**

Formally, an item m is represented as a quadruple

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

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

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

TABLE I: Table of Notations

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 item. 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.

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) 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.

Problem 1: 4) **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 $\mathcal{P}_{soft}$.

B. Instantiation of the Framework

1) Course Planning

Item. For course planning, $type^m$ represents if a course is a core vs. elective. cr^m represents the 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 consists of 6 courses, as presented in Table II. 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, 5 primary and 5 secondary courses. The integer constant *gap* specifies the *distance* between m and pre^m in the recommended sequence: if a student takes 3 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 1, *Classification, Clustering, Neural Network, and Linear System* are the topics that the student wishes to learn. Therefore, for Example 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 3 permutations of primary (core courses) and secondary items (elective courses) listed in Table II:

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

Using Example 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.

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 8 topics/themes: Museum, Art Gallery, Cathedral, Palace, River, Street, Restaurant, 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 8 topics.

Hard Constraints. For the trip planning, a day long trip in Paris to visit 2 primary and 3 secondary POIs, with a total visit time of 6 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 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 soft constraints 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 2, Louvre Museum $\rightarrow$ Le Cinq $\rightarrow$ Eiffel Tower $\rightarrow$ Rue des Martyrs $\rightarrow$ River Seine is a sequence that fully satisfies the permutation I_1 of the aforementioned IT.

III. 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.

A. Modeling TPP

In general, a Constrained MDP (CMDP) [10] is designed to learn a policy of an agent with the goal to

$$max_{\pi} E^{\pi}[R(H)] \text{ s.t. } E^{\pi}[D(H)] \leq c \tag{1}$$

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 [12], [13] (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 6 hours, the itinerary will be terminated if the total visitation time exceeds 6 hours. Thus $H = 6$.

B. 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},

$$R(s_i, e_i, s_{i+1}) = \theta \times [\delta \times AvgSim(s_{i+1}, IT_{i+1}) + \beta \times weight_{type^m}] \tag{2}$$

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

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

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

We also look at the scenario where our reward function (Equation 2) uses minimum similarity rather than average similarity.

We now describe the different components of this overall reward.

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 $|T^{ideal} \bigcap \{\mathcal{T}^{current}_{i+1} \setminus \mathcal{T}^{current}_{i}\}| \geq \epsilon$, 0 otherwise.

Formally speaking,

$$r_1 = \begin{cases} 1, & |T^{ideal} \bigcap \{T^{current}_{i+1} \setminus T^{current}_{i}\}| \geq \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$CourseId$	$CourseName$	$type^m$	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]

TABLE II: Course Information

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 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.

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 the 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,

$$r_2 = \begin{cases} 1, & Dist(pre^m, m) \geq gap \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

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

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

3) Combining Topic Coverage and Prerequisite Reward

$\theta = 1$ if the conditions on topic coverage AND antecedents are satisfied, and 0, otherwise. Therefore,

$$\theta = r_1 \times r_2 \tag{5}$$

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* [18]. 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 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 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 consecutive match ($\zeta \in [0, k]$) and normalize by dividing it by k.

$$Sim(s, I)^k = \frac{\zeta \times \sum_{\forall c} length(c_I)}{k} \tag{6}$$

Finally,

$$AvgSim(s, IT)^k = \frac{\sum\limits_{\forall I \in IT} Sim(s, I)^k}{|IT|} \tag{7}$$

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

Theorem 1: The designed reward function (Equation 2) satisfies $\mathcal{P}_{hard}$ of the **TPP**.

Proof: (Sketch) We note that there are actually 4 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 Section III-A 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:

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

Case I. Is consistent, as a core course could be construed as an elective, thereby satisfies 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 denoting a higher preference for 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 2: **(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.

$$\pi^* = argmax_{\pi} E[\sum_{i=1}^{H} R(s_i, e_i, s_{i+1})|\pi] \tag{8}$$

C. 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 [19], or Monte Carlo method, or more popular Temporal Difference based approach which is a combination of Monte Carlo and Dynamic Programming [14], [20], [21]. 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 [22] that policy iteration is computationally more efficient and requires a smaller number of iterations to converge. So, we adapt model-free RL [14], [20], [21] 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 gets periodically updated as the agent learns from the environment based on the reward function (recall Equation 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 items (except for the ones that are chosen already) given our complete graph $\mathcal{G}$.

We use the popular SARSA algorithm [14] 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 9.

$$Q(s_i, e_i) \leftarrow Q(s_i, e_i) + \alpha[r_{i+1} + \gamma Q(s_{i+1}, e_{i+1}) - Q(s_i, e_i)] \tag{9}$$

where the reward r_{i+1} is computed using Equation 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 1.

IV. 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 2.4 GHz Quad-Core Intel Core i5 Processor and 16 GB RAM. Our code and data are available on GitHub.[2]

A. 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?

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

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

TABLE III: `RL-Planner` Default Parameters Values

Algorithm 1 Algorithm in `RL-Planner`

```
1: **Learning Policy:**
Require: $\mathcal{P}_{hard}$, $\mathcal{P}_{soft}$, $\mathcal{G} = (\mathcal{I}, E)$, number of episodes $N$, size of an episode $H$, $\alpha$, $\gamma$
Ensure: a policy $\pi$ satisfying $\mathcal{P}_{hard}$
2: for $i \leftarrow 1$ to $N$ do
3:     $s_i \leftarrow m$
4:     $e_i \leftarrow argmax_{\forall md, d \in \{\mathcal{I}-m\}} R(m, md, d)$
5:     $r \leftarrow Equation\ 2$
6:     $W \leftarrow \{m \bigcup d\}$
7:     $s_j \leftarrow d$
8:     for $j \leftarrow 2$ to H do
9:         $e_j \leftarrow argmax_{\forall s_j w, w \in \{\mathcal{I}-W\}} R(s_j, s_j w, w)$
10:        $W \leftarrow \{W \bigcup w\}$
11:        $Q(s_j, e_j) \leftarrow Q(s_j, e_j) + \alpha[r_{j+1} + \gamma Q(s_{j+1}, e_{j+1}) - Q(s_j, e_j)]$
12:        $s_j \leftarrow s_{j+1}$
13:        $e_j \leftarrow e_{j+1}$
14: Return $Q$
15: **Recommending a plan:**
Require: policy $\pi$, a starting item $m$
Ensure: A sequence $rec$ of $H$ items starting with $m$
16: $s_i \leftarrow m$
17: $W \leftarrow \{m\}$
18: $rec \leftarrow m$
19: for $i \leftarrow 2$ to $H$ do
20:     $e_i \leftarrow argmax_{\forall j \in Q(s_i, j)} Q(s_i, j)$
21:     $rec \leftarrow [rec \rightarrow s_{i+1}]$
22:     $W \leftarrow \{W \bigcup s_{i+1}\}$
23:     $s_i \leftarrow s_{i+1}$
24: Sequence of items in $rec$
```

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 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.

1) Datasets.

Course Planning: We consider datasets extracted from the NJIT (Univ-1) and Stanford (Univ-2) websites. The Univ-1 (NJIT) dataset contains 1216 courses comprising 126 degree programs through 6 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 4 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 units constraints in the following 6 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 the data science. The actual number of courses per program are 31, 30, 32, 36 for DS-CT for NJIT, MS Cybersecurity NJIT, MS CS NJIT, 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 the CS. We obtain 73 topics from Univ-2.

Trip Planning: Akin to a prior work of ours [23], 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 are 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 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.

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 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 for transfer learning.

1. *OMEGA*: To the best of our knowledge, the existing sequence recommendation algorithms leverage historical data. For the purpose of comparison, we chose a recent sequence recommendation algorithm *OMEGA* [16], that leverages 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.

2. *EDA*: To the best of our knowledge, there does not exist any exploratory data analysis (*EDA*) based solution [17] that satisfies a multitude of complex constraints such as ours. We adapt the *EDA* paradigm by implementing a greedy method that chooses the action with the highest reward based on Equation 2 in each step. If two actions provide the same result, one will be picked at random.

3) Default Parameter Settings.

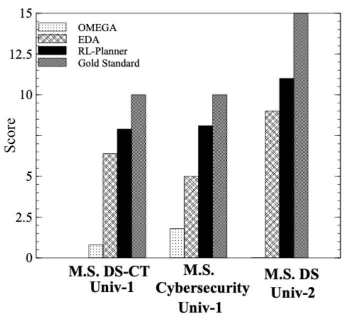
Table III contains default values for all parameters of our model. After consultation with students and academic advisors $|\mathcal{T}^{ideal}|$ is set to 60 for DS-CT, 61 for Cybersecurity, 100 for M.S. CS for Univ-1, and 73 for Univ-2 M.S. DS. For NYC and Paris, these are 21 and 16, respectively.

4) Summary of Results.

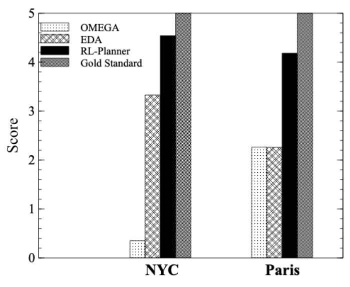
Our results demonstrate that: (a) Existing fully automated approaches are not capable to adapt to sequence recommendations with a multitude of complex constraints. Both *OMEGA* [16] and *EDA* are unable to generate course plans and trip plans that satisfy the hard constraints most of the time, `RL-Planner` 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 the 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 2), and `RL-Planner` outperforms all baselines in both scenarios. 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.

B. 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 Section IV-A2. Figure 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 lower scores compared to our proposed solutions.



(a) Course Planning



(b) Trip Planning

Fig. 1: `RL-Planner`, *OMEGA*, *EDA*, and Gold Standa

C. User Studies

We measure user satisfaction of the sequence generated by `RL-Planner` against the gold standard. *OMEGA* and *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 (5 for NYC and 5 for Paris) generated by domain experts and by `RL-Planner` by involving 50 unique AMT workers. Each itinerary is validated by 5 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

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

TABLE IV: Average Ratings: `RL-Planner` User Study

standard without revealing which one is which. Each volunteer is asked to provide a rating for 4 questions on a scale of $1-5$ with 5 being the best. Our results are summarized in Table IV. `RL-Planner` produces course/trip plans that are highly comparable to the gold standards across all 4 questions, demonstrating its effectiveness.

D. Case Study: Transfer Learning

We present the effectiveness of `RL-Planner` in transfer learning through a small number of case studies using 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 V presents the results (the full mapping to course titles is in Table VI). 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 to Paris and vice versa. The results are shown in Table VII. 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 VIII 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.

E. Robustness of `RL-Planner`

We vary one parameter at a time while all other parameters are kept at the default values (see Table III). 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 X, IX, XI and Tables XII, XIII, XIV for Univ-1 and Univ-2, respectively. Tables XV, XVI 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.

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

Course Planning: For Univ-1 (Table XI), we notice that starting with any of the acceptable starting core courses, has minimal impact on the performance of the 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 the 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 XII, XIII, XIV). We observe that `RL-Planner` performs well for all parameter values that are 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 XV) does not have high impact on the final score and the results are stable with respect to reward's weights (δ, β).

F. 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 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 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.

V. Related Work

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

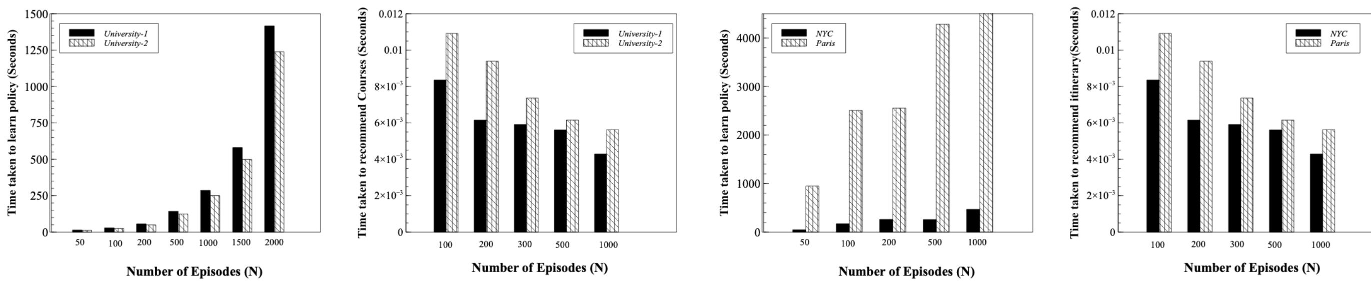
A. Sequence Recommendation

Sequence recommendation is explored in [16], [24], [25] , [26] and [27]. In [24], the authors have recently developed a computational framework to generate a sequence of sessions to improve user satisfaction in the web applications. In [16], a Directed Acyclic Graph is used to model sequential dependencies and a novel class of utility functions are 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 [25]. However, both [16] and [25] do not account for constraints or prerequisites when producing a sequence of items, which is crucial for **TPP**.

In [28], the authors 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 [29] that leverages Convolutional Neural Network for capturing both general preferences and sequential patterns. SASRec [30] balances Markov Chains and Recurrent Neural Network approaches to make sequence recommendations.

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

TABLE V: `RL-Planner` for Course Planning: Transfer Learning between M.S. CS and M.S. DS-CT



(a) Time to Learn Policy (b) Time to Recommend Course Plan (c) Time to Learn Policy (d) Time to recommend Trip Plan

Fig. 2: `RL-Planner` Scalability Results

	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 VI: Course IDs & Descriptions for `RL-Planner`

In [26], the authors 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 [27], 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.

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

B. Course Planning

The use of recommendation systems for course predictions for students has been extensively studied in [1], [2], [7], [31]–[37]. CourseRank [2], 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. The authors 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 acquires expertise on a topic whilst also completing their degree. In [1], the authors 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 [31] 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 [35] using collaborative filtering and popularity ranking, and a follow up work proposes a hybrid of the Random Forest and Matrix Factorization in [36] for this. In [37], the authors 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, unlike our work.

C. Trip Planning

How to recommend trips that are personalized have been studied in many recent related works [4], [5], [8], [38]–[40], including our own [3]. Unlike our solutions, most of these do

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

TABLE VII: `RL-Planner` for Trip Planning: Transfer Learning between NYC and Paris

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 VIII: `RL-Planner` for Trip Planning : Itinerary Description

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	—		—		—		—		—	

TABLE IX: `RL-Planner` vs. *EDA*: 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 X: `RL-Planner` 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 XI: `RL-Planner` vs. *EDA*: Parameter Tuning Results Univ-1 M.S. DS-CT

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**

TABLE XII: `RL-Planner` vs. *EDA*: 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 XIII: `RL-Planner` 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 XIV: `RL-Planner` vs. *EDA*: Parameter Tuning Results Univ-2 M.S. DS

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 XV: `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	

TABLE XVI: `RL-Planner` vs. *EDA*: Parameter Tuning Results Trip Planning

not account for a multitude of constraints, except for [38]. 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 accordingly.

From the perspective of solution design, the existing approach designs combinatorial solutions, heavily rely on logs or training data to model user preference, unlike our work.

D. RL and Guided EDA

Our guided task planning bears similarity to guiding users in performing Exploratory Data Analysis (EDA), a well-studied problem [41]–[44].

Numerous works proposed next-step recommendations [17], by using logs of previous operations (e.g., [45]), or by relying on real-time feedback [46], [47]. The fully automated generation of EDA sessions has been examined in [21], [48]. The use of Reinforcement Learning (RL) for recommending user groups or sequences of EDA techniques has been documented in [48] and [21] respectively. In [48], 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 [21], 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 [49], the authors 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 adapt a next-step EDA to create a baseline for our experiments.

The closest to our work is the area of safe Reinforcement Learning [42], 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 a reasonable system performance and/or respect safety constraints. In particular, our work borrows inspiration from Constrained MDPs [11], [13] , for which linear programming, weighted sum, state space extension, and recursive formulation of the constraints based approaches are known. In [11], 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 to the weighted approach, although traditional weighted RLs do not have to deal with multiple soft and hard constraints, such as **TPP**.

VI. 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 rating (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.

Acknowledgment

The work of Sepideh Nikookar, Paras Sakharkar, Baljinder Smagh, and Senjuti Basu Roy are supported by the NSF, CAREER Award #1942913, IIS #2007935, IIS #1814595, PPoSS: Planning #2118458, and by the Office of Naval Research Grants No, N000141812838, N000142112966.

References

[1] A. Parameswaran *et al.*, "Recommendation systems with complex constraints: A course recommendation perspective," *TOIS*, 2011.

[2] B. Bercovitz *et al.*, "Courserank: a social system for course planning," in *SIGMOD*, 2009, pp. 1107–1110.

[3] S. B. Roy *et al.*, "Interactive itinerary planning," in *ICDE*, 2011, pp. 15–26.

[4] S. Amer-Yahia *et al.*, "Grouptravel: Customizing travel packages for groups," in *EDBT*, 2019, pp. 133–144.

[5] M. D. Choudhury *et al.*, "Automatic construction of travel itineraries using social breadcrumbs," in *Hypertext*, 2010, pp. 35–44.

[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*, vol. 51, pp. 1–14, 2013.

[7] K.-K. Chu *et al.*, "Designing a course recommendation system on web based on the students' course selection records," in *EdMedia+ Innovate Learning*. AACE, 2003, pp. 14–21.

[8] Z. Friggstad *et al.*, "Orienteering algorithms for generating travel itineraries," in *WSDM*, 2018, pp. 180–188.

[9] C. Severance, "Teaching the world: Daphne koller and coursera," *Computer*, vol. 45, no. 8, pp. 8–9, 2012.

[10] E. Altman, *Constrained Markov decision processes*. CRC Press, 1999, vol. 7.

[11] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *JAIR*, vol. 24, pp. 81–108, 2005.

[12] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *ICML*, 2017.

[13] P. Geibel, "Reinforcement learning for mdps with constraints," in *European Conference on Machine Learning*. Springer, 2006, pp. 646–653.

[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[15] S. Ravichandiran, *Hands-on Reinforcement Learning with Python: Master Reinforcement and Deep Reinforcement Learning Using OpenAI Gym and TensorFlow*. Packt Publishing Ltd, 2018.

[16] S. Tschiatschek, A. Singla, and A. Krause, "Selecting sequences of items via submodular maximization." in *AAAI*, 2017, pp. 2667–2673.

[17] T. Milo and A. Somech, "Next-step suggestions for modern interactive data analysis platforms," in *SIGKDD*, 2018, pp. 576–585.

[18] F. P. Miller *et al.*, "Levenshtein distance: Information theory," 2009.

[19] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[20] M. Seleznova *et al.*, "Guided exploration of user groups," *PVLDB*, 2020.

[21] O. Bar El *et al.*, "Automatically generating data exploration sessions using deep reinforcement learning," in *SIGMOD*, 2020, pp. 1527–1537.

[22] E. Pashenkova and Others, "Value iteration and policy iteration algorithms for markov decision problem," in *AAAI Workshop*, 1996.

[23] S. B. Roy *et al.*, "Interactive itinerary planning," in *ICDE*. IEEE, 2011, pp. 15–26.

[24] M. 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*, 2021, pp. 1928–1936.

[25] I. Benouaret *et al.*, "An efficient greedy algorithm for sequence recommendation," in *DEXA*. Springer, 2019, pp. 314–326.

[26] M. Quadrana, P. Cremonesi, and D. Jannach, "Sequence-aware recommender systems," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[27] M. Stratigi, E. Pitoura, J. Nummenmaa, and K. Stefanidis, "Sequential group recommendations based on satisfaction and disagreement scores," *Journal of Intelligent Information Systems*, pp. 1–28, 2021.

[28] S. Jiang *et al.*, "Personalized travel sequence recommendation on multi-source big social media," *IEEE Transactions on Big Data*, vol. 2, no. 1, pp. 43–56, 2016.

[29] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *ICDM*, 2018, pp. 565–573.

[30] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *ICDM*, 2018.

[31] W. Chen, Z. Niu, X. Zhao, and Y. Li, "A hybrid recommendation algorithm adapted in e-learning environments," *World Wide Web*, vol. 17, no. 2, pp. 271–284, 2014.

[32] W. Jiang, Z. A. Pardos, and Q. Wei, "Goal-based course recommendation," in *Proceedings of the 9th International Conference on Learning Analytics & Knowledge*, 2019, pp. 36–45.

[33] G. Engin *et al.*, "Rule-based expert systems for supporting university students," *Procedia Computer Science*, vol. 31, pp. 22–31, 2014.

[34] N. Bendakir *et al.*, "Using association rules for course recommendation," in *AAAI Workshop*, vol. 3, 2006, pp. 1–10.

[35] A. Elbadrawy and G. Karypis, "Domain-aware grade prediction and top-n course recommendation."

[36] M. Sweeney *et al.*, "Next-term student performance prediction: A recommender systems approach," *arXiv preprint arXiv:1604.01840*, 2016.

[37] M. Ebner *et al.*, "First steps towards an integrated personal learning environment at the university level," in *ICTL*, 2011.

[38] A. Gionis *et al.*, "Customized tour recommendations in urban areas," in *WSDM*, 2014, pp. 313–322.

[39] H. Chang *et al.*, "ATIPS: automatic travel itinerary planning system for domestic areas," *Comput. Intell. Neurosci.*, vol. 2016, pp. 1 281 379:1–1 281 379:13, 2016.

[40] Z. Friggstad *et al.*, "Orienteering algorithms for generating travel itineraries," in *WSDM*, 2018, pp. 180–188.

[41] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[42] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *JMLR*, vol. 16, no. 1, pp. 1437–1480, 2015.

[43] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *SIGKDD*, 2018, pp. 2496–2505.

[44] 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*, 2002, pp. 259–268.

[45] M. Eirinaki, S. Abraham, N. Polyzotis, and N. Shaikh, "Querie: Collaborative database exploration," *IEEE Transactions on knowledge and data engineering*, vol. 26, no. 7, pp. 1778–1790, 2013.

[46] K. Dimitriadou, O. Papaemmanouil, and Y. Diao, "Aide: an active learning-based approach for interactive data exploration," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 11, pp. 2842–2856, 2016.

[47] E. Huang, L. Peng, L. Di Palma, A. Abdelkafi, A. Liu, and Y. Diao, "Optimization for active learning-based interactive database exploration," Ph.D. dissertation, Ecole Polytechnique; University of Massachusetts Amherst, 2018.

[48] M. Seleznova, B. Omidvar-Tehrani, S. Amer-Yahia, and E. Simon, "Guided exploration of user groups," *Proc. VLDB Endow.*, vol. 13, no. 9, pp. 1469–1482, 2020.

[49] S. Narvekar, J. Sinapov, and P. Stone, "Autonomous task sequencing for customized curriculum design in reinforcement learning." in *IJCAI*, 2017, pp. 2536–2542.