Accepted for publication in Knowledge and Information Systems

Efficient Itinerary Recommendation via Personalized POI Selection and Pruning

Sajal Halder[*], Kwan Hui Lim[†], Jeffrey Chan[*], Xiuzhen Zhang[*]
[*]School of Computing Technologies, RMIT University, Australia
[†] Singapore University of Technology and Design, Singapore
sajal.halder@student.rmit.edu.au,kwanhui_lim@sutd.edu.sg,
jeffrey.chan@rmit.edu.au,xiuzhen.zhang@rmit.edu.au

Abstract. Personalized itinerary recommendation has garnered wide research interests for their ubiquitous applications. Recommending personalized itineraries is complex because of the large number of points of interest (POI) to consider in order to construct an itinerary based on visitors' interest and preference, time budget, and uncertain queuing time. Previous studies typically aim to plan itineraries that maximize POI popularity, visitors' interest and minimize queuing time. However, existing solutions may not reflect visitor preferences because when creating itineraries, they prefer to recommend POIs with short prior visiting periods. These recommendations can conflict with real-life scenarios as visitors typically spend less time at POIs that they do not enjoy, thus leading to the inclusion of unsuitable POIs. Moreover, constructing itineraries based on selected POIs is a challenging and time-consuming process. Existing approaches involve searching through a large number of non-optimal, duplicate itineraries that are time-consuming to review and generate. To address these issues, we propose an adaptive Monte Carlo Tree Search (MCTS) based reinforcement learning algorithm *EffiTourRec* using an effective POI selection strategy by giving preference to POIs with long visiting times and short queuing times along with high POI popularity and visitor interest. In addition, to reduce non-optimal and duplicated itineraries generation, we propose an efficient MCTS search pruning technique to explore a smaller, more promising portion of solution space. Experiment results in real theme park datasets show clear advantages of our proposed method over baselines, where our method outperforms the current state-of-the-art by 20.89% to 52.32% in precision, 8.36% to 21.35% in F1-score and 40.00% to 67.64% in execution time.

Keywords: Itinerary Recommendation, Personalization, Points of Interest, Queuing Time, Search Pruning, Monte Carlo Tree Search.

Received 19 Jan 2020
Revised 13 May 2021
Accepted 26 Nov 2021

1. Introduction

Tourism is one of the popular leisure activities for humans where the aim is to visit interesting attractions in new locations. Visiting all the attractions is typically not possible as visitors tend to have limited time budgets. Thus, a critical task for visitors is to plan an itinerary that contains popular and interesting POIs, which must be completed within the visitor's specific time limit. This personalized tour planning is complex due to various constraints: (i) the degree to which a visitor has a preference for popular POIs, POIs aligned to his/her individual interests or some combination of popularity and interest-aligned POIs; (ii) visitors do not like to queue, but if they have to, they prefer the POI/attraction to last longer than conveys the earlier visitors interest; and (iii) visitor has limited time budget to complete the tour. In particular, neglecting queuing time and solely maximizing visitor's interest can create a frustrating experience for visitors as they spend an unnecessarily long time queuing rather than enjoying the attractions. In contrast, if visitors focus mainly on POIs with short queuing times, this can also create a frustrating experience and possibly miss preferred attractions in their itineraries. Considering these issues, it is difficult to manually construct suitable itineraries that satisfy these constraints. Therefore, an efficient and personalized tour/ itinerary recommendation approach is needed to maximize visitor's preferences, POI popularity and minimize queuing time within the visitor's time budget.

We will repeatedly use three kinds of time duration terminology to introduce these important concepts: queuing time, visiting time, and traveling time in Fig. 1. Queuing time means the time to wait to enter a POI, visiting time means the time visitors spend at the POI after getting access and traveling time measures the amount of time to travel from a POI to another.

Previous research has focused on developing itinerary recommendation algorithms based on an optimally-scheduled path based on POI popularity [48], group pleasure [20, 38], mandatory POI categories [5, 35], demographic features [6], and geographical check-in impact [8]. The prior studies [40] and [12] show that travelling and visiting time are important factors for tour planning. However, these approaches did not consider the queuing time of POIs. Considering queuing time, Lim et al. [36] proposed a Monte Carlo Tree Search (MCTS) based algorithm whose objective is maximizing the POI popularity, user interest, and minimizing the queuing time. However, they prefer a short visiting time POI that indicates earlier visitors do not like it very much, which may lead to inappropriate POI recommendations when there is queuing to access POIs.

As an illustration, Fig. 2(a) shows the POIs average visiting times of California Adventure Theme Park in the US. The results suggest that visitors prefer POIs with long visiting times rather than ones with short visiting times. As a proxy of user's interests, we use the number of visitors to gauge the general interest of each POI.

Fig. 2(b) shows the Pearson correlation between the number of visitors and visiting time of each POIs. The correlation coefficient is 0.77, which means the number of visitors and visiting time are positively correlated. Thus, visiting time could be a consideration for visitors to prefer specific POIs. We hypothesize that

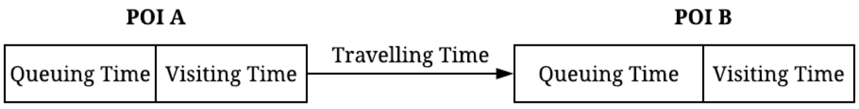
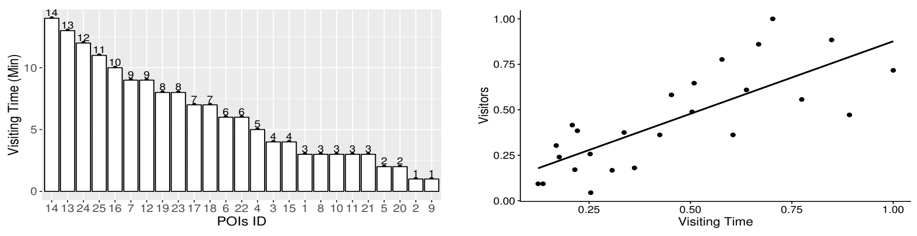


Fig. 1. Example of various time duration's.



(a) Average visiting time of POIs

(b) Correlation between number of visitors and visiting time

Fig. 2. Illustration of prior visitors visiting time influence of each POI

visitors have to wait in a queue to enter a POI. In the California Adventure theme park scenario, visitors have to spend on average more than 60% time as queuing time. In research on the psychology of waiting in queues [43, 9], people will wait longer for more valuable service. If the queue times are roughly the same across POIs, then all things being equal, they prefer the POI with a longer average visiting time to justify the wait. To consider this psychology, in our proposed method, longer visiting POIs has been preferred in the itinerary recommendation along with shorter queuing time.

Previous methods [2, 35, 36] directly or indirectly preferred to recommend POIs with shorter prior visitor visit duration, as they aim to visit as many POIs in the time budget to maximize the predicted interest and popularity. However, in light of our analysis, it suggests this might lead to POIs, which are of less interest to visitors. In addition, previous methods consider all POIs with the same significance in their evaluation metrics, i.e., precision, recall, and F1-score calculation, because all these metrics count the number of POIs. They do not consider visiting time, thus in Fig. 2(a), POI 9 and POI 14, whose visiting time is one minute and fourteen minutes, respectively. They are considered equal in desirability. In addition, visitors have to endure long queuing times before being able to enjoy the attraction/ride. However, selecting POIs with shorter visiting times may increase recall value, but it can use a large proportion of time on queuing, which can make visitors bored. In this paper, we have given special consideration to visiting time that results in our approach may recommend less number of POIs to the visitors within budget time than the existing research. However, these POIs are likely of higher interest to visitors as our evaluation shows that our precision and F1-scores, computed against ground truths, are significantly higher than other works.

The problem of personalized itinerary recommendation considering visiting and queuing time involves optimizing over POIs across time. The current approach of using integer programming to optimize is not sufficient due to its long-running times. This paper proposes *EffiTourRec* algorithm to construct POI itineraries for the visitors based on enhanced MCTS with an effective heuristic and efficient pruning technique. An effective heuristic is introduced that suggests POIs optimized to the visitor's interest, POI popularity, prior visitors visiting time, and queuing time. To reduce non-optimal itineraries generation, we introduced a reward-based pruning strategy that makes our approach efficient than the existing algorithms. The main contributions of this paper are discussed as follows:

- We propose a new POI selection heuristic for personalized itinerary recommendation by considering visitors' various preferences, i.e., POIs popularity, visitors interest, prior visitors visiting, queuing, and traveling time.
- We introduce an effective itinerary reward function-based level of trade-off among prior visitors visiting time, POI popularity, visitor interests, and queuing time.
- We propose adaptive MCTS based reinforcement learning algorithm *EffiTourRec* to make visitor itinerary recommendation more effective and efficient by using the new POI selection heuristic and reward function strategy.
- In addition, to improve the time efficiency of our proposed method, we use MCTS pruning technique to reduce search space by filtering out non-optimal and duplicate itineraries in the early stages.
- We evaluate our proposed algorithm on five theme parks datasets to show our proposed method's effectiveness and efficiency against various state-of-the-art baselines. We test on theme parks datasets because that is what [36] did to compare the itinerary recommendation with queues.
- Experiment results demonstrate the effectiveness of our proposed method over baselines; our method outperforms the current state-of-the-art by 21.04% to 50.24% in precision, 7.83% to 21.23% in F1-score, 8.36% to 43.96% in visiting time ratio with total time. It also shows the efficiency of our proposed *EffiTourRec* algorithm outperforms than the baselines by reducing 51.62% to 66.59% execution time and 47.49% to 61.99% moves in MCTS compared to the best baseline *PersQ* algorithm.

The remaining parts of this paper are organized as follows. Section 2 describes the related works while some preliminary concepts are discussed in Section 3. Problem definition is formulated in Section 4. Our proposed *EffiTourRec* method is described in Section 5. Experiments to evaluate the proposed method's performance compared with the existing baselines are demonstrated in Section 6. Finally, we conclude our proposed method with some future directions in Section 7.

2. Related Work

Personalized itinerary recommendation research has recently attracted significant attention due to its various applications. There are a variety of existing research works that cover different aspects of the itinerary recommendation problem. In the following subsections, we discuss various related works and highlight the difference between our work and the existing works.

2.1. General Itinerary Recommendation

Many existing research on itinerary recommendation [10, 11, 33] are based on Orienteering Problem (OP) [49, 22] whose main aim is to maximize a global reward point within user-defined time budget. POI popularity is frequently used as a global reward in theme park [23, 51] and city [24, 33] itinerary recommendation. Many significant tourism-related works utilize geo-tagged photos [27, 10] for identifying popular POIs to analyzing tourist interest. While these recommendations are interesting, they do not use visitor's personalized interest preferences as a fact of global reward points. Yoon et al. [54] introduced an efficient and balance intelligent tour recommendation using global positioning system (GPS) itineraries. Lim et al. [37] described a social media data-based tour recommendation and itinerary planning models in detail as a survey study.

2.2. Personalized Itinerary Recommendation

Research in itinerary recommendations has focused on discovering different types of itinerary recommendations based on the impacts of various constraints. These existing works' objectives are to recommend itineraries based on visitors' interest preference [50], particular POI visit order [21], group pleasure [20, 38], mandatory POI categories [5, 35], demographic features [6], geographical check-in impact [8], etc. Lim et al. [40] introduced *PersTour* system for personalized itinerary recommendations based on trip constraints and visitor interest preferences using modified Ant Colony Optimization [15] algorithm. Debnath et al. [12] presented a time-aware and preference-aware routes recommendation system. Quan et al. [18] presented a spatial-temporal context-aware mixture model to a user within a geospatial range. Travel time is one of the important factors of tour planning. To focus on travel time, Irina et al. [14] designed an adaptive orienteering problem with stochastic travel times (AOPST) that finds the path between the reward POIs in an integral component of the decision space. Time constraints-based framework *pirT* [25] proposed a personalized itinerary in which social network features and social relationships are used to define visitor preference. Zhixue et al. [34] introduced a hybrid heuristic-based $RS\text{-}H^2A$ algorithm that applies a random simulation-based hybrid evolution strategy in a time-dependent stochastic environment to handle risk awareness of the tourists. A collaborative filtering-based *WMF-CR* [46] approach has been proposed to the personalized landmark nontrivial recommendations using geo-tagged photos. Bowen et al. [16] developed a crowdedness-aware route recommendation model for predicting the passenger transfer volumes of a specific location at a specific time duration. These itinerary recommendation approaches do not consider the queuing time of attractions.

2.3. Personalized Itinerary Recommendation with Queuing time Awareness

The queuing time has significant effects on a personalized recommendation system where a visitor has to wait a long time to get rides, i.e., a theme park tour. The theme park ride access requires a long waiting time which can generate a frustrating experience for the users. Lim et al. [36] incorporated queuing time to geo-tagged photos and proposed the *PersQ* algorithm by modifying the Monte Carlo Tree Search (MCTS) for recommending personalized itineraries. The *PersQ* algorithm's objectives are maximizing the POI popularity, user interest and minimizing the queuing time. However, the method shows that recommended POIs are inversely proportioned to prior visitor's visit duration. In the actual scenario, POIs visit duration expresses visitors' interest that should be proportion for POIs selection of an itinerary.

The *PersQ* [36] algorithm first used MCTS in itinerary recommendation like single-player game [42]. Before that, most of the researchers applied Monte Carlo Tree Search for two-player games. MCTS space exponentially increases with the number of iterations and nodes in the tree. Different kinds of pruning techniques: probability-based [44, 17], heuristic-based [45] reduce MCTS space in two players game. Neil et al. [4] designed a single-player game whose objective is to transform an initial phase into a set of goal conditions phase using automatic move pruning. Although our work is based on MCTS, it differs from the earlier research works. In our work, we consider a different heuristic for potential POI selection and a new reward function for ensuring visitor's preferences in the itinerary recommendation. In addition, our proposed personalized itinerary recommendation system is like one player game, and we apply an efficient pruning technique in adaptive Monte Carlo Tree Search to reduce the tree search space.

2.4. Top-k POI Recommendation

Most of the top-k POI recommendation works are based on collaborative filtering (CF) or matrix factorization approaches. Their main objective is to make a ranked list and recommend top-k POI to the visitors. User-based collaborative filtering (UBCF) for itineraries [53] has been proposed to recommend a set of top-k POIs for visitors considering the social influence and spatial influence. Kotiloglu et al. [32] proposed "Filter-First, Tour-Second" framework where the first phase finds the top-k optional set of POIs using CF [13] that are added to mandatory visited POIs to create a possible recommendation. An iterative heuristic approximation (IHA) [55] method is proposed that makes attractions set based on profits and recommends these attractions to the visitor until the budget time is reached. Hu et al. [26] proposed travelogues and check-in information based multi source data to capture user's interest and find top-ranked itineraries and recommend that to the users.

2.5. Sequences of Locations

A set of locations are ordered to generate an itinerary recommendation where the order of locations is important. Baral et al. [1] proposed a context-aware personalized POI sequence recommendation approach by extending the recurrent neural network and its variants. Multi-source based personalized travel sequence recommendation [28] has been presented, which can recommend a travel sequence rather than individuals POIs using heterogeneous metadata. Lou et al. [41] focused on sentimental characteristics of POIs and then recommended these POIs using SPR algorithm. The geographical position has a remarkable impact on POI recommendation [53] that visitors tend to visit nearby POIs around their homes or office. The research works [7, 52] proposed probability-based recommendations

Table 1. Comparison between proposed algorithm and various baselines, in terms of considering constraints. Here max and min represent maximize and minimize constraints value, respectively.

Algorithms	Popularity based	Interest based	Queue Time	Visit Time	Travel Time	Heuristics based	Construct Itinerary	Pruning Technique
UBCF [53]	✓	✓						
IHA [55]	✓	✓		min	min	Heuristic		
TripBuilder [2]	✓	✓					✓	
pirT [25]	✓	✓		min	min	A*	✓	
Lim et al. [39, 35]	✓	✓		min	min		✓	
PersQ [36]	✓	✓	min	min	min	MCTS	✓	
EffiTourRec	✓	✓	min	max	min	MCTS	✓	✓

that a closer distance between visitors and locations has a higher probability for a recommendation. To explore the impact of spatial, temporal, and social influence, *STSCR* [19] model has been proposed to handle user's behaviors properly in sequential attractions recommendation.

2.6. Discussion of differences with previous works

Our proposed *EffiTourRec* algorithm differs from these previous works in various aspects. Table 1 shows the main features of tour itinerary recommendation works and the significant constraints faced by existing methods. First, unlike top-k POI recommendations, our proposed method recommends a set of POIs and constructs an itinerary considering the traveling time between POIs, visiting time of POI, queuing time of POI, and completing this itinerary within user-defined time budget. Second, in contrast to existing works, our proposed visitor personalized interest is proportional to visitors' prior visit duration (e.g., more spending time means more interest). While existing works' personalized interest is inversely proportional to the prior visit duration (e.g., more spending time means less interest), visitors can board. Third, we formulate a personalized itinerary reward function by giving prior visitors' time-based POI visit preferences. In contrast, existing works did not differentiate long and short POIs visiting time that has significant meaning in itinerary recommendation shown in Fig 2. Fourth, previous algorithms were not concerned about various time ratios with total expending time. We have shown the performance analysis based on various time ratios between our proposed approach and existing baselines. Last but not least, shorter itinerary construction time is an important feature of the itinerary recommendation system because visitors would prefer to get their itinerary recommendations without excessive delays. Thus, our proposed *EffiTourRec* algorithm uses an efficient pruning technique to reduce the search space and makes the system more than 40% time efficient than the existing baseline algorithms.

3. Preliminaries

Suppose that there are some visiting POIs in a city or theme park. To recommend these POIs based on visitor interest, POIs popularity and time constraints, we introduce the following terms, some of which have been defined previously.

Definition 1. Point of Interest (POI): Let a set of tourist points be $P = \{p_1, p_2, p_3, \cdots, p_n\}$ in the theme park or city. Each point $p_i \in P$ can have properties e.g. points area, points category, travel time from other POIs, visiting time and queuing time.

Definition 2. Popularity of POI (PoP): Let a POI attraction be $p_i \in P$, the popularity of p_i is defined as the number of times p_i has been visited by the visitors U and it is defined as:

$$PoP(p_i) = \sum_{u \in U} \delta(u, p_i) \tag{1}$$

where, $\delta(u, p_i) = 1$ if the visitor $u \in U$ visits POI p_i in his/her tour, otherwise $\delta(u, p_i) = 0$.

Nowadays, photo sharing and social media sites provide many avenues for users to share photos of their daily experiences, many of which involve interesting places they have visited. Thus, visitors are interested in taking photos in their preferred attraction and the number of taking pictures reflects the level of interest of any attraction. Although POI popularity is the same for all visitors, the interest relevance of a POI differs from visitor to visitor. Each visitor will have their independent interest preferences. We can find user's interest in particular categorical POIs in three different ways: (i) number of taken photos; (ii) amount of spending time; and (iii) number of repeated visits. Therefore, in this paper, we consider user interest based on the number of taken photos, which has been applied in the baseline algorithm *PersQ* [36]. User's interest feature selection may change the efficiency of the proposed and baselines algorithms. Effective user interest features selection or concatenation approach of multiple features to express users interest may be another research direction. However, this research's main aim is to propose an effective and efficient itinerary recommendation considering similar features like baselines.

Definition 3. Interest of POI (IoP): Suppose that C represents the set of all POI categories and each $p_i \in P$ is associated with a certain category $c \in C$. Furthermore, F is the set of photos taken by visitor u in these POIs. Therefore, the interest level of visitor u in category c is defined as:

$$IoP_u(c) = \frac{1}{|F|} \sum_{q \in F} \lambda(c_q = c), \forall c \in C \tag{2}$$

where, $\lambda(c_q = c) = 1$ if the visitor u takes photo q which category is c_q in a POI that belong to category c, otherwise $\lambda(c_q = c) = 0$. The intuition of this definition is that the visitor takes more photos of a POI (category) if he/she likes it.

Definition 4. Itinerary History (IH): Suppose a visitor u visits k number of POIs, we can represent visitor itinerary history as an order of visiting points sequence,

$$IH_u = \Big((p_1, t^a_{up_1}, t^d_{up_1}), (p_2, t^a_{up_2}, t^d_{up_2}) \cdots, (p_k, t^a_{up_k}, t^d_{up_k})\Big) \tag{3}$$

in which the triplet $(p_i, t^a_{up_i}, t^d_{up_i})$ conveys the visited POI p_i, the arrival time and departure time at POI p_i are $t^a_{up_i}$ and $t^d_{up_i}$, respectively.

Definition 5. Itinerary Sequence: Based on the itinerary history IH_u of a visitor u, we further process this extended travel history into smaller travel sequences. Thus, we divide this itinerary history into multiple itinerary sequences i.e. sub-sequences of IH_u if the travel duration between two constitutive POI is $TDoP(p_x, p_{x+1}) > \tau$. In this paper, we consider $\tau = 8$ hours. For a visitor u with n number of itinerary sequences, we use $IH^1_u, IH^2_u, \cdots, IH^n_u$ that represent travel sequences in temporal order i.e. IH^1_u visited before IH^2_u.

Definition 6. Visit Duration of POI (VDoP): Suppose a visitor $u \in U$ visits $p_i \in P$ at time $t^a_{up_i}$ and leaves at time $t^d_{up_i}$, then the visit duration of visitor u at p_i is $VDoP(u)_{p_i} = t^d_{up_i} - t^a_{up_i}$ which represents one visit duration. For all visitors $W \subseteq U$ who have visited p_i, we can determine the average visit duration of POI p_i as follows.

$$VDoP(p_i) = \frac{1}{|W|} \sum_{u \in W} (t^d_{up_i} - t^a_{up_i}) \tag{4}$$

where $|W|$ is the number of visitors who visit the POI p_i.

Definition 7. Travel Duration of POI (TDoP): Suppose a visitor u completes his/her visit at POI p_q at time $t^d_{up_q}$ and start/arrival visit at POI p_r at time $t^a_{up_r}$, then we define the travel duration of two sequential POI from p_q to p_r as $TDoP(p_q, p_r) = t^a_{up_r} - t^d_{up_q}$.

Definition 8. Queuing time of POI: In general, each POI $p_i \in P$ can also be associated with a queuing time after arriving at POI p_i, e.g., queuing to buy a ticket, to ride a roller coaster, etc. Suppose that visitors $W \subseteq U$ who visit POI p_i and t is the visiting time, then we can designate the queuing time as a function of timestamps t and POI p_i as follows.

$$Queue^t_{p_i} = \frac{1}{|W|} \sum_{u \in W} \mu(p_i)\Big((t^d_{up_i} - t^a_{up_i}) - VDoP_{p_i}\Big), \quad \text{where} \quad t^a_{up_i} \leq t \leq t^d_{up_i} \tag{5}$$

where $\mu(p_i) = 1$ if visitor u visit POI p_i at timestamps t, otherwise $\mu(p_i) = 0$. In short, we find the queuing time at POI p_i based on the total time spent at an attraction subtracted by its average visiting time.

4. Problem Definition

We define the personalized itinerary recommendation problem, with the main objectives of maximizing the popularity, visitor interest, visiting time of each POI visited/recommended, and minimizing the queuing and travel times. As visitor interest is determined by the visitors spending time and visitors taking photos of visiting attractions, it is relevant to construct a personalized itinerary.

We denote this problem as the *EffiTourRec* problem. Another objective is to recommend attractions to the visitor in a shorter time than the existing systems.

Given a set of POIs $P = \{p_1, p_2, p_3, \cdots, p_n\}$, a time budget T, a starting POI p_1 and destination POI p_n. Our main goal is to recommend itinerary $I = \{p_1, \cdots, p_n\}$ that maximize the total reward ensuring the itinerary is completed within the given time budget T. The reward function is calculated based on POI popularity, visitors' interest, visiting time and queuing time of POI p_j as follows.

$$max \sum_{p_i \in I} \sum_{p_j \in I, p_i \neq p_j} Path^t_{p_i,p_j} * \Big(\frac{PoP(p_j) * IoP(c_{p_j}) * VDoP(p_j)}{Queue^t_{p_j}} \Big) \tag{6}$$

where $Path^t_{p_i,p_j} = 1$ if visitor visits p_i and p_j as a sequence in an itinerary history, and $Path^t_{p_i,p_j} = 0$ otherwise. $PoP(p_j)$ represents the popularity of POI p_j, while $IoP(c_{p_j})$ indicates user interest based on number of taken photos in the category of POI p_j and $VDoP(p_j)$ indicates the prior visitors average visiting time of p_j. Moreover, $Queue^t_{p_j}$ conveys average queuing time at timestamp t in POI p_j. The main objective of this research is to maximize popularity, interest, visiting time and minimize queuing time. Therefore, we maximize popularity, visitor interest and prior visiting time as the numerator and queuing time as the denominator in equation 6. To emphasise the balance among popularity, interest and visiting time, we use multiplication among them. The idea is similar to probability multiplication, which is known to work well when different measures are independent and have equivalent value ranges.

Moreover, the following constraints are aggregated to solve the Equation 6.

$$\sum_{p_i \in I, i \neq 1} Path^t_{p_1,p_i} = \sum_{p_j \in I, j \neq n} Path^{t+d}_{p_j,p_n} = 1 \tag{7}$$

Constraint 7 ensures that the recommended itinerary starts at a particular POI p_1 and ends at specific POI p_n.

$$\sum_{p_i \in I, k \neq n} Path^t_{p_i,p_k} \leq 1 \quad \text{and} \quad \sum_{p_j \in I, k \neq 1} Path^{t+d}_{p_k,p_j} \leq 1 \tag{8}$$

Constraint 8 indicates that all selected POIs in an itinerary are connected and no POIs are included more than once.

$$\sum_{p_i \in I} \sum_{p_j \in I, p_i \neq p_j} Time^t(p_i, p_j) Path^t_{p_i,p_j} \leq T \tag{9}$$

Constraint 9 confirms that the recommended itinerary will be completed within budget time T. Here the time function is calculated based on travel time, visiting time, and queuing time using the following Equation 10.

$$Time(p_i, p_j) = TDoP(p_i, p_j) + VDoP(p_j) + Queue^t_{p_j} \tag{10}$$

The main goal of this research is to propose an effective and efficient personalized itinerary recommendation system that can conduct practical and significant tourism parameters e.g. POIs popularity, visitors' interest, starting and ending POIs, and time constraints including budget time, queuing time, visiting time and travel time.

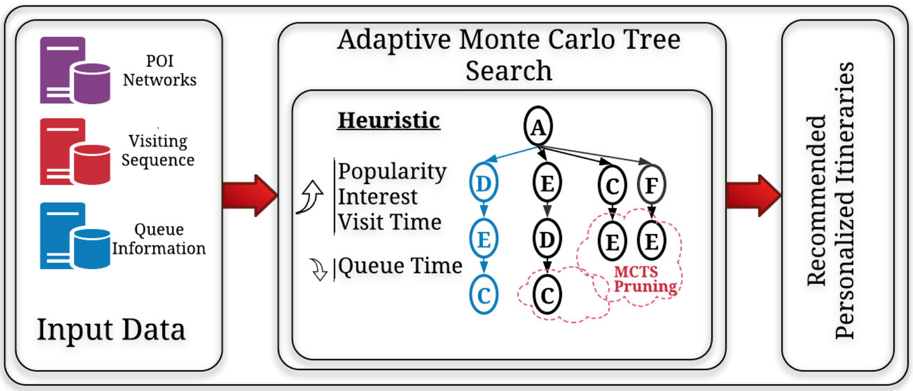


Fig. 3. Working steps of proposed *EffiTourRec* recommendation system

5. Proposed EffiTourRec Method

In this section, we introduce our proposed approach. We design an adaptive Monte Carlo Tree Search (MCTS) based **Effi**cient **Tour Rec**ommendation (*EffiTourRec*) system for personalized itinerary recommendation using realistic effective heuristic and efficient pruning technique. The main purpose of using adaptive MCTS is that it can be adaptive to run only a fixed number of iteration or a fixed amount of time and has been shown [36] to be a good search strategy for showing itinerary paths. Fig. 3 depicts our proposed *EffiTourRec* recommendation system working steps. At the first step, the system takes the data from the input databases, which consists of a POI network, visiting sequence and queue information. Second, the *EffiTourRec* system finds a set of POIs for the visitor by using an effective heuristic. These POIs create a personalized itinerary recommendation that visitors get maximum entertainment by maximizing visitors' interest, POI popularity, visiting time and minimizing queuing time. Our proposed effective heuristic method is applied to select potential POI at the adaptive Monte Carlo Tree Search selection step. We also applied an efficient pruning technique that allows us to explore a smaller, more promising portion of the solution space, thus our method requires a shorter running time. The following sub-section formalizes the concepts and presents a detailed description of adaptive MCTS that is used in our proposed *EffiTourRec* method.

5.1. Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is a well-known reinforcement learning algorithm that not only has been successfully applied to artificial intelligent (AI) games (e.g., chess, go, Othello) [3] but also has been applied in personalized itinerary recommendation [36]. In the personalized recommendation system, each move from a node to a child node is considered as a variable cost, such as tour time, in contrast to the uniform cost of board games. This variable cost is time-dependent because queuing time at the same attraction depends on the crowd, which is high at peak time and low at the off-peak time. Thus, the itinerary recommendation result is measured by the complex reward function that aims to maximize POIs popularity, visitors' interest, visiting time and minimize queuing time instead of board game binary reward win or lose. MCTS and our approach run over a number of iterations and each iteration consist of four steps.

Selection step is used for traversing the POIs from starting POI to last visit POI. After that, the expansion step is used to decide how many and which POIs are considered as the next potential POIs in the itinerary recommendation. Then, the simulation step is used to complete the itinerary recommendation starting POI to ending POI. Finally, the reward function of Monte Carlo evaluation is propagated back to the starting POI using a back-propagation strategy. We describe each part of our proposed adaptive MCTS steps in detail.

Step 1: Selection: At each iteration, the MCTS begins at starting POI as a tree root node and moves recursively through the tree child node to expand based on the tree policy until it reaches destination POI or unvisited POI. In the most common tree policy, the next node is selected by the Upper Confidence Bound Tree (UCT) [29], which is improved and applied in the personalized itinerary recommendation by the *PersQ* [36] algorithm, using an additional heuristic for selecting the next potential POI p_j.

$$UCT_{p_j}^{PersQ} = \underbrace{\overbrace{\frac{PoP(p_j) + IoP(c_{p_j})}{VDoP(p_j) + TDoP(p_i, p_j) + Queue_{p_j}^t}}^{\text{Potential POI}} + \overbrace{\frac{totalReward_{p_j}}{visitCount_{p_j}}}^{\text{Path Reward}}}_{\text{Exploitation}} + \underbrace{2C_p \sqrt{\frac{2 ln visitCount_{p_j}}{visitCount_{p_i}}}}_{\text{Exploration}} \tag{11}$$

In Equation 11, the first part Potential POI used a heuristic to select the next potential POI to favor the POIs with higher popularity and interest but with the lower associated traveling, visiting and queuing time. The second part Path Reward used for existing path rewards based on POI popularity, visitor interest and queuing time from starting POI to POI p_i. These two parts express the Exploitation of itineraries from source to potential candidate POI p_j. The Exploitation ensures the best itinerary path is found thus far. The third part, Exploration controls the number of POIs that have not been visited previously. The parameter C_p fixed the value of exploration of POIs which the best value is $\frac{1}{\sqrt{2}}$ determined by Kocsis and Szepesvári [30] as it satisfies Hoeffiding's inequality.

We observe that the proposed UCT in the earlier *PersQ* suffers from three shortcomings. First, the *PersQ* [36] system did not consider visitor visiting time as visitors' interest or preferences, it only considers the number of visitors who visited the POI and the number of taken photos in that POI as the visitor interest. Second, *PersQ* cannot differentiate the persons' interest who spend more time in an attraction than the others because the popularity of attraction is measured by the number of visitors, not their spending time. Third, in the proposed *PersQ* algorithm next potential POI selection is inversely proportional to prior visit time, which means if prior visit time is large, then POI selection probability will be less. This is contradictory with real-world behaviour because prior visit time is typically proportional to the visitors' interest. That means if prior visitors spend more time in a POI, they will likely like it very much.

To overcome these three shortcomings, we propose an effective heuristic-based potential POI selection method which is based on the POI popularity, visitors' interest, POI visiting time and POI queuing time.

$$UCT_{p_j}^{EffiTourRec} = \underbrace{\overbrace{\frac{PoP(p_j) * IoP(c_{p_j}) * VDoP(p_j)}{TDoP(p_i, p_j) + Queue_{p_j}^{t}}}^{\text{Potential POI}} + \overbrace{\frac{totalReward_{p_j}}{visitCount_{p_j}}}^{\text{Path Reward}}}_{\text{Exploitation}} + \underbrace{2C_p\sqrt{\frac{2lnvisitCount_{p_j}}{visitCount_{p_i}}}}_{\text{Exploration}} \quad (12)$$

The main reason for the effectiveness of our proposed method compared to the existing *PersQ* method is due to its Exploitation part. Exploration part is the same for both methods. The exploitation part consists of two parts that ensure the best itinerary path is found from starting POIs to current POI p_i and select next potential POI p_j, respectively. In the first part, Potential POI ensures the selected p_j maximize popularity, visitor interest and prior visitors' preferences and minimize queuing time and traveling time. Here, POI popularity depends not only on specific user previous visits but also on previous users' repeated visits. Similarly, POI visiting time does not depend on signal users, and it also depends on other users spending time on that particular POI. That is why user interest measured by the number of taken photos does not depend on POI popularity and POI visiting time. On the other hand, POI popularity and POI visiting time are not dependent each other. Thus, we consider these three factors and there are potential overlaps between them. Maximizing these three factors, we cover potential overlap and balance the multi-factors preferences, which finds the strong relationship among the factors. It will cover both user's interest independent and dependent factors. We have used multiplication among these features that objectives are maximizing values. This multiplication value trends to be affected by one feature which value is very large. To avoid this unexpected trend, we use normalization and all factors values within 0 to 1 range. In the second part, Path Reward mentions the existing itinerary from starting POI to current POI p_i is the best itinerary based on the reward function. The reward function measures itinerary rating considering POIs popularity, visitors' interest, prior visitors' preferences and queuing time.

Fig. 4 illustrates a toy example that highlights the difference between our proposed and existing heuristics. Assume that visitor u completes POI A visit. The next POI to be visited will be selected based on UCT values among these three POIs. Consider Path Reward and Exploration are the same at this moment. Therefore, the value of UCT to select the next potential POI depends on the only Potential POI part. The existing *PersQ* selects next potential POI C because it potential POI value is larger than others. The potential POI values of these POIs are like this: POI B = (250+200)/(10+5+30) = 10, POI C = (250+200)/(5+5+30) = 11.25, and POI D = (250+200)/ (15+5+25) = 10. On the other hand, our proposed heuristic recommends the next potential POI D because it potential POI value is maximum among these three POIs. These values are: POI B = $\big((200/200) * (250/250) * (10/15)\big)/((5 + 30)/35) = 0.667$,

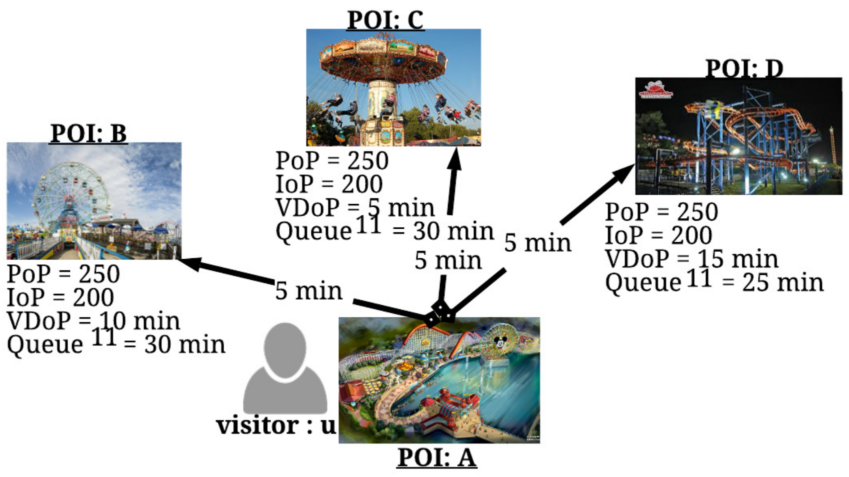


Fig. 4. The example illustrating the effectiveness of our proposed heuristic compared to the existing *PersQ* heuristic. It shows that prior visiting time is proportion to user interest in itinerary recommendation.

POI C $= \big((200/200) * (250/250) * (5/15)\big)/((5+30)/35) = 0.33$ and POI D $= \big((200/200) * (250/250) * (15/15)\big)/((5+25)/35) = 1.17$.

In the above discussion, it is clear that the existing *PersQ* method always recommends minimum visiting, travel and queuing time in which visitor spends only 5 minutes for enjoying POI C and 35 minutes for traveling and queuing purposes. This recommendation spends much waiting and traveling time. In contrast, our proposed method recommends POIs whose visiting time is maximum and traveling and queuing time are minimized. Hence, a visitor spends 15 minutes to enjoy POI D and 25 minutes for traveling and queuing purposes that makes visitors happier than the state of the art *PersQ*. For the readers' easy understanding, we consider all POIs popularity is same and visitor interest to all these three categories are uniform.

Step 2: Expansion: If the selected POI is not the ending POI, then one POI is added to the itinerary to represent this move by selecting one unvisited POI child, which is selected randomly. All these unvisited POIs are not significant. There are some POIs with long queuing times that can make visitors bored. To avoid these POIs, we have used a pruning technique that will be described in detail in the next section.

Step 3: Simulation: The selection and expansion steps are repeated until the visitor reaches the destination POI or exceeds the time budget. If the process continues and does not reach the destination POI within a fixed budget time, then the system cannot create a recommended itinerary. These itineraries, which are not ending with destination POI, are not considered as successful itineraries.

Step 4: Back-propagation: The itineraries of simulation results are back-propagated from the current POI to starting POI through the ancestor-selected POIs, updating the reward function until it reaches the starting POI. We choose a reward function that reflects the POI popularity, interest, visiting time and queuing time. The reward function is associated with each iteration, and it differs from the existing *PersQ* [36] algorithm, and it is defined as:

$$Reward = \sum_{p_j \in IH_{temp}} \frac{PoP(p_j) * IoP(c_{p_j}) * VDoP(p_j)}{Queue^t_{p_j}} \tag{13}$$

In this reward function, our main objectives are to maximize POI popularity, user interest and visiting time and minimize queuing time. This function is more effective than the existing function because the existing reward function does not consider visiting time as a factor to determine the reward function.

5.2. Efficient Move Pruning

MCTS runs a fixed number of iterations and each iteration generates an itinerary, which may be successful or unsuccessful. An itinerary that ends at the specified destination POI is a successful itinerary, while one that fails to do so is an unsuccessful one. All these successful itineraries do not achieve the same reward points. Therefore, the itineraries that achieve non-optimal we want to avoid to reduce unnecessary search and expansion time. On the other hand, the proposed method can generate the same successful itineraries multiple times, which is again redundant and time-consuming. To solve these time-consuming itinerary generations, we propose a new MCTS pruning technique that can prune duplicate itineraries and non-optimal itineraries at the early stage. This pruning technique reduces search space remarkably without the loss of successful itineraries. This pruning reduces time complexity or space complexity and makes the system more efficient.

We consider cumulative itinerary reward point (IR) and cumulative itinerary time (IT) data structures to remove these low-quality and duplicate itineraries. The cumulative itinerary reward point and cumulative itinerary time are represented by $IR^i_{p_j}$ and $IT^i_{p_j}$, where i indicates the itinerary POIs length and p_j mentions POI in the itinerary. Then, whenever the algorithm creates a new itinerary from starting POI to destination POI, its prune factor (PF), defined by the ratio of cumulative reward points and itinerary time will be checked. If the itinerary consists of the same POIs at the same length position with less or equal prune factor value compared to the existing optimal one, the algorithm prunes search space considering low rewarded itinerary or duplicate itinerary using the following equation.

$$MCTS_{Prune} = If\Big(current\_PF^i_{n_j} <= existing\_PF^i_{n_j}\Big) \tag{14}$$

where, $current\_PF^i_{n_j}$ and $existing\_PF^i_{n_j}$ mentions current prune factor and existing prune factor at POI n_j in position i, respectively. The following equation represents the prune factor.

$$PF^i_{p_j} = \big(\sum_{1,p_j \in I}^{|i|} IR^i_{p_j}\big)/\big(\sum_{1,p_j \in I}^{|i|} IT^i_{p_j}\big) \tag{15}$$

where $IR^i_{p_j}$ mentions cumulative itinerary reward and $IT^i_{p_j}$ indicates the

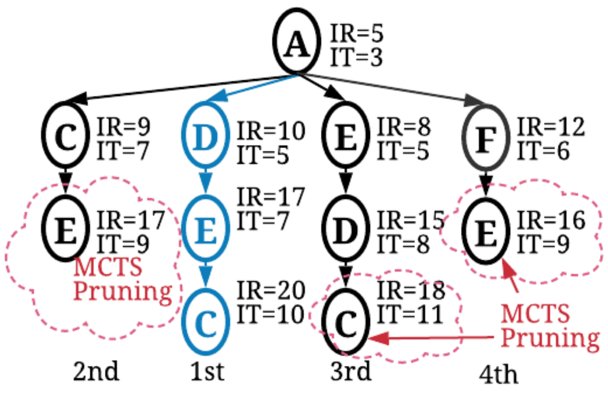


Fig. 5. The example illustrating the efficiency of our proposed pruning technique. It prunes MCTS space which makes non-optimal and duplicate itineraries.

cumulative itinerary time from starting POI to POI p_j and i presents the length of the creating itinerary.

We further elaborate on this pruning technique using the following example. Suppose that, in Fig. 5 a successful itinerary is $I = \big(A \to D \to E \to C\big)$, which length is 4. The cumulative itinerary reward points and cumulative itinerary time are $IR_A^1 = 5$, $IT_A^1 = 3$, $IR_D^2 = 10$, $IT_D^2 = 5$, $IR_E^3 = 17$, $IT_E^3 = 7$ $IR_C^4 = 20$ and $IT_C^4 = 10$. Then, the algorithm creates a second itinerary $A \to C$ as POI C is not visited yet as the second POI in the itinerary. After that, when POI E is selected as the next potential POI it creates low prune factor $(17/9 = 1.89)$ than the existing POI E prune factor $(17/7 = 2.43)$ at the third position in the first itinerary. Thus, the system can prune remaining MCTS traversing, which generates low-quality itineraries. In this same process, the algorithm can prune remaining tree search whether they find a low prune factor at POI C and POI E in third and fourth itineraries, respectively. This approach ensures low-quality itineraries generation are pruned early, making our proposed system faster without losing any information. Besides this, sometimes MCTS generates duplicate itineraries with the same prune factor. Suppose the fifth iteration generates a duplicate itinerary. In that case, the algorithm can prune this itinerary generation whenever it finds that the prune factor of POI D is the same as the existing value. Therefore, the system can avoid huge tree searching using this pruning technique. This example exhibits that itinerary reward and time are small in which the ratio value may be the same for different levels. In reality, it is infrequent to find the same prune factor in different levels of itineraries because the prune factor depends on five constraints e.g. POI popularity, visitor interest, traveling, visiting and queuing time. In this way, our proposed MCTS pruning technique reduces the MCTS space, making the algorithm more efficient than the existing system.

5.3. EffiTourRec Algorithm

Algorithm 1 shows an overview of our proposed *EffiTourRec* algorithm in which inputs are POIs information (e.g. popularity, queuing time, visiting time and traveling time etc.) represented by $P = \{p_1, p_2, \cdots, p_n\}$, starting POI $p_1 \in P$, ending POI $p_n \in P$, starting time t_s, and time budget T for completing the visitor itinerary. The output of this algorithm is recommended itinerary $I = \{p_1, \cdots, p_n\}$, which starts from POI p_1 at time t_s and reaches at POI p_n within the time limit $t_s + T$.

Algorithm 1: EffiTourRec-Overview of Algorithm

```
Data: $P = \{p_1, p_2, \cdots, p_n\}$: POI information; $Queue^t_P$: Queuing time
      of POI at different times; $p_1 \in P$: Staring POI; $p_n \in P$: Ending
      POI; $t_s$: Staring time of itinerary, T: Total time budget;
      maxLoop: Number of Iterations
Result: $I = \{p_1, \cdots, p_n\}$: Recommended Personalized Itinerary
1  $T_{visits} \leftarrow emptyTree$; $T_{reward} \leftarrow emptyTree$;    /* Initialize visit
     count and reward tree */
2  $T_{prune} \leftarrow emptyTree$;   /* Initialize MCTS pruning tree */
3  $I_{list} \leftarrow NULL$;   /* Initialize list of itineraries */
4  for $Iterations \leftarrow 1\ to\ maxLoop$ do
5      $I_{temp} \leftarrow p_1$;
6      $p_i \leftarrow p_1$; $p_j \leftarrow \emptyset$; $totalTime \leftarrow 0$; $tempReward \leftarrow 0$;
7      while $totalTime \leq T$ do
8          $p_j \leftarrow SelectNextPOI(p_i, T_{visits}, T_{reward}, I_{temp})$;
9          $I_{temp} \leftarrow I_{temp} \bigcup p_j$;   /* Append $p_j$ to temporary itinerary
             */
10         $totalTime \leftarrow totalTime + TDoP(p_i, p_j) + Queue^t_{p_j} + VDoP(p_j)$;
11         $tempReward \leftarrow tempReward + Reward(p_j)$
12         if $T_{prune}[j, p_j] \neq null\ and\ T_{prune}[j, p_j] \geq \frac{tempReward}{totalTime}$ then
13             Break Loop ;   /* Prune low rewarded and duplicate
                 itineraries */
14         if $p_j == p_n$ then
15             Break Loop;
16         $p_i \leftarrow p_j$;
17     $BackPropagationC(I_{temp}, T_{visits})$;
18     if $p_j == p_n$ then
19         $Reward \leftarrow Simulate(I_{temp})$;
20         $BackPropagationR(I_{temp}, T_{visits}, T_{reward})$;
21         for $\forall p_j \in I_{temp}$ do
22             $T_{prune}[j, p_j] = \frac{CumulativeReward(I_{temp}, p_j)}{CumulativeTime(I_{temp}, p_j)}$;   /* Update
                 Pruning Tree */
23         $I_{list} \leftarrow I_{list} \bigcup I_{temp}$;
24 $I \leftarrow maxReward(I_{list})$;
25 $Return$ I;   /* Return best itinerary */
```

To apply MCTS in our itinerary recommendation system, we consider the root node as the starting POI of an itinerary and its child nodes as next potential POIs that can be visited. The start of Algorithm 1, it initializes three similar trees for measuring the amount of spending time, reward points and prune POI named T_{visits}, T_{reward} and T_{prune} by the *emptyTree* (in lines 1 and 2). The *emptyTree* root node is POI p_1 and child nodes are set of POIs $p_i \in P$, up to a depth of $|P|$ (the number of POI in the visiting theme park). Here, the traversal of POI from root to leaf represents an itinerary $I = \{p_1, p_2, \cdots, p_n\}$, with p_1 as the selected POI at depth 1, p_2 at depth 2, and so on until it reaches leaf p_n at depth n. Initially, I_{list} set is empty at line 3 that is used to store a set of exploring itineraries from iteration 1 to *maxLoop*. This algorithm runs a fixed number of iterations *maxLoop* in line 4. Each iteration (lines 4-29) executes a single run of adaptive MCTS and generates a possible recommended itinerary. The possible itinerary I_{temp} is initialized by first visiting p_1 at line 5. Then, line 6 set current POI is staring POI p_1, next POI is empty and expending time *totalTime* and *tempReward* are zero. The procedure *SelectNextPOI()* selects next potential POI using effective Upper Confidence Bound heuristics until the visitor expending time exceeds the given budget time or next POI selects ending p_n in lines 7-19. This *SelectNextPOI()* method in line 8 reflects the selection and expansion steps of our proposed adaptive MCTS and describes further in Algorithm 2. Every time the selected POI appends to the temporary itinerary and visitors expending time and temporary itinerary reward update in lines 9-11. Then, the temporary itinerary is checked whether it has been explored already or has generated low quality or duplicate itinerary in line 12. If it generates low quality or duplicates itinerary, the search stops and prunes the remaining POI traversing in line 13. On the other hand, if the temporary itinerary is new or creates a high-quality itinerary, select next POI as the current POI and continue until it reaches budget time or find the destination POI as selected POI in lines 15-18.

After that, *BackPropagation()* method updates visited POIs in T_{visits} based on the recommended POIs in temporary itinerary I_{temp} at line 20. If the temporary itinerary I_{temp} completes with the ending p_n, the model applies the obtained reward to the itinerary I_{temp} at line 22. Then it updates the accumulated rewards of visited POIs in T_{reward} at line 23. Then, the pruning tree T_{prune} is updated based on cumulative reward and cumulative time in lines 24-26. Here, $CumulativeReward(I_{temp}, p_j)$ and $CumulativeTime(I_{temp}, p_j)$ calculate cumulative reward and cumulative time spending from staring POI to POI p_j in itinerary I_{temp}, respectively. After that, the temporary itinerary I_{temp} appends to the itinerary list I_{list} at line 27 and finds the best (highest rewarded) itinerary I at line 30. Finally, the algorithm *EffiTourRec* returns the best itinerary I to the visitor at line 31.

Algorithm 2 illustrates the selection process of next potential POI by *SelecNextPOI()* function that takes current p_i, visit count tree, reward tree as input, current creating itinerary and returns next potential POI (p_p) which maximizes the Upper Confidence Bound denoted as $UCT_{p_p}^{EffiTourRec}$. At first, the algorithm counts the number of visitors who visit p_i based on T_{visit} tree at line 1. Initially, the next potential POIs, upper confidence bound and existing itinerary are initialized in lines 2-4. To select the best next potential POI, each p_j connected with p_i but does not exist in the current creating itinerary I_{temp} calculates upper confidence bound and finds a best potential POI in the lines 5-16. Each POI p_j connects to POI p_i counts the number of visitors at line 6. Then, the total reward of these POIs is measured based on popularity, interest, visiting time and queuing time in lines 7 and 8. The exploitation value of p_j represents the itineraries with high rewards, relative to the number of chosen POI at line

Algorithm 2: *SelectNextPOI*$(p_i, T_{visit}, T_{reward}, I)$

```
Data: $p_i \in P$: Current Point of Interest; $T_{visits}$: Tree of visit count;
      $T_{reward}$: Tree of reword count; $I_{temp}$: Existing Itinerary;
Result: $p_p$: Next potential POI of personalized itinerary
1  $visitCount_{p_i} \leftarrow GetVisitCount(p_i, T_{visit})$;
2  $p_p \leftarrow \emptyset$;                /* Next potential POI is empty */
3  $UCT_{max}^{EffiTourRec} \leftarrow 0$; $UCT_{max} \leftarrow 0$;
4  $I_{temp} \leftarrow I_{temp} \bigcup p_i$;
5  for $p_j \in P$ and $p_j \notin I_{temp}$ do
6      $visitCount_{p_j} \leftarrow GetVisitCount(p_j, T_{visit})$;
7      $totalReward_{p_j} \leftarrow \sum_{p_j \in T_{reward}} (\frac{PoP(p_j)*IoP(c_{p_j})*VDoP(p_j)}{Queue_{p_j}^t})$;
8      $GetTotalReward(p_j, T_{reward})$;
9      $exploit_{p_j} \leftarrow \frac{PoP(p_j)*IoP(c_{p_j})*VDoP(p_j)}{TDoP(p_i,p_j)+Queue_{p_j}^t} + \frac{totalReward_{p_J}}{visitCount_{p_j}}$;
10     $explore_{p_j} \leftarrow 2C_p \sqrt{\frac{2ln(visitCount_{p_j})}{visitCount_{p_i}}}$;
11     $UCT_{p_j}^{EffiTourRec} \leftarrow exploit_{p_j} + explore_{p_j}$;
12     if $UCT_{p_j}^{EffiTourRec} > UCT_{max}$ then
13         $p_p \leftarrow p_j$;
14         $UCT_{max} \leftarrow UCT_{p_j}^{EffiTourRec}$;
15 Return $p_p$;                          /* Return next potential POI */
```

9 considering POI popularity, interest, visiting time, queuing time and reward. The explore part controls the POIs that have not been selected previously, thus ensuring the different POIs are considered in line 10. After combining exploitation and exploration, we find our proposed *EffiTourRec* upper confidence bound at line 11. If the upper confidence bound $UCT_{p_j}^{EffiToruRec}$ is greater than the previous bound, it has been stored as maximum upper bound and considered the p_p as the next selected POI in lines 12-15. Finally, the algorithm returns the best potential next selected POI p_p within all connected POIs at line 17.

6. Experiments

In this section, we present and discuss the experimental datasets, baseline algorithms and evaluation metrics. For these comparisons, our proposed *EffiTourRec* algorithm and the existing baseline methods are implemented in the R language. The experiments are run on 2.50 GHz Intel Core i5 with 8GB RAM, in Windows 10.

6.1. Datasets

For our experiments, we used geo-tagged photos in five real theme parks from August 2007 to August 2016 that was used in [36]: Disney Land, Epcot, California Adventure, Disney Hollywood, Magic Kingdom. These datasets were collected

Table 2. Parameters description of various theme park datasets

Theme Park	# Photos	POI Visits	# Visitors	# POIs	# Visit Sequences
Disney Land (disland)	181,735	119,987	3,704	31	11,758
Epcot (epcot)	90,435	38,950	2,725	17	5,816
California Adventure (caliAdv)	193,069	57,177	2,593	25	6,907
Disney Hollywood (disHolly)	57,426	41,983	1,972	13	3,858
Magic Kingdom (MagicK)	133,221	73,994	3,342	27	8,126

in four steps: First, Flickr API [1] is used to retrieve all geo-tagged photos with visitor ID, geo-coordinates and timestamp within the theme parks. Second, each photo geo coordinates maps to a POI coordinates if its Haversine [47] distance is less than 100m. If there are multiple POI coordinates within 100m range, then the photo maps to the nearest POI coordinate. Third, visit sequences are constructed based on photos taken time of these POIs. If the time gap between two consecutive taken photos is greater than 8 hours, it is considered as a new visit sequence. Finally, in these visit sequences, POI popularity, visitors' interest, queuing time, visiting time,and traveling time are determined by the number and timestamp of taken photos. The variation of five theme parks including the number of POI, number of photos, number of visitors, number of POI visits and visit sequences are shown in Table 2.

6.2. Baseline Algorithms

In the personalized itinerary recommendation system, practical constraints play a significant role in effective and efficient tour planning. Our main baseline is the existing work *PersQ* [36] as it also considers POI popularity, visitor interest, starting and ending POIs and time constraints including traveling, visiting and queuing time simultaneously. Moreover, there are various state-of-art baselines in which all these constraints are considered separately. The baseline algorithms related to our proposed algorithm are as follows.

– Personalized Tours with Queuing Time Awareness **(PersQ)** [36]: Itinerary starts at starting POI, selects potential POIs based on maximum popularity and interest, minimizing queuing time and completes the itinerary at ending POI within a time budget.
– Personalized Tour Recommendation **(PersTour)** [39]: This algorithm recommends itineraries based on POI popularity and time-based visitor interest within budget time.
– Tour Recommendation with Interest Category **(TourRecInt)** [35]: The algorithm introduces mandatory visit POI category-based tour recommendation system in which visitors are most interested in visit frequently visited POIs category.
– Trip Builderer Algorithm **(TripBuilder)** [2]: The main objective is to define visitor interest by the spending time of visited POI of a certain category, corresponding to his/her total visiting time.

[1] https://www.flickr.com/services/api/

– Iterative Heuristic Approximation **(IHA)** [55]: A heuristic based recommendation system that starts with a tour staring p_i, completes at p_n and repetitively adds new POI into the itinerary until the budget time is reached.
– User Based Collaborative Filtering for Itineraries **(UBCF)**[53]: One of the popular user-based collaborative filtering variations is proposed to recommend a set of top-k POIs for another user to utilize user interest similarities.

6.3. Performance Evaluation

To evaluate the performance of our proposed *EffiTourRec* algorithm and existing baseline algorithms, we consider visitors who have at least two visit sequences and each sequence has at least three visiting POIs, including starting and ending POIs. Then we apply the leave-one-out evaluation [31] strategy into these sequences where one visit sequence is used for evaluation and the other visit sequences are used to determine visitor preferences. For each visit sequence, we use starting and ending POIs as input to the algorithm and the budget time is determined by the actual time spent in the visit sequence. After that, we evaluate the performance of our proposed algorithm against the various baselines using the following standard metrics [36, 39].

– **Precision:** The ratio of POI recommended itinerary I that are present in a visitor's real life visit sequence. Let P_{real} be the set of POIs in the real visit sequence and P_{rec} be the set of POIs recommended in itinerary I, the tour precision is defined as: $Precision = \frac{|P_{real} \bigcap P_{rec}|}{|P_{rec}|}$.
– **Recall:** The proportion of POI visits in a visitor's real-life visit sequence that also be present in the recommended itinerary I. Using the same notation for P_{real} and P_{rec}, the tour recall is defined as: $Recall = \frac{|P_{real} \bigcap P_{rec}|}{|P_{real}|}$.
– **F1-Score:** The harmonic mean of both tour recall IR_I and tour precision IP_I of an itinerary I, defined as: $F1 - Score = \frac{2 * IP_I * IR_I}{IP_I + IR_I}$.
– **Popularity:** The average popularity of all POIs in a recommended itinerary I, exposed as: $Popularity = \frac{1}{|I|} \sum_{p_i \in I} PoP(p_i)$.
– **Interest:** The average interest of all POIs in a recommended itinerary I, exposed as: $Interest = \frac{1}{|I|} \sum_{p_i \in I} IoP(p_i)$.
– **Rank**: The average rank of our proposed *EffiTourRec* algorithm is defined based on popularity and interest scores ranked compare with other algorithms, exposed as: $Rank = \frac{1}{|I|} \sum_{p_i \in I} MaxRange - \big(\frac{Norm(PoP(p_i)) + Norm(IoP(p_i))}{2} \times MaxRange\big) + 1$, where *Norm(.)* is a normalization function converts score within [0,1] and *MaxRange* is maximum Rank value (we assume 1 = best and 12 = worst).
– **Visiting Time Cost Ratio** ($VTCR$)**:** The ratio of visiting time of an itinerary I, relative to itinerary total time, defined as:
$VTCR = \sum_{p_i \in I, i \neq 1} \frac{VDoP(p_i)}{TDoP(p_{i-1}, p_i) + VDoP(p_i) + Queue^t_{p_i}}$.
– **Queuing Time Cost Ratio** ($QTCR$)**:** The ratio of queuing time an itinerary I, relative to itinerary total time, defined as:
$QTCR = \sum_{p_i \in I, i \neq 1} \frac{Queue^t_{p_i}}{TDoP(p_{i-1}, p_i) + VDoP(p_i) + Queue^t_{p_i}}$.

- **Queue Time Popularity Ratio ($QTPR$):** The ratio of queuing time an recommended itinerary, relative to the popularity of an itinerary I, defined as: $QTPR = \sum_{p_i \in I} \frac{Queue^t_{p_i}}{PoP(p_i)}$.
- **Maximum Queuing Time (MQT):** The average queuing time of POIs in itinerary I, relative to their maximum queuing time, defined as: $MQT = \frac{1}{|I|} \sum_{p_i \in I} \frac{Queue^t_{p_i}}{Max_{t \in T}(Queue^t_{p_i})}$.
- **Execution Time:** Execution time means algorithm run time to recommend itineraries for particular datasets.
- **Number of Moves:** The total number of POI moves require to create itineraries for particular datasets.

6.4. Results and Discussion

In this section, we describe our proposed *EffiTourRec* algorithm results compared to the existing baseline algorithms.

6.4.1. Precision, Recall and F1-score

The main evaluation process of an itinerary recommendation method is how well the recommended itineraries satisfy visitors' requirements. The evaluation metrics precision, recall and F1-scores measure how well the recommended POIs match with real-life user preferences.

The results in Fig. 6 show the overview of the average precision, recall and F1-score, respectively in five theme parks, for our proposed *EffiTourRec* and the baseline algorithms. The results show that the proposed *EffiTourRec* outperforms the six baselines in terms of precision and F1-scores. In terms of recall scores, the proposed method outperforms all baselines except *PersQ* in California Adventure and Magic Kingdom datasets among the five datasets. Now, we explain the performance of our *EffiTourRec* algorithm and the baseline algorithms in detail.

The first column of Fig. 6 depicts the proposed *EffiTourRec* algorithm performance on the itinerary precision score is maximum 79.10% in Hollywood and minimum 62.90% in Disney Land dataset whereas, the best baseline algorithm performs maximum 64.23% in Hollywood and minimum 43.31% in California Adventure. Our proposed algorithm shows improvement results compared to the baselines maximum 52.32% (the proposed method is 65.97% and existing *PersQ* is 43.31%) in California Adventure dataset and minimum 20.89% (*EffiTourRec* method is 70.61% and existing *PersQ* is 58.41%) in Epcot dataset.

The second column of Fig. 6 presents the itinerary recall score performance analysis on our proposed *EffiTourRec* algorithm and baselines. The results show that the proposed method under-performed 5.87% to 11.11% than the existing *PersQ* algorithm, whereas it outperformed than the other baselines. The main reason for these results is that *PersQ* algorithm prefers minimum visiting times POIs. Even though these POIs may not show visitors' interest accurately, they create long itineraries that may be likely makes recall value high. To distinguish our performance analysis, we focus on F1-score that provides a balanced representation of itinerary precision and recall scores.

The third column of Fig. 6 illustrates the result of F1-scores for all theme parks and *EffiTourRec* out-performs all baselines from 8.36% to 21.35%. The

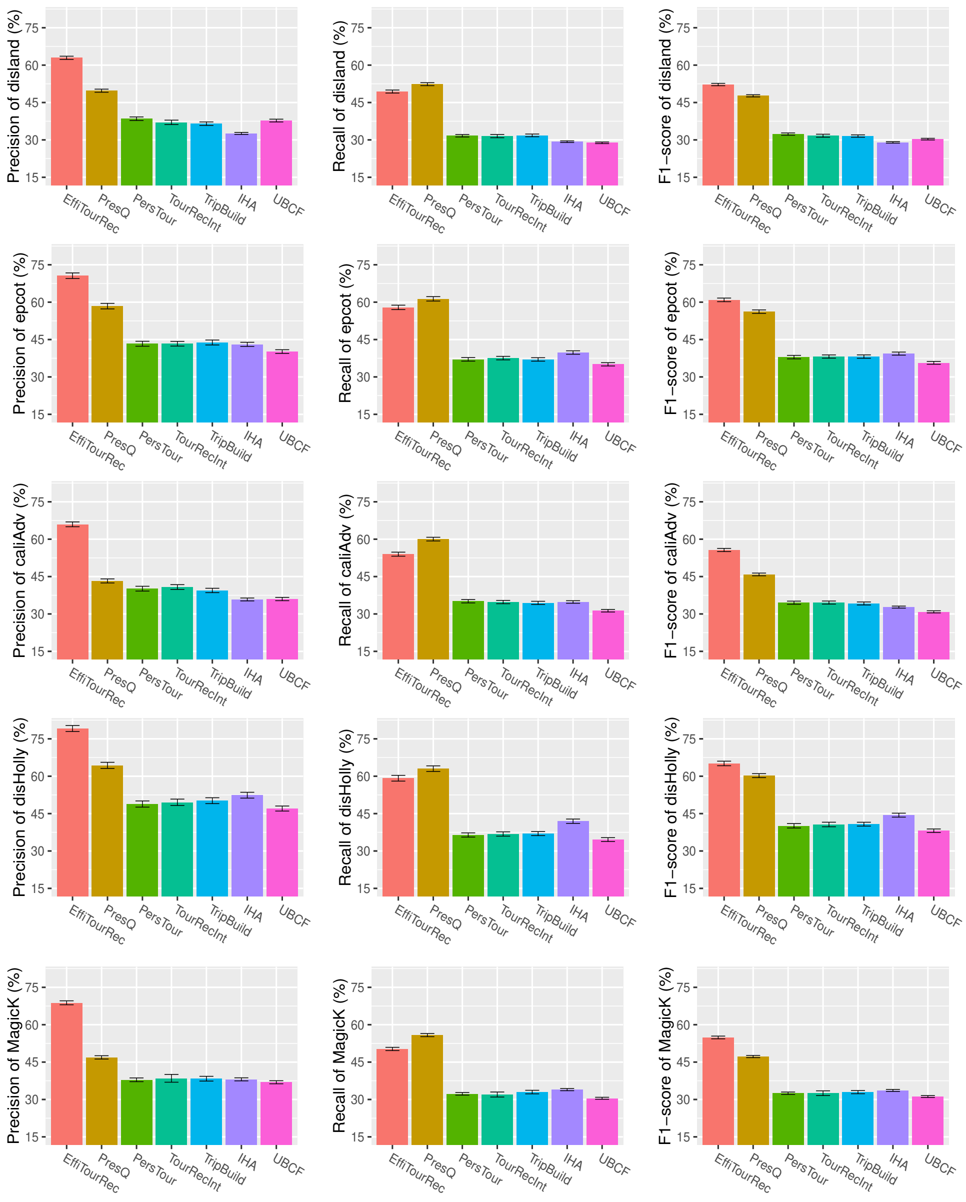


Fig. 6. Comparison among proposed *EffiTourRec* and various baselines, in terms of precision, recall and F1-score (1st to 3rd column) of tours recommendation for Disney Land, Epcot, California Adventure, Hollywood and Magic Kingdom (1st to 5th row) datasets. The x-axis of the graph shows the algorithms analyzed, namely: EffiTourRec, PersQ, PersTour, TourRecInt, TripBuild, IHA and UBCF (left to right).

Table 3. Comparison between EffiTourRec and various baselines, in terms of the mean and standard errors of Popularity and Interest whose higher values are better, and lower values of Rank is better. In each metric, bold blue numbers express the best result.

Datasets		EffiTourRec	PersQ	PersTour	TourRecInt	TripBuilder	IHA	UBCF
disland	Popularity	**3822±64**	2860±49	2443±41	3263±48	2940±41	1651±33	2314±42
	Interest	157.8±20.2	147.5 ±15.9	100.9±11.1	122.8±21.1	109.8±12.5	**173.2±18.6**	127.3±11.3
	Rank	5.65 ± 0.06	6.05± 0.07	5.75 ± 0.09	5.76 ±0.07	5.98 ± 0.10	**4.47 ±0.04**	5.68 ± 0.05
Epcot	Popularity	**2376±64**	1927±44	1415±31	1376±29	1377±33	1666±35	1175±28
	Interest	**23.21±2.4**	22.61±1.9	15.1±1.3	15.1±1.2	18.8±2.0	21.7±1.5	12.2±1.0
	Rank	**4.60 ±0.11**	4.89 ± 0.12	5.38 ± 0.09	5.08 ± 0.10	5.50 ± 0.10	4.69 ± 0.08	6.06 ± 0.08
caliAdv	Popularity	**2834±37**	1603±26	1518±41	1554±43	1560±50	1446±33	1416±35
	Interest	**242.3±37.8**	238.2±34.1	145.0±33.2	188.3±34.1	161.9±29.9	239.2±32.7	130.5±18
	Rank	**4.14 ±0.09**	5.71 ±0.08	5.58 ± 0.08	5.3 ± 0.08	5.56 ± 0.13	4.19 ± 0.06	5.84 ± 0.07
disHolly	Popularity	**3142±74**	2480±54	1990±51	2016±52	1803±54	2976±44	1737±47
	Interest	15.6±1.3	**16.2±1.4**	12.0±1.2	12.4±1.2	11.9±1.1	16.1±1.4	11.4±1.1
	Rank	**3.61±0.13**	4.33 ± 0.14	4.73 ± 0.10	4.33 ± 0.12	4.53 ± 0.10	3.65 ± 0.08	4.98 ± 0.09
MagicK	Popularity	**3724±78**	1960±35	1767±79	1629±70	1591±68	2125±26	1616±36
	Interest	**31.16±2.7**	30.41±2.3	18.4±0.4	18.2 ±2.6	26.2±2.4	27.6±2.4	14.2±0.9
	Rank	**4.27 ±0.09**	5.70 ± 0.09	5.84 ± 0.12	5.91 ± 0.15	6.09 ± 0.14	4.36 ± 0.06	6.07 ± 0.06

results show that the proposed method under-performs on tour recall scores but out-performs on F1-score scores. The results improve performance minimum 8.36% in Epcot dataset (proposed *EffiTourRec* method is 60.90% and best existing *PersQ* is 56.2%) to maximum 21.35% in Disney Hollywood dataset (proposed *EffiTourRec* and best existing *PersQ* are 55.70% and 45.86%, respectively). These results prove that the recommended itineraries of our proposed *EffiTourRec* algorithm are highly significant to the real-life visitors than the other baselines.

6.4.2. Popularity, Interest and Rank of POI

The evaluation metrics' popularity and interest reflect how well the visitors like these recommended POIs and how well the category of recommended POIs match with real-life visited POIs category, respectively. The results in Table 3 show that the maximum average popularity of POI of our proposed *EffiTourRec* method is 3822 in Disney Land dataset, whereas the existing best *PersQ* method is 2860. On the other hand, the minimum average popularity of the proposed method is 2376 in Epcot dataset, which is also best among all baseline algorithms. The results show that our proposed algorithm recommends the most popular POI for all theme parks datasets compared to the baselines, while *PersQ* offers the second-best performance. It shows that our proposed *EffiTourRec* creates itineraries based on POIs that are 23.30% to 81.17% more popular than the existing baselines.

Table 3 illustrates proposed *EffiTourRec* method recommends the highest interest scores for three datasets among the five theme parks while *PersQ* and *IHA* algorithms lead POI interest for one theme park.

Besides these, ranking expresses the user's interest and POI popularity values as a function to show the POIs ranks. We can see that our proposed method's average POIs rank value is best compared to the other baselines. We consider an effective heuristic to maximize popularity, user interest, and visiting time, along with minimizing queuing time. In this evaluation, we consider rank value from 1 to 12, and the smallest rank value expresses the better result.

6.4.3. Visiting, Travel and Queuing Time Metrics

Time constraints are important parameters for an itinerary recommendation because every visitor has specific time limits. Thus, the recommended itinerary should be more effective in which the visitor gets maximum visiting time to visit POIs rather than traveling and queuing time to access that POI. These time metrics indicate the effectiveness of visitor spending time.

Visiting time cost ratio (VTCR) represents the proportion of POI visiting time compared to the total spending time of visitors. Table 4 shows the comparison of VTCR for the five datasets among our proposed method and existing six baseline methods. All results show that our proposed *EffiTourRec* algorithm outperforms the baselines with an increment of 6.69% to 44.93% in VTCR. The main reason for these performances is that our proposed potential POI selection focuses on visiting time as a proportion factor to show the visitor preferences.

Travel time cost ratio (TTCR) means the part of the traveling time from one POI to another POI with the total spending time of visitors to complete their itineraries. Most of the datasets travel time cost ratio is 5-7% as traveling time makes up a small part of the itinerary in this theme park context. Thus, as the results for the baselines and *EffiTourRec* are almost similar and we do not present this in the table.

The queuing time cost ratio (QTCR) in Table 4 presents visitors waiting time ratio with total expending time. More QTCR means the recommended itinerary spends more time as waiting time than the visitor preferences POI visiting time. The table clearly shows that queuing time is an important factor in visitors' theme park travel because it requires more than 58% budget time for all datasets. Thus, less value of QTCR elicits the visitors spend less time as waiting time. Therefore, it points out that the proposed method *EffiTourRec* outperforms all baselines with a reduction of queuing time. Table 4 depicts the queue time popularity ratio (QTPR) of itineraries. We know that long queuing time makes visitors disappointed, which means the small value of the queuing time popularity ratio indicates visitor preferences. The results show that the proposed method outperforms QTPR than the existing baseline algorithms in the five datasets, reducing 6.97% to 23.56%. The maximum queuing time (MQT) conveys the queuing time association of each POI in an itinerary. The larger value represents that the recommended POI is more similar to queuing time than the less valued queuing time. It shows that our proposed method outperforms all baseline algorithms except the disland dataset, where *UBCF* performs well than the proposed method.

In the above analysis, we see that our proposed *EffiTourRec* method outperforms 37 evaluation values for different datasets, whereas existing *PersQ* outperforms 6 values and *IHA* and *UBCF* outperform one evaluation value. Thus, it is clear that the superior performance of *EffiTourRec* is due to its effective itinerary creation approach using POI popularity, user interest, queuing and visiting times.

Table 4: Comparison between EffiTourRec and various baselines, in terms of Visiting time cost ratio (VTCR), Queuing time cost ratio (QTCR), Queuing time popularity ratio (QTPR) and Maximum Queuing time (MQT). Higher value of VTCR is better, while lower values of QTCR, QTPR and MQT are preferred. In each metric, bold blue numbers express the best result.

		EffiTourRec	PersQ	PersTour	TourRecInt	TripBuilder	IHA	UBCF
disland	VTCR	**0.180±0.002**	0.129±0.002	0.118±0.003	0.120±0.002	0.123±0.003	0.07±0.001	0.11±0.002
	QTCR	**0.765±0.003**	0.809±0.003	0.849±0.004	0.853±0.003	0.847±0.004	0.89±0.002	0.83±0.003
	QTPR	**0.760±0.012**	0.91± 0.02	1.42±0.05	1.08±0.004	1.19±0.03	1.71±0.03	1.32±0.04
	MQT	0.154±0.002	0.121±0.002	0.172±0.005	0.152±0.006	0.152±0.005	0.151 ±0.002	**0.118±0.002**
Epcot	VTCR	**0.328±0.008**	0.264±0.006	0.18±0.004	0.18±0.004	0.195±0.005	0.13±0.003	0.21±0.005
	QTCR	**0.586±0.009**	0.636±0.007	0.77±0.004	0.77±0.004	0.76±0.005	0.80±0.003	0.71±0.005
	QTPR	**0.893±0.02**	0.96±0.025	1.68±0.042	1.74±0.044	1.74±0.051	1.47±0.03	2.25±0.09
	MQT	**0.121±0.004**	0.125±0.004	0.183±0.004	0.181±0.004	0.175±0.004	0.195 ±0.003	0.158±0.004
caliAdv	VTCR	**0.30±0.006**	0.207±0.004	0.127±0.004	0.126±0.004	0.138±0.005	0.08±0.002	0.13±0.003
	QTCR	**0.631±0.008**	0.682±0.005	0.829±0.006	0.829±0.005	0.809±0.006	0.85±0.004	0.79±0.005
	QTPR	**0.623±0.03**	0.815±0.02	1.86±0.04	1.33±0.06	1.49± 0.07	2.02±0.16	1.88±0.09
	MQT	**0.096±0.003**	0.097 ±0.002	0.143±0.008	0.146±0.005	0.143±0.004	0.159±0.003	0.139±0.003
disHolly	VTCR	**0.255±0.007**	0.239±0.006	0.20±0.005	0.20±0.005	0.20±0.004	0.114±0.004	0.194±0.005
	QTCR	**0.693±0.007**	0.696±0.006	0.769±0.006	0.77±0.005	0.768±0.005	0.849±0.004	0.754±0.006
	QTPR	**0.769±0.04**	0.915±0.05	1.99±0.13	1.96±0.12	2.22±0.14	0.98±0.03	2.01±0.09
	MQT	**0.143±0.004**	0.150±0.004	0.18±0.005	0.18±0.005	0.18±0.005	0.208±0.005	0.20±0.006
MagicK	VTCR	**0.254±0.004**	0.199±0.003	0.126±0.003	0.123±0.003	0.155±0.006	0.07±0.002	0.16±0.003
	QTCR	**0.676±0.004**	0.70±0.004	0.842±0.004	0.834±0.006	0.802±0.005	0.87±0.002	0.77±0.004
	QTPR	**0.669±0.02**	0.83±0.02	1.59±0.03	1.47 ±0.04	1.42±0.06	1.14±0.03	1.58±0.04
	MQT	**0.111±0.003**	0.12±0.002	0.18±0.006	0.173±0.004	0.148±0.005	0.198 ±0.003	0.161±0.003

Table 5. Execution time (Minutes) of each successful itinerary comparison among the proposed *EffiTourRec* and existing algorithms of various datasets. The bold blue numbers express the best result.

Datasets	EffiTourRec	PersQ	PersTour	TourRecInt	TripBuilder	Efficiency
disland	**0.34**	0.69	4.2	5.06	5.74	50.72%
Epcot	**0.21**	0.35	1.57	1.59	1.87	40.00%
caliAdv	**0.26**	0.64	3.6	3.8	4.01	59.39%
disHolly	**0.16**	0.31	0.74	0.88	1.02	48.39%
MagicK	**0.22**	0.68	8.95	6.93	5.63	67.64%

6.4.4. Runtime Analysis between Existing Methods and EffiTourRec

Previously, we have shown the effectiveness of our method. This part analyzes why our method run time is faster than the baselines. The result on Table 5 shows that the proposed algorithm *EffiTourRec* finds recommended itineraries from theme park datasets more efficiently than the existing baselines. The existing *IHA* and *UBCF* algorithm recommend top-k POIs based on heuristic and they do not make these POIs as an itinerary recommendation nor they consider the various practical spatial and temporal constraints which are associated with the itinerary plan. Thus, in this itinerary planning, we do not consider these two algorithms (*IHA* and *UBCF*) time complexity analysis. We consider time comparison among our proposed algorithm and other baseline algorithms that recommend itinerary paths. We have used adaptive Monte Carlo Tree Search with effective heuristics to select potential next POI for creating itinerary paths in our proposed method. We also used MCTS search pruning that avoids non-optimal and duplicate itineraries generation. Thus, the time efficiency of the proposed *EffiTourRec* algorithm is the best among all baseline algorithms for all datasets. We set $maxLoop = 1000$ as the value allows the algorithms to complete itinerary in a reasonable time.

The results show that the run time of algorithms to find a successful itinerary changes with the number of sequences and number of POIs. The run time is proportional to the number of sequences and the number of POIs. Table 5 shows that the average itinerary constructing time of the proposed method is only 0.16 minutes at Hollywood dataset that consists of 13 POIs. The execution time depends on number of POIs and number of sequences in each dataset. The runtime of different datasets shows that the proposed *EffiTourRec* algorithm finds personalized recommended itineraries more efficiently than the existing baseline algorithms. We can see the ratio(%) efficiency values with the best existing algorithm and our proposed *EffiTourRec* algorithm in the last column of Table 5. It is clear that the proposed algorithm is a minimum 40.00 % faster than the bests existing baselines in Epcot dataset and maximum 67.64% faster in Magic Kingdom dataset. All other baselines are more time-consuming than the proposed method.

Table 6. Number of moves comparison of proposed *EffiTourRec* and existing *PersQ* algorithm of various datasets

Algorithms	disland	Epcot	caliAdv	disHolly	MagicK
PersQ	6087171	2069796	4838537	1334913	5165181
EffiTourRec	**3492072**	**997044**	**1795576**	**638150**	**2068937**
Moves Reduce	42.63%	51.83%	62.89%	52.19%	59.77%

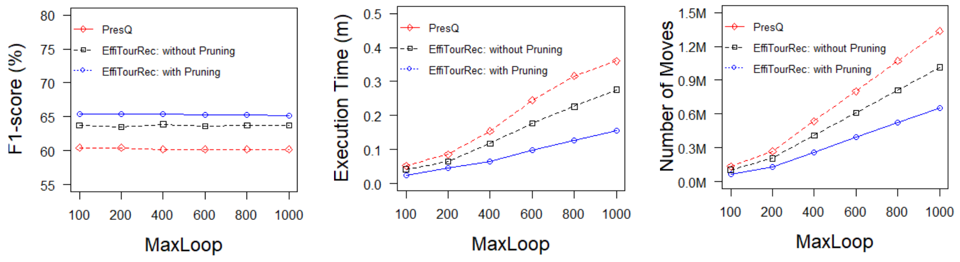


Fig. 7. Effects of *MaxLoop* threshold value in Disney Hollywood dataset.

6.4.5. Number of Moves Analysis between EffiTourRec and Existing Method

Number of moves indicates the efficiency of the algorithms. If number of moves is less number that means it requires less time to generate an itinerary to the visitor. The number of potential POI moves of proposed *EffiTourRec* and existing *PersQ* algorithms are viewed in Table 6 for five datasets. All the values show that our proposed algorithm requires fewer number moves than the existing algorithm. The last row of the table 6 depicts the number of moves reduce ratio in percentage. We can see that our proposed algorithm efficiently prunes unnecessary moves using pruning rules and makes our model time efficient. It shows that compared to *PersQ* algorithm, our proposed method is able to reduce the number of moves by 42.63% to 62.89% in various datasets.

6.4.6. Effectiveness and Efficiency Analysis based on MaxLoop

In this work, the execution time and other evaluation metric values change with *MaxLoop* values change, which is significant for our proposed *EffiTourRec* and existing *PersQ* algorithms. These two algorithms used adaptive Monte Carlo Tree Search considering *MaxLoop*, whereas other baseline algorithms are maximum loop-free. The effects of the maximum loop iteration threshold on F1-score, execution time and the number of moves of our proposed method and existing method are shown in Fig. 7. In this research paper, we have two-fold contributions: effective heuristic and efficient pruning technique. To show the impact of heuristic and pruning technique, we have used two different approaches that are *EffiTourRec: without Pruning* and *EffiTourRec: with Pruning*. The results shows that *EffiTourRec:without Pruning* method is effective and efficient than the existing *PersQ* method. On the other hand, *EffiTourRec: with Pruning* technique is the most effective than the without pruning technique and existing *PersQ* algorithm. Although all datasets show the same result trends, Fig. 7 shows Disney Hollywood dataset results. First, second and third figure in Fig 7 depicts F1-score, the execution time of a successful itinerary and the total number of moves, respectively.

6.5. Discussion

The vast application of personalized itinerary recommendation system increases due to the importance of tour planning researches in real life theme parks tours or unknown city tours. The proposed *EffiTourRec* algorithm outperforms because of its effective heuristic and efficient pruning technique. In the proposed heuristic, we focus on POI popularity, user's interest, travel time, visiting time, and queuing time which are essential fundamental factors. Previous studies were concerned about the shortest visiting time POI visit priority, but they did not notice each POI queuing time relatively high. That is why the queuing time ratio increases instead of the POI visiting time ratio, which makes visitors bored. On the other hand, the user's interest measurement depends not only on the number of POI visits and the taken photos but also on spending time on that particular POI. Existing models did not consider spending time impact to measure users interest. These two issues make our algorithm more effective than the baselines. Moreover, we use the pruning technique, which avoids low rewarded POI path exploration and avoids repeated exploration in the early stage. This technique saves a huge number of moves and time. Table 5 and 6 show the time and move efficiency, respectively.

These results present the best performance of our proposed *EffiTourRec* algorithm over the baseline algorithms. The baseline algorithms were included to depict only effectiveness based on evaluation metrics, which did not consider the efficiency measurement of these algorithms. To compare the time efficiency of our proposed algorithm and existing baseline algorithms, we show that our proposed algorithm is at least fifty percent faster than the baseline algorithms. Moreover, the proposed algorithm is able to reduce around fifty percent of POI moves because of its efficient pruning technique. However, our proposed method is able to consistently recommend itineraries that consist of higher precision, recall, F1-score and visiting time, and lower queuing time. Moreover, the itineraries recommended by the *EffiTourRec* algorithm utilize the visiting time more efficiently than the existing baselines.

7. Conclusion and Future Work

In this work, our main goal is to propose an effective and efficient itinerary recommendation that users get the maximum reward within budget time. We have designed a new algorithm *EffiTourRec* to recommend personalized itineraries based on adaptive Monte Carlo Tree Search using effective heuristic and efficient pruning techniques. Our proposed *EffiTourRec* shows that visitors' preferences proportion to prior visit time along with POI popularity and visitor interest. We evaluate experimental results on five theme park real datasets and show that proposed algorithm outperforms the state of the art in terms of tour precision, recall, F1-score, visiting time, queuing time, POI popularity and visitor interest alignment of POI visits. It also shows that proposed method is 40.00% to

67.64% faster than the existing baselines and prunes 42.63% to 62.89% moves than the existing algorithm. To our knowledge, it is the first work in personalized itineraries recommendation that considered both effectiveness and efficiency evaluation metrics simultaneously.

In our future work, we will consider dynamic queuing time managements and queuing capacity in each POI, instead of static queuing time management. We also intend to address a visitor's dynamic sentiment and activity-based interests for personalized tour recommendations.

References

[1] R. Baral, T. Li, and X. Zhu. Caps: Context aware personalized poi sequence recommender system. *arXiv preprint arXiv:1803.01245*, 2018.

[2] I. R. Brilhante, J. A. Macedo, F. M. Nardini, R. Perego, and C. Renso. On planning sightseeing tours with tripbuilder. *Information Processing & Management*, 51(2):1–15, 2015.

[3] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[4] N. Burch and R. C. Holte. Automatic move pruning in general single-player games. In *Fourth Annual Symposium on Combinatorial Search*, 2011.

[5] L. Castillo, E. Armengol, E. Onaindía, L. Sebastiá, J. González-Boticario, A. Rodríguez, S. Fernández, J. D. Arias, and D. Borrajo. Samap: An user-oriented adaptive system for planning tourist visits. *Expert Systems with Applications*, 34(2):1318–1332, 2008.

[6] A.-J. Cheng, Y.-Y. Chen, Y.-T. Huang, W. H. Hsu, and H.-Y. M. Liao. Personalized travel recommendation by mining people attributes from community-contributed photos. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 83–92. ACM, 2011.

[7] C. Cheng, H. Yang, I. King, and M. R. Lyu. Fused matrix factorization with geographical and social influence in location-based social networks. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[8] C. Cheng, H. Yang, I. King, and M. R. Lyu. A unified point-of-interest recommendation framework in location-based social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):10, 2016.

[9] E. C. Daniels, J. B. Burley, T. Machemer, and P. Nieratko. Theme park queue line perception. *línea]. Disponible en: http://www. iaras. org/iaras/filedownloads/ijch/2017/017-0012*, 2017.

[10] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, pages 35–44. ACM, 2010.

[11] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Constructing travel itineraries from tagged geo-temporal breadcrumbs. In *Proceedings of the 19th international conference on World wide web*, pages 1083–1084. ACM, 2010.

[12] M. Debnath, P. K. Tripathi, A. K. Biswas, and R. Elmasri. Preference aware travel route recommendation with temporal influence. In *Proceedings of the 2nd ACM SIGSPATIAL Workshop on Recommendations for Location-based Services and Social Networks*, page 2. ACM, 2018.

[13] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[14] I. Dolinskaya, Z. E. Shi, and K. Smilowitz. Adaptive orienteering problem with stochastic travel times. *Transportation Research Part E: Logistics and Transportation Review*, 109:1–19, 2018.

[15] M. Dorigo, M. Birattari, and T. Stützle. Ant colony optimization. *IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE*, 1556(603X/06), 2006.

[16] B. Du, Y. Cui, Y. Fu, R. Zhong, and H. Xiong. Smarttransfer: Modeling the spatiotemporal dynamics of passenger transfers for crowdedness-aware route recommendations. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(6):70, 2018.

[17]J. Duguépéroux, A. Mazyad, F. Teytaud, and J. Dehos. Pruning playouts in monte-carlo tree search for the game of havannah. In *International Conference on Computers and Games*, pages 47–57. Springer, 2016.

[18]Q. Fang, C. Xu, M. S. Hossain, and G. Muhammad. Stcaplrs: A spatial-temporal context-aware personalized location recommendation system. *ACM Transactions on Intelligent systems and technology (TIST)*, 7(4):59, 2016.

[19]R. Gao, J. Li, X. Li, C. Song, J. Chang, D. Liu, and C. Wang. Stscr: Exploring spatial-temporal sequential influence and social information for location recommendation. *Neurocomputing*, 319:118–133, 2018.

[20]I. Garcia, L. Sebastia, and E. Onaindia. On the design of individual and group recommender systems for tourism. *Expert systems with applications*, 38(6):7683–7692, 2011.

[21]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. ACM, 2014.

[22]A. Gunawan, H. C. Lau, and P. Vansteenwegen. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315–332, 2016.

[23]A. GUNAWAN, Z. YUAN, and H. C. LAU. A mathematical model and metaheuristics for time dependent orienteering problem. In *Proceedings of the 10th International Conference of the Practice and Theory of Automated Timetabling*, pages 202–217, 2014.

[24]H.-P. Hsieh, C.-T. Li, and S.-D. Lin. Triprec: recommending trip routes from large scale check-in data. In *Proceedings of the 21st International Conference on World Wide Web*, pages 529–530. ACM, 2012.

[25]Y.-L. Hsueh and H.-M. Huang. Personalized itinerary recommendation with time constraints using gps datasets. *Knowledge and Information Systems*, pages 1–22, 2018.

[26]G. Hu, Y. Qin, and J. Shao. Personalized travel route recommendation from multi-source social media data. *Multimedia Tools and Applications*, pages 1–16, 2018.

[27]R. Ji, X. Xie, H. Yao, and W.-Y. Ma. Mining city landmarks from blogs by graph modeling. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 105–114. ACM, 2009.

[28]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.

[29]L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[30]L. Kocsis, C. Szepesvári, and J. Willemson. Improved monte-carlo search. *Univ. Tartu, Estonia, Tech. Rep*, 1, 2006.

[31]R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on Artificial intelligence-Volume 2*, pages 1137–1143. Morgan Kaufmann Publishers Inc., 1995.

[32]S. Kotiloglu, T. Lappas, K. Pelechrinis, and P. Repoussis. Personalized multi-period tour recommendations. *Tourism Management*, 62:76–88, 2017.

[33]X. Li. Multi-day and multi-stay travel planning using geo-tagged photos. In *Proceedings of the second ACM SIGSPATIAL international workshop on crowdsourced and volunteered geographic information*, pages 1–8. ACM, 2013.

[34]Z. Liao and W. Zheng. Using a heuristic algorithm to design a personalized day tour route in a time-dependent stochastic environment. *Tourism Management*, 68:284–300, 2018.

[35]K. H. Lim. Recommending tours and places-of-interest based on user interests from geo-tagged photos. In *Proceedings of the 2015 ACM SIGMOD on PhD Symposium*, pages 33–38. ACM, 2015.

[36]K. H. Lim, J. Chan, S. Karunasekera, and C. Leckie. Personalized itinerary recommendation with queuing time awareness. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 325–334. ACM, 2017.

[37]K. H. Lim, J. Chan, S. Karunasekera, and C. Leckie. Tour recommendation and trip planning using location-based social media: A survey. *Knowledge and Information Systems*, 60(3):1247–1275, 2019.

[38]K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera. Towards next generation touring: Personalized group tours. In *ICAPS*, pages 412–420, 2016.

[39]K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera. Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency. *Knowledge and Information Systems*, 54(2):375–406, 2018.

[40]K. H. Lim, X. Wang, J. Chan, S. Karunasekera, C. Leckie, Y. Chen, C. L. Tan, F. Q. Gao, and T. K. Wee. Perstour: A personalized tour recommendation and planning system. In *HT (Extended Proceedings)*, 2016.

[41]P. Lou, G. Zhao, X. Qian, H. Wang, and X. Hou. Schedule a rich sentimental travel via sentimental poi mining and recommendation. In *2016 IEEE second international conference on multimedia big data (BigMM)*, pages 33–40. IEEE, 2016.

[42]F. Maes, D. L. St-Pierre, and D. Ernst. Monte carlo search algorithm discovery for single-player games. *IEEE Transactions on Computational Intelligence and AI in Games*, 5(3):201–213, 2013.

[43]D. H. Maister et al. *The psychology of waiting lines.* Harvard Business School Boston, MA, 1984.

[44]M. Oshima, K. Yamada, and S. Endo. Effect of potential model pruning on different-sized boards in monte-carlo go. *Procedia Computer Science*, 12:146–151, 2012.

[45]N. Sephton, P. I. Cowling, E. Powley, and N. H. Slaven. Heuristic move pruning in monte carlo tree search for the strategic card game lords of war. In *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*, pages 1–7. IEEE, 2014.

[46]Y. Shi, P. Serdyukov, A. Hanjalic, and M. Larson. Nontrivial landmark recommendation using geotagged photos. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(3):47, 2013.

[47]R. W. Sinnott. Virtues of the haversine. *Sky Telesc.*, 68:159, 1984.

[48]Y. Sun, H. Fan, M. Bakillah, and A. Zipf. Road-based travel recommendation using geotagged images. *Computers, Environment and Urban Systems*, 53:110–122, 2015.

[49]T. Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35(9):797–809, 1984.

[50]P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. Van Oudheusden. The city trip planner: an expert system for tourists. *Expert Systems with Applications*, 38(6):6540–6546, 2011.

[51]P. Varakantham and A. Kumar. Optimization approaches for solving chance constrained stochastic orienteering problems. In *International Conference on Algorithmic Decision-Theory*, pages 387–398. Springer, 2013.

[52]M. Ye, P. Yin, and W.-C. Lee. Location recommendation for location-based social networks. In *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, pages 458–461. ACM, 2010.

[53]M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 325–334. ACM, 2011.

[54]H. Yoon, Y. Zheng, X. Xie, and W. Woo. Smart itinerary recommendation based on user-generated gps trajectories. In *International Conference on Ubiquitous Intelligence and Computing*, pages 19–34. Springer, 2010.

[55]C. Zhang, H. Liang, and K. Wang. Trip recommendation meets real-world constraints: Poi availability, diversity, and traveling time uncertainty. *ACM Transactions on Information Systems (TOIS)*, 35(1):5, 2016.

Author Biographies

Sajal Halder is currently working toward a Ph.D. degree in Computer Science at RMIT University, Australia. He has been appointed as a faculty member in the Department of Computer Science and Engineering at Jagannath University, Bangladesh, since August 2016. He received B.Sc in Computer Science and Engineering from University of Dhaka, Bangladesh in 2010 and Master of Engineering from Kyung Hee University, South Korea in 2013. His research interests are in Personalized Itinerary Recommendation, Deep Learning, Reinforcement Learning and Periodic Behavior Mining in Large Scale Graph.

Kwan Hui Lim is an Assistant Professor at the Information Systems Technology and Design Pillar, Singapore University of Technology and Design, and leads the Social and Urban Analytics Lab. Previously, he was a Research Fellow at the School of Computing and Information Systems, University of Melbourne, Research Engineer at the Living Analytics Research Centre, Singapore Management University, Research Intern at IBM Research - Australia, and various visiting appointments. He received his PhD from the University of Melbourne, and MSc (Research) and BCompSci (1st Class Honours) from the University of Western Australia. He is a recipient of the 2016 Google PhD Fellowship in Machine Learning. His research interests are in Data Mining, Machine Learning, Artificial Intelligence, Social Network Analysis, and Social Computing.

Jeffrey Chan is a Senior Lecturer at RMIT University, Melbourne, Australia. He has a BEng, BSci and Ph.D., all from the University of Melbourne, and has interned at the National Institute of Informatics in Japan. He was previously a senior post-doctoral fellow at the Digital Enterprise Research Institute in Ireland and a research fellow at the University of Melbourne. He has over 100 publications in graph mining, social network analysis and data mining and his research interests are in Data Analytics, Analysing Graphs and Social Networks and learning about new and exciting research.

Xiuzhen (Jenny) Zhang is a Professor in the School of Science (Computer Science and IT), RMIT University, Australia. She has received the Ph.D. degree in computer science from The University of Melbourne, Australia. She has been working in data mining and machine learning for more than 15 years and has published more than 100 papers in reputable journals and conferences including the IEEE Transactions on Knowledge and Data Engineering, KDD and PAKDD. She has been a PC member for international and Australian conferences including KDD, PAKDD, DEXA, and ADC. Her research has been supported by Australian Research Council, Victorian State government, and industry grants.