Top-k Context-Aware Tour Recommendations for Groups

Frederick Ayala-Gómez[1,2], Barış Keniş[2], Pınar Karagöz[3], and András Benczúr[4]

[1] Faculty of Informatics, Eötvös Loránd University, Pázmány P. sny. 1/C., H-1117 Budapest, Hungary
fayala@caesar.elte.hu
[2] Computer Science Department, Aalto University, 02150 Espoo, Finland
baris.kenis@aalto.fi
[3] Computer Eng. Department, METU, A-404, 06800, Ankara Turkey
karagoz@ceng.metu.edu.tr
[4] Inst. Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), H-1111, Budapest, Hungary
benczur@sztaki.mta.hu

Abstract. Cities offer a large variety of Points of Interest (POI) for leisure, tourism, culture, and entertainment. This offering is exciting and challenging, as it requires people to search for POIs that satisfy their preferences and needs. Finding such places gets tricky as people gather in groups to visit the POIs (e.g., friends, family). Moreover, a group might be interested in visiting more than one place during their gathering (e.g., restaurant, historical site, coffee shop). This task is known to be the orienteering under several constraints (e.g., time, distance, type ordering). Intuitively, the POI preference depends on the group, and on the context (e.g., time of arrival, previously visited POIs in the itinerary). Recent solutions to the problem focus on recommending a single itinerary, aggregating individual preferences to build the group preference, and contextual information does not affect the scheduling process. In this paper, we present a novel approach to the following setting: Given a history of previous group check-ins, a starting POI, and a time budget, find top-k sequences of POIs relevant to the group and context that satisfy the constraints. Our proposed solution consists of two primary steps: training a POI recommender system for groups, and solving the orienteering problem on a candidate set of POIs using Monte Carlo Tree Search. We collected a ground-truth dataset from Foursquare, and show that the proposed approach improves the performance in comparison to a Greedy baseline technique.

Keywords: Recommender Systems for Groups · Tour Recommendation · Orienteering

1 Introduction

The web changed the way people interact with each other and with organizations [13]. As a part of the web, Location-Based Social Networks (LBSNs) help

people to track their leisure and tourism activities. In populous cities, LBSNs help users navigate the large variety of Points of Interest (POI) such as parks, museums, restaurants, etc. Due to the large amount of POIs, finding venues that are relevant to the user preferences is time-consuming. To help with this task, POI recommendation is a popular research problem. Most of the research in POI recommendation focus on recommending to one user [25]. However, as people gather in groups – which is a frequent situation in social life (e.g., holidays, weekend, birthdays), the preferences of the individuals might change and conflict. Research on POI recommendations for groups is scarse but a promising and emerging area. In this paper, we focus on building recommendations for groups (i.e., gatherings of more than one user). More specifically, groups have at least two members. A user might be a member of many groups. And groups are considered to be different if any of their members is added or removed.

When groups are planning to visit more than one POI in their gathering, to a sequence of POIs, the task is more challenging, as additional constraints (e.g., time) must be satisfied. This task is known as the group orienteering problem. In the group orienteering problem, the preferences of a group of people are considered. Intuitively, group's preferences vary according to the members. For example, the group might prefer different venues when it is a *best friends gathering* compared to a family gathering or meeting with colleagues after work. Moreover, the group may have specific constraints (e.g., category sequence, duration, time). In this work, we investigate the group orienteering problem. More specifically, we study the problem *recommending top-k sequences of POIs for a known group, given a starting POI, time constraint, and contextual information (e.g., popularity of the venue at time of arrival)*. Our study focuses on LBSNs as the data source, as they enable groups to share their activities by doing *check-ins* on the visited venues. Therefore, it is possible to find group traces of visits as well as individual visits. These traces provide important insights of group preferences.

The proposed solution differs from related studies in the literature in several ways. Recent work on group orienteering (i.e. itinerary planning for group ([9], [2])) modeled groups preference in terms of individual preferences. In our work, we model group preference on the basis of observed group behavior. The motivation behind this choice is that, in [3], it is reported that recommending POIs for groups based on the group preferences performs better than the aggregation of individual preferences. Thus, we model group preferences directly from the group behavior observations and use. Additionally, building the itinerary depends on contextual information such as the venue's popularity at the time of arrival, previous visited POIs, and POI category transitions. For example, assume that the group is visiting a restaurant. It is likely that the next POI will not be a restaurant, but rather a coffee shop, or a plaza. Moreover, if at the time of expected arrival to the next POI, the venue is unpopular, selecting the venue is less likely, such as arriving to a night-club in the afternoon. Recent work does not incorporate contextual information during the itinerary planning. In our work, we incorporate such features in our solution. Finally, recent work focus on

recommending one itinerary for the group, and our work focus on recommending top-k itineraries.

Our proposed approach for building top-k tour recommendations for groups consists of two main steps. In the first step, we model group preferences using group check-ins and train a recommender systems to learn the group preferences. In the second step, we generate a candidate set of POIs near the starting POI, rank the POIs using the recommender system, and solve the orienteering problem using Monte Carlo Tree Search (MCTS), where the expected reward of the POIs depend on the POI recommendation score and on contextual features.

We conducted the experiments on a real dataset of check-ins collected from Swarm by Foursquare, a popular LBSN, and use it as a ground-truth. The collected dataset contains group check-ins as well as individual check-ins. The proposed solution differs from related studies in the literature in several ways. First, we analyze the coverage of the generated POI candidates near a given venue. Second, we compare the performance of various recommender systems and different individual aggregation strategies. Implicit matrix factorization for groups performed better than content-based, item-to-item. Moreover, we show that training a recommender systems using group profiles perform better than aggregating individual recommendations. Finally, our results show that MCTS constructs better itineraries in terms of recall and precision, than the greedy baseline.

The paper is organized as follows. Section 2, summarizes previous studies on group recommendation and itinerary planning. Section 3 presents our proposed approach for building top-k itineraries for groups. Section 4 explains the data used as ground truth, and a data analysis. Section 5 presents the evaluation methodology, and experimental results. Finally, Section 6 presents findings and conclusions.

2 Related Work

We categorize related studies in the literature into two: studies on item recommendations for groups, and studies on itinerary recommendations. In this section, we summarize the literature for both of these problems.

2.1 Item Recommendation for Groups

In [1], as one of the earliest works on group recommendation, the problem is defined as a consensus function aiming to maximize relevance and minimize disagreement. Their algorithm uses a disagreement list, and several variations of the algorithm are presented. In [19], personal impact of groups members on group decisions are modeled as a latent variable. The model is further refined by exploring the relationship between personal impact and other features (e.g., social network). In [26], the proposed method for group recommendation is based on aggregating preferences of the group member with different weights. In [12], rather than aggregating member preferences, group preference is modeled as a

distribution of member's ratings. On the basis of this distribution, item selection is solved through a multi-criteria decision making process. The study in [6] focuses on diversity and novelty aspects in group recommendation. The different preferences of the group members are considered as group hesitation, and group preference is modeled through Hesitant Fuzzy Sets (HFSs). In accordance with the focus of the paper, evaluation involves diversity and novelty measurement in addition to accuracy. The study in [24] models the group recommendation problem as a multi-criteria optimization problem to maximize the satisfaction of each group member and to minimize the unfairness within the group. Fairness metric employed aim to minimize the individual utility differences in the group. In the study, optimality is defined as *Pareto-optimality*, and to solutions are presented in order to achieve Pareto-efficient solution. In [21], in addition to location context, they focus on passive users who do not explicitly specify preferences. Such users are represented through topic modeling. In order to improve time efficiency of the proposed method, the authors additionally propose an index structure, namely *Topic-aware R-tree*. The work in [3] proposes a class of hybrid recommendation algorithms that combine a set of features including geographical preferences, venue category and group check-in history in order to recommend a list of POIs to a group. Empirical analysis reveal that the individual preferences deviate from preference of groups. Hence recommendation under group preference model performs better than aggregation of individual preferences.

2.2 Itinerary Recommendation

Research on itinerary recommendation mostly focus on recommendation for individuals. In [8], it is aimed to generate inter-city travel itineraries by using geo-temporal traces (i.e., breadcrumbs) left by travelers in social media platforms. They use Flickr[5], by mapping the photos taken by users to POIs used to construct paths. The generated paths are mined under several constraints to construct travel itineraries. The work given in [10] also uses geo-temporal traces of travelers on LBSNs. Their aim is to construct tour itinerary within a city. The authors propose two algorithms to construct itinerary under several constraints, including those on venue types and distance. In [23], authors consider itinerary planning as an iterative process. At each step, the user gives feedback on the POIs selected by the system. This feedback is used to pick the next steps in the itinerary. The problem is modeled as a rooted orienteering problem, and the proposed optimization solution is based on finding a Hamiltonian path in a hypercube. For time-efficiency, heap-based data structures are used. The work presented in [7] aim to construct more general itineraries involving several days. A two-step solution is proposed. In the first step, all possible single-day itineraries are constructed. In the second step, the single-day itineraries are combined selectively for optimal multi-day itinerary. Hence, the itinerary planning problem is modeled as a set-packing problem with good approximate algorithms, instead

[5] https://www.flickr.com/

of NP-complete tour orienteering problem. In [17], itinerary planning problem is considered as orienteering problem as in the previous studies. However, the major difference in the paper is that the user preference/interest for a POI is weighted though visit duration rather than visit frequency. This weighting approach is further enhanced by considering recency of past visits. In [16], the same authors studied the itinerary planning problem under queuing time consideration. The objective function of this new version of the problem aims to maximize user interest and POI popularity whereas to minimize queuing time.

Recently in the literature, itinerary recommendation for groups has attracted attention as a research problem. In [9], a multi-user itinerary planning solution is proposed. The proposed algorithm is based on aggregating individual preferences while considering agreement and disagreement among the group members. The authors propose two algorithms: the first constructs the optimal itinerary under limitations, and the second one generates an approximate solution with bounded approximation ratio. In [2], the authors consider the task to be the *orienteering* problem, as well, where the goal is to find a path in a graph within a time budget, that has the best value for the group. The group value is an objective function that aggregates individual preferences of the POIs in the path. Their solution for *TourGroup* has two steps. In the first step, they compute the individual preference for the POIs using a content-based recommender system. In the second step, they build the itineraries for groups by solving *TourGroup* for different objective functions: *i) TourGroupSUM*, where they added the individual values; *ii) TourGroupMIN*, where the goal is to find the maximum of the least liked individual path; *iii) TourGroupFAIR*, where the score is the average value of the individual paths minus a weighted standard deviation. For these problems, the authors applied various search methods including, greedy heuristics, dynamic programming, and ant colony optimization. They found the best results for itinerary construction using Ant Colony, although it did not vary much from the greedy approach

In our work, one of the basic differences from the last two summarized work is that rather than aggregating group members' preferences, we use the known group preferences. In that sense, we have a common approach with the study in [3]. Moreover, we focus on recommending a list of itineraries instead of a single itinerary. In terms of the methods used to solve the orienteering task, we use Monte Carlo Tree Search (MCTS), and added contextual features during the itinerary planning such as time and transition probabilities.

3 Proposed Approach

At a glance, our proposed approach consists of two steps. In the first step, we train a recommender system using the known check-ins from a ground-truth data set. In the second step, from a starting POI, we generate a set of K neighbor POIs used as candidate set, and score them with the recommender system (from step one). These ranked candidates are used as input to solve the orienteering

problem with Monte Carlo Tree Search. The overview of the process is presented in Figure 1.

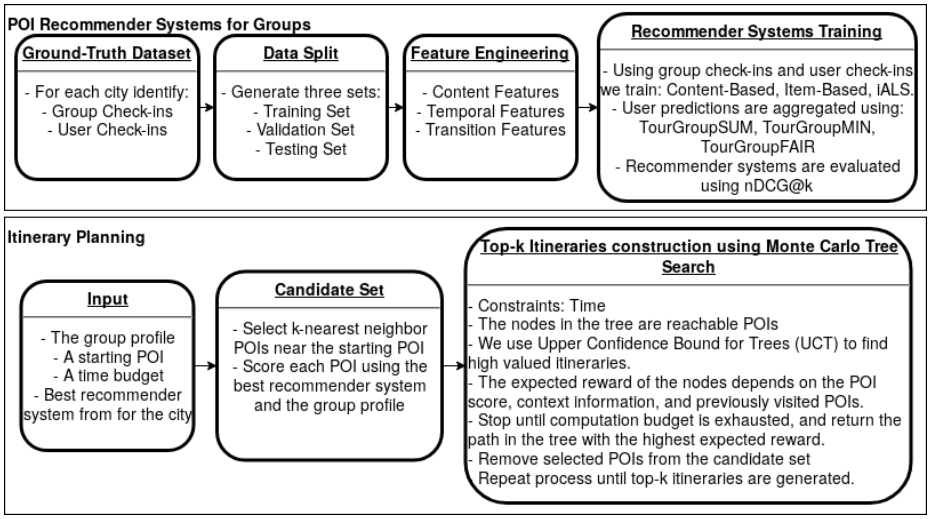


Fig. 1: Overview of the proposed approach.

We define the problem of top-k tour recommendations for groups as follows. Given a group G of LBSN users, a set of POIs P, and a POI p_0 as a starting point, the aim is to construct K paths $I =< p_0, ..p_n >$, where $p_i \in P$, such that I fulfills a given time budget. Additionally, the intersection of all K paths equals to p_0.

3.1 Candidate Set

In [3], the authors observed that groups do not travel long distances between check-ins. Based on this observation, we included a candidate set generation step that takes into account POI near the *starting POI* of the group. That is, for a given location, we consider K number of nearest neighbors. We measure the quality of the candidate set through coverage, such that, checking whether the *next POI* that the group goes to is in the candidate set or not. Figure 2 presents the coverage obtained at different candidate set sizes. As seen in the figure, as the number of neighbors increases, the coverage gets closer to 1.0. However, high number of neighbors incurs computational cost in scoring phase (through recommender system), therefore we need to limit the number while retaining good enough coverage.

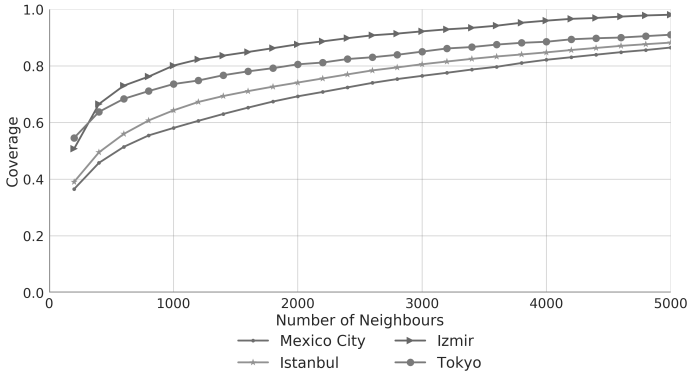


Fig. 2: Coverage of the next venue at different number of neighbors for the cities. As we get a larger set of near venues it is more likely to find the next POI that the group visited next.

3.2 Recommender Systems

We model the group preference for each of the POIs in the candidate set using well-known recommender systems from the literature. We experiment with content-based recommender system [22] based on the content features described in Section 4.3, item-based [18], and collaborative filtering model for implicit feedback (iALS) [14].

We build recommendations for groups using two approaches. The first approach is to create a profile for each of the groups using the observed check -ins in the ground-truth dataset. Then, we train the recommender systems to learn the preferences of the group using the history of check-ins. The seconds approach is to build a profile for each user using the known check-ins in the ground-truth dataset. Then, the group recommendation is computed as an aggregation of the predicted score for each of the users in the group. Similar to [2], we use three different aggregation functions: *i) IndividualSUM* is the sum of the group members' recommendations, *ii) IndividualMIN* the min value of the group members' recommendations, and *iii) IndividualFAIR* assigns the mean value of the scores minus one standard deviation.

3.3 Itinerary Construction

Our proposed solution for the orienteering problem uses Monte Carlo Tree Search (MCTS) to efficiently find a promising itinerary. MCTS is built on the idea of combining tree search with random sampling, and finds optimal decision paths [5].

There are different variations of MCTS [5]. We focus on Bandit-Based Methods, as they are a well-known class of sequential decision problems. Specifically, we use Upper Confidence Bound for Trees (UCT), "the most popular algorithm

in the MCTS family"[5]. UCT iteratively builds a tree by expanding nodes in each iteration, selecting the most promising node, estimating the expected reward of the selected node, and back-propagating the reward to the parents. This process is repeated until certain computational budget is reached. Finally, starting from the root, we pick the child node with highest expected rewards until we reach a terminal node. In our case, each node represents visiting a POI. The main function of the UCT algorithm is defined in Algorithm 1, and the following is an explanation of each of the functions.

Tree policy expands a node if it is not fully expanded or terminal, otherwise it picks the *best child*, which is the node that requires more attention at the current iteration, and controls the trade between exploration and exploitation. The UCT algorithm picks the *best child* that maximizes the Upper Confidence Bounds defined in Equation 1.

$$UCT(j) = \bar{X}_j + 2C_p\sqrt{\frac{2\ln n}{n_j}}, \tag{1}$$

where $\bar{X}_j$ is the average expected reward of a candidate node j calculated during default policy step, C_p controls the exploration-exploitation trade-off. For expected rewards between [0,1], $\frac{1}{\sqrt{2}}$ is recommended [15], n is the total number of visits to all nodes, and n_j the visits to node j.

Default policy simulates an expected reward, if the node is non-terminal. A terminal node is the case when the time budget is exhausted and no more POIs can be visited. In the simplest case, the algorithm picks uniformly random unseen reachable nodes, until a terminal node is found. The expected reward is the average of the nodes visited during simulation. If the node is terminal, the expected reward is score of the node.

Backup is the process of updating the parents of the selected best-child according to the expected reward.

```
UCTSearch(s_0)
create root node v_0 with starting POI
while within computational budget do
    v_j ← TreePolicy(v_0)
    Δ ← DefaultPolicy(s(v_j))
    Backup(v_j, Δ)
end
return BestChilds(v_0)
```

Algorithm 1: The UCT Algorithm.

To generate top-k recommendations, we run the UCT algorithm, select the best itinerary, remove the selected POIs in the itinerary, and run again UCT until K itineraries are generated. We propose three versions of the MCTS method.

MCTS_Simple selects uniformly random the POIs. The expected reward is defined in Equation 2.

$$x_j = \frac{\hat{r_j}}{f_{c_j} \cdot \lambda + 1}, \tag{2}$$

where $\hat{r_j}$ is the recommendation score, f_{c_j} the number of occurrences of the venue category, and λ is a category diversity parameter greater than 1.

MCTS_Score replaces the uniformly random sampling with a greedy search, using the score of the recommender system. The expected reward is defined in Equation 2.

MCTS_Context also replaces the uniformly random sampling with a greedy search, using the score of the recommender system, and contextual information. The expected reward is calculated as Equation 3.

$$x_j = \frac{\hat{r_j} \cdot popularity(p_j, t_j) \cdot ecp(c_j|c_i)}{f_{c_j} \cdot \lambda + 1}, \tag{3}$$

where $\hat{r_j}$ is the recommendation score, $popularity(v_j, t_j)$ is the proportion of check-ins observed in p_j at time of arrival t_j, $ecp(c_j|c_i)$ is the empirical conditional probability of visiting a venue of type c_j after a venue of type c_i (e.g., going to a coffee shop after a park), f_{c_j} the number of occurrences of the venue category, and λ is a category diversity parameter greater than 1.

4 Data

We start by describing our working dataset, the cleaning steps, and data statistics, followed by presenting data exploration results. Afterwards, we describe the feature engineering.

4.1 Working Dataset

Our approach requires a dataset including both individual users' preferences and group preferences. For this purpose, we collected a dataset of check-ins from *Swarm by Foursquare*, a popular LBSN. *Check-ins* are posts in LBSN with information of the venue that a user visited. Moreover, check-ins may mention other users that are with the user who checked-in. Following the idea of [3], we collected check-ins by searching for public check-ins shared on *Twitter*, a popular social network. Then, we follow a *Snowball* sampling approach [11]. For each of the group check-ins found in Twitter, we search for additional public check-ins in the group members Twitter time-line. This collection has approximately 2.7M individual check-ins, 1M group check-ins, 200K users, 400K groups, 116K venues, and 547 categories.

We followed the cleaning steps of [3]. We considered groups with maximum 12 participants, removed users and groups with high geographical dispersion (i.e.,

according to the standard deviation of their geographical mobility), and removed irrelevant venue categories (i.e., Residence, States & Municipalities, Professional & Other Places and Event, College & University, Travel & Transport, Event, Shop & Service). Table 1 presents statistics of the groups, users, venues, and categories for the Top-4 cities.

City	Check-ins	Venues	Categories	Group Check-ins	Group Venues	Group Categories
Istanbul	763K	32K	423	205K	18	362
Izmir	262K	14K	373	46K	5K	271
Mexico City	179K	19K	370	71K	12K	340
Tokyo	108K	19K	392	7K	3K	264

Table 1: Statistics for the Top-4 cities in our working dataset.

4.2 Data exploration

LBSN data is rich in contextual features such as time and location. Regarding time, there are seasonality effects for the daytime, day of the week and month. Intuitively, popularity of venues changes depending on the week, the hour, and the season. For example, a restaurant by seaside in Istanbul is more popular during summer where temperatures are around 25°C, than winter where temperatures are around 5°C. An example of the effects that the week-day, time, and venue type have on the popularity is shown in Figure 3.

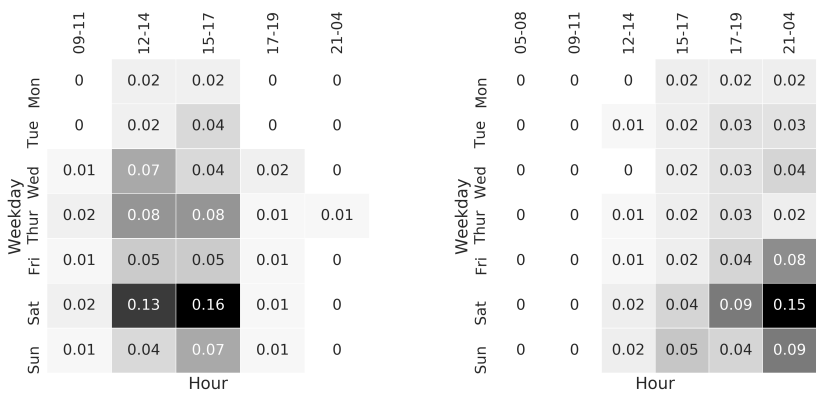
The dataset contains groups who transit from one venue to another in the same day. These item to item transitions help us to understand popular venue type transitions in a city. Table 4 shows an extract from the transition matrix for Istanbul.

4.3 Feature Engineering

From the dataset, we construct content, transition and temporal features to represent different aspects of the venues, groups, users, and context.

Content Features: Foursquare provides textual information that helps know more about the venues. These are *tips* left by other users, *phrases* that describe the venue, and *attributes* (e.g., romantic, cozy). We performed conventional preprocessing steps on these textual content by removing stop words and stemming the tokenized terms. For Spanish, Portuguese and English, we used Snowball stemmer from the Natural Language Toolkit by [4]. For Turkish, we used TurkishStemmer [6] and for Japanese, we used TinySegmenter [7]. Then, we compute

[6] https://github.com/otuncelli/turkish-stemmer-python
[7] http://tinysegmenter.tuxfamily.org/



(a) Check-in distribution for museums (b) Check-in distribution for bars

Fig. 3: Distribution of check-ins per weekday and hour for museums and bars in Istanbul. The rows are the parts of the day, and the columns are days of the week. A darker color represent higher distribution of check-ins.

tf-idf [20] on the tips, phrases, attributes, and on the venues' category tree [8]. Additionally, we include the price tier (e.g., cheap, moderate, expensive), the venue rating, and the number of check-ins as features. These are the content features used by the content-based recommender system.

Transition Features: We use pair-wise transition probabilities between the current POI category c_i and a possible next POI category c_j as transition feature. The empirical conditional probability for the venues categories observed in the training dataset is defined as $ecp(c_j|c_i) = \frac{n_{ij}}{n_i+1}$, where n_{ij} is the number of observed transitions between c_i and c_j, and c_i is the number of check-ins at c_i.

Temporal Features: To include temporal context as a feature, we split the day into 6 segments. 05-08 (Early Morning), 09-11 (Late Morning), 12-14 (Early Afternoon), 15-17 (Late Afternoon), 17-19 (Early Evening), 21-04 (Night). The distribution of check-ins for the venue and category in each of the day-segments are added as features.

5 Experiments

5.1 Data Split

For each of the cities presented in Table 1, the dataset is split into training, validation and testing datasets with respect to time order. The training dataset

[8] https://developer.foursquare.com/docs/resources/categories

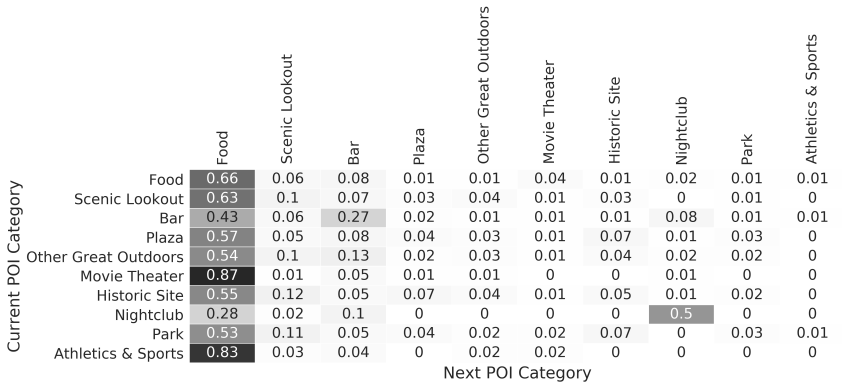


Fig. 4: Part of the transition matrix for activity types in Istanbul. The numbers represent the transition probability between two categories. Intuitively, moving from a venue of certain category conditions the next venue category. For example, if the group is in a Park, it is more likely that they will go to a Food place, than to a Nightclub. And, if the group is in a Nightclub, they are more likely to go to a Food place after than a Park.

contains the 80% of the data, from the training set we take 5% as validation set. The testing dataset is 20% of the dataset. As the ground truth, we collect itineraries by looking for check-in sequences of the same group on the same day, in the data sets.

5.2 Metrics

Our proposed approach consists of two phases: The recommender systems, which finds relevant POIs to the group; and the itinerary algorithm, which constructs the sequence of POIs according to the constraints. We evaluate the results of these phases independently.

The quality of the group recommender system is measured by evaluating the ranking and the relevance of the recommended POIs. For this, we use normalized discounted cummulative gain. This metric rewards relevant POIs that appear higher in the rank and penalizes relevant POIs that appear lower in the rank. The nDCG@100 is defined as given in Equation 4.

$$nDCG@100 = \frac{\sum_{i}^{100} \frac{2^{rel_i}-1}{log_2(i+1)}}{IDCG}, \tag{4}$$

such that rel_i is 1 if the venue was visited. The $IDCG$ is the ideal ordering of the ranking as defined in Equation 5.

$$IDCG = \sum_{i}^{|REL|} \frac{2^{rel_i}-1}{log_2(i+1)}, \tag{5}$$

where $|REL|$ is the *next* POIs visited by the group.

The itinerary construction is evaluated on how well the constrains are satisfied while maximizing the predicted preference of the group for the candidate POI. The quality of the itinerary is measured by three metrics. The value of the itinerary, as the sum of the recommendations scores in the itinerary, precision, as defined in Equation 6, and recall, as defined by Equation 7.

$$\text{Precision} = \frac{|\ \{\text{Visited POIs}\} \cap \{\text{POIs in the Itinerary}\}\ |}{|\ \{\text{POIs in the Itinerary}\}\ |} \tag{6}$$

$$\text{Recall@}K = \frac{|\ \{\text{Visited POIs}\} \cap \{\text{POIs in the Itinerary}\}\ |}{|\ \{\text{Visited POIs}\}\ |} \tag{7}$$

5.3 Baselines

The recommender systems that are used within this study are compared with each other in terms of nDCG@100 per city. For itinerary construction evaluation, MCTS_Context is compared against MCTS_Simple and MCTS_Score. Additionally, Greedy Algorithm is used another baseline for comparison. In itinerary construction, for each city, the best recommender system to score the candidate sets is used. MCTS_Context, and the three baselines include the penalization over repeating venue types in the itinerary.

Greedy Algorithm A basic approach to solve the itinerary construction problem is to use a greedy approach. Just like the MCTS, our greedy algorithm also utilizes the recommender system to score the POIs. Starting from a POI, the algorithm asks for top k candidates from the recommender system, filters out the ones that are already included, or break some constraint, and chooses the one that gives the highest score according to the recommender system. The algorithm continues next POI selection until the budget is consumed.

5.4 Recommender Systems Evaluation Results

We used Apple's Turi Create [9] recommender module for the implementation of recommender systems. We fine-tuned the parameters using grid search on the validation set. The code for generating the candidate sets, training, and predictions, was executed on a single machine with 16 GB of RAM, 16 cores.Table 2 shows the results of the different recommender systems.

5.5 Itinerary Construction Evaluation Results

We generate Top-10 itineraries for each of the group tours in the testing set. For all the itineraries we used a budget of 8 hours, and assumed that the group walks to commute between the POIs at a speed of 1.4 m/s. We took the first

[9] https://github.com/apple/turicreate

Using Next Category	Istanbul w/o	Istanbul w/	Izmir w/o	Izmir w/	Mexico City w/o	Mexico City w/	Tokyo w/o	Tokyo w/
iALS - Group	**0.546**	**0.700**	**0.550**	**0.705**	**0.531**	**0.716**	0.460	0.666
iALS - IndividualSum	0.395	0.567	0.454	0.643	0.508	0.692	0.455	0.695
Item-to-Item - IndividualSum	0.364	0.532	0.344	0.531	0.418	0.615	0.428	0.655
Item-to-Item - Group	0.502	0.650	0.271	0.441	0.204	0.395	**0.500**	**0.753**
Content - Group	0.269	0.434	0.271	0.441	0.204	0.396	0.282	0.494
Content - IndividualFair	0.246	0.414	0.261	0.430	0.186	0.384	0.145	0.350
Content - IndividualMin	0.246	0.414	0.261	0.430	0.186	0.384	0.145	0.350
Content - IndividualSum	0.246	0.414	0.261	0.430	0.186	0.384	0.145	0.350
Item-to-Item - IndividualMin	0.175	0.321	0.229	0.390	0.212	0.387	0.146	0.363
Item-to-Item - IndividualFair	0.152	0.292	0.132	0.272	0.183	0.338	0.169	0.378
iALS - IndividualMin	0.057	0.206	0.088	0.242	0.134	0.293	0.239	0.409
iALS - IndividualFair	0.034	0.163	0.060	0.163	0.075	0.215	0.112	0.326

Table 2: Results of evaluating the *Top 100* recommendations of the next POI to visit. The scores are the mean nDCG@100. In bold are the best models for each city. The abbreviation *w/* stands for recommendations filtered to a specific *next* POI category (i.e., Top 100 nearest POI of the given category), and *w/o* when the next category is not specified.

	Greedy	MCTS_Context	MCTS_Score	MCTS_Simple
Istanbul	0.318	0.357	0.356	**0.361**
Izmir	0.276	0.317	0.315	**0.322**
Mexico City	0.417	**0.445**	**0.445**	**0.445**
Tokyo	0.320	0.363	0.363	**0.371**

Table 3: Mean value of the itineraries. The value of an itinerary is the sum of the POIs recommendations.

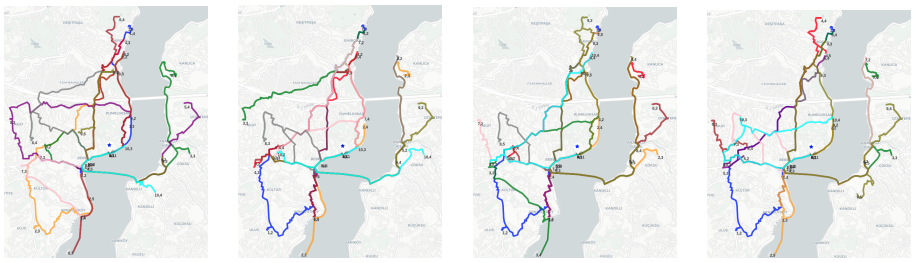
	Greedy	MCTS_Context	MCTS_Score	MCTS_Simple
Istanbul	0.008	**0.009**	**0.009**	**0.009**
Izmir	0.005	**0.007**	**0.007**	**0.007**
Mexico City	0.024	0.024	0.023	**0.026**
Tokyo	0.044	**0.054**	0.048	0.048

Table 4: Mean precision of the *Top 10* itineraries recommendation. In bold are the highest values for each city.

POI as the first check-in of the group sequence in our testing set, the starting POI is excluded from the evaluations. We experimented with λ values of 1, 2, 3, and 4. The best λ values for each city are 4 for Istanbul, 3 for Izmir, 3 for Mexico City, and 2 for Tokyo.

	Greedy	MCTS_Context	MCTS_Score	MCTS_Simple
Istanbul	**0.011**	**0.011**	**0.011**	0.010
Izmir	**0.013**	**0.013**	0.012	0.012
Mexico City	0.024	0.023	0.023	**0.025**
Tokyo	**0.042**	0.041	0.038	0.037

Table 5: Mean recall of the *Top 10* itineraries recommendation. In bold are the highest values for each city.



(a) Greedy (b) MCST_Simple (c) MCTS_Context (d) MCTS_Score

Fig. 5: Examples of itineraries generated by different algorithms for a group in Istanbul: Greedy, MCTS_Simple, MCTS_Context, and MCTS_Score. Each line represent one of the Top-10 recommended itineraries.

The results of comparing the greedy baseline with the MCTS models are presented in Table 3 for the itinerary values, Table 4 for average precision, and Table 5 for average recall. Figures 5 presents an example of the Top 10 itineraries generated for a group in Istanbul.

5.6 Findings on Experiments

The following findings summarize the experimental results.
Finding 1: LBSNs provide enough clues to determine group preferences for venues and contextual information.
Finding 2: In Table 2 we observe that for most of the cases, recommendation models constructed with group history perform better than the models including the aggregation of individual recommendations. However, aggregating individual recommendations could be useful for cold start groups (i.e., groups without check-ins history).
Finding 3: Predicting the next POI using a fixed category type (i.e., using next category) improves the quality of the recommender systems considerably.
Finding 4: Except for content based recommender, aggregation of individual recommendations under different functions effect the the quality of the recommender systems. Among three aggregation functions, *summation* of individual recommendation scores provide the highest performance.
Finding 5: Collaborative filtering with implicit feedback (under group) gives

the highest performance for three of the four cities, whereas for Tokyo item-to-item based recommender has a better performance. These result hint that collaborative filtering based methods perform well in next POI recommendation, however, there is no single winner, possibly due to the characteristics of the cities.
Finding 6: In itinerary construction, the MCTS methods outperform the greedy baseline in most of the cases in terms of POI scoring, precision and recall.
Finding 7: *MCTS_Context* performed similarly or better than *MCTS_Simple*, and *MCTS_Score*. Hence, the benefit of using context is not very clear and possibly depends on the group behavior characteristics per city. For instance, Tokyo dataset contains longer ground truth group itineraries, which enables MCTS_Context to capture and use context information. On the other hand, in Mexico City data set, the itineraries are much shorter, of length two on average. In such a case, simple reward function performs better in itinerary construction.
Finding 8: Sample constructed itineraries for Istanbul in Figure 5 include high overlap. Hence, it is hard to conclude about the characteristics of the constructed itineraries. On the basis of the mean itinerary scores given in Table 3, the slight differences in itineraries, especially on the leftmost part of the figures, are due to differences in POI selection with different scores.

6 Conclusions and Future Work

We present an approach for solving the top-k context-aware tour recommendations for groups problem by using MCTS and contextual information. Our solution integrates selection of a candidate set of neighbor POIs, a group recommender system, and Monte Carlo Tree Search. Given a starting POI, building top-k itineraries consists of these steps: selecting and ranking a candidate set of next POIs, scoring the POIs by recommender system, and solving the orienteering problem using MCTS under time budget constraint.

In itinerary construction, MCTS performs better than greedy algorithm in terms of POI scoring and accuracy. On the other hand, benefits of using the contextual information in MCTS_Context in comparison to MCTS_Score and MCTS_Simple is hard to measure due data sparsity.

The main limitation of our work is the difficulty of evaluating the top itineraries for groups using the ground-truth dataset. Even though we collected approximately 300 K group check-ins, the number of publicly available sequences of check-ins remain limited, and most of the sequences are of length two.

We envision further work on several research direction. Understanding what makes different recommender systems to perform better for characteristics of the cities is a potential problem to work on. In order to improve the itinerary construction performance, using category type as constraint during itinerary generation, experimenting with different candidate selection strategies, and using more advanced scoring functions that combine contextual information with recommendations can be further investigated. Additionally, studying itinerary recom-

mendations for groups with no check-ins (cold start) is an interesting follow-up research problem.

Acknowledgments

The authors would like to thank Aristides Gionis, and Levente Kocsis for their input to our work. Frederick Ayala-Gómez was supported by the Mexican Postgraduate Scholarship of the Mexican National Council for Science and Technology (CONACYT). This work is partially supported by TUBITAK under the project grant number 117E566, by the Hungarian Government project 2018-1.2.1-NKP-00008: Exploring the Mathematical Foundations of Artificial Intelligence and by the Momentum Grant of the Hungarian Academy of Sciences.

References

1. Amer-Yahia, S., Roy, S.B., Chawlat, A., Das, G., Yu, C.: Group recommendation: Semantics and efficiency. Proceedings of the VLDB Endowment **2**(1), 754–765 (2009)
2. Anagnostopoulos, A., Atassi, R., Becchetti, L., Fazzone, A., Silvestri, F.: Tour recommendation for groups. Data Mining and Knowledge Discovery **31**(5), 1157–1188 (2017)
3. Ayala-Gómez, F., Daróczy, B., Mathioudakis, M., Benczúr, A., Gionis, A.: Where could we go?: Recommendations for groups in location-based social networks. In: Proceedings of the 2017 ACM on Web Science Conference. pp. 93–102. WebSci '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3091478.3091485, http://doi.acm.org/10.1145/3091478.3091485
4. Bird, S., Klein, E., Loper, E.: Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc." (2009)
5. Browne, C.B., Powley, E., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of monte carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in games **4**(1), 1–43 (2012)
6. Castro, J., Barranco, M.J., Rodriguez, R.M., Martinez, L.: Dealing with diversity and novelty in group recommendations using hesitant fuzzy sets. In: Proceedings of the FUZZ-IEEE International Conference on Fuzzy Systems (2017)
7. Chen, G., Wu, S., Zhou, J., Tung, A.K.: Automatic itinerary planning for traveling services. IEEE transactions on knowledge and data engineering **26**(3), 514–527 (2014)
8. De Choudhury, M., Feldman, M., Amer-Yahia, S., Golbandi, N., Lempel, R., Yu, C.: Automatic construction of travel itineraries using social breadcrumbs. In: Proceedings of the 21st ACM conference on Hypertext and hypermedia. pp. 35–44. ACM (2010)
9. Fan, L., Bonomi, L., Shahabi, C., Xiong, L.: Multi-user itinerary planning for optimal group preference. In: Advances in Spatial and Temporal Databases. pp. 3–23 (2017)
10. Gionis, A., Lappas, T., Pelechrinis, K., Terzi, E.: Customized tour recommendations in urban areas. In: Proceedings of the 7th ACM International Conference on Web Search and Data Mining. pp. 313–322 (2014)

11. Goodman, L.A.: Snowball sampling. The annals of mathematical statistics pp. 148–170 (1961)
12. Guo, Z., Tang, C., Niu, W., Fu, Y., Xia, H., Tang, H.: Beyond the aggregation of its members—a novel group recommender system from the perspective of preference distribution. In: Knowledge Science, Engineering and Management. pp. 359–370 (2017)
13. Hall, W., Tiropanis, T.: Web evolution and web science. Computer Networks **56**(18), 3859–3865 (2012)
14. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on. pp. 263–272. Ieee (2008)
15. Kocsis, L., Szepesvári, C., Willemson, J.: Improved monte-carlo search. Univ. Tartu, Estonia, Tech. Rep **1** (2006)
16. Lim, K.H., Chan, J., Karunasekera, S., Leckie, C.: Personalized itinerary recommendation with queuing time awareness. In: Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval (SIGIR17) Google Scholar (2017)
17. Lim, K.H., Chan, J., Leckie, C., Karunasekera, S.: Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency. Knowledge and Information Systems pp. 1–32 (2017)
18. Linden, G., Smith, B., York, J.: Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing **7**(1), 76–80 (2003)
19. Liu, X., Tian, Y., Ye, M., Lee, W.C.: Exploring personal impact for group recommendation. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management. pp. 674–683 (2012)
20. Luhn, H.P.: A statistical approach to mechanized encoding and searching of literary information. IBM Journal of research and development **1**(4), 309–317 (1957)
21. Qian, Y., Lu, Z., Mamoulis, N., Cheung, D.W.: P-lag: Location-aware group recommendation for passive users. In: Advances in Spatial and Temporal Databases. pp. 242–259 (2017)
22. Ricci, F., Rokach, L., Shapira, B.: Recommender Systems Handbook. Springer Publishing Company, Incorporated, 2nd edn. (2015)
23. Roy, S.B., Das, G., Amer-Yahia, S., Yu, C.: Interactive itinerary planning. In: Data Engineering (ICDE), 2011 IEEE 27th International Conference on. pp. 15–26. IEEE (2011)
24. Xiao, L., Min, Z., Yongfeng, Z., Zhaoquan, G., Yiqun, L., Shaoping, M.: Fairness-aware group recommendation with pareto-efficiency. In: Proceedings of the Eleventh ACM Conference on Recommender Systems. pp. 107–115 (2017)
25. Yu, Y., Chen, X.: A survey of point-of-interest recommendation in location-based social networks. In: Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence. vol. 130 (2015)
26. Yuan, Q., Cong, G., Lin, C.Y.: Com: A generative model for group recommendation. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 163–172 (2014)