Improving Personalized Trip Recommendation by Avoiding Crowds

Xiaoting Wang[1], Christopher Leckie[1], Jeffrey Chan[2], Kwan Hui Lim[1],
Tharshan Vaithianathan[3]
[1]Department of Computing and Information Systems, University of Melbourne, Australia
[2]Department of Computer Science and Information Technology, RMIT University, Australia
[3]Chisholm Institute, Australia
{wangx5@student., limk2@student., caleckie@}unimelb.edu.au,
jeffrey.chan@rmit.edu.au, Tharshan.Vaithianathan@chisholm.edu.au

ABSTRACT

There has been a growing interest in recommending trips for tourists using location-based social networks. The challenge of trip recommendation not only lies in searching for relevant points-of-interest (POIs) to form a personalized trip, but also selecting the best time of day to visit the POIs. Popular POIs can be too crowded during peak times, resulting in long queues and delays. In this work, we propose the Personalized Crowd-aware Trip Recommendation (PersCT) algorithm to recommend personalized trips that also avoid the most crowded times of the POIs. We model the problem as an extension of the Orienteering Problem with multiple constraints. We extract user interests by collaborative filtering and we propose an extension of the Ant Colony Optimisation algorithm to merge user interests with POI popularity and crowdedness data to recommend trips. We evaluate our algorithm using foot traffic information obtained from a real-life pedestrian sensor dataset and user travel histories extracted from a Flickr photo dataset. We show that our algorithm out-performs several benchmarks in achieving a balance between conflicting objectives by satisfying user interests while reducing the crowdedness of the trips.

Keywords

Orienteering problem; location-based social network; trip recommendation

1. INTRODUCTION

Location-based social networks (LBSNs) has been a rapidly developing field in the last few years. The volume of data generated by LBSNs allows data miners to extract accurate user information to provide better service in target applications. One application heavily influenced by LBSNs is trip recommendation for tourists. Mobile-based pocket tour guides have been deployed for small scale applications like museum tour guides [2] or large city guides [19]. Despite these successes, trip recommendation is still a non-trivial problem due to the following challenges:

1. Relevant Points-of-Interest (POIs) must be selected from a large collection of POIs. A naive approach is to select the top k most relevant POIs and list the results, as in POI recommendation using Collaborative Filtering (CF) [22][24]. However, organizing a trip from such a list can yield a solution that is far from optimal, as the POIs can be spatially distant and the user might not have enough time to visit all POIs in a single trip.

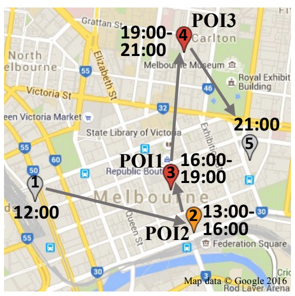
2. Constructing an optimal solution requires all permutations of the POIs to be computed, which is an NP-hard problem. The computational cost will be prohibitive even for a small number of POIs.

3. For POIs of different categories, different peak hours may apply. Visiting a POI during the peak time may result in a long wait time, poor service and sometimes a higher price. Previous studies on temporal POI recommendation have focused on recommending popular times at a POI [23][24] rather than avoiding crowded times.

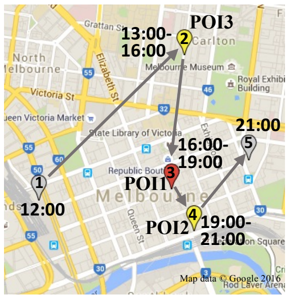
Various trip recommenders have been proposed to recommend personalized trips [11][25]. However, previous studies have failed to consider that some POIs may satisfy a user's interest but can be too crowded at times. Fortunately, pedestrian traffic data from sensor deployments [14] is making it possible to refine tour recommendation based on how crowded places are at different times of the day. In this paper, we propose the personalized crowd-aware trip recommendation (PersCT) framework, which recommends personalized trips that avoid crowded areas to users. To illustrate an example, we show two trips manually planned with three POIs in the city of Melbourne, Australia (Figure 1).

Figure 1(a) and 1(b) show the trips marked on a map and Figure 1(c) shows the normalized pedestrian foot traffic captured by nearby sensors at the POIs (details in Section 5). Colors of the POIs in Figure 1(a) and 1(b) correspond to different crowdedness levels in Figure 1(c). POI1 (David Jones) is a shopping mall and the peak foot traffic is between 12:00 and 17:00. POI2 (Flinders Street) is near a famous laneway, a visitor center and a train station/historical site whose pedestrian volume peaks at 17:00. POI3 (Lygon Street) is a popular Italian dining destination and the peak time occurs after 20:00. Assuming a user leaves the airport coach terminal at location 1 at 12PM and the destination is a hotel at location 5, trip 1 might be a typical trip where the user interest is maximized: the user visits POI2 between 13:00 - 16:00 for sightseeing,

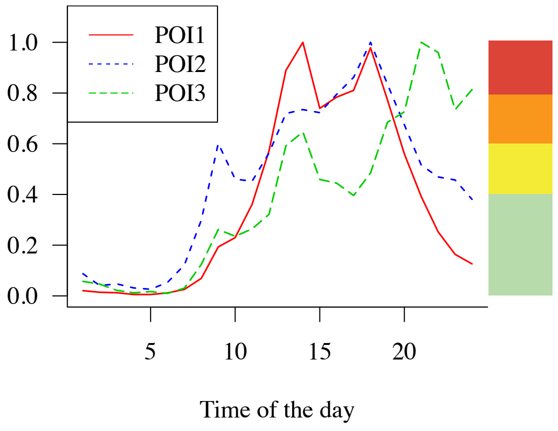
CIKM'16 , October 24-28, 2016, Indianapolis, IN, USA
© 2016 ACM. ISBN 978-1-4503-4073-1/16/10. . . $15.00
DOI: http://dx.doi.org/10.1145/2983323.2983749



(a) Trip 1



(b) Trip 2



(c) Pedestrian Volume

Figure 1: An example of two trips planned manually in the City of Melbourne with 3 POIs. (a) Trip 1: maximizing interest. (b) Trip 2: avoiding crowds. (c) Normalized pedestrian volume from [14]. Legend indicates crowdedness in (a) and (b).

POI1 between 16:00 - 19:00 for shopping and POI3 between 19:00 - 21:00 for dining and relaxation. However, the crowdedness of this trip would be very high as shown in Figure 1(c). In contrast, trip 2 offers a different plan: visit POI3 between 13:00 - 16:00 for lunch and sightseeing, POI1 between 16:00 - 19:00 for shopping and POI1 between 19:00 - 21:00 for dinner and a walk by the Yarra river. By slightly shuffling the order of the visits, crowds can be avoided while the user interest can still be satisfied. (And lunch at popular restaurants is usually cheaper than dinner!)

With only three POIs, it is possible albeit difficult to manually plan a trip that avoids crowds. However, the number of POIs in a city is typically much larger than three, and the search space increases exponentially. Therefore, algorithmic approaches must be efficient to perform trip recommendation. In this work, we define the problem as an extension of the Orienteering Problem (OP) [8], which models each POI with a profit score and finds the trip that collects the maximum total profit subject to certain time constraints. We define the profit to be a time-varying function that combines POI popularity, user interest and crowdedness of the POIs at various times of the day. With this information, we formulate the personalized crowd-aware tour recommendation problem as a multi-objective time-dependent Orienteering Problem. We propose the PersCT algorithm, which is an extension of the Ant Colony Optimisation (ACO) metaheuristic, to solve the problem. Using this algorithm, trips that avoid the crowds while still satisfying user interests can be found efficiently. Our main contributions are:

- We formulate the personalized crowd-aware tour recommendation problem.
- We combine multiple objectives and propose the PersCT algorithm that can efficiently find a solution that achieves a balance between conflicting objectives such as user interest and crowdedness of trips.
- We evaluated the effectiveness of our algorithm using a Flickr photo dataset and pedestrian count dataset, both in the city of Melbourne.

The rest of the paper is organized as follows. Section 2 discusses related work in tour recommendation and planning. Section 3 presents relevant definitions and formally defines the problem. In Section 4 our trip recommendation framework is discussed. In Section 5 we present the evaluation and discussion.

2. RELATED WORK

Suggesting interesting locations to tourists is a popular problem in data mining. We classify previous studies in this field into three categories: (1) POI recommendation, (2) itinerary recommendation, (3) the orienteering problem in operations research.

POI recommendation. In POI recommendation, the problem is to suggest a list of the top k most relevant POIs to a user [22]. Techniques from traditional recommender systems such as user-based collaborative filtering [23], item-based collaborative filtering [21] and matrix factorization [1] have been extensively studied. More recent work also considered recommending the best time of visit [24]. For this category of studies, the POIs are often treated as spatial items and the main goal is to find correlated user-item pairs. This differs from our work and the other two categories (itinerary recommendation and trip planning in operations research) as the ranking of POIs in the list suggests relevance but not the order of visit. Moreover, there is no overall time budget constraint, and travelling time is not considered. In this work, we find relevant POIs as well as ordering the POIs into a trip to satisfy constraints such as maximum allowed trip time.

Itinerary recommendation. Itinerary recommendation is to suggest a sequence of POIs to visit, which is similar to our problem. One of the early state-of-the-art itinerary recommenders was proposed in [3] where the authors used geo-tagged Flickr photos to infer user interest and recommend multi-day itineraries using a recursive greedy algorithm. In [12], user check-in data were mined and an apriori-based algorithm was proposed to find optimal trips under multiple constraints, despite that the computational cost was high. In [7], customized tours in urban areas were recommended by utilizing POI categories and suggesting different types of venues. More recently, [11] proposed time-based user interest and demonstrated advantages over the use of frequency-based popularity measures in tour personalization. In [10], personalized travel sequences in different seasons were recommended by merging textual data and view point information extracted from images. Our work differs from the above since we additionally focus on balancing the crowdedness of a trip with various objectives. To the best of our knowledge, this work is the first that utilizes crowdedness information for trip recommendation, in conjunction with POI popularity and user interests.

Orienteering problem in operations research. We formulated our problem based on the Orienteering Problem (OP) in operations research. The name orienteering problem was initially derived from the sporting game of orienteering [18]. Given the origin and destination, the objective is to traverse a subset of a graph and select paths with the maximum award while still satisfying a time constraint. OP has been proven to be NP-hard [4]. A survey of the tour planning algorithm with an OP formulation can be found in [5]. Since our problem has a time-varying objective function, we only review time-dependent variations of OP. In [20], a fast solution for the time-dependent OP was proposed, although the authors assumed that early starters will also arrive early at their destinations. In [19], a city trip designer was proposed to plan tours that consider the opening and closing times of a POI. In [9], the uncertain wait time problem was addressed using a metaheuristic algorithm to plan tours for theme park visitors. In [6], a time-varying travel cost was modelled and trips were recommended for a group of users. These studies mainly focused on the trade-off between efficiency and optimality of the solution rather than the relevance of the trips to the users. Moreover, the trips are not personalized and the same origin destination pair will result in the same trip for all users.

The most recent work relevant to ours is [25], where the authors proposed a tree-based algorithm to solve the personalized trip recommendation problem. While [25] focused solely on personalization of trips, our focus is on balancing and merging multiple objectives such as crowdedness of the POIs and user interest,

3. PRELIMINARIES

In this section, we give the necessary definitions and formally define the problem.

Definition 1: POI Graph. For a region with W POIs, we construct an undirected complete weighted graph G with W nodes being the POIs, and edge weights $e(i,j)$ being the travel time between two POIs. For simplicity, we assume travel time is symmetric, the mode of transport is walking and the distance to travel between two nodes is a straight line, as in [11][1]. Travel time between two POIs can then be computed by dividing their distance by the speed of walking. For a certain user, a trip $R = <O, P_1, P_2, ..., P_M, D>$ is a non-cyclic path on the graph G where O = origin, D = destination and P_i, $i = 1,...,M$ $M \leq W-2$, are the POIs visited. The time that the user arrives at each POI is $T = <T_O, T_1, ..., T_M, T_D>$. The duration that the user spent visiting each POI is $Du = <Du_O, Du_1, ..., Du_M, Du_D>$. The travel history H of a user $H = <R_1, R_2, ..., R_L>$, where $L \geq 1$.

Definition 2: Time Cost. The time cost $C(R)$ of a trip R is the sum of all travel times and the time spent visiting each of the POIs of a trip:

$$C(R) = \sum_{i=1}^{M-1} \sum_{j=2}^{M} e(R(i),R(j)) + \sum_{k=1}^{M} Du_k,\ j = i+1 \tag{1}$$

where $R(i)$ is the ith POI in trip R.

Definition 3: POI Popularity. For POI i, we calculate the POI popularity $Pop(i)$ by counting its occurrence $Ocr(i)$ in the travel history of all users and normalize the values to [0,1]:

$$Pop(i) = \frac{Ocr(i)}{max(Ocr(j, j \in W))} \tag{2}$$

For example, if there are 3 POIs in total, and $Ocr(1) = 10$, $Ocr(2) = 50$, $Ocr(3) = 20$, then $Pop(1) = 0.2$, $Pop(2) = 1$, $Pop(3) = 0.4$.

Definition 4: User Interest. We define the interest score $Int(u,i)$ of a user u to a POI i as the similarity between POI i and the past visiting history of the user. More on this in Section 4.2.

[1]Other travelling distance functions and modes of transport can also be incorporated into the problem

Definition 5: POI Crowdedness. For POI i, we define the crowdedness at time t as the foot traffic volume U divided by the maximum foot traffic detected during a certain period of time T_p:

$$Crd(i,t) = \frac{U(i,t)}{max(U(i,t \in T_p))} \tag{3}$$

For example, a sensor reported the pedestrian foot traffic at a POI to be <100, 200, 300> at 9:00, 10:00 and 11:00. The maximum pedestrian volume observed for this POI is 500. Therefore the crowdedness during these three hours is <0.2, 0.4, 0.6>.

3.1 Problem Definition

Given a user u with an origin O, a destination D, a start time t and a time budget c, we recommend a trip R such that the following objective function is maximized:

$$max\left(\sum_{i=1}^{W-1} \sum_{j=2}^{W} x(i,j,t) pr(u,j,t)\right) \tag{4}$$

where the profit, $pr(u,j,t)$, is defined as

$$pr(u,j,t) = \frac{(Pop(j) + Int(u,j))^{\gamma}}{Crd(j,t)} \tag{5}$$

The function $x(i,j,t)$ equals to 1 if P_i and P_j are consecutively visited POIs in R, and $x(i,j,t) = 0$ otherwise. The parameter t is the arrival time at POI j. We optimize the objective function such that the following constraints are satisfied: (1) the total time cost is no greater than the time budget, (2) no POIs are visited more than once, and (3) the trip always starts at the origin and finishes at the destination.

Equation 4 is a multi-objective time-varying function that merges POI popularity Pop, user interest Int and POI crowdedness $Crd(i,t)$ information at the time of the visit. As we aim to maximize Pop and Int while minimize Crd, we use a parameter γ to control the bias towards popularity and interest or crowdedness. A higher value of γ will place more weight on selecting relevant POIs and a lower γ will encourage searching for less crowded locations. We show in Section 5 the effect of tuning γ on the behaviour and performance of the proposed algorithm. Based on the above definitions, we present our trip recommendation framework in the following section.

4. TRIP RECOMMENDATION

In this section, we present our trip recommendation framework.

4.1 Overview

There are three main stages to our trip recommendation framework, namely POI popularity modelling, user interest modelling and the PersCT algorithm. Figure 2 shows an overview of our system. POI popularity is given by Equation 2, which defines a normalized popularity score for each POI by counting its occurrence in the travel histories of all past users. The reason for including such information is that when a new user with no past travel histories is using the system, popular locations can still be used to find a solution. This addresses the "cold start" problem frequently faced by user modelling algorithms such as collaborative filtering. The output of Stage 1 is a vector of POI popularities, Pop. The second step is the modelling of user interest. For users who have at least three visited POIs in the past, we infer user interest for unvisited POIs by performing User-Based Collaborative Filtering (UBCF, more on this in the next section). The result of UBCF is a vector of scores Int corresponding to user interest for each POI. The sum of Pop and Int are used as part of the objective function that has to be

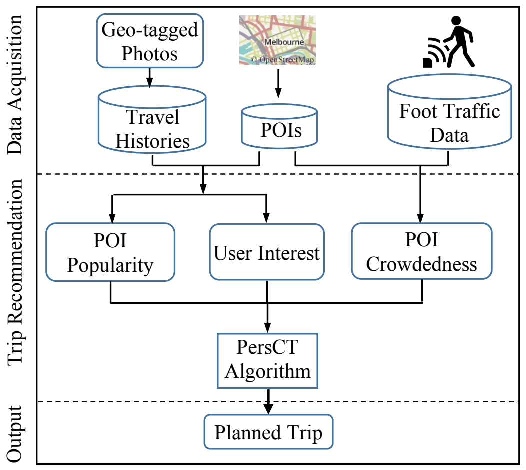


Figure 2: System Overview

maximized (Equation 5). The third stage is to combine foot traffic information with *Pop* and *Int* to obtain the objective function for the multi-objective time-dependent orienteering problem, which is then solve under various constraints as defined in Section 3.1. Since OP is NP-hard, our trip recommendation problem is also NP-hard. In addition, the crowdedness objective is a time-varying function, which is difficult to optimize using exact algorithms like dynamic programming or off-the-shelf optimisation solvers. To this end, we adapt and extend the Ant Colony meta-heuristic to find a solution (Section 4.3 to 4.5).

4.2 Modelling User Interest

We implement a user-based collaborative filtering (UBCF) algorithm to personalize user interest based on previous travel histories due to its simplicity and proven performance in POI recommendation [23]. We create a user-POI matrix by counting the number of times a user has visited a POI. The ratings user u gives to item i is calculated as an aggregation of the ratings given by k most similar users of u:

$$r_{u,i} = aggr(r_{u',i}),\ u' \in U$$

where U is the set of k most similar users. The cosine similarity was used as the similarity measure between user u and u':

$$sim(u,u') = \frac{\sum r_{u,i}\ r_{u',i}}{\sqrt{\sum r_{u,i}^2}\sqrt{\sum r_{u',i}}^2} \tag{6}$$

In addition, we selected the mean function as the aggregation function, and set $k = 15$ (see Section 5). We compute a user-specific interest score $Int(u,i)$ for every unvisited POI of each user. Since this only needs to be computed once, the running time is negligible.

4.3 The Ant Colony Meta-heuristic

In this work, we extend the Ant Colony Optimization Metaheuristic (ACO) [4] to solve the multi-objective time-dependent optimization problem. For completeness, we briefly introduce the ACO algorithm in this section (Algorithm 1). In ACO, software agents, or the ants, search for good solutions to a given optimization problem. The ants are involved in two procedures: (1) heuristic solution construction (2) pheromone trail update. In the solution construction phase, a heuristic search probability function is computed by each ant to find the next destination. In the pheromone

Algorithm 1: Ant Colony Optimization Metaheuristic

```
Data: $G$: a graph
Result: $P$: a path on G
1  begin
2      Initialize pheromone trail over $G$
3      while termination condition not met do
4          Initialize ants
5          //construct solution
6          while $\exists$ ant $\in$ ants not finished do
7              for ant $\in$ ants do
8                  Find next node
9                  // online trail update
10                 Update pheromone trail
11                 Check if ant has finished
12         Update global best ant
13         Perform local search
14         // offline trail update
15         Update pheromone trail with global best ant
16     return path $P$ traversed by the best ant
```

trail update phase, the ants "communicate" with one another by updating a trail matrix TM. The heuristic search function can be as simple as a nearest neighbour search. The trail update involves two parts, online and offline update. In online update, after finding the next destination P_{i+1}, an ant immediately updates $TM(P_i, P_{i+1})$, reinforcing locations with large visiting probabilities. In offline update, only the best ant updates the trail matrix with every path visited in its trip. Local heuristic search can also be used with ACO. After a number of iterations, the trip of the best ant is output as the result.

4.4 PersCT Algorithm

In this section, we discuss the details of our PersCT algorithm in three subsections: heuristic search function, local search and trail update strategy.

4.4.1 Heuristic Search Function

For each ant, the next POI to visit is found using a heuristic probabilistic search function. The original Ant Colony algorithm suggested a search function in the following form:

$$Prob(i,j) = \frac{tr(i,j)^a pr(j)^b}{\sum_i^W \sum_j^W tr(i,j)^a pr(j)^b} \tag{7}$$

where j belongs to the set of all unvisited POIs, $tr(i,j)$ is the trail matrix and $pr(j)$ is the profit function. The denominator normalizes the values into a proper probability score. Using the trail matrix encourages traversing frequently used paths by previous ants that are also close to the current node. The parameters a and b set the preference for using trails versus exploring new nodes. In this work, we incorporate the following into the PersCT framework:

- popularity and user interest objectives
- time-varying crowdedness information
- a distance re-weighting function

Our search probability function is defined as the following:

$$Prob(i,j,t) = \frac{tr(i,j,t)^a pr(u,j,t)^b f_{dist}(i,j)}{\sum_i^W \sum_j^W tr(i,j,t)^a pr(u,j,t)^b f_{dist}(i,j)} \tag{8}$$

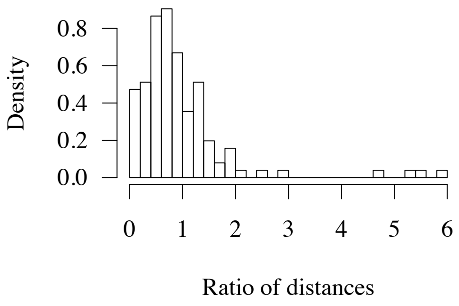


Figure 3: Histogram of the ratio dist(i+1,D)/dist(i,D).

The search function is a product of three terms: (1) trail left by past ants (2) computed profit score of unvisited POIs for this user (3) distance re-weighting. The pheromone trail matrix $tr(i,j,t)$ stores the contributions from ants in the previous iterations. Since we must consider the time-dependence of profit scores, the trail contains a time dimension rather than a purely spatial formulation.

The profit function $pr(u,i,t)$ takes into account (1) POI popularity as defined in Equation 2, (2) user interest as defined in Section 4.2, and (3) time varying crowdedness. We define a distance re-weighting function f_{dist} that re-weights the visiting probability based on the distance to the current POI and the final destination. Let i be the current POI, j the next POI to select from unvisited POIs, and D the final destination, then the probability of visiting j is re-weighted by a factor f_{dist}:

$$f_{dist}(i,j) = \frac{1}{\exp(dist(i,j) + \frac{dist(j,D)}{dist(i,D)})} \tag{9}$$

The search function is re-weighted such that these POIs are preferred: (1) POIs that are close to the last visited POI, (2) POIs that are closer to the final destination than the last visited POI. Rule 1 is based on the first law of geography, which states "everything is related to everything else, but near things are more related than distant things" [17]. Rule 2 is based on the observation that people tend to move towards the final destination and visit POIs on the way rather than move away from it. Figure 3 shows a histogram of the ratio $\frac{dist(i+1,D)}{dist(i,D)}$ for a Flickr dataset in Melbourne, Australia (see Section 5). Approximately 68% of the ratios are less than 1, and the vast majority are less than 1.5, which supports the re-weighting scheme. In Section 5, we show that distance re-weighting is effective in recommending trips.

4.4.2 Local Heuristic Search

We apply a local heuristic search when each iteration finishes and all ants have reached their destination. The best ant is updated and we generate a temporary ant to store the best ant. We randomly swap all pairs of the visited POIs of the best ant. If a better solution is found, then we update the temporary ant with current best ant. Otherwise we undo the previous swap and re-start from the previous POI. Next we randomly swap an unvisited POI with one POI in the trip. This scheme can increase the search space with little computational cost and we can avoid being attracted to a certain local optimum.

4.4.3 Trail Update Strategy

For both online and offline trail update steps, previous trails are first "evaporated", or multiplied by a constant $\rho \in [0,1]$, to limit influence by previous ants [4]. During each iteration, we implement the online trail update rule as the following:

$$trail(i,j,t) = (1-\rho)trail(i,j,t) + \frac{\rho}{W} \tag{10}$$

where W is the number of unvisited nodes in the POI graph.

After all ants have stopped, the global best ant is updated by finding the ant with the maximum objective score (Equation 4), and pheromone trails are also updated using the offline update rule as in [4]:

$$trail(i,j,t) = (1-\rho)trail(i,j,t) + \rho\, pr(bestAnt) \tag{11}$$

where $pr(bestAnt)$ is the profit score of the trip produced by best ant (Equation 4).

4.5 Algorithmic Implementation

In this section, we describe the implementation details of the PersCT algorithm, which consists of the follow three steps.

Step 1: Mapping POI visits. We extract user travel histories by mapping geo-tagged photos to the list of POIs. In particular, we map a photo to a POI if their coordinates differ by <200m based on the Haversine formula [15], which is used for calculating spherical (earth) distances. We calculate the popularity of POIs and the rating scores of a user to each of their visited POIs using user-based collaborative filtering.

Step 2: Calculating crowdedness. Due to practical sensor placement issues, foot traffic sensors might not be placed at the exact POI locations. Therefore, we estimate the foot traffic volume at the location of POIs using a nearest neighbour based estimation method. Specifically, we find the three nearest sensors within 200m of a POI, and compute the mean pedestrian volume weighted by the inverse of their distances to the POI. As defined in Equation 3, we calculate the crowdedness of the POIs at each time instance by dividing the current volume by the recorded maximum volume. For each POI, the maximum pedestrian volume is found by scanning through the data in the whole dataset to normalize between [0,1]. We then average the crowdedness of each POI using its weekly values at the same time. Although in this work we do not implement a real-time system, our framework can be used when real-time data are available.

Step 3: Recommending trips. We compute the travel time matrix as in Equation 1. Using POI popularity, user interest, crowdedness, travel time, start time and maximum trip time as input parameters, the tours are generated by giving the current location and the expected destination of the user to the PersCT algorithm (Algorithms 2 and 3). Lines 3 to 12 in Algorithm 2 are the main loop where the solution space is explored. In each iteration, all ants are re-initialized with the given origin and destination. The set *stopped* contains ants that either have reached their destination or exceeded time limit (Line 4). If at least one ant has not stopped, then it enters the inner while loop to search for the next POI to visit (line 8-10), and Algorithm 3 is called.

Lines 2-5 of Algorithm 3 compute the visiting probability to all unvisited POIs using Equation 8. Line 6 initializes a random floating point number v in the range [0,1] and uses it to set a threshold to determine the next POI *to*. Lines 8-10 compute the cumulative probability score and when the sum is greater than v, *to* is set to the index of the last value. Lines 11-20 update the objective function of the ant and also performs trail update using Equation 5 and Equation 10. If the stopping criteria is met (lines 18 and 20), ant is added to the list of stopped ants and will not search for another POI in this iteration. The function SelectNextPoi then returns the ant.

Line 11 of Algorithm 2 updates the global best ant by finding the maximum objective score. On line 12, offline trail update is

Algorithm 2: PersCT

```
Data: $O$ = origin, $D$ = destination; $T$ = time limit; $Cost$ =
      travel time matrix; $Pop$ = popularity array; $Int$ = user
      interest array; $Crd$ = crowdedness array; $P$ = list of
      POIs
Result: $R$ = Best tour
begin
    $trail \longleftarrow$ matrix of 1.0, $bestAnt \longleftarrow \emptyset$
    while $iteration < maxIterations$ do
        Initialize $ants$ with $< O, D >$, $stopped \longleftarrow \emptyset$
        while $stopped \neq ants$ do
            for $ant \in ants$ do
                if $ant \notin stopped$ then
                    $from \longleftarrow$ last visited POI
                    $depTime \longleftarrow$ departure time at $from$
                    SelectNextPoi($ant$);
        Update $bestAnt$
        Update $trail$ with $bestAnt$ (Equation 11)
    while $iteration < maxIterations$ do
        Swap $bestAnt.i$ with $bestAnt.j$
        Update $bestAnt.obj$ (Equation 4)
        if $bestAnt.obj < tempAnt.obj$ then
            Swap $bestAnt.j$ with $bestAnt.i$
    return $bestAnt.tour$
```

Algorithm 3: SelectNextPoi

```
Data: $ant$; $P$ = list of POIs; $T$ = time limit
Result: $ant$
begin
    $prob \longleftarrow \emptyset$
    for $p \in P$ do
        if $p \in unvisited$ then
            $prob[p] \longleftarrow$ Equation 8
    $v \longleftarrow random \in [0,1]$, $s \longleftarrow 0$
    for $pr \in prob$ do
        $s \longleftarrow s + pr$
        if $s > v$ then
            $to \longleftarrow$ index of $pr$ in $prob$
    if $to \neq ant.Destination$ then
        Add $to$ to $ant.trip$
        $c \longleftarrow$ cost of $ant$ from Equation 1
        if $c < T$ then
            Update $ant.obj$ (Equation 4)
            Update trail with Equation 10
        else
            $ant.stopped \longleftarrow$ TRUE
    else
        $ant.stopped \longleftarrow$ TRUE
    return $ant$
```

performed (Equation 11). Lines 13-17 perform the swapping local search discussed in Section 4.4.2. Lastly, the trip by the best ant is returned.

5. EXPERIMENTAL EVALUATION

In this section, we present the experimental evaluation of our framework.

5.1 Datasets and Pre-processing

The evaluation was performed using datasets collected in Melbourne, Australia (Table 1). A list of 242 POIs in Melbourne was downloaded from [13]. User travel histories were extracted from the Yahoo! Flickr Creative Commons 100M (YFCC100M) dataset [16], which contains 100 million photos and videos taken by real users. Visits within 200 meters of the POIs were kept and the rest were removed. For each user, photos within eight hours were grouped into a tour, producing a total of 3975 tours and 17087 visits.

We extracted pedestrian foot traffic from the Melbourne Open Data Portal [14]. Hourly pedestrian counts at 41 locations in the CBD were captured. Four sensors were found to be frequently non-operational and their data were removed. Missing entries were detected and filled with counts from the previous hour. This accounts for only 0.13% of the total data and does not affect the results.

We combined the Flickr trip data with pedestrian traffic data and performed further filtering to keep trips within 200 meters of the pedestrian sensors. This reduces the dataset to 2586 tours containing 9123 visits to 72 POIs. The pedestrian counts at the POIs were estimated from the mean of the three nearest pedestrian sensors. It should be noted that the counts were street foot traffic, which could partially correlate with the actual number of visitors at the POIs. Due to the challenge of obtaining exact POI visitor counts, we proposed this alternative method of obtaining an estimated POI visitor count. Nevertheless, an estimation of the street foot traffic can still provide travellers with some degree of guidance to infer the actual crowdedness of the venues. Furthermore, this system can be readily deployed in venues where exact counts are available, such as theme parks and museums.

Flickr Photos		
No. Users	No. Trips	No. Visits
911	3975	17087
Pedestrian Foot Traffic Data		
No. Sensors	Data Rate	Period
41	1/hour	1/May/2015-30/Sep/2015
POI data: 242 POIs		

Table 1: A summary of the datasets.

We took a 50-50 split approach for training and testing, i.e. for each user, the first 50% of the trips are used as the training set and the remaining trips as the test set. For each POI, the visiting time was set to 1 hour, as in related previous work [25]. Travel time between two POIs were estimated using their Euclidean distance and a walking speed of 4km/hour, which is also from the literature [11]. Although more accurate travel time estimation models are available, it is not the focus of our study and can be decoupled from our problem.

5.2 Benchmark algorithms

Since our work is the first to consider the crowdedness criterion, we could not find a benchmark algorithm that performs the same task. Nevertheless, we implement six benchmarks that have been reported in the literature [11][25].

Random (RD): This algorithm randomly selects an unvisited POI as the next POI.

5-Nearest Neighbour (5NN): This algorithm finds the five nearest unvisited POIs and chooses the most popular as the next POI.

10-Nearest Neighbour (10NN): This algorithm finds the ten nearest unvisited POIs and chooses the most popular as the next POI. The reason for two nearest neighbour algorithms is to investigate the impact of search range.

Iterative Heuristic Approximation (IHA): This is an iterative heuristic search algorithm proposed in [25] which was adapted to our problem. In each iteration, a trip is found by inserting POIs between the origin and destination. The POI inserted must satisfy all the constraints shown in Section 3.1. POIs are ranked and selected by computing the ratio of squared profit and cost: $s(i,j) = \frac{Prof(j)^2}{cost(i,j)}$, where $Prof = Pop$ (Equation 2) + Int (Section 4.2) and $cost(i,j)$ is the sum of travel time from POI i to j and stay time at j (Equation 1). After each iteration, the trip is recorded and after all iterations have finished, the best trip is selected as the final output.

Greedy algorithm (GD): This algorithm selects the most popular unvisited POI as the next POI.

Integer Programming (IP): This is an integer programming based optimisation algorithm proposed in [11] which finds the optimal solution of an orienteering problem. Since it could not solve the time-dependent orienteering problem, only the popularity information was used.

5.3 Variants of PersCT

In addition to the above benchmarks, we also evaluate four variants of the proposed PersCT algorithm.

Vanilla: This variant purely maximizes POI popularity without considering crowdedness or personalization.

Crowdedness-aware (CR): This variant maximizes a combined objective of POI popularity and crowdedness.

Personalized (CF): This variant uses the user-based collaborative filtering to perform personalization and maximizes a combined objective of POI popularity and user interest.

PersCT (CR+CF): This variant is the proposed algorithm that combines CR and CF.

5.4 Evaluation Metrics

We assume a trip is planned before a user leaves the origin and we evaluate the performance of the algorithms after the user reaches the final destination. Each trip is evaluated independently and the results of all trips are averaged together to produce the final results. The following metrics are computed for each trip:

1. **Precision (Pre)**: The proportion of true positives that are also found in the recommended trip itinerary. If S_r is the set of POIs recommended, and S_a is the set of POIs actually visited, then precision $Pre = \frac{S_r \cap S_a}{S_r}$
2. **Recall (Rec)**: The proportion of true positives that are also found in the actual trip itinerary. If S_r is the set of POIs recommended, and S_a is the set of POIs actually visited, then recall $Rec = \frac{S_r \cap S_a}{S_a}$
3. **F Score (F_1)**: The harmonic mean of Pre and Rec: $F = 2\frac{Pre \times Rec}{Pre + Rec}$
4. **Crowdedness (Crd)**: The mean crowdedness of the POIs at the time of visit in the itinerary.
5. **Popularity (Pop)**: The sum of the popularity of all POIs in the itinerary.
6. **User Interest (Int)**: The sum of user interest of all POIs in the itinerary for a user.

Algorithm	Pre	Rec	F_1	Crd	Pop	Int
RD	0.038	0.059	0.046	**0.401**	1.848	1.920
5NN	0.202	0.259	0.227	0.480	4.134	2.683
10NN	0.194	0.268	0.226	0.525	4.794	2.901
IHA	0.202	0.294	0.239	0.508	4.588	**3.446**
GD	0.220	0.295	0.252	0.533	**5.126**	2.985
PersCT	**0.239**	**0.322**	**0.274**	0.485	4.368	3.111
IP	N/A	N/A	N/A	N/A	N/A	N/A

Table 2: Comparison of the proposed PersCT algorithm and various benchmark algorithms.

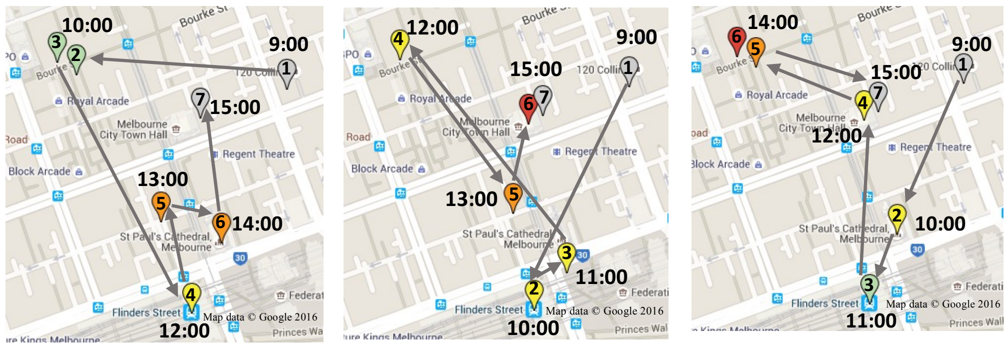
Metrics 1-3 are selected to evaluate how accurately users are modelled using PersCT vs baselines. Metrics 4-6 evaluate the capability of the algorithms to make the best trade-off between finding interesting places for a user and avoiding the most crowded times.

5.5 Results and Discussion

5.5.1 PersCT vs Benchmarks

Table 2 shows a comparison of the PersCT algorithm versus the benchmarks. Given the origin and destination of the trip, PersCT was run 50 times with different random seeds and we report the mean of all trips. The parameter settings are $\gamma = 5$, $\alpha = 2$, $\beta = 1$, $\rho = 0.8$ (for a parameter selection analysis see Section 5.5.5). For each trip, we stopped the execution of an algorithm if the search time was longer than 10 seconds due to real-life usability considerations. As our problem is NP-hard, the integer programming algorithm (IP) [11] failed to find a solution within 10 seconds for all trips tested, and thus we excluded its results. It can be seen that PersCT out-performs all other heuristic algorithms in precision, recall and F score. Interestingly, although GD performs the best in popularity, and IHA wins in user interest, PersCT is the most effective algorithm in finding attractive trips for users. This could be due to the difference in the search strategy. GD only selects the most popular POIs and thus its search space is very limited. IHA searches for POIs that give the highest scores per unit time spent, and thus it also tends to select POIs close-by. In contrast, PersCT explores a much larger solution space due to the use of the trail variable to update trajectories for the ants. Additionally, the distance re-weighting scheme implemented by PersCT also gives a boost in performance, as we show in Section 5.5.2. For other benchmarks, 5NN and 10NN perform slightly worse than IHA, which is expected since the search space is limited to the nearby POIs. Random performs the worst in <Pre, Rec, F1>, which is also not surprising.

In terms of the crowdedness objective, the random method finds the least crowded POIs with the lowest popularity. The second best algorithm is 5NN, and the third best is PersCT. Despite high accuracy in recommendation, PersCT still manages to find trips that are less congested. The greedy algorithm finds trips with the highest popularity, and interestingly they are also the most crowded. 10NN and IHA perform better in crowdedness than Greedy but underperform PersCT in <Pre,Rec,F1>. These results indicate that crowdedness is highly correlated with popularity, and we found the Pearson Correlation score between popularity and crowdedness values of Table 2 to be 0.982. This justifies an intuitive belief that more popular places are more crowded, despite the fact that the data are collected by two completely different systems (pedestrian sensors vs user-uploaded photos). PersCT shows that the crowdedness of trips can be reduced by avoiding visiting most popular locations at their busiest times to maximize user satisfaction.



(a) PersCT (b) Greedy (c) CF (d) Color Legend

Color	Crowdedness Value
🟥	0.8 < Crd ≤ 1
🟧	0.6 < Crd ≤ 0.8
🟨	0.4 < Crd ≤ 0.6
🟩	Crd ≤ 0.4
⬜	None

Figure 4: A case study showing the trips planned using (a) PersCT, (b) Greedy method, (c) PersCT without crowdedness objective (the CF only variant). Crowdedness of the trips: (a) 0.478, (b) 0.563, (c) 0.598.

Since PersCT is a stochastic algorithm while the benchmarks are deterministic, we evaluated the statistical significance of the results. Table 3 shows a summary of the distribution of F_1 scores generated by 50 runs of the algorithm. It can be seen that none of the benchmarks in Table 2 are within the 95% range of the sample mean. Table 4 shows the P-values of the one-sample t-test by using the F_1 scores of the benchmark algorithms with PersCT results. All values are significantly less than 0.05, implying that the improvement in F_1 score of PersCT is statistically significant.

DF	95% LB	Sample Mean	95% UB
49	0.269	0.274	0.281

Table 3: Distributional information of F_1 scores of the proposed algorithm.

	Greedy	Random	5NN	10NN	IHA
P-value	2.1e-10	2.2e-16	2.2e-16	2.2e-16	7.8e-13

Table 4: One sample t-test results of F_1 scores of the benchmarks against the proposed PersCT method.

5.5.2 Variants of PersCT

Table 5 compares different variations of the PersCT algorithm. Vanilla relies solely on popularity to generate itinerary recommendations, and consequently the results are more similar to GD in Table 2. However, Vanilla still out-performs GD by having a higher F_1 score and less crowded trips. When the crowdedness information is combined with Vanilla (the CR variant), less crowded locations are found, despite lower precision and recall values. This is expected since popularity is highly correlated with the crowds. When CF is used with Vanilla, the best precision and recall scores are observed, consolidating our previous observation about personalization. Meanwhile, trip crowdedness is also high for these itineraries, suggesting that the POIs favoured by the users are also popular. Combining crowdedness and personalization, a slight reduction in user interest is observed. However, it is compensated by a large reduction in the level of congestion at the POIs, which can be more desirable for users.

Table 6 compares variants of PersCT that includes or excludes the distance re-weighting term (Equation 9). A significant increase in <Pre, Rec, F1> and a slight reduction in popularity and user interest can been observed when the re-weighting scheme is applied. This could suggest that for some trips, the most popular or interesting POIs are far away from the user. However, real users tend to prefer POIs that are close to the final destination or at least in the same direction of travel. The results have shown the effectiveness of PersCT and the importance of using geo-graphical information in trip recommendation.

Variant	Pre	Rec	F_1	Crd	Pop	Int
Vanilla	0.220	0.306	0.256	0.526	**4.973**	3.018
CR	0.206	0.285	0.240	0.492	4.842	3.010
CF	**0.241**	**0.330**	**0.279**	0.523	4.567	**3.148**
CR + CF	0.239	0.322	0.274	**0.485**	4.368	3.111

Table 5: Comparison of variants of PersCT. Vanilla: No crowdedness or collaborative filtering. CR: only crowdedness. CF: only collaborative filtering. CR+CF: both crowdedness and collaborative filtering.

Variant	Pre	Rec	F_1	Crd	Pop	Int
No f_{dist}	0.209	0.284	0.241	**0.480**	**4.391**	**3.131**
With f_{dist}	**0.239**	**0.322**	**0.274**	**0.485**	4.368	3.111

Table 6: Comparison of variants of PersCT. No f_{dist}: does not use distance re-weighting. With f_{dist}: include distance re-weighting.

5.5.3 Case Study

We illustrate the effectiveness of the proposed PersCT algorithm with a real life case study. In Figure 4, we compare three algorithms, namely PersCT, greedy and the CF variant, to recommend a trip for a user who starts at 9:00 and has a time limit of 7 hours. The colors of the POIs indicate its crowdedness at the time of the visit. The first observation is that most POIs are more crowded in the afternoon than the morning, which is expected. It can be seen that the PersCT algorithm plans the trip such that POIs that are more crowded in the afternoon are visited in the morning instead (POI 2 and 3, which is in a busy shopping region) and POIs less sensitive to temporal foot traffic variations are scheduled in the afternoon (Figure 4(a)). All POIs visited have a crowdedness value of less than 0.8, which is acceptable to most users. For the Greedy

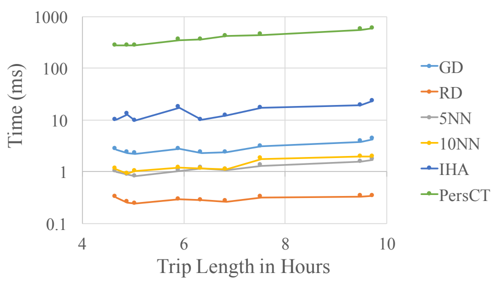


Figure 5: Running time in milliseconds.

method 4(b), since the most popular locations are selected first, the distance travelled for the trip is far from optimized. Moreover, it is not surprising that the crowdedness is high as a consequence of recommending popular POIs first. POI 7, which is very congested in the afternoon, significantly increased the overall crowdedness and it is avoided in the trip recommended by PersCT. A second example, Figure 4(c) shows the resulting trip if only the collaborative filter is used in PersCT. The crowdedness of the trip is even higher than the greedy solution. This is also expected as the algorithm has no information about the crowd profile at each POI.

5.5.4 *Running Time*

We implemented PersCT in Java7 and we performed the evaluation on a 2.7GHz Macbook Pro with 8GB of RAM. We set the number of iterations to be 20 as no significant improvement in performance was observed for higher values. We compare the running time of PersCT with various benchmarks in Figure 5. The running time of all evaluated algorithms increase with the trip length, which is expected due to the growth of the search space. As the time complexity of the Ant Colony meta-heuristic is quadratic in the number of nodes, PersCT is slower than the benchmarks. However, the longest running time recorded was at trip length = 9 hours and it is still less than one second in Java without any code optimisation. Given the improvements of PersCT over the benchmarks in terms of Pre, Rec, F1 and Int, we consider that the small trade-off in running time is negligible. Moreover, this running time is sufficient for most real-life applications.

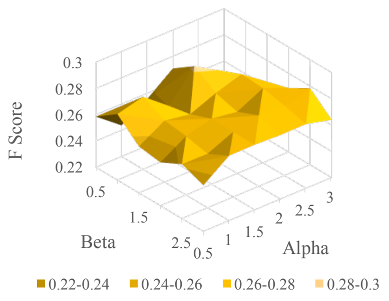
5.5.5 *Parameter Selection*

We performed a grid search for the two important parameters, α and β, of the PersCT algorithm (see Section 4). The results are shown in Figure 6(a). Each experiment was ran 10 times and the results were averaged. The best performance was observed at $\alpha = 2$ and $\beta = 1$. We also evaluated the impact of changing γ, which determines the balance between maximizing the profit function and minimizing crowdedness. A high γ will instruct PersCT to search for POIs with higher popularity and user interest level, whereas a lower value will results in less congested venues to be selected. As can be observed in Figure 6(b), the highest F score can be achieved when γ was set to 5.0.

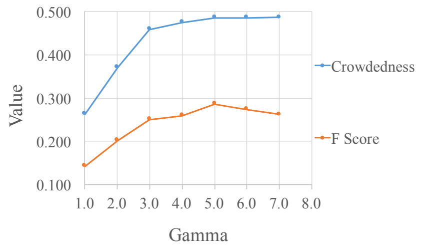
6. CONCLUSIONS AND FUTURE WORK

In this paper, we formulated the personalized crowd-aware trip recommendation problem based on the Orienteering Problem and modelled our problem as a multi-objective time-dependent OP with time-dependent objectives. Given user travel histories and foot traffic information collected by sensors, we proposed the PersCT algorithm, which extends the Ant Colony Optimisation meta-heuristic and extracts POI popularity, user interest and crowdedness information to recommend low congestion trips that also suit the users. We evaluated PersCT using real life datasets from Flickr geo-tagged photos and 6 months of pedestrian traffic data in Melbourne, and we showed that the proposed PersCT algorithm out-performs a number of benchmark algorithms in precision, recall, F_1 score, as well as achieving a good balance in crowdedness.

We acknowledge the following limitations of this study and propose some directions for future work. (1) Due to data availability issues, we were unable to obtain pedestrian count datasets in other cities to further evaluate our algorithm. A potential alternative option is to use pedestrian simulation models and evaluate accordingly. However, some calibration data are still needed in many simulation models. Obtaining a reliable pedestrian model could be the next step. Other data sources such as mobile phone logs could also be explored. (2) We have assumed that the number of users of this system is a small fraction of the overall population and does not affect the pedestrian distribution. In real-life situations where a POI has limited availability, this might not be true. A load balancing scheme may be the next step in this direction.



(a) F_1 vs α and β



(b) F_1 vs γ

Figure 6: Performance of the PersCT algorithms with varying parameter settings. (a) Changing α and β. (b) Changing γ.

7. ACKNOWLEDGEMENTS

National ICT Australia (NICTA/Data61) is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

8. REFERENCES

[1] B. Berjani and T. Strufe. A recommendation system for spots in location-based online social networks. In *Proceedings of the 4th Workshop on Social Network Systems*, page 4. ACM, 2011.

[2] F. Bohnert, I. Zukerman, and J. Laures. Geckommender: Personalised theme and tour recommendations for museums.

In *User Modeling, Adaptation, and Personalization*, pages 26–37. Springer, 2012.

[3] 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.

[4] M. Dorigo and M. Birattari. Ant colony optimization. In *Encyclopedia of Machine Learning*, pages 36–39. Springer, 2010.

[5] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou. A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics*, 20(3):291–328, 2014.

[6] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, and N. Vathis. Heuristics for the time dependent team orienteering problem: Application to tourist route planning. *Computers and Operations Research*, 62:36 – 50, 2015.

[7] 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*, WSDM '14, pages 313–322, New York, NY, USA, 2014. ACM.

[8] B. L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics (NRL)*, 34(3):307–318, 1987.

[9] A. Gunawan, H. C. Lau, and Z. Yuan. *A Mathematical Model and Metaheuristics for Time Dependent Orienteering Problem*. Helmut-Schmidt-Univ., Univ. der Bundeswehr Hamburg, 2014.

[10] S. Jiang, X. Qian, T. Mei, and Y. Fu. Personalized travel sequence recommendation on multi-source big social media. *IEEE Transactions on Big Data*, PP(99):1–1, 2016.

[11] K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera. Personalized tour recommendation based on user interests and points of interest visit durations. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1778–1784. AAAI Press, 2015.

[12] E. H.-C. Lu, C.-Y. Chen, and V. S. Tseng. Personalized trip recommendation with multiple constraints by mining user check-in behaviors. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 209–218, New York, NY, USA, 2012. ACM.

[13] Melbourne. Melbourne landmarks and places of interest, 2015. https://data.melbourne.vic.gov.au/Assets-Infrastructure/Landmarks-and-Places-of-Interest/j5vt-ppat, Accessed: 2015-10-20.

[14] Melbourne. Melbourne pedestrian data, 2015. http://www.pedestrian.melbourne.vic.gov.au, Accessed: 2015-10-20.

[15] R. W. Sinnott. Virtues of the Haversine. *Sky and Telescope*, 68:158, Dec. 1984.

[16] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015.

[17] W. R. Tobler. A computer movie simulating urban growth in the detroit region. *Economic Geography*, 46:234–240, 1970.

[18] T. Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, pages 797–809, 1984.

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

[20] C. Verbeeck, K. SÃűrensen, E.-H. Aghezzaf, and P. Vansteenwegen. A fast solution method for the time-dependent orienteering problem. *European Journal of Operational Research*, 236(2):419 – 432, 2014.

[21] 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.

[22] Y. Yu and X. Chen. A survey of point-of-interest recommendation in location-based social networks. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[23] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann. Time-aware point-of-interest recommendation. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 363–372, New York, NY, USA, 2013. ACM.

[24] Q. Yuan, G. Cong, and A. Sun. Graph-based point-of-interest recommendation with geographical and temporal influences. CIKM '14, pages 659–668, New York, NY, USA, 2014. ACM.

[25] C. Zhang, H. Liang, K. Wang, and J. Sun. Personalized trip recommendation with poi availability and uncertain traveling time. CIKM '15, pages 911–920, New York, NY, USA, 2015. ACM.