Noname manuscript No.
(will be inserted by the editor)

Tour Recommendation for Groups

Aris Anagnostopoulos · Reem Atassi · Luca Becchetti · Adriano Fazzone · Fabrizio Silvestri

Received: date / Accepted: date

Abstract Consider a group of people who are visiting a major touristic city, such as NY, Paris, or Rome. It is reasonable to assume that each member of the group has his or her own interests or preferences about places to visit, which in general may differ from those of other members. Still, people almost always want to hang out together and so the following question naturally arises: *What is the best tour that the group could perform together in the city?* This problem underpins several challenges, ranging from understanding people's expected attitudes towards potential points of interest, to modeling and providing good and viable solutions. Formulating this problem is challenging because of multiple competing objectives. For example, making the entire group as happy as possible in general conflicts with the objective that no member becomes disappointed. In this paper, we address the algorithmic implications of the above problem, by providing various formulations that take into account the overall group as well as the individual satisfaction and the length of the tour. We then study the computational complexity of these formulations, we provide effective and efficient practical algorithms, and, finally, we evaluate them on datasets constructed from real city data.

Keywords Group recommendation · Tour recommendation for groups · Orienteering problem

1 Introduction

Planning an itinerary is one of the most time-consuming travel-preparation activities. For a popular touristic city, the planning process may involve careful examination of

This research was partially supported by the Google Focused Research Award "Algorithms for Large-Scale Data Analysis" and by the EU FET project MULTIPLEX 317532.

Aris Anagnostopoulos, Reem Atassi, Luca Becchetti, Adriano Fazzone
Sapienza University of Rome, Italy
E-mail: {aris, atassi, becchetti, fazzone}@dis.uniroma1.it

Fabrizio Silvestri
ISTI CNR, Pisa, Italy
E-mail: fabrizio.silvestri@gmail.com

tens, if not hundreds, *Points of Interest (PoIs)*, to select those that are most likely to make up a gratifying experience and to figure out the order in which they should be visited. Planning tours is done both by individual tourists as well as professionals, and it has also received a lot of interest lately by the data-mining community (see Section 2).

The goal of this paper is to automatize this process for *groups of people*. The plan, in addition to meeting overall group's tastes, should also ensure that the overall time to perform the tour does not exceed a given time budget constraint B. Especially when we are dealing with major touristic cities, which contain hundreds of PoIs, the interplay between these two aspects naturally forces a selection of the most interesting PoIs to visit, that is, those that are most likely to meet average group's expectations within the given time budget. Many online travel services provide packaged itineraries to their clients. However, such packages suffer from two main drawbacks. First, they are often pre-made and not tailored to one's own interests or they lack flexibility [52]. Second, suggested itineraries may not fit the particular time budget B a group usually has (e.g., a group of young adults can typically walk for 10h overall; more senior visitors may be able to walk for at most half the time). The former problem is magnified when designing itineraries that are intended for a group of tourists who are traveling together. In particular, designing a tour that meets the group members' overall expectations (however we measure this) is far from trivial. In fact, even close friends may not share the same tastes. As a result, every PoI in a given city may have different potential interest to different group members and this disagreement among members must be taken into account when planning a tour. These ingredients make the scenario more complex to model as an optimization problem compared to providing individual tourist itineraries.

Informally speaking, by an optimization perspective the goal is to try and satisfy as many members of the group as possible. The main challenges are how to model this general objective in the first place and how to efficiently compute solutions of good quality in the presence of constraints. To make our ideas clear, consider Figure 1. It depicts the ideal tours of two hypothetical visitors to the city of Rome. If the first visitor were traveling alone, he would prefer the blue route because he is interested in nice fountains. Instead the second visitor would prefer the red route, since she is more interested in churches. Still, they both want to travel together as a group. Thus the green route selects spots that may make *both* members of the group content, even if they are not their favorites, if visitors are singularly considered. As one can observe, the optimal solution for a group can substantially differ from the optimal solutions for individual members. In Section 3 we formalize how we measure the quality of individual and group routes.

Our contribution. In this paper, we formalize the group tour recommendation problem as a generalization of the *orienteering* problem [13]. The first challenge we face is quantifying the satisfaction of a "group of people," which depends on the satisfaction of its members. Thus, we consider several optimization objectives that provide a mathematical formulation of the aforementioned concept.

The problem we consider entails two major technical challenges. The first is designing itineraries with total required time not exceeding a given time budget. This constraint already makes the problem hard to approximate even for the simplest objective functions in the case of a single user [13]; it is the *orienteering* problem, which has been studied by the computer-science and the operations-research communities. The second aspect is the problem of recommending itineraries to groups, whose members

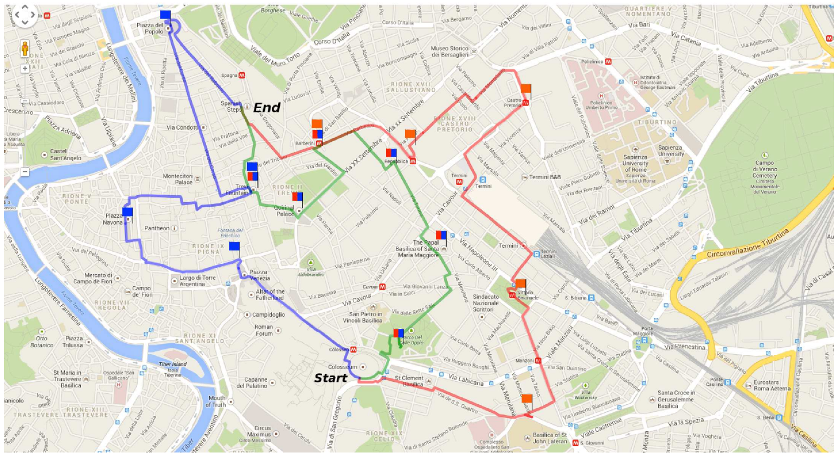


Fig. 1: The ideal tours in Rome, starting at a given spot and finishing in another one for two hypothetical visitors, the first one in blue and the second in red. Instead in green is a tour that keeps both of them satisfied.

may have conflicting preferences. Here, the challenge is how to cast such diversity into an optimization framework that is computationally tractable, at least in practice.

Our formulation and solution approaches are based on these two aspects. We combine ideas from the prior research on the orienteering problem and from previous approaches on recommendation of items to groups of users. As we will see, this combination creates hard problems, for which we design efficient heuristics.

In summary, in this work we make the following contributions: (1) In Section 3 we formalize the problem of tour planning for groups by defining different objective functions to measure the agreement of the group members on the suggested itinerary. (2) In Section 3.3 we prove that one natural version of this problem is NP-hard to approximate within any factor. (3) Given the hardness of the problem, in Section 4 we consider a variety of approaches to solve it: a dynamic-programming approach, some greedy heuristics, as well as some meta-heuristic based approaches. (4) In Section 5, we create three realistic datasets based on real data describing tourist movement within three renowned touristic cities in Italy, (Rome, Florence, and Pisa). (5) Finally, in the same section, we conduct a comprehensive experimental evaluation of the problem formulation and of our algorithmic approaches.

2 Related Work

The general problem of computing itineraries of potential interest to single users or groups has been analyzed through both a data mining and an optimization viewpoint.

Tour optimization. The problem we consider is related to important combinatorial optimization problems on networks. Here we restrict our attention to the contributions that are closest in spirit to the scenario we consider in this paper. The problem of computing a tour over a weighted graph, optimizing some function of the subset of nodes visited under a time or length constraint is the *orienteering problem*; Vansteenwegen and Van Oudheusden [50] describe some features of the mobile touristic guides (MTGs) of the next generation. They discuss the services that MTGs must provide, and they define the tourist trip design (TTD) problem as an extension of the orienteering problem. Souffriau et al. [49] put forward a combined artificial intelligence and meta-heuristic approach to solve the TTD problem. The approach enables fast decision support for tourists on small mobile devices. Wang et al. [51] use a genetic algorithm to solve a generalized version of the orienteering problem. Their goal is to maximize the total path score, given a fixed amount of time. Schilde et al. [47] design heuristic solution techniques for a multi-objective orienteering problem. The motivation stems from the problem of planning individual tourist routes in a city. Each point of interest in a city provides different benefits for different categories (e.g., culture, shopping). Each tourist has different preferences for the different categories when selecting and visiting the points of interests (e.g., museums, churches). Hence, a multi-objective decision situation arises. The authors use a Pareto ant-colony optimization technique and extend the design of the variable neighborhood search method to the multi-objective case to determine all the Pareto-optimal solutions.

Chekuri and Pal [13] proved that the orienteering problem is hard to approximate better than by a logarithmic factor, even for simple objective functions in the presence of time windows. The problem is still APX-hard without time windows, even when the objective is maximizing the sum of the values of the nodes visited by the tour [4, 8].[1]. The latter is essentially our problem when applied to a group consisting of a single user. The approach of [13] was applied to the computation of travel itineraries for single users by De Choudhury [19]. Gionis et al. [23] consider the setting in which each PoI comes with a type and the goal is to maximize the overall level of satisfaction of a user, under budget preference constraints on the types of PoIs the user is willing to visit. Similarly, Brilhante et al. [10, 11] model the recommendation of a personalized tour to a user as a coverage problem (called the *generalized coverage problem*) where the goal is to maximize a measure that depends on the personal interests of the user, subject to time-budget constraints. Roy et al. [5] model itinerary planning as an interactive process where at each step the user has the possibility of giving her feedback on the recommendation given. On the basis of this feedback, their method recommends subsequent PoIs.

The main difference, which makes our problem also more challenging (even to just formalize), is that we are dealing with groups of users and we attempt to find tours that are satisfying for the entire group as opposed to a single user.

Recommendation to groups. Group recommendation has been designed for various domains such as news pages [45], tourism [21], music [17], restaurants [36], and TV programs [53]. There exist two dominant strategies for group recommendations [1, 7]: The first approach creates a pseudo-user representing the group and then makes recommendations to that pseudo-user, whereas the second strategy computes a recommendation list for each group member and then combines these lists to produce a recommendation for the group.

[1] We remind that a problem is APX-hard if there exists a polynomial-time approximation scheme (PTAS) reduction to it from any problem in APX. Practically, it means that unless $P = NP$, there exists a constant c such that it is impossible to design a polynomial-time algorithm that solves the problem with approximation ratio better than c.

As examples on the first approach, Hu et al. [27] design a deep-architecture model built with collective deep belief networks and dual-wing restricted Boltzmann machines, to represent group preference using high-level features that are learned from lower-level features. Yuan et al. [54] propose a probabilistic model to model the generative process of group activities and make group recommendations. They consider the fact that users in a group may have different influences, and that those who are expert in topics relevant to the group's interests are usually more influential. Similarly, Zhang et al. [55] proposed GroupBox: a generative model for group recommendation, which can be applied to both group users and individual users.

The second strategy for group recommendation, a widely adopted approach, is to apply an aggregation function to obtain a consensus group preference for a candidate item. Popular aggregation functions, such as, *least misery* or *sum* are used in existing works [1,28,46]. Ntoutsi et al. [43] pre-cluster the users and, subsequently, generate the individual recommendations for group members using that member's cluster. Finally, they apply a group-aggregation function.

Although not applied yet to group recommendation, there exists a related line of work, which views the group recommendation process as a decision problem. In particular, the field of multiple criteria decision analysis (MCDA) deals with problems that are inherently multidimensional and it aims at supporting the decision-making process by evaluating alternatives that have multiple points of view [31] (see also below for more on multi-criteria optimization). Lakiotaki et al. [31,32] apply MCDA to recommendation systems, attempting to capture the multidimensional nature of user preferences. Although their approach provides recommendations to individual users, it is based on the selection of a recommendation (decision) that is able to satisfy the different criteria of the user preferences. The techniques for aggregating users and analyzing their preference profiles are similar to some of the approaches for group recommendation. In Section 5 we perform some similar aggregation for evaluating our approach.

There is a big difference between all these works in group recommendation and our approach: In all the aforementioned works, the general goal is to compute a list of top-k items, that is, those with maximum relevance to all members in a group. Instead, in our work we provide not a single item but an ensemble of items (a set of PoIs) that must satisfy the group as a package (even if there may be an item that does not satisfy some users) subject to an additional tour constraint.

Predicting PoIs. In the past, one of the first approaches to solve the PoI prediction problem has used a data-mining approach, namely trajectory pattern mining, to extract temporally annotated frequent movement patterns. Trajectory-based models are exploited by Monreale et al. [38] and by Krumm and Horvitz [30] to predict the most interesting locations to a single user. Monreale et al. propose "WhereNext" [38], a data-mining method that is used to predict the next location of a moving object. A similar approach, which uses machine learning instead of pattern mining, is defined in Baraglia et al. [40]. The goal is to predict, in a personalized way, the next PoI that will be visited by the tourist in a given city. The prediction function sought is on a feature space containing, among others, a set of features capturing the historical visits made by the user; these historical features are used for personalization purposes. Likewise, Noulas et al. [41] study the problem of predicting the next venue a mobile user will visit, by exploring the predictive power offered by different aspects of the user behavior. The authors propose a set of 12 features that aim to capture the main factors that may drive users' movements. They model transitions between types of places, mobility flows between venues, and spatio-temporal characteristics of user check-in patterns. Furthermore, they exploit such features in two supervised learning models, based on linear regression and M5 model trees, resulting in higher overall prediction accuracy.

The behaviors of tourists and local citizens wandering around a city are quite different: in the case of touristic attractions it is easy to recommend popular sites, but it is much harder to predict particular and niche attractions that tourists would enjoy.

Other related problems. The problem that we study is an example of a *network-design problem*. There are many variants of network-design problems, a class of them in which our problem falls is, given a graph, to select a subset of edges that optimizes some objective and satisfies some constraint.

One of the most famous examples is the travelling salesman problem (TSP), which can be viewed as a dual version of the orienteering problem. The goal is to select a tour that covers all the nodes and minimizes its length. It is one of the most studied NP-hard problems. When the underlying graph forms a metric space, the problem is APX-hard and the best approximation algorithm designed by Christofides [14] achieves an approximation ratio of $3/2$. A more general version of the problem also has as input two nodes s and t, and the objective is to find a path from s to t that visits each node exactly once (unless $s = t$, in which case s has to be visited twice). For this problem the best approximation algorithm known is by Hoogeveen [26], which achieves approximation ratio of $5/3$. Recently there have been some advancements, improving the approximation ratio for some special cases, for instance for the *graphic TSP* [48]. Much harder is the asymmetric TSP (ATSP). For this the best known approximation algorithm is by Asadpour et al. [3], which gives an approximation ratio of $O(\log n/ \log\log n)$, where n is the number of nodes in the graph.

Lappas et al. [33] introduced the problem of finding a team of experts: Given a network representing people and their social distances, in which people have particular skills, the goal is to select a subset of the nodes that covers each skill and has a small connection distance. This problem can be casted as the group Steiner-tree problem: the nodes belong in one or more groups and the goal is to connect at least one node from each group with a minimal cost. The best algorithm by Chekuri et al. [12] achieves an approximation ratio of $O(\log^2 n \log\log n \log \ell)$, where n is the number of nodes in the network and ℓ the number of groups. Anagnostopoulos et al. [2] extend the work of Lappas et al. [33] in an online setting. To do that they solve a series of bi-criteria problems, in each of which the goal is to cover the skills of incoming jobs while keeping low both the connection cost and the number of solutions in which a node has participated.

The problem that we study requires to address multiple goals: find a tour with high value for the group, maintain fairness, and satisfy budget constraints. There exists a rich literature in the area of *multi-criteria* or *multi-objective optimization*, in which the goal is to model appropriately problems that address all the objectives and design algorithms that solve them. Papadimitriou and Yannakakis [44] designed a very general technique for constructing in polynomial-time an approximate *Pareto curve*, which is the set of solutions for which there does not exist a solution that is strictly better. The number of solutions on the Pareto curve can be exponential in the input, but Papadimitriou and Yannakakis showed that there exists a nearby curve that has polynomial number of solutions, and show how to compute it. There may exist multiple such curves; Bazgan et al. [6] provided a 3-approximation algorithm to computing the approximate curve with the minimum number of solutions for the bi-objective case. Grandoni et al. [24] model multi-criteria problems by treating all but one objectives as constraints and optimize for the last one. This procedure defines a set of *k-budgeted problems* (k is the number of constraints). The versions for $k \geq 2$ are usually significantly harder than $k = 1$. They, nevertheless, present a mechanism for solving these problems in polynomial time but at the cost of violating slightly the budget constraints. There also exists a sequence of more applied works, which applies heuristics for solving multi-criteria combinatorial problems. For instance, Legriel et al. [34], propose an approach for computing an approximate Pareto curve using a search-based methodology that consists in submitting queries to a constraint solver. Coello et al. [15] formulate multi-criteria optimization problems as max–min problems and solve them using genetic algorithms. Czyzżak and Jaszkiewicz [18] take a weighted sum of the different objectives and they try to solve it using simulated annealing.

3 Problem Definition

In this section we define our problem. To do this we first present the orienteering problem, which is the special case of our problem for a single user.

3.1 The Orienteering Problem

In the orienteering problem (we refer henceforth to a tourist scenario), we are given a directed weighted graph $G = (V, E)$ where V is a set of n nodes representing *Points of Interest (PoIs)* in a city (a museum, a church, a restaurant, etc.) and E is a set of m edges connecting PoIs, representing the set of available routes among them. Each node u has an associated waiting time $d_V(u)$, and a distance $d_E(u,v)$ is naturally defined between each pair of PoIs u and v, as their shortest path length on G, thus inducing a metric space. As a consequence, in the remainder, without loss of generality, we consider the *metric completion* of G, that is, we assume an edge between every pair of nodes in G with length equal to the length of the shortest path between them. Note that by considering directed graphs, we can include the waiting time in a weight associated to each (directed) edge. Namely, we define $w(u,v) = d_V(u) + d_E(u,v)$ and we use this as the cost to go from u to v. Furthermore, each PoI has an associated value $p : V \mapsto \mathbb{R}_+$, quantifying the importance (or profit) to a user of visiting that PoI.

A *path* or *tour* is a sequence of PoIs. We are interested in paths T starting and terminating at two (not necessarily distinct) specified PoIs s and t. Given a path $T = (s, v_1, \ldots, v_\ell, t)$ with $v_i \neq v_j$ for $i \neq j$, we define its *value* as the sum of the values of its nodes $\sum_{i=1}^{\ell} p(v_i)$. Note that, although source and destination can be the same, the rest of the nodes have to be distinct, and these are the only ones that contribute to the value of the path. Given a tour $T = \{s, v_1, \ldots, v_\ell, t\}$, we denote by $t(T)$ its tail, namely v_ℓ and by $\text{len}(T) = w(s,v_1) + w(v_1,v_2) + \cdots + w(v_\ell,t)$, its overall length. Finally, we denote by $T \bigoplus \{v\}$ the tour $T' = \{s, v_1, \ldots, v_\ell, v, t\}$.

The goal is, given a time budget $B \in \mathbb{R}_+$ and initial and final PoIs s and t, to find a path $T = (s, v_1, \ldots, v_\ell, t)$, such that the budget constraint is satisfied, that is, $\text{len}(T) \leq B$ and the value of the path is maximized.

3.2 Tour Recommendations for Groups

In this paper we define and study the more general problem in which we have a *group of individuals* performing a tour together, rather than a single person. The goal is to compute a tour that is satisfactory to the group as a whole. When solving this problem, one has to reconcile the different preferences of the group members. Some solutions may well maximize the overall total utility, but they might leave some group members unhappy. Thus, one major challenge is devising objective functions whose optimization results in satisfying tours. To this end we consider various options.

We consider a given group of k members $\{P_1,\dots,P_k\}$. As in the orienteering problem, we are given a graph $G = (V,E)$ with edge weights $w(\cdot,\cdot)$ as above, but now each PoI v_i in general has a different value $p_j(v_i)$ for each member P_j. Put differently, for each node (PoI) v_i we have a vector of values associated with it
$\mathbf{p}(v_i) : V \mapsto {\mathbb{R}_+}^k$, whose jth value is $p_j(v_i)$.

As before, we are interested in paths $T = (s, v_1, \dots, v_\ell, t)$ starting and terminating at two (not necessarily distinct) nodes $s, t \in V$, and we are interested in finding those whose cost does not exceed a given budget B (as before, $v_i \neq v_j$ for $i \neq j$). Each path has some value for each person P_j, which (abusing notation) we define as $p_j(T) = \sum_{i=1}^{\ell} p_j(v_i)$. We will say that the *satisfaction* of user P_j for path T is $p_j(T)$. Optimizing the overall group's satisfaction can be mathematically formalized in several ways. First we define the general problem and subsequently three special cases corresponding to different optimization criteria.

Problem 1 (TourGroup) Given a weighted directed graph $G = (V, E, w)$, two nodes $s, t \in V$, a value $B \in \mathbb{R}_+$, an integer k, for each $v \in V$ a vector $\mathbf{p}(v) \in {\mathbb{R}_+}^k$, and a function $\Phi : \mathbb{R}^k \mapsto \mathbb{R}_+$, compute a path $T = (s, v_1, \dots, v_\ell, t)$ such that $\operatorname{len}(T) \leq B$ and $\Phi(p_1(T), p_2(T), \dots, p_k(T))$ is maximized.

TourGroup describes an entire class of problems, depending on the objective function Φ, with different functions providing different tradeoffs between overall group satisfaction and individual fairness. To study this tradeoff, we consider three specializations of the general problem, corresponding to different definitions of $\Phi(\cdot)$. The first maximizes the sum (or average) of the values:

Problem 2 (TourGroupSum) In this case, we maximize
$\Phi(p_1(T), p_2(T), \dots, p_k(T)) = \sum_{j=1}^{k} p_j(T)$.

Clearly, this formulation optimizes the overall group satisfaction, without taking individual preferences into account. Note that this problem can be reduced to the standard orienteering problem. Let us now define $[k] = \{1, \dots, k\}$. A second approach is to create a max-min formulation:

Problem 3 (TourGroupMin) In this case, we maximize
$\Phi(p_1(T), p_2(T), \dots, p_k(T)) = \min_{j \in [k]} \{p_j(T)\}$.

This formulation represents the other extreme of the spectrum, in which we try to make the least satisfied person as happy as possible. However, this may lead to solutions that provide little value to the entire group, just to make a single person slightly happier. Thus we also consider a smoother formulation (see also Jameson and Smyth [28]).

Problem 4 (TourGroupFair) In this case, we maximize $\Phi(p_1(T), p_2(T), \ldots, p_k(T)) = \text{avg}_{j\in[k]}(p_j(T)) - \alpha \cdot \text{std}_{j\in[k]}(p_j(T))$, for a fixed parameter $\alpha \in \mathbb{R}_+$, a weight that reflects the relative importance of fairness.

Here, we define $\text{avg}_{j\in[k]}(p_j(T))$ and $\text{std}_{j\in[k]}(p_j(T))$ to be the average and standard deviation of the k values $p_j(T)$. The idea behind this last formulation is that we try to optimize overall group satisfaction, however we penalize overly unfair solutions, exhibiting high variance in individual satisfaction. In the rest of the paper we use the terms *Sum*, *Min*, and *Fair* to refer to the corresponding objective functions.

3.3 Hardness of the Problem

The orienteering problem is APX-hard [8] in general and remains APX-hard even when the objective is the sum of the prices collected. In particular, Bansal et al. [4] designed a 3-approximation algorithm when the tour has given start and end points. As a consequence, we expect the problems we consider to be at least as hard (for reasonable objective functions $\Phi(\cdot)$).

Let us look with more detail into the three specific versions of the problem that we consider. TourGroupSum is equivalent to the orienteering problem. TourGroupMin is strictly harder: in Theorem 1 we prove that, unless $P = NP$, no polynomial-time algorithm with a bounded approximation ratio exists. Finally, TourGroupFair is also APX-hard (if we consider a group with a single member, then the problem is reduced to the orienteering problem); we conjecture that it is much harder but we have not been able to show a hardness results similar to TourGroupMin because of the complicated form of the objective function.

Theorem 1 *There does not exist a polynomial-time algorithm with bounded approximation ratio for* TourGroupMin*, unless $P = NP$.*

Proof We prove the theorem via a reduction from the set-cover problem. Consider an instance of set cover, with a universe $U = \{e_1, \ldots, e_k\}$ of k elements and m sets $S^1, \ldots, S^m$, with $S^i \subset 2^U$. Assume that the optimal solution has value ℓ, that is, there exist ℓ sets that together cover all elements in U. We show that if there exists a c-approximation algorithm for TourGroupMin for some $c > 0$, then we can solve the set-cover problem in time that is polynomial in the input size. This is enough to prove the theorem, given that the latter is not possible unless $P = NP$ [22].

The outline of the reduction (and the proof) is easy. Given an instance of set cover, the corresponding instance of TourGroupMin consists of a clique graph with unitary weights on the edges and a number of vertices equal to the number of sets. Members of the group are the elements of the universe U, whereas each set S^i corresponds to a vertex of the clique. The satisfaction vector for each vertex will be an element in $\{0,1\}^k$ and will indicate which elements belong to the corresponding set. That is, if an element of e_j of U belongs to some set S^i, the corresponding member of the group will experience a value 1 of satisfaction when visiting the vertex corresponding to S^i, whereas this value will be 0 if $e_j \notin S^i$. Given this reduction, if we can find a tour of length $\ell + 1$ that has positive value then we can solve the set-cover problem: a positive value for the tour means that each member (element) is covered at least once. We next formalize these arguments.

Given an instance of set cover as defined above, the corresponding TourGroupMin instance has a group of k members. The underlying graph G is a clique with node set $v^0, v^1, \ldots, v^m$ and all pairwise distances equal to 1. Let $\mathbf{p}(v^0) = (0,0,\ldots,0)$ and for $i \in \{1,\ldots,m\}$ let $\mathbf{p}(v^i) = (p_1(v^i), \ldots p_k(v^i))$ with

$$p_j(v^i) = \begin{cases} 1, & \text{if } e_j \in S^i \\ 0, & \text{otherwise.} \end{cases}$$

Let us call the set of vertices $v^1, \ldots, v^m$ *useful vertices*. Let $s = t = v^0$ and assume that we have some budget B. This means that we are searching for a tour that visits at most $B-1$ useful vertices (the first and last vertices must be v^0).

Consider some tour T of length B that visits a set of useful vertices $v^{r_1}, \ldots, v^{r_{B-1}}$.[2] Notice that we have $\Phi(T) > 0$ if and only if for each $j \in \{1,\ldots,k\}$ there exists an $r \in \{r_1,\ldots,r_{B-1}\}$ such that $p_j(v^r) = 1$. If we can find such a tour, then we can use it to obtain a solution to the original set-cover problem that uses $B-1$ sets; this solution is precisely the family of sets $S^{r_1}, \ldots, S^{r_{B-1}}$.

Conversely, assume we can find a set cover that uses $B-1$ sets. Then, this immediately yields a solution to TourGroupMin with positive value of the objective that uses budget B, from the reduction we used.

Now, assume we can solve the TourGroupMin problem with an approximation ratio of c. Then we can try each possible budget $2, \ldots, m+1$. Let B^* be the minimum budget that gives a tour that has positive value. This implies that the optimal solution with budget B^*-1 has value 0 because c is finite, whereas the optimal solution with budget B^* has value at least 1. Summarizing, (1) any tour with positive value using budget B results in a feasible set cover of size $B-1$, (2) if we need budget at least B^* to find a tour with positive value, the optimal tour for B^*-1 must have 0 value and (3) if no positive tour value can be achieved with budget B, then no set-cover of size $B-1$ or less exists. The above facts imply that for $B^*-1 = \ell$ we will be able to recover the optimal set-cover solution of cost ℓ.

3.4 Discussion

Before presenting the algorithms for solving the three specific problems that we have introduced, we would like to discuss about these formulations.

The TourGroup problem can be seen as a combination of the single-user tour-recommendation problem and of the problem of recommending items to groups of users. As such it attempts to trade off various objectives: limited time availability, overall group satisfaction, and individual fairness. Given that our problem recommends tours to groups it faces the challenges of any group-recommendation problem, in particular it has to consider both the total group satisfaction as well at the individual one (fairness). The TourGroupSum is the most simple extension of the single user to the group, and it maximizes the overall satisfaction. On the other extreme, the TourGroupMin problem is another natural formalization and it provides the best guarantees for each individual: Consider an optimal solution T to the TourGroupMin problem. There does not exist any other tour that would make happier every person in the group. In

[2] This is without loss of generality, since if T visits fewer than $B-1$ useful vertices, we can obviously return a shorter tour T' achieving the same value of the objective.

other words, the TourGroupMin problem attempts to find (weakly) *Pareto-optimal* solutions. These two objectives are the ones that have been primarily considered in the literature of item-recommendation to groups; see for instance [1, 46]. The TourGroupSum version is often referred to as optimizing the *average* (optimizing for the sum or the average are equivalent with each other), and the TourGroupMin is often referred to as the *least-misery* objective.

Starting from those two extremes one can consider various alternatives. For instance, we can consider a convex combination of the two objectives. Or we can consider a more *socialist* approach: find the optimal tour for the group that at the same time satisfies each member by at least some amount. The latter introduces an additional constraint to the problem and it makes it even harder to optimize. The approach we followed, captured by the TourGroupFair problem, uses a function described by Jameson and Smyth [28]. It does try to optimize the total satisfaction, but by penalizing high variance in a controllable way (using the parameter α) it leads towards solutions that have high value yet are not too unbalanced, leading to satisfaction of individual members as well. In Section 5 we measure the individual satisfaction for the different objectives.

One could consider different approaches to formulate the problems. For instance, we can compute the best solution for each user and combine them using some aggregation function over paths (and not collection of PoIs as we do now); since, however, paths are likely to be very different, such an approach may fail to find a good trade-off solution. A more general approach along these lines could compute *sets* of good solutions for each of the users and then aggregate the paths by more elaborate methods, for instance, by keeping segments of tours that intersect solutions of multiple users. It is not straightforward how to define such a set of problems, but it may be an interesting direction for future work.

Note that until this point (and in the rest of the paper) we have assumed that the budget captures time constraints. Yet it could be generalized to other types of constraints, for instance monetary constraints (compute a tour that is not very expensive). It only suffices to redefine the distance function. However, such an approach could possibly create very long and impractical tournaments. Thus, what one would desire would be a tournament that satisfies both time and monetary constraints. This is what is known as the *k-budgeted* version of the problem. The k budgeted versions are usually much harder than the simple ones, even without the presence of groups [24].

There are various generalization that one can consider, which have been proposed for the case of tour recommendation for single users: time windows, non-deterministic stay times, and coverage constraints (for instance, one would like to visit a restaurant and an ice cream shop) (e.g., [23, 25]). We can combine such generalizations to our framework and create group versions of such problems. However, the added complexity of the existence of a group is likely to make those problems harder as well. We leave such extensions for future work.

4 Algorithms

As we mentioned earlier, the orienteering problem admits an approximation scheme [13], whose running time becomes prohibitive as the underlying graph grows. Even though TourGroupSum can be reduced to it, this does not hold for the other two variants of the TourGroup problem.

In this section we present various heuristics to solve the three variants of the Tour-Group problem we consider. First, we briefly describe the exhaustive-search approach, which we use as an ideal baseline on small instances of the problem.

4.1 Exhaustive Search (ES)

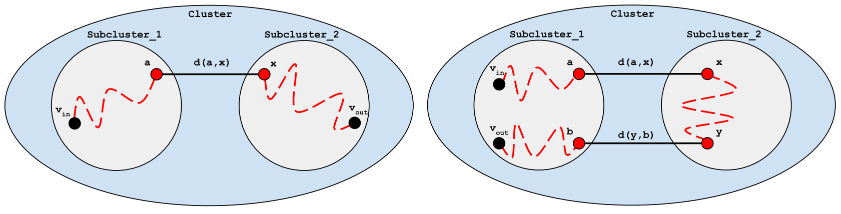
Though generally infeasible for this problem, we have implemented an exhaustive search algorithm as a benchmark, to assess the quality of the solutions computed by the heuristics we propose. ES starts from the initial route connecting the starting and the termination PoIs only. It then iteratively enumerates all possible candidate tours, by adding new PoIs to already computed tours, as long as the new tours meet the given budget constraint.

To reduce computational cost, candidates are pruned when (1) they contain a subtour that already exceeds the budget or (2) there is an alternative tour that traverses the same PoIs in a different order at a lower cost.

4.2 Dynamic Programming Algorithm (DP)

We next present a dynamic-programming heuristic, first describing it for the *Sum* objective function. This algorithm has pseudo-polynomial cost but the order of the polynomial is high and thus we can use it only for small instances. For such instances, in Section 5 we see that DP gives solutions very close to the optimal one. We provide a high-level description in the remainder of these paragraphs. This algorithm first performs hierarchical agglomerative clustering to obtain a dendrogram tree, which is a full binary tree in which each node represents a cluster of PoIs, with the entire city as the root and the single PoIs as the leaves. We have chosen a minimum Euclidean distance between sub-clusters centroids as agglomerative policy for merging two sub-clusters but, of course, other choices are possible. This dendrogram organizes our dynamic program, so that we can increasingly find solutions including more and more PoIs as we perform a post-order visit of the tree, from the leaves to the root. Namely, the idea behind this algorithm is to compose solutions for two sub-clusters (corresponding to two sibling nodes of the dendrogram) into a solution for the super-cluster that corresponds to their parent node. A cell of the DP is indexed by $(V', v_{\text{in}}, v_{\text{out}}, b)$ where $V' \subseteq V$ corresponds to a node of the dendrogram, v_{in},$v_{\text{out}} \in V'$, and it contains our best estimate for the subtour T' that includes PoIs in V', starts at v_{in}, ends at v_{out}, and requires budget at most b. To compute it, we split the tour T' into smaller subtours. Figure 2 shows the two possible scenarios considered by the algorithm, depending on whether v_{in} and v_{out} belong to the same node (i.e., the corresponding subcluster) of V' in the dendrogram; it highlights one of the reasons why the solution returned by this algorithm might be suboptimal: an optimal tour for the super-cluster might traverse the borders between the sub-clusters, multiple times.

Let us see in detail how we compute the value of the DP cell $(V', v_{\text{in}}, v_{\text{out}}, b)$ if it corresponds to the second scenario, with the first one being similar and simpler. Refer to Figure 2. Let V'' be the nodes in `Subcluster_1` of the figure and V''' the nodes in `Subcluster_2`. To compute the solution for cell $(V', v_{\text{in}}, v_{\text{out}}, b)$ the algorithm concatenates the solutions of these three family of cells: $(V'', v_{\text{in}}, a, b_1)$, (V''', x, y, b_2), and $(V'', b, v_{\text{out}}, b_3)$, where V'' and V''' are the two children of V' in the dendrogram



(a) First scenario (b) Second scenario

Fig. 2: Dynamic Programming scenarios.

tree and b_1, b_2 and b_3 are three budget values such that $b_1 + d(a,x) + b_2 + d(y,b) + b_3 = b$. Among all possible sub-solution combinations, given by all possible values of $(a, b, x, y, b_1, b_2, b_3)$, the algorithm selects the one that gives the concatenated route of highest value. According to the *Sum* objective function, the value of the concatenated solution is equal to the sum of the values of the three subcells. The algorithm for the first scenario is similar.

Till now, we have considered the *Sum* objective function. For general objective functions, such as *Min* and *Fair*, we need to store the values of different group members. A slight modification shows that we can do that for k constant, however in practice the algorithm is completely impractical. Unfortunately, even for the *Sum* objective function, although the algorithm is pseudo-polynomial, the running time of DP is extremely large: $O(n^7[B]^4)$, where $[B]$ is the granularity of the budget (i.e., we discretize B into $[B]$ levels). On the other hand, it offers a highly parallelizable alternative, the main reason why we consider this heuristic.

To see why the bound of $O(n^7[B]^4)$ is true, we note that the time complexity of the DP algorithm can be bounded by the number of cells of the dynamic-programming table that must be considered to obtain the final solution multiplied by the number of steps required in the second scenario (the first scenario requires fewer steps than the second one). The number of cells of the dynamic-programming table is $O(n^3[B])$: the dendrogram tree has $O(n)$ nodes, the number of possible pairs of PoIs is $O(n^2)$, and we consider each possible granularity of the budget. For the second scenario the number of steps required is $O(n^4[B]^3)$: there are at most $O(n^4)$ choices for the four nodes `a`, `b`, `x`, `y` (see Figure 2), and we must consider each possible budget assignment (out of the $[B]$ possible budget assignments) for each of the three subtours $\texttt{v}_{\texttt{in}}$-`a`, `x`-`y`, and `b`-$\texttt{v}_{\texttt{out}}$, giving an additional term of $O([B]^3)$.

4.3 Greedy Heuristics

We present several natural greedy heuristics, which both serve as baselines and (as we show in Section 5.2) are significantly faster than the other approaches. In the greedy heuristics that we describe here the system recommends the next PoI only on the basis of the current, partial tour. We first present three basic ones, whose running time is at most $O(n^2)$, making them efficient for our application scenarios. We then propose more sophisticated variants which, however, increase the running time.

To minimize the solution cost, we exploit the properties of the underlying metric space by applying the Hoogeveen approximation algorithm [26] together with our greedy heuristics. The cost of the solution produced by Hoogeveen algorithm is within 5/3 of the optimal minimum cost. As a result of saving budget, our algorithms select additional PoIs.

Best Value (BV). The first greedy heuristic we consider is the `BestValue` algorithm, illustrated in Figure 3. In a nutshell, this algorithm constructs an itinerary connecting s to t incrementally. Assuming T is the currently computed (feasible) tour, it tries to append a new PoI that maximizes the overall benefit to the participants, without violating the budget constraint. The other heuristics are variants of this algorithm and are briefly described in the paragraphs that follow.

```
BestValue(V, s, t)
Require: PoI set V, source s, destination t
T = {s, t}
S = V \ T
while S not empty do
  X' = T
  for all v in S do
    X = T $\bigoplus$ {v}
    if ($\Phi$(X) > $\Phi$(X')) and (len(X) $\leq$ B) then
      v' = v
      X' = X
  if X' = T then
    return T
  T = X'
  S = S \ {v'}
return T
```

Fig. 3: The basic Best Value algorithm.

Best Distance (BD). According to the best-distance heuristic, the selection of the next PoI on the route is the nearest PoI to the current location of the group, regardless to its satisfaction. This is a plain naïve baseline.

Best Ratio (BR). The best ratio heuristic is motivated by the knapsack-like nature of the budget constraint. It appends the PoI v that maximizes the ratio between the overall value of the tour constructed so far including v and the total distance of the tour constructed so far (including v). Of course it considers only PoIs that can be extended to a tour that will not violate the budget constraint.

Best Ratio Plus (BR+). This heuristic follows exactly the same rules of the BR heuristic with the only difference that, when the algorithm cannot improve any more applying only the BR heuristic, it tries to improve the route replacing a PoI in the route with another PoI not in the route (it tries all the possible pairs for PoI to remove–insert and it selects the one that offers the best ratio value of path over total distance of path). It is easy to see that BR+ dominates BR (but it is slower).

Best Ratio Plus Plus (BR++). This heuristic follows exactly the same rules of the BR+ with the only difference that, when the algorithm cannot improve any more applying only the BR+ heuristic, it tries to improve the current route by replacing a PoI in the route with a new PoI and at the same time inserting another new PoI to the route's tail (i.e., between the last node before t and t). It considers all the possible triplets of nodes and it selects the triplet that offers the best ratio of the value of the path over the total distance of the path. It is easy to see that BR++ dominates BR+ (but it is slower).

4.4 Best User Meta-Algorithm (BUMA)

This meta-algorithm solves the tour recommendation problem for groups of users performing these three phases:

– (Recommendation phase): Find (an estimate to) the best tour for each individual user in the group.
– (Evaluation phase): Evaluate the quality of each solution obtained in the previous step for the group: one solution for each group member.
– (Selection phase): Return as solution that one with the highest quality value for the whole group.

The first phase of this meta-algorithm (recommendation phase) can be performed using any algorithm proposed in the paper or, more generally, by any algorithm for the orienteering problem (due to the fact that the recommendation is for a single user and not for an entire group). The evaluation of each single user solution, performed in the second phase (evaluation phase), is done by applying the chosen objective function to each of the k candidate solutions. We have used this meta-algorithm mainly as a benchmark for our methods, since it performs group tour recommendation by solving orienteering problem for each group member (single user tour recommendation).

For the experiments, we have collected data for five versions of this meta-algorithm, which differ in the algorithm used in the first (recommendation) phase. The five f algorithms are: ACO (see the next section), BR, BR+, BR++ and BV, which we use for a group of size 1 (a single user). Finally, from the five variants of BUMA we consider the one that gives the best solution. For each single test instance, we have considered only the BUMA version among all the five versions with the highest solution quality for the particular test instance.

4.5 Ant-Colony–Optimization Algorithms (ACO)

ACO, first proposed by Dorigo and Gambardella [20] as an algorithm for solving the traveling-salesman problem is inspired by the behavior of ants when finding a short path from a food source to their nest by exploiting pheromone information. It is a heuristic local-search approach, which is used for problems with complicated constraints and objective function such as ours. In ACO a set of artificial agents, called ants, cooperate in parallel to find a good solution for the problem by exchanging information via pheromone deposited on graph edges. Ant colony optimization algorithms have been applied to many combinatorial optimization problems, ranging from quadratic assignment to protein folding [9]. They have also been used to solve graph problems similar to ours. Ant-colony optimization was first proposed for the traveling-salesman problem [20] and was shown to perform better than other heuristics. It has subsequently been used to solve vehicle-routing problems [16] as well as the orienteering problem and extensions [29, 37, 39]. It is therefore natural to try it for our setting. As we will see in Section 5 it performs very well compared to the other approaches.

For a path T, let $\mathrm{Val}(T)$ be the value of the solution according to one of the three objective functions that we consider in Section 3.2. Each edge (v,v') will have a *pheromone* value $\tau_{v,v'}$; higher values indicate that such edges have higher chance to be part of the solution. Initially, we set $\tau_{v,v'}$ to the ratio between the value of the shortest feasible route ($T_{min} = (s,t)$) and the weight of the edge itself, namely,

$$\tau_{v,v'} = \tau^0_{v,v'} = \mathrm{Val}(T_{min})/w(v,v').$$

For a parameter h, our algorithm will use h *ants*, which act as agents for generating solutions. We generate h feasible solutions according to the rules explained in the following paragraphs.

Generating a set of feasible solutions. For $j = 1, \dots, h$, the jth ant generates a feasible solution, starting from the tour (s,t) and adding new PoIs into the second-to-last position until no further improvement is possible. In more detail, assuming that for the jth ant the current feasible solution is $T = (s, v_1, \dots, v_i, t)$, this ant generates a new feasible solution $T \bigoplus \{v'\}$ selecting the PoI v' in the following way.

Let $J_j(T)$ be the set of PoIs v not yet visited by the jth ant and able to be reached with the available budget: $B - \mathrm{len}(T) + w(v_i,t)$. Let $\eta(T,v)$ be the local heuristic

$$\eta(T,v) = \frac{\mathrm{Val}(T \bigoplus \{v\})}{\mathrm{len}(T \bigoplus \{v\})}.$$

Finally, recall that $\tau(v_i,v)$ is the amount of pheromones on the edge (v_i,v).

Then, with some fixed probability q_0,

$$v' = \operatorname*{argmax}_{v \in J_j(T)} (\tau(v_i,v))^\alpha \cdot (\eta(T,v))^\beta,$$

for two constants α and β, and with probability $1 - q_0$, v' is selected with probability

$$P_j(T,v') = \frac{(\tau(v_i,v'))^\alpha \cdot (\eta(T,v'))^\beta}{\Sigma_{u \in J_j(T)} (\tau(v_i,u))^\alpha \cdot (\eta(T,u))^\beta}.$$

In words, PoI u has higher chance to be selected as the next PoI if the edge (v_i,u) has a high amount of pheromones (see next paragraph) and if it offers high additional value per cost.

Online pheromone update. To increase the exploration of the search space within each iteration, every time that an ant selects a new PoI, the algorithm decreases the amount of pheromones of the edge selected. In this way, this edge will be less desirable for the following ants. The following formula describes the online pheromone update rule followed by our implementation:

$$\tau_{v_i,v'} \leftarrow (1 - \varphi) \cdot \tau_{v_i,v'} + \varphi \cdot \tau^0_{v_i,v'},$$

for a constant $\varphi \in [0,1]$.

Local optimization. Within each iteration, whenever an ant returns a solution with a value greater than or equal to the current value of the best solution found so far (S_{best}), a local-search optimization is applied to further improve the new solution. In particular, the applied local search is a combination of the BR greedy heuristic (Section 4.3) with the well known tour cost improvement heuristic 2-opt [35].

Offline pheromone update. At the end of each iteration, the best solution (S_{best}) is used to update the amount of pheromones of all the edges according to the following rule (elitist approach). For every edge in the solution S_{best}, the best solution among all solutions found by the h ants at the end of the current iteration, we set

$$\tau_{v,v'} \leftarrow (1-\varphi) \cdot \tau_{v,v'} + \varphi \cdot \text{Val}(S_{best}) \frac{B}{\text{len}(S_{\text{best}})},$$

whereas, for every other pair of edges in the graph we set

$$\tau_{v,v'} \leftarrow (1-\varphi) \cdot \tau_{v,v'}.$$

Parameters setting. The best values for the parameters depend on the problem to be studied. After experimenting, we selected the following values: $\alpha = 1$, $\beta = 3$, $\varphi = 0.1$, $q_0 = 0.2$. We used $h = 200$ ants and the algorithm stops when all the 2,000 best solutions associated to the latest 2,000 iterations are equal to each other. We experimented with values up to 5,000 and we observed experimentally that 2,000 iterations are sufficient to obtain stability on the final solution.

5 Experiments

This section presents our experimental results. Our goal is to address a variety of questions: (1) What is the quality obtained by our algorithms? (2) How do the different objectives compare with each other? (3) What is the price (in terms of satisfaction) that a person pays when he is part of a group? (4) Does optimizing for one member produces high-quality solutions for the others? (5) Does the diversity of the preferences within the group affect the quality of the solution? (6) Are we able to satisfy the group's members reasonably well if the tour only visits famous PoIs?

We start by describing our dataset and then we proceed addressing these questions. Recall that B is the time budget in minutes, k is the group size, and n is the number of PoIs that we consider. When we omit mentioning the value of n, it means we are using all PoIs in the city; see Section 5.1. *Sum*, *Min*, and *Fair* refer to the three objective functions that we consider (see Section 3.2), and ES, DP, ACO, BR, BR+, BR++, BUMA, BV, and BD to the algorithms that we consider. In particular, ES is the exponential-time algorithm that returns the optimal solution, and which we use for comparison, and BV and BD are two very simple greedy approaches, which we consider as baselines. For BUMA we run all five versions (see Section 4.4) and we use the best value for each instance. In each plot, each data point is the average of 500 executions, in each of which we generate a new group. This number of repetitions gives us a high confidence on the results: we performed two-sided t-tests as well as two-sided Wilcoxon tests and obtain p-values less than 10^{-8} whenever the relative difference of the values that we compare is more than 5%.

City	#PoIs	#Users	#Photos
Rome	671	13722	234616
Florence	1022	7049	102888
Pisa	124	1825	18170

Table 1: Datasets description. For each city we show the number of PoIs, the number of tourists, and the total number of photos that these users have uploaded on Flickr and which are used for estimating the trajectories and for creating the user profiles.

All experiments have been conducted on the Microsoft Azure computing platform (standard machine with 16 virtual CPUs, 56 GB memory) and on two servers with 12 virtual CPUs and 2 GB of memory. We have exploited the multi CPU architecture of these machines, by running at the same time different instances of the problem on the same machine. To reduce the running time of a single DP execution we have run a multi-threaded version of the algorithm on the two servers with the 12 virtual CPUs.

5.1 Datasets

To evaluate experimentally our model and our algorithms, we needed a realistic way to create groups of users each of whom has different preferences for different PoIs, based on their type. We created such a set of datasets[3] in the following way. We started by using a set of individual trajectories obtained by Baraglia et al. [40] (created by mining and aggregating information about PoIs from Wikipedia and locations visited by Flickr) in the Italian cities of Pisa, Rome, and Florence. For each PoI we associated a profile vector on various features constructed through a data/text mining approach on aggregated information coming from several sources (e.g. Wikipedia articles) with the process that we explain later. For each user, we also associate a profile vector, which depends on the PoIs that appear in her trajectory.

Eventually, for each user and each PoI, we can obtain a match using the corresponding profile vectors. By selecting randomly from this pool of users we can create groups of different sizes. The distance between PoIs is their Euclidean distance (note though that our algorithms are designed to work with any metric), and for each PoI we also estimated a waiting time based on how much time real users spent on that PoI (we used average time spent, after excluding the 5% longest and shortest stays). To summarize the dimensions of our dataset, the number of PoIs and users in each of the cities are respectively 124 and 1,825 for Pisa, 671 and 13,773 for Rome, and 1,022 and 7,049 for Florence. In some cases (we specify it in the text) we consider smaller sets of PoIs. In these cases we choose PoIs selected independently and uniformly at random among all the PoIs. Table 1 summarizes the dimension of our datasets.

PoI profiling. We want to associate a profile with each PoI such as to enable the evaluation of how much a given PoI matches with a given user. Profiling is a classical technique used in machine learning and data mining and consists in *embedding* an entity (in this case a PoI) in a vector space where each dimension represents a particular feature of the entity. In this work we mined important words out of the text

[3] The datasets are available at
`http://wadam.dis.uniroma1.it/datasets/Tour_Recommendation_for_Groups_Dataset.tgz`.

of Wikipedia pages corresponding to the PoI in consideration, as Wikipedia text contains rich and important information about each PoI [42][4]. We apply latent semantic indexing (LSI) to extract the most important concepts from the Wikipedia pages.

LSI is a text indexing and retrieval method, which is based on the singular value decomposition. It exploits the idea that words that appear in the same documents tend to be semantically related. LSI is able to extract important concepts from the documents and map words and documents into this concept space. It offers several advantages, for instance it addresses the problem of *synonymy*, which is the phenomenon that two or more words express the same concept. For example, the terms *church*, *basilica*, and *cathedral* are equivalent for our purpose and they hindered our profiling task before the employment of LSI, as syntactic-only approaches cannot capture the relationship between those three terms.

In detail these are the steps that we perform:

- For each PoI we extract its Wikipedia page.
- We perform preprocessing, in particular we remove punctuation symbols, hyperlinks, stopwords, words that appear fewer than twice or in more than 95% of the documents in our collection, and we stem each remaining term.
- We represent each page as a vector of tf-idf values and we build a document–term matrix.
- We apply LSI, also tuning for the best number of components using the elbow method; as a result, we use 8 components for Rome, 9 for Pisa, and 10 for Florence.

User profiling. To embed users in a vector space allowing further processing steps, we consider the number of photos taken by a user at each PoI she has visited as an explicit indicator of her interest in that PoI. We exploit the vectors built for each PoI and we represent each user as the average of the vectors of the PoIs that appear on her trajectory, weighted by the number of photos (taken by her) at each PoI.

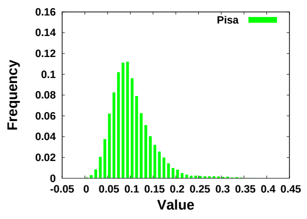
To have informative profiles that are able to give more information about the preferences of the user, we only consider the users with a minimum number of visited PoIs equal to 10, for both Rome and Florence, and equal to 8 for Pisa. The final number of users are 1,872 for Rome, 905 for Florence, and 134 for Pisa.

Score of PoIs. Having profiled each PoI and each user by a vector, we can now estimate the value of a PoI to a user. We do this by taking their dot product. In Figure 4 we see the distribution of the values we obtain for the PoIs among all users for each of the three cities in our datasets.

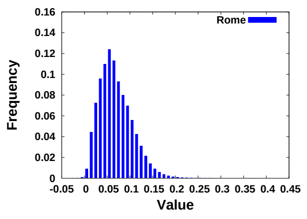
5.2 Quality of Solutions

First we present some results on the performance of the various algorithms presented in Section 4. In this first part, we are interested in assessing the quality of solutions that we obtain by the various heuristics with respect to the best possible. In Figure 5 we

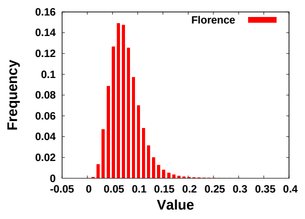
[4] We attempted to also use the Wikipedia categories; however they turned out not to be appropriate for our purpose: they tend to be very specific and they refer mostly to the architectural features or the historical era of construction. For instance, it is common to find two churches belonging to completely different categories, even at higher levels of the Wikipedia ontology.



(a) Pisa.

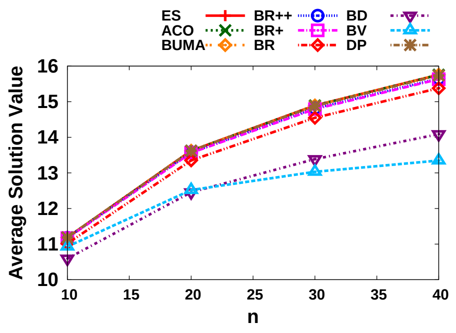


(b) Rome.

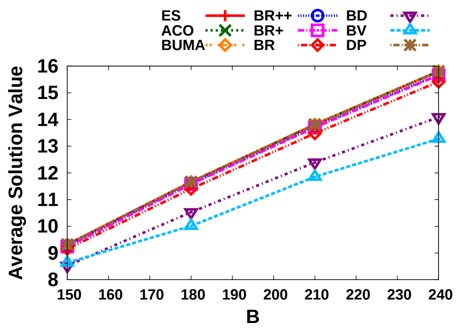


(c) Florence.

Fig. 4: Distribution of the PoIs value among all the users.



(a) $B = 240$



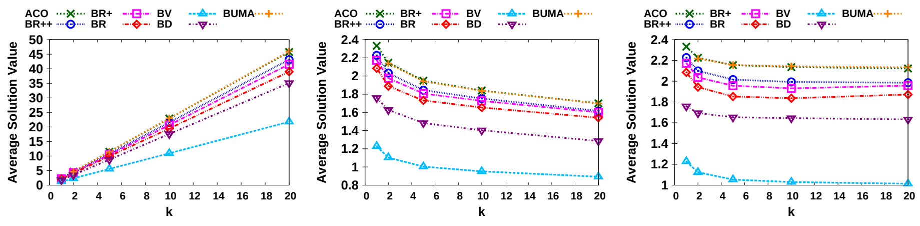
(b) $n = 40$

Fig. 5: Comparison of all the algorithms with the optimal solution in the city of Rome for the *Sum* objective; $k = 20$.

can view how the solution returned by the various algorithms compares with respect to the optimal, for the *Sum* objective function for the city of Rome for groups of size $k = 20$.

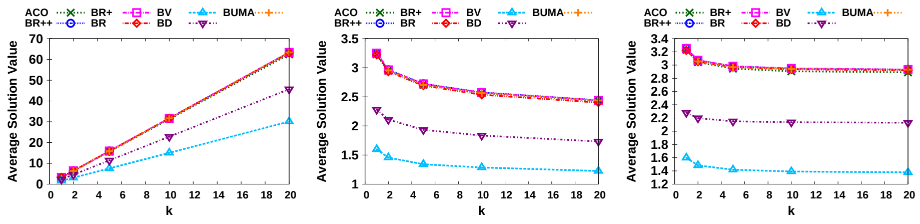
We start by comparing all the algorithms, including the time-intensive ones; thus we use a small number of PoIs. In Figure 5(a) we set the budget to $B = 240$ minutes and we vary the number of PoIs, and in Figure 5(b) we set the number of PoIs to 40 and we vary the budget B. The optimal solution is the one returned by the ES algorithm, which for instances larger than 40 PoIs fails to terminate in a reasonable time. However, the comparison with the optimal solution for these small instances gives an indication of what happens in larger instances as well. Later we observe the values of the algorithms for larger instances without comparing to the optimal solution. We observe that the DP, BUMA, and the ACO algorithms most of the time give the optimal solution. Among the greedy approaches, the BR++ provides very close solutions to the optimal one. As expected, the simple baselines, BV and BD perform notably worse. We obtain similar results for the *Min* and *Fair* objectives (with the exception that we have not implemented the DP because it is completely impractical, as we explain in Section 4.2) and we omit them.

Now we compare the algorithms also for larger instances, omitting ES and DP because they do not complete in a reasonable amount of time. In Figures 6, 7, and 8 we have plotted the values of the solutions returned by different algorithms as we vary the group sizes k when we run them on the entire set of PoIs for each city. We observe



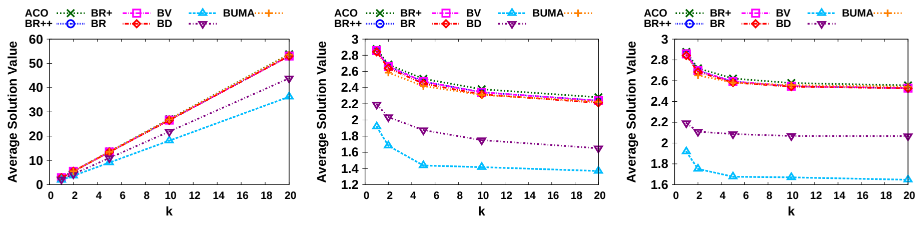
(a) Obj. = *Sum*, $B = 420$ (b) Obj. = *Min*, $B = 420$ (c) Obj. = *Fair*, $B = 420$

Fig. 6: Comparison of the algorithms for the city of Rome.



(a) Obj. = *Sum*, $B = 420$ (b) Obj. = *Min*, $B = 420$ (c) Obj. = *Fair*, $B = 420$

Fig. 7: Comparison of the algorithms for the city of Florence.

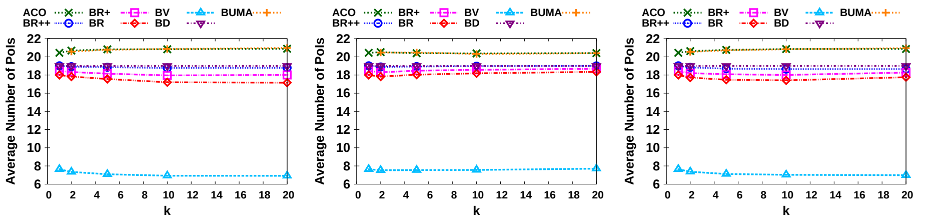


(a) Obj. = *Sum*, $B = 420$ (b) Obj. = *Min*, $B = 420$ (c) Obj. = *Fair*, $B = 420$

Fig. 8: Comparison of the algorithms for the city of Pisa.

the same behavior as previously: ACO and BUMA behave better than the other heuristics; in Florence, also the BR and its variants returns high-quality solutions, whereas in Pisa for the *Min* and *Fair*, the ACO is (marginally) better than BUMA as well. In Figure 9 we report the average number of PoIs returned by the different algorithms for the city of Rome, and we omit the other two because they follow the same trend.

Running time. In Table 2 we show the running times for the various algorithms that are able to terminate fast for the entire city datasets. Thus, ES and DP are missing as they fail to complete for the entire cities. We notice that the best algorithms ACO and BUMA are also the slowest ones. However for most of the cases ACO is significantly faster, thus it is our algorithm of choice.



(a) Obj. = *Sum*, $B = 420$ (b) Obj. = *Min*, $B = 420$ (c) Obj. = *Fair*, $B = 420$

Fig. 9: Average number of PoIs in the solution. Comparison of the algorithms for the city of Rome.

Alg.	Pisa			Rome			Florence		
	Sum	*Min*	*Fair*	*Sum*	*Min*	*Fair*	*Sum*	*Min*	*Fair*
ACO	52.56	75.81	98.51	276.48	410.78	556.09	1318.52	1728.61	2321.65
BUMA	1051.20	1516.20	1970.20	5529.60	8215.60	11121.80	1775.60	2551.40	1454.20
BR++	10.42	9.86	7.73	52.95	41.60	37.80	88.78	127.57	72.71
BR+	4.17	3.38	4.09	11.65	10.83	12.32	26.18	34.32	24.45
BR	1.41	1.26	1.48	3.35	3.54	3.37	12.15	12.50	12.49
BV	0.18	0.22	0.20	0.65	0.61	0.72	1.31	1.58	1.33
BD	3.44			10.19			41.25		

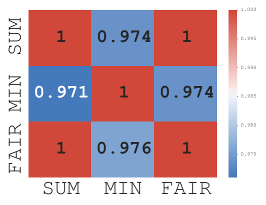
Table 2: Algorithms' execution time (sec); $B = 420$, $k = 20$.

5.3 Comparison of Objective Functions

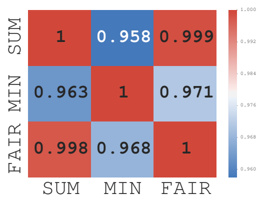
The choice of objective function to optimize depends on what is the tradeoff that we are ready to accept. Optimizing for the sum might make some people very unhappy, or, on the contrary, trying to make every single person as happy as possible might incur a very large penalty to the group as a whole. Thus, here we study the following: We optimize with respect to one objective function and we check the value of the output solution with respect to the other objectives. In Figure 10 we compare the different objectives for all three cities for a budget of seven hours, if we optimize running the ACO algorithm. In each table, each row corresponds to the objective function that we optimize for, and each column corresponds to the objective that we are observing normalizing so that the diagonal is 1. For instance, in the heatmap 10(b) of Florence, the value 0.963 means that if we compare two solutions, the one obtained by the ACO algorithm optimizing for the *Min* objective and the one obtained by the ACO algorithm optimizing for the *Sum* objective, and we observe the value of the *Sum* objective, the former gives a solution that is 0.963 times better (so it is slightly worse) than the latter. We observe that the differences are small, indicating that typical tourists have similar preferences. In the next section we study in more detail the effect of the group diversity on individual satisfaction.

5.4 Solution for Group Versus Solution for Individuals and Effect of Group Size

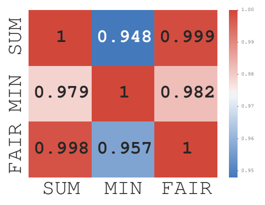
The goal of this section is to measure the tradeoff (sacrifice) that group members make for participating in a group. We start by comparing the various algorithms. In Figure 11



(a) Rome

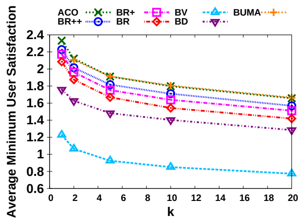


(b) Florence

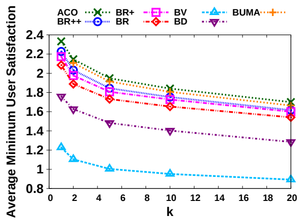


(c) Pisa

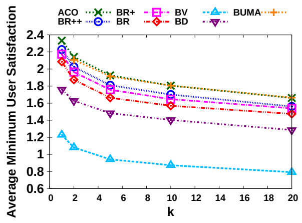
Fig. 10: Heatmap; Algorithm: ACO, $k = 20$, $B = 420$.



(a) Obj. = *Sum*, $B = 420$



(b) Obj. = *Min*, $B = 420$



(c) Obj. = *Fair*, $B = 420$

Fig. 11: The effect of the group size on individual user satisfaction; city: Rome.

we observe for each group the average value (among our 500 runs) of the minimum satisfaction (with respect to the PoIs visited) among each group's members. We depict the satisfaction given by the different algorithms. Again we present the results only for the city of Rome. It is natural that the value decreases as the group size increases: the more the people in the group, the higher the sacrifice that a user will typically have to make.

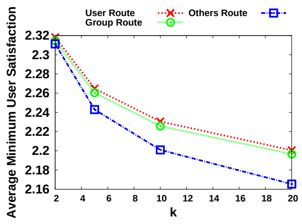
Let us study this phenomenon in more detail. In the following we consider only one algorithm (ACO) and only the city of Rome. We compare three solutions:

1. the best route for the user;
2. the best route for the group;
3. the best route for the others (i.e., we consider all the ordered distinct pairs of users, we optimize for one and we look at the satisfaction of the other, and we take the minimum among all pairs).

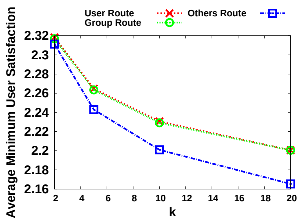
Of course, the satisfaction depends on how diverse the group is. Therefore, we cluster the users and we consider three levels of diversity:

1. groups with all members selected from the same cluster (groups with very similar members);
2. groups of random users;
3. groups with members who are all from different clusters (groups with very diverse members).

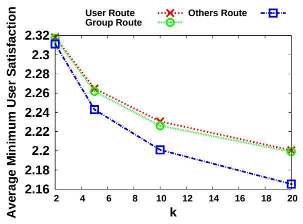
In Figures 12, 13, and 14 we compare the values of the solutions. In particular, we compare the value of the solution computed for the group with the solution computed for a user, which is good for the user that is being optimized for but not for the other members, especially for diverse groups. We observe that the minimum user satisfaction



(a) Obj. = *Sum*, $B = 420$

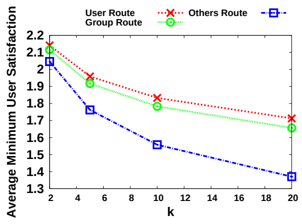


(b) Obj. = *Min*, $B = 420$

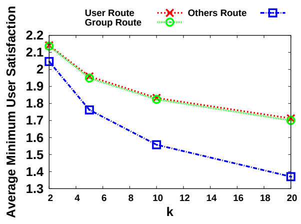


(c) Obj. = *Fair*, $B = 420$

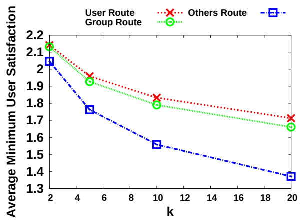
Fig. 12: Solution for Group versus Solution for Individuals, Similar Group Members; city: Rome.



(a) Obj. = *Sum*, $B = 420$

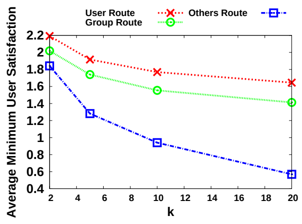


(b) Obj. = *Min*, $B = 420$

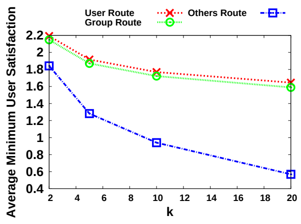


(c) Obj. = *Fair*, $B = 420$

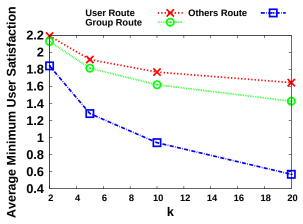
Fig. 13: Solution for Group versus Solution for Individuals, Random Group Members; city: Rome.



(a) Obj. = *Sum*, $B = 420$



(b) Obj. = *Min*, $B = 420$

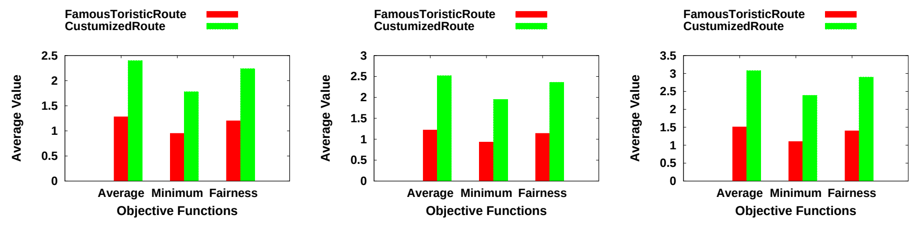


(c) Obj. = *Fair*, $B = 420$

Fig. 14: Solution for Group versus Solution for Individuals, Diverse Group Members; city: Rome.

for our group solution is maintained close to the optimal solution for the user as the group size grows. Instead, if we optimize to the other group members we can obtain a solution that is worse.

Comparing the plots with each other vertically (e.g., Figures 12(a) 13(a) 14(a)) we can see also the effect of the group diversity. We can see that the plots for similar group members (Figure 12) are higher in value than those of random group members (Figure 13), which in turn are higher than those of diverse group members (Figure 14).



(a) Rome, $B = 518$min (b) Florence, $B = 490$min (c) Pisa, $B = 691$min

Fig. 15: Comparison of commercial touristic route with custom route for the group; $k = 20$

5.5 Touristic Route vs. Customized Route

How do our solutions compare to commercial ones? In this section we estimate this difference. Our hypothesis is that commercial tours, being generic, perform significantly worse. Indeed our findings confirm our hypothesis.

We obtained the tours constructed by a commercial service that provides sightseeing tours to tourists through a hop-on hop-off bus. (We considered multiple companies per city and the results we obtained are very similar.)

We first constructed 500 random groups of $k = 5$ members. Using as starting and ending points those of the commercial route and as budget the time required for the commercial route—estimated using our metric $w(\cdot,\cdot)$—we computed our solution for each group. Then, for both our group solution and the commercial one, we compared (1) the average user satisfaction (calculated dividing the *Sum* objective function value by the number of users in the group), (2) the value of satisfaction of the least satisfied user (i.e., the *Min* objective function value), and (3) the *Fair* objective function value. We present the comparison of the commercial with the custom route in Figure 15.

Observe that profiling and optimization pays off significantly, with our solutions being much better from the commercial ones.

5.6 Some Concrete Examples

Finally, we want to show some concrete examples of how optimizing for a group can trade off between user preferences. In Table 3 we depict the tours of user 65 when he forms a group with either member 1531 or member 496. To present the results in a clean and compact way, we have categorized (manually) the PoIs into 5 broad categories. The table shows how many PoIs are in each category for the routes that the users selected and what PoIs our system recommends for the group of two people (we use the *Fair* objective function and the ACO algorithms). Naturally, we can observe how the system attempts to create a balanced route.

In Figure 16 we show the routes of the three tours corresponding to users 65 and 1531. (The routes for the other pair are not well separated pointless to plot.) The red markers in Figure 16 indicate to the PoIs that been visited by user 65; they are museums or galleries, historical monuments, and a natural garden. Instead, the blue line on the map shows twelve PoIs, those visited by user 1531; seven PoIs out of twelve are churches. The route of the group recommended by our algorithm is the green line.

ID	C	H	M	P	S
65		5	1	1	1
1531	7	4			1
Group	5	6	1		2

ID	C	H	M	P	S
65	1	4	1	1	1
496	1	8			3
Group	4	8	2		1

Table 3: The routes of two users and the one for the group (C: Churches, H: Historical monuments, M: Museums/Galleries, P: Parks, S: Squares).

ID	C	H	M	P	S
1042	9	10		3	
451	6	5	3	2	1
505	6	9	1	1	4
1353	4	9	2	2	3
111	7	10	3		3
Group	8	8	3	1	2

ID	C	H	M	P	S
102	5	8	1		4
244	6	5	3	2	1
272	5	11	1	1	2
773	5	2	2		4
1512	3	9	2	1	3
Group	5	9	2	1	2

Table 4: The routes of five users and the one for the group (C: Churches, H: Historical monuments, M: Museums/Galleries, P: Parks, S: Squares).

It is a route of fourteen PoIs, five of them are churches, the rest are museums, historical monuments and squares. The categories of PoIs on the group route and the number of them from each category show that the preferences of both members of the group are considered so as to maximize the users' satisfaction.

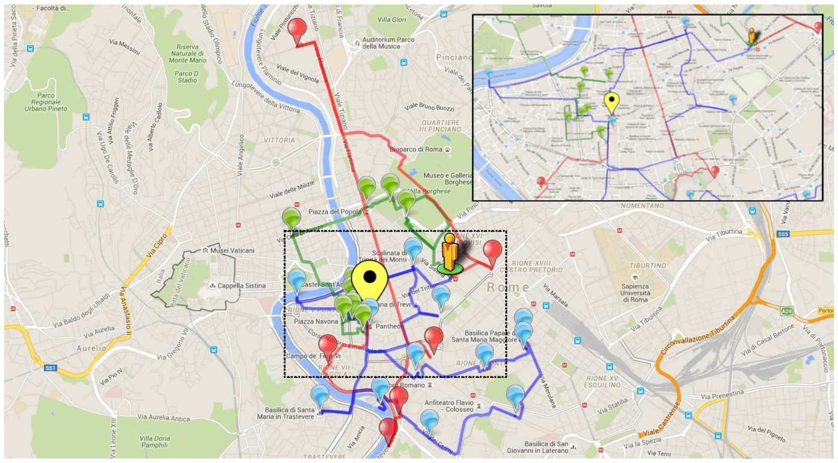

Piazza del Popolo
Musei Vaticani
Cappella Sistina
Piazza Navona
Campo de' Fiori
Pantheon
Basilica Papale di Santa Maria Maggiore
Anfiteatro Flavio - Colosseo
Basilica di Santa Maria in Trastevere
Basilica di San Giovanni in Laterano
Rome
Auditorium Parco della Musica
Bioparco di Roma
Museo e Galleria Borghese
Sapienza Università di Roma

Fig. 16: The routes of two tourists in Rome (red and blue lines), with the green line representing the route for the group that is formed with both of them.

In Table 4 we show our solutions for groups of five tourists. Here as well, we can observe how the recommended tours attempt to satisfy the majority, but are also modified to accommodate for the users who have different tastes.

6 Conclusion

In this work we formulated and formalized a novel computational problem, TourGroup, to automatically build tours for groups of tourists, respecting a given time budget. The problem models a given city as a graph whose edges represent connections between two PoIs, which are, in turn, represented as nodes in the graph. The cost of traversing an edge is the weight of the edge itself. Each node is weighted by a vector of preference scores (representing preferences of each person in the group). Depending on the objective function we presented three different formulations of the problem: TourGroupSum, TourGroupMin, and TourGroupFair. All three problems are NP-Hard even for a single person (being the orienteering problem) and we showed that TourGroupMin is significantly harder to approximate. We gave several heuristics to solve them and we performed extensive experiments to test the algorithms as well as multiple dimensions of our problems. We showed that an ant-colony heuristic seems to always give high-quality solutions with reasonable execution time. We also showed that our approach can provide solutions of much higher satisfaction for the group members compared to fixed ones offered by commercial services. As part of our experiments, we performed an elaborate method for profiling PoIs and users, and we believe that the datasets that we created will be of value for other researchers in the area.

As future work there are some open questions on the theoretical front. As we explained, the TourGroupSum problem is APX-hard even when the underlying space is a metric. What is the complexity when instead the graph is directed, as in our case? Regarding the TourGroupMin problem, in Theorem 1 we proved an unbounded approximation ratio for any polynomial-time algorithm that tries to solve it (assuming that $P \neq NP$). Even though an approximation algorithm does not exist, one may hope for a bi-criteria approximation. What happens if we are allowed to violate the budget constraint? Our proof shows that we cannot obtain an approximation, even if we are allowed to violate the budget constraint by a factor of $o(\log k)$. Is it possible to design an algorithm with finite approximation ratio if we are allowed to violate the constraint by a factor of $\Theta(\log k)$? We conjecture that it is not. Finally, can we extend the hardness result of TourGroupMin to the TourGroupFair problem?

Acknowledgements We thank Fabrizio Grandoni for useful discussions on the problem complexity. We also thank Microsoft for awarding us with credits on their Azure cloud-computing platform, providing us in this way the required infrastructure to run our experiments. Finally, we want to thank the anonymous reviewers, whose comments helped to improve significantly our paper.

References

1. Amer-Yahia, S., Roy, S.B., Chawlat, A., Das, G., Yu, C.: Group recommendation: Semantics and efficiency. Proc. VLDB Endow. **2**(1), 754–765 (2009). DOI 10.14778/1687627.1687713. URL http://dx.doi.org/10.14778/1687627.1687713
2. Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A., Leonardi, S.: Online team formation in social networks. In: Proc. of the 21st International World Wide Web Conference 2012 (WWW 2012), pp. 839–848. ACM Press (2012)
3. Asadpour, A., Goemans, M.X., Mądry, A., Gharan, S.O., Saberi, A.: An O(Log N/ Log Log N)-approximation algorithm for the asymmetric traveling salesman problem. In: Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '10, pp. 379–389. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2010). URL http://dl.acm.org/citation.cfm?id=1873601.1873633

4. Bansal, N., Blum, A., Chawla, S., Meyerson, A.: Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, pp. 166–174. ACM (2004)
5. Basu Roy, S., Das, G., Amer-Yahia, S., Yu, C.: Interactive itinerary planning. In: Proceedings of the 2011 IEEE 27th International Conference on Data Engineering, ICDE '11, pp. 15–26. IEEE Computer Society, Washington, DC, USA (2011). DOI 10.1109/ICDE.2011.5767920. URL `http://dx.doi.org/10.1109/ICDE.2011.5767920`
6. Bazgan, C., Jamain, F., Vanderpooten, D.: Approximate pareto sets of minimal size for multi-objective optimization problems. Operations Research Letters **43**(1), 1 – 6 (2015). DOI http://dx.doi.org/10.1016/j.orl.2014.10.003. URL `http://www.sciencedirect.com/science/article/pii/S0167637714001412`
7. Berkovsky, S., Freyne, J.: Group-based recipe recommendations: analysis of data aggregation strategies. In: Proceedings of the fourth ACM conference on Recommender systems, pp. 111–118. ACM (2010)
8. Blum, A., Chawla, S., Karger, D.R., Lane, T., Meyerson, A., Minkoff, M.: Approximation algorithms for orienteering and discounted-reward TSP. SIAM J. Comput. **37**(2), 653–670 (2007). DOI 10.1137/050645464. URL `http://dx.doi.org/10.1137/050645464`
9. Blum, C.: Ant colony optimization: Introduction and recent trends. Physics of Life reviews **2**(4), 353–373 (2005)
10. Brilhante, I., Macedo, J.A., Nardini, F.M., Perego, R., Renso, C.: Where shall we go today?: Planning touristic tours with TripBuilder. In: Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management, CIKM '13, pp. 757–762. ACM, New York, NY, USA (2013). DOI 10.1145/2505515.2505643. URL `http://doi.acm.org/10.1145/2505515.2505643`
11. Brilhante, I.R., Macedo, J.A., Nardini, F.M., Perego, R., Renso, C.: On planning sightseeing tours with TripBuilder. Information Processing & Management **51**(2), 1–15 (2015)
12. Chekuri, C., Even, G., Kortsarz, G.: A greedy approximation algorithm for the group steiner problem. Discrete Appl. Math. **154**(1), 15–34 (2006). DOI 10.1016/j.dam.2005.07.010. URL `http://dx.doi.org/10.1016/j.dam.2005.07.010`
13. Chekuri, C., Pal, M.: A recursive greedy algorithm for walks in directed graphs. In: Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on, pp. 245–253. IEEE (2005)
14. Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University (1976)
15. Coello, C.A.C.: Two new approaches to multiobjective optimisation using genetic algorithms. In: Adaptive Computing in Design and Manufacture, pp. 151–160. Springer (1998)
16. Coltorti, D., Rizzoli, A.E.: Ant colony optimization for real-world vehicle routing problems. SIGEVOlution **2**(2), 2–9 (2007). DOI 10.1145/1329465.1329466. URL `http://doi.acm.org/10.1145/1329465.1329466`
17. Crossen, A., Budzik, J., Hammond, K.J.: Flytrap: Intelligent group music recommendation. In: Proceedings of the 7th International Conference on Intelligent User Interfaces, IUI '02, pp. 184–185. ACM, New York, NY, USA (2002). DOI 10.1145/502716.502748. URL `http://doi.acm.org/10.1145/502716.502748`
18. Czyzżak, P., Jaszkiewicz, A.: Pareto simulated annealing–a metaheuristic technique for multiple-objective combinatorial optimization. Journal of Multi-Criteria Decision Analysis **7**(1), 34–47 (1998)
19. 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, HT 2010, pp. 35–44. ACM, New York, NY, USA (2010). DOI 10.1145/1810617.1810626. URL `http://doi.acm.org/10.1145/1810617.1810626`
20. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. Evolutionary Computation, IEEE Transactions on **1**(1), 53–66 (1997)
21. Garcia, I., Sebastia, L., Onaindia, E.: On the design of individual and group recommender systems for tourism. Expert systems with applications **38**(6), 7683–7692 (2011)
22. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1979)
23. 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. ACM (2014)

24. Grandoni, F., Ravi, R., Singh, M., Zenklusen, R.: New approaches to multi-objective optimization. Math. Program. **146**(1-2), 525–554 (2014). DOI 10.1007/s10107-013-0703-7. URL `http://dx.doi.org/10.1007/s10107-013-0703-7`
25. Gupta, A., Krishnaswamy, R., Nagarajan, V., Ravi, R.: Approximation algorithms for stochastic orienteering. In: Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '12, pp. 1522–1538. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2012). URL `http://dl.acm.org/citation.cfm?id=2095116.2095237`
26. Hoogeveen, J.A.: Analysis of Christofides' heuristic: Some paths are more difficult than cycles. Oper. Res. Lett. **10**(5), 291–295 (1991). DOI 10.1016/0167-6377(91)90016-I. URL `http://dx.doi.org/10.1016/0167-6377(91)90016-I`
27. Hu, L., Cao, J., Xu, G., Cao, L., Gu, Z., Cao, W.: Deep modeling of group preferences for group-based recommendation. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI'14, pp. 1861–1867. AAAI Press (2014). URL `http://dl.acm.org/citation.cfm?id=2892753.2892811`
28. Jameson, A., Smyth, B.: Recommendation to groups. In: P. Brusilovsky, A. Kobsa, W. Nejdl (eds.) The Adaptive Web: Methods and Strategies of Web Personalization, pp. 596–627. Springer, Berlin (2007)
29. Ke, L., Archetti, C., Feng, Z.: Ants can solve the team orienteering problem. Comput. Ind. Eng. **54**(3), 648–665 (2008). DOI 10.1016/j.cie.2007.10.001. URL `http://dx.doi.org/10.1016/j.cie.2007.10.001`
30. Krumm, J., Horvitz, E.: Predestination: Inferring destinations from partial trajectories. In: Proceedings of the 8th International Conference on Ubiquitous Computing, UbiComp'06, pp. 243–260. Springer-Verlag, Berlin, Heidelberg (2006). DOI 10.1007/11853565_15. URL `http://dx.doi.org/10.1007/11853565_15`
31. Lakiotaki, K., Matsatsinis, N.F., Tsoukias, A.: Multicriteria user modeling in recommender systems. IEEE Intelligent Systems **26**(2), 64–76 (2011)
32. Lakiotaki, K., Tsafarakis, S., Matsatsinis, N.: UTA-Rec: a recommender system based on multiple criteria analysis. In: Proceedings of the 2008 ACM conference on Recommender systems, pp. 219–226. ACM (2008)
33. Lappas, T., Liu, K., Terzi, E.: Finding a team of experts in social networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, pp. 467–476. ACM, New York, NY, USA (2009). DOI 10.1145/1557019.1557074. URL `http://doi.acm.org/10.1145/1557019.1557074`
34. Legriel, J., Le Guernic, C., Cotton, S., Maler, O.: Approximating the pareto front of multi-criteria optimization problems. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, pp. 69–83. Springer (2010)
35. Lin, S.: Computer solutions of the traveling salesman problem. Bell System Technical Journal **44**(10), 2245–2269 (1965). DOI 10.1002/j.1538-7305.1965.tb04146.x. URL `http://dx.doi.org/10.1002/j.1538-7305.1965.tb04146.x`
36. McCarthy, J.F.: Pocket restaurant finder: A situated recommender systems for groups. In: Proceeding of Workshop on Mobile Ad-Hoc Communication at the 2002 ACM Conference on Human Factors in Computer Systems (2002)
37. Mocholí, J., Jaén, J., Canós, J.H., et al.: A grid ant colony algorithm for the orienteering problem. In: Evolutionary Computation, 2005. The 2005 IEEE Congress on, vol. 1, pp. 942–949. IEEE (2005)
38. Monreale, A., Pinelli, F., Trasarti, R., Giannotti, F.: WhereNext: A location predictor on trajectory pattern mining. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, pp. 637–646. ACM, New York, NY, USA (2009). DOI 10.1145/1557019.1557091. URL `http://doi.acm.org/10.1145/1557019.1557091`
39. Montemanni, R., Weyland, D., Gambardella, L.: An enhanced ant colony system for the team orienteering problem with time windows. In: Computer Science and Society (ISCCS), 2011 International Symposium on, pp. 381–384. IEEE (2011)
40. Muntean, C.I., Nardini, F.M., Silvestri, F., Baraglia, R.: On learning prediction models for tourists paths. ACM Trans. Intell. Syst. Technol. **7**(1), 8:1–8:34 (2015). DOI 10.1145/2766459. URL `http://doi.acm.org/10.1145/2766459`
41. Noulas, A., Scellato, S., Lathia, N., Mascolo, C.: Mining user mobility features for next place prediction in location-based services. In: Proceedings of the 2012 IEEE 12th International Conference on Data Mining, ICDM '12, pp. 1038–1043. IEEE Computer Society, Washington, DC, USA (2012). DOI 10.1109/ICDM.2012.113. URL `http://dx.doi.org/10.1109/ICDM.2012.113`

42. Nourashrafeddin, S., Milios, E., Arnold, D.V.: An ensemble approach for text document clustering using Wikipedia concepts. In: Proceedings of the 2014 ACM symposium on Document engineering, pp. 107–116. ACM (2014)
43. Ntoutsi, E., Stefanidis, K., Nørvåg, K., Kriegel, H.P.: Fast group recommendations by applying user clustering. In: Proceedings of the 31st International Conference on Conceptual Modeling, ER'12, pp. 126–140. Springer-Verlag, Berlin, Heidelberg (2012). DOI 10.1007/978-3-642-34002-4_10. URL http://dx.doi.org/10.1007/978-3-642-34002-4_10
44. Papadimitriou, C.H., Yannakakis, M.: On the approximability of trade-offs and optimal access of web sources. In: Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on, pp. 86–92 (2000). DOI 10.1109/SFCS.2000.892068
45. Pizzutilo, S., De Carolis, B., Cozzolongo, G., Ambruoso, F.: Group modeling in a public space: methods, techniques, experiences. In: Proceedings of the 5th WSEAS International Conference on Applied Informatics and Communications, pp. 175–180. World Scientific and Engineering Academy and Society (WSEAS) (2005)
46. Roy, S.B., Thirumuruganathan, S., Amer-Yahia, S., Das, G., Yu, C.: Exploiting group recommendation functions for flexible preferences. In: Data Engineering (ICDE), 2014 IEEE 30th International Conference on, pp. 412–423. IEEE (2014)
47. Schilde, M., Doerner, K.F., Hartl, R.F., Kiechle, G.: Metaheuristics for the bi-objective orienteering problem. Swarm Intelligence **3**(3), 179–201 (2009)
48. Sebő, A., Vygen, J.: Shorter tours by nicer ears: 7/5-approximation for the graph-TSP, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. Combinatorica pp. 1–34 (2014). DOI 10.1007/s00493-011-2960-3. URL http://dx.doi.org/10.1007/s00493-011-2960-3
49. Souffriau, W., Vansteenwegen, P., Vertommen, J., Berghe, G.V., Oudheusden, D.V.: A personalized tourist trip design algorithm for mobile tourist guides. Appl. Artif. Intell. **22**(10), 964–985 (2008). DOI 10.1080/08839510802379626. URL http://dx.doi.org/10.1080/08839510802379626
50. Vansteenwegen, P., Van Oudheusden, D.: The mobile tourist guide: an OR opportunity. OR Insight **20**(3), 21–27 (2007)
51. Wang, X., Golden, B.L., Wasil, E.A.: Using a genetic algorithm to solve the generalized orienteering problem. In: The vehicle routing problem: latest advances and new challenges, pp. 263–274. Springer (2008)
52. Xie, M., Lakshmanan, L.V., Wood, P.T.: IPS: an interactive package configuration system for trip planning. Proceedings of the VLDB Endowment **6**(12), 1362–1365 (2013)
53. Yu, Z., Zhou, X., Hao, Y., Gu, J.: TV program recommendation for multiple viewers based on user profile merging. User Modeling and User-Adapted Interaction **16**(1), 63–82 (2006). DOI 10.1007/s11257-006-9005-6. URL http://dx.doi.org/10.1007/s11257-006-9005-6
54. 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. ACM (2014)
55. Zhang, C., Gartrell, M., Minka, T.P., Zaykov, Y., Guiver, J.: GroupBox: A generative model for group recommendation. Tech. Rep. MSR-TR-2015-61, Microsoft Research (2015). URL http://research.microsoft.com/apps/pubs/default.aspx?id=251683