Distributing Tourists Among POIs with an Adaptive Trip Recommendation System

Sara Migliorini, Damiano Carra, and Alberto Belussi

Abstract—Traveling is part of many people leisure activities and an increasing fraction of the economy comes from the tourism. Given a destination, the information about the different attractions, or *points of interest* (POIs), can be found on many sources. Among these attractions, finding the ones that could be of interest for a specific user, who may have different constraints – such as the available time or budget – represents a challenging task. Travel recommendation systems deal with this type of problems. Despite the vast literature on this topic, most of the solution does not take into account the impact of the suggestions on the level of crowding of POIs.
This paper considers the trip planning problem focusing on user balancing among the different POIs. To this aim, we consider the effects of the previous recommendations, as well as estimates based on historical data, while devising a new recommendation. The problem is formulated as a multi-objective optimization problem, and a recommendation engine has been designed and implemented for exploring the solution space in near real-time, through a distributed version of the Simulated Annealing approach. We test our solution using a real dataset of users visiting the POIs of a touristic city, and we show that we are able to provide high quality recommendations, yet maintaining the attractions not overcrowded.

Index Terms—Trip recommendation, tourist balancing, simulated annealing, MapReduce, SpatialHadoop

1 Introduction

City sightseeing is one of the most popular activity for tourists, especially for short-term trips. Not only well-know destinations, such as New York or Paris, but many small cities are selected as popular destinations, for instance, during the weekend, or as a part of a more complex travel. Within a city, it is possible to identify a set of *attractions*, or *Points of Interests* (POIs): they are usually collected in different sources, such as travel guides and websites. The former provide general suggestions about which POIs should be visited based on the available time. The latter, such as Location Based Social Networks (LBSNs) [1], offer tailored suggestions based on other travellers' experiences. In addition, many cities have tourist offices, which provide general information about the city and its POIs. In order to promote the different POIs in a city, and consequently improve the visibility of some minor POIs, sometimes the tourist offices offer a "city pass" that bundles together different attractions. Despite the different sources of infomations, the tourists need to answer to a fundamental question: given the set of available POIs, how can we make the most of them given our current constraints?

The systems that recommend wich POIs should be visited given a set of constraints – e.g., user personal interests, available time, or budget [2] – deal with this type of problem. In practice, they need to solve an optimization problem [3], such as the Traveling Salesman Problem, which is NP-hard. Sometimes, due to the multiple constrains, rather than a single objective function, the recommendation systems have to take into account a multi-objective optimization problem, whose complexity is further increased. Thanks to the efficient heuristics that are able to provide close-to-optimal solutions to such optimization problems, it is possible to formulate very complex queries, where the number of constraints may be very high: for instance, the user previous experiences may influence the POI choice, or the POI opening hours may determine its position in the sequence of POIs to be visited [4]. Despite such complexity, the aim of the trip recommendation system is to *tailor* the suggestions to the specific user.

Limitation of the prior works. There is a vast literature on trip recommendation systems (see Sect. 2 for a review), but the proposed solutions have one aspect in common: they focus their attention on the user needs and viewpoints. The information about the POIs, such as the the opening hours, or an estimate of the busy periods, are "static", i.e., they do not change as the time goes by. The fact that the suggestions have an impact on the status of the POIs is not considered in the recommendation engine. For instance, if the recommendation systems take into account, for a given POI, the average busy hour of the last week, it may try to divert the user to that POI at different hours. Nevertheless, this approach may generate over time different busy hours. This oscillatory dynamics have been observed in routing algorithms that take into accounts the current state of the routes [5]. To avoid such dynamics, the system should estimate the *effect of the trip recommendation on the POIs*. Such a system would dynamically balance the users considering the level of crowding in the POIs.

Proposed approach. In this paper we consider the trip planning problem that takes into account, besides the user preferences and the system constraints, the balancing of users among the different POIs. The recommendation engine needs to consider the prediction of the user presence

• *S. Migliorini, D. Carra and A. Belussi are with the Computer Science Department, University of Verona, Italy.*
E-mail: {name.surname}@univr.it

This work was partially supported by the Italian National Group for Scientific Computation (GNCS-INDAM) and by "Progetto di Eccellenza" of the Computer Science Dept., Univ. of Verona, Italy.
Manuscript received . . . ; revised . . . .

at the POIs. The quality of the prediction determines the quality of the recommendation: the prediction should include historical data, as well as the recommendations made so far by the system itself. There are a number of challenges that need to be faced to design such a system. First, the user requests are usually issued by a mobile application, where the user expects a near real-time response: the solution space, therefore, should be explored in a limited timeframe. Another issue regards the necessity to understand the impact of the estimation error – due to some unpredictable user behavior – on the effectiveness of the balancing process. Finally, in order to increase the effectiveness of the recommendations, the constraints used by the system for comparing possible solution instances should include spatial properties, like for example the total trip distance computed on a network with different traveling modes.

Motivating example. Fig. 1 shows an example taken from a real-world case regarding the city of Verona in northern Italy. The map reports a set of POIs covered by a "city pass", called *VeronaCard*. Each POI is represented by a circle of variable size which indicates an estimation of the current number of visiting tourists. The figure displays also a trip performed by a tourist (the green one), who travels from p_1 = "Arena" to p_2 = "Casa di Giulietta", without immediately visiting it; in fact, she goes over to reach p_3 = "Chiesa di San Fermo Maggiore" and comes back to p_2 at the end. The POI p_2 is visited after p_3 due to the number of tourists currently on p_2, producing a trip which is not optimal w.r.t. to the travelled distance. Namely, if the tourist is able to know in advance the level of crowding in each POI, she can plan a better trip (e.g., the blue one).

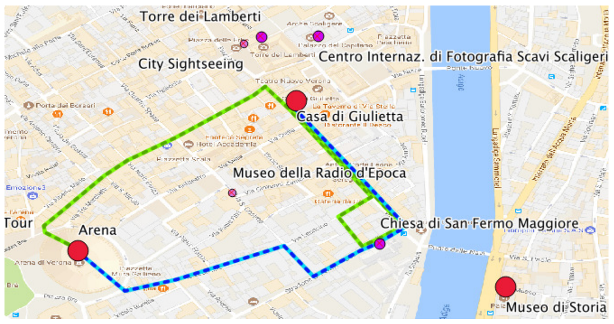

Torre dei Lamberti
City Sightseeing
Centro Internaz. di Fotografia Scavi Scaligeri
Casa di Giulietta
Museo della Radio d'Epoca
Tour
Arena
Chiesa di San Fermo Maggiore
Museo di Storia

Fig. 1. Example of trip performed by using the "city pass" VeronaCard and influenced by the number of tourists visiting the POIs. The level of crowding is represented by the circle size. The green line is the real trip, and the blue line is a recommended trip that avoids the crowded POI.

Key contributions. The contributions of our work are the following: (1) we formulate the online optimization problem, where we consider the current estimation of the user visiting the different POIs as part of the input of the recommendation system. (2) We design and implement an efficient recommendation engine that works in near real-time. The solution is based on a parallel version of the Simulated Annealing approach, using the MapReduce programming framework. (3) We evaluate the trip recommendation system with a dataset collected from the tourist information office of the city of Verona.

This article extends the previous conference version [6] in several respects: (i) we introduce additional optimization criteria that take care also of spatial aspects of the problem, (ii) we provide a detailed description of the implemented framework (algorithms used in the offline analysis, the exploration of the solution space, and the profile updates), and (iii) we enriched the experimental section considering the impact on the POIs of the user variable behavior (i.e., how closely the users follow the recommendations).

The remainder of this paper is organized as follows. In Sect. 2 we discuss the related works. Sect. 3 is devoted to the problem formulation. In Sect. 4 we present the system, the algorithms for trip recommendation, and their implementation in MapReduce. Finally, in Sect. 5 we discuss the experimental results and we conclude the paper in Sect. 6.

2 Related Work

This section reviews the related works focusing on two main topics: (i) trip or itinerary recommendation, and (ii) computational aspects of the solution of optimization problems.

Recommendation systems. This topic has received a lot of attention in recent years, therefore the related literature is vast. Here, due to space constraints, we highlight some representative works based on the taxonomy provided in two recent surveys [1] [7]. The interested reader can find more details in such surveys and the references therein.

The main problem to consider is the identification of the POIs and their relevance. The data used to find POIs can be gathered from different sources, such as user check-in behaviours [8], [9], crowdsourced digital footprints [10], [11], GPS data [12], [13], or it can be inferred by using geographical or social correlations of visited POIs [14], [15], [16]. Once the system has the list of POIs, it needs to select the subset of POIs that are relevant to the user. The recommendation may take into account multiple constraints [3], [17] or constraints related to time [4], [18]. The POIs can be used to build semantically enriched trajectories – for a survey on the topic, please refer to [19]. All the above systems are focused on the user viewpoint to provide a tailored recommendation. Only some of them include spatial consideration in building the itinerary, and none of them adapts the solution considering the number of users that can be present in the POIs.

The only work that considers how much a POI may be crowded is [2]. Nevertheless, the proposed system bases its recommendations on instantaneous information, thus it may generate new peak hours at the different POIs. Moreover, the authors do not consider the geographical aspects in building itineraries. To the best of our knowledge, our work (along with the conference version [6]) is the first that takes into consideration the impact of the recommendations on the current and future level of crowding, so that to balance the users among the POIs.

Recommendations systems have been proposed also for guiding tourists within a museum [20], [21]. In [20] the authors study the user behavior, but they do not propose a recommendation system that suggests the sequence of rooms to visit, i.e., they do not build a whole path tailored to the visitor needs and the level of crowding as we do. Lykourentzou et al. [21] is tailored to the user experience, but it does not consider the impact of the recommendation on the crowding level of each room. For instance, given the crowd tolerance of a user, the system may divert her/his path to avoid a room. But this may increase the crowd in another room, i.e., there could be oscillatory dynamics. In our system, instead, we consider the estimation of the level of crowding, and the system will constantly try to balance the users in order to control the crowding.

Optimization problem. Approximate solutions of optimization problems have been extensively treated in the literature. Hoos et al. [22] provide a broad view of the techniques and the solutions adopted so far. Since we are interested in a near real-time system, we focus on some works that deal with the parallel implementation of a specific technique, i.e., the Simulated Annealing (SA).

A common idea is to adopt an Asynchronous Approach [23], [24], where different workers execute independent SA using different starting solutions, and the best solution among them is reported. Inspired by such results, the authors in [25], [26] propose different MapReduce implementations, where the computations is divided among MAP and REDUCE tasks in different ways. The solution of multi-objective optimization problems using SA have been considered in [27], [28], and its parallel implementation in [29]. To the best of our knowledge, these parallel implementations have never been adapted to the MapReduce framework. In our work, we take inspirations from the above mentioned works to design a MapReduce implementation of the solution of a multi-objective optimization problem.

3 PROBLEM FORMULATION

In this section we provide the necessary definitions and formalize the trip planning problem we want to consider. The notation we used is similar to the one adopted by [13].

Definition 1 (Point of interest). *A point of interest (POI) p represents an attraction reachable by users. It is characterized by several attributes, such as the admission fee, or the opening hours. Among these, we consider in particular the spatial coordinates defining its position on the Earth surface, which we denote with p^c. We also consider the* duration of a visit, *denoted by $p^v(t)$, which depends on the instant t when the visit starts.*

The dependency on t is necessary since $p^v(t)$ is influenced by many factors, such as the day of the week, and the number of people currently visiting the POI p. We will show in Sect. 4 how we compute (and update) the value of $p^v(t)$. For the purposes of this paper, the set of POIs that can be considered for building a trip is assumed to be known and fixed, and is denoted by $\mathcal{P}$.

Definition 2 (Trip). *A trip τ is an ordered collection of POIs, i.e., $\tau = \langle p_1, p_2, \ldots, p_n \rangle$, where n indicates the number of POIs contained in τ, $|\tau| = n$.*

Given the set of POIs, $\mathcal{P}$, the set of all possible trips, denoted by $\mathcal{T}$, contains all the possible ordered combinations of POIs, for any cardinality of τ.

Definition 3 (Path). *Given any two spatial coordinates c_i and c_j, and a travel mode m (e.g, walking, public transportation), a path $\pi(c_i, c_j, m)$ is a continuous portion of a transport network that connects the points whose location is defined by c_i and c_j. The path is characterized by the travel distance, $\pi_{td}(c_i, c_j, m)$, and by the travel time, $\pi_{tt}(c_i, c_j, m)$.*

In order to maintain the notation simple, we may not indicate the dependency of π_{td} (π_{tt}) on the travel mode, which is specified by the user when she/he submits the query to the system. In general, the travel time and the travel distance are not necessarily correlated, since the same path can be done with different travel modes that result in different travel time.

Notice that, several alternative paths may be defined between two POIs using the same transportation network and travel mode m. In particular, a path can include a kind of streets, called *scenic routes*, that are attractive for the tourist even if they are not included in the shortest path between POIs. As suggested in [30], in fact, a person is willing to trade shortest paths with pleasant ones, especially if the difference in terms of distance is limited.

Definition 4 (Scenic route). *Given a transport network, a scenic route is a linear portion of it that represents a tourist attraction. Given a path $\pi(c_i, c_j)$ we denote as $\pi_\sigma(c_i, c_j)$ the set of scenic routes that are spatially contained in the path.*

The number of scenic routes contained in a trip τ is dented as $\sigma(\tau)$. The set of possible scenic routes considered by the recommendation system is denoted as $\mathcal{R}$.

Definition 5 (Smoothness). *Given a trip τ its smoothness is the directional consistency over the sequence of paths $\pi(c_i, c_{i+1})$ between its POIs. The smoothness of a trip τ is given by the mean ($sm_\mu(\tau)$) and the standard deviation ($sm_{sd}(\tau)$) of the angular attributes of τ. Given a set of paths $\Pi = \{\pi(c_i, c_{i+1})\}_{i=1}^{|\tau|-1}$ for a trip τ, the vector $\Theta = (\theta_i)$ of angular attributes contains the smallest angle between each pair of paths $\pi(c_i, c_{i+1})$ and $\pi(c_{i+1}, c_{i+2})$:*

$$\begin{aligned}\theta_i = \min(&\angle(\pi(c_i, c_{i+1}), \pi(c_{i+1}, c_{i+2})),\\ &360^\circ - \angle(\pi(c_i, c_{i+1}), \pi(c_{i+1}, c_{i+2})))\end{aligned} \tag{1}$$

Notice that, given with s_i and s_j collinear, $\angle(s_i, s_j)$ is equal to 180°.

A large mean and a small standard deviation of the angular attributes of τ, denote a smooth trip, conversely a small mean and a large standard deviation denotes a very jagged trip [31]. The smoothness can be considered as an indication of how pleasant is a path: rather than going back and forth using the same portion of a path, it is interesting to explore alternative paths that touch different roads.

Definition 6 (Recommendation query). *Users looking for a recommendation submit a query Q to the system by specifying the following constraints:*

- *the initial coordinates c_0 where the trip begins (user position);*
- *the time at which the trip will start t_0;*
- *the desired trip duration as an interval (d_{min}, d_{max});*
- *the mandatory maximum trip duration $TD_{\max}$;*
- *the travel mode m.*

In order to reply to such a query, the system needs to compute a set of values that drives the trip selection. We start considering the main constraint, i.e., the total time of the duration of the trip should be less than $\mathrm{TD}_{\max}$. To this aim, we introduce a fictional POI p_0, which corresponds to the user initial position, and we set $p_0^c = c_0$ and $p_0^v(t_0) = 0$.

We denote with t_i the time of arrival at p_i, the i-th POI of the trip, which can be computed considering the time t_{i-1}, the visit time of the previous POI and the travel time between the two POIs, i.e.,

$$t_i = t_{i-1} + p^v_{i-1}(t_{i-1}) + \pi_{tt}(p^c_{i-1}, p^c_i), \quad i \geq 1. \tag{2}$$

Note that $t_1 = t_0 + p^v_0(t_0) + \pi_{tt}(p^c_0, p^c_1) = t_0 + \pi_{tt}(c_0, p^c_1)$, which represents the starting time of the trip plus the travel time between the user position and the first POI. We can now define the total trip time λ_τ for a trip τ as:

$$\lambda_\tau(c_0, t_0) = \sum_{i=1}^{n} \left(\pi_{tt}(p^c_{i-1}, p^c_i) + p^v_i(t_i)\right), \tag{3}$$

where $n = |\tau|$. When exploring the solution space, the system will consider the trips for which $\lambda_\tau(c_0, t_0) < \mathrm{TD}_{\max}$. The exploration is guided by the values of the objective function that can be defined starting from a set of possible optimization criteria. We focus mainly on six objective functions that have to be minimized (adding more objective functions is cumbersome). Note that we do not consider here additional information, such as specific topics of interests a user expresses, or previously visited POI to be excluded. These information, in fact, has an impact solely on the set of POIs $\mathcal{P}$ to be considered in our optimization, i.e., some POIs will not be included; the overall process described in this paper remains unchanged.

Definition 7 (Objective functions)**.** *Given a trip τ, the objective functions f_n, f_d, f_{tt}, f_{td}, f_{sr} and f_{sm} denote the number of locations* not *visited during the trip, the estimated trip duration, the estimated trip travel time, the total distance travelled during the trip, the number of scenic routes in the trip and the smoothness of the trip, respectively. They are computed as:*

$$\begin{aligned}
f_n &= |\mathcal{P}| - |\tau| \\
f_d &= \begin{cases} w_a \cdot (d_{min} - \lambda_\tau) & \text{if } \lambda_\tau < d_{min} \\ w_b \cdot (\lambda_\tau - d_{max}) & \text{if } \lambda_\tau > d_{max} \\ d_{max} - \lambda_\tau & \text{otherwise} \end{cases} \\
f_{tt} &= \sum_{i=1}^{|\tau|} \pi_{tt}(p^c_{i-1}, p^c_i) \\
f_{td} &= \sum_{i=1}^{|\tau|} \pi_{td}(p^c_{i-1}, p^c_i) \\
f_{sr} &= (|\tau| - 1) - \sigma(\tau) \\
f_{sm} &= 1 - sm_{sd}(\tau)/sm_\mu(\tau)
\end{aligned} \tag{4}$$

When defining f_n, we use the number of locations *not* visited (instead of the ones visited), so that all objective functions need to be minimized. The definition of f_d uses the weights w_a and w_b to discourage trips that have an estimated duration not included in the desired interval. Such weights can be chosen in order to consider more or less bad a trip with a duration less than the minimum desired one, or a trip with a duration greater than the maximum desired one, respectively. Functions f_{tt} and f_{td} represent the sum of the travel time and travel distance of the paths that compose the trip. As stated when we have presented Definition 3, the travel time and distance depends on the travel mode (walking, public transportation, etc.). Function f_{sr} counts the number of paths that do not contain a scenic routes (notice that the number of paths in a trip τ is equal to $|\tau| - 1$).

We are now ready to define the trip planning problem, which can be cast as an optimization problem:

$$\begin{aligned} \underset{\tau}{\text{Minimize}} \quad & \langle f_n, f_d, f_{tt}, f_{td}, f_{sr}, f_{sm} \rangle, \\ \text{subject to} \quad & \lambda_\tau(c_0, t_0) < \mathrm{TD}_{\max} \end{aligned} \tag{5}$$

Note that the global objective function we would like to minimize is a composition of objective functions, and it can be defined as $\bar{f} : \mathcal{T} \to \mathbb{R}^6$. We are therefore in the context of *multi-objective* optimization, in which it is not possible to define a total order. We need to introduce a *dominance* relation to partially order the set of possible solutions. A trip τ_i dominates a trip τ_j, denoted $\tau_i \prec \tau_j$, if at least one of the composing objective functions is smaller for τ_i than for τ_j, while the other are equivalent. The results of the optimization problem will be the set of mutually non-dominating trips, i.e.,

$$res(\mathcal{Q}) = \{\tau \in \mathcal{T} \mid \nexists \tau_0 \in \mathcal{T} \text{such that } \tau_0 \prec \tau\}$$

The equivalent, mutually non-dominating solutions will be presented to the user, who will choose the one she prefers.

Considering the cardinality of the set containing all the possible trips, $\mathcal{T}$, the solution space to explore in order to provide a recommendation is very large. In addition, note that the total trip time depends on the POI visit duration, which depends on the time when the visit starts, therefore the solution space further increases. For this reason, our search for the solution is based on well known heuristics for solving the optimization problem.

4 Proposed system

This section provides a detailed description of the proposed system starting from an overview of its architecture, then presenting the algorithms for trip recommendation, and their implementation in MapReduce.

4.1 Overview

Our proposed recommendation system has two main components: an offline analysis of the user presences at different POIs, and a recommendation engine based on a parallel implementation divided into two main stages. Fig. 2 shows the overall architecture. User presence at the POIs is collected in a database. Overnight, the new records are processed by an offline procedure that extracts the main information and updates the data structures that are used by the recommendation engine. In particular, a set of common trips are stored back in the database, while the POI profiles (see Sect. 4.2) are stored in a distributed cache. The recommendation engine itself is composed by two steps: given a query, it first extracts a set of possible solutions from the set of common trips, then it explores the solution space looking for improved solutions by taking into account the optimization criteria, which include the time spent in each POI.

Once the system has issued the recommendation, it updates the POI profiles to reflect the impact of such recommendation on the POI crowding. Therefore, when the next user will submit a query, the recommendation engine will use the updated values of POI profiles while computing the optimal solution. In the following sections, we describe in detail the different system components.

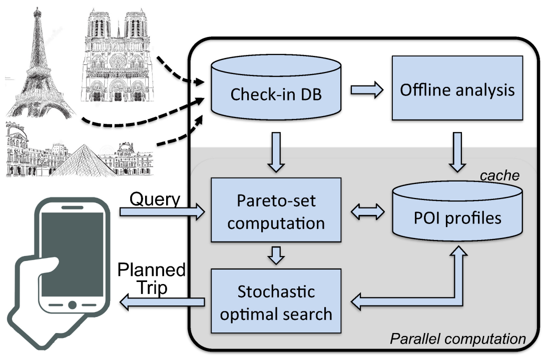


Fig. 2. System architecture. Tourist check-ins are registered in a database that are analyzed offline to build the POI profiles, and online to provide the starting point to the exploration of the solution space.

4.2 Offline analysis of the check-ins

The POIs record the entering and the exiting visitors: in fact, for security reasons, it is necessary to know how many people are inside a POI. Every time a new entrance is recorded, the POI sends to the database a record $r = \langle t^{ci}, p, u^{id}, n^p \rangle$, where t^{ci} is the check-in (entrance) timestamp, $p \in \mathcal{P}$ is the POI, u^{id} is the user identifier (if it exists), and n^p is the current number of visitors inside the POI p.

There are two types of users: anonymous and registered. Registered users are people who use, for instance, a bundle offer, where they receive an identifier and they can access to a set of POIs with reduced prices (e.g., a city-pass, or an app for their smartphones). All the other users that cannot be identified, such as local users who visit a single POI, are anonymous.

Having registered users is important, since it is possible to reconstruct the set of POIs that they visited. The offline analysis of the registered users allows to build a set of *popular trips*. We build this set using the algorithms developed for the Frequent Itemset Mining problem [32], [33], [34]: given a set of "baskets" (in our case, the trips of the registered users), we mine the most common "items" (in our case, POIs), i.e., the set of n items that appear most frequently in the baskets, with $n = 1, 2, \ldots$. The analysis can be done easily in parallel with a Mapreduce approach [33]. Such set of popular trips are stored back in the main database: they are accessed by the recommendation engine as a starting point in the search of the optimal solution when replying to a query.

Another advantage derived from registered users is the possibility to compute an important metric: the visiting time. Given a registered user u^{id}, for any two consecutive visited POIs, p_j and p_k, the offline analysis can compute the actual time spent at POI p_j by subtracting from the interval between the timestamps t^{ci}_j and t^{ci}_k the travel time from p_j to p_k, assuming a given travel mode m (e.g., the most used m). In such way, the record corresponding to the check in of u^{id} at p_j, formally $r_j = \langle t^{ci}_j, p_j, u^{id}, n^p_j \rangle$ can be enriched by a new element, $vt_j = t^{ci}_k - t^{ci}_j - \pi_{tt}(p^c_j, p^c_k, m)$, where t^{ci}_k is taken from the record $r_k = \langle t^{ci}_k, p_k, u^{id}, n^p_k \rangle$. The enriched records now contain a direct relation between the number of people inside a POI and the visiting time for that POI.

In summary, given a POI p_i, it is possible to build a set of characterizing measures, which we call *profiles*:

- **Average Time Occupancy**, $\text{ATO}(d, h)$: for each day d of the week, it represents the average number of visitors inside the POI at time h (h has the granularity of hours); the average is calculated considering the same day for a given interval (e.g., last year);
- **Average Visiting time**, $\text{AVT}(n^p)$: it provides the average visiting time given a number of visitors inside the POI; the average is computed considering all the enriched records, by grouping the records for the same value of n^p.

The Average Time Occupancy $\text{ATO}(d, h)$ reflects how much crowded a POI is on average, and it should show some peaks at specific hours (e.g. mid morning). As for the Average Visiting Time $\text{AVT}(n^p)$, intuitively it should be an increasing function, i.e, as the number of user inside a POI increases, the visiting time should increase, since the crowding slows the visit.

With the profiles defined above, the duration of a POI visit at time t, introduced in Def. 1 and denoted by $p^v(t)$, can be computed as follow. Given the time t we can derive the day d and the hour h, we then compute the average number of user for that day at that hour, $n^p = \text{ATO}(d, h)$, and then we derive the average visiting time from $\text{AVT}(n^p)$:

$$p^v(t) = \text{AVT}(\text{ATO}(d, h)), \tag{6}$$

with d day and h hour of the request derived from t.

Algorithms in Fig. 3-4 illustrate the implementation of two MapReduce jobs for computing the ATO and AVT statistics, respectively. In particular, given a user record r_j, the Mapper in Fig. 3 builds a tuple by isolating from its timestamp t^{ci}_j: the day d of the week and the hour h. The key of each produced value is represented by these information together with the POI identifier p_j, while the value is the number of visitors currently in the POI. More than one reducer can be concurrently executed, each one working on the values related to one key at time. In particular, for each key it produces the average number of visitors that occupies a particular POI in a given time (hour) of a week day.

Fig. 4 describes the construction of AVT for each possible number of visitors in a POI. In particular, given a user record, each Mapper produces a pair having as key the user identifier and as value the tuple composed of the timestamp, the POI, and the number of users currently in the POI. More

Mapper

```
1: procedure MAP($r_j = \langle t^{ci}_j, p_j, u^{id}, n^p_j \rangle$)
2:     $d \longleftarrow \text{DAYOFWEEK}(t^{ci}_j); h \longleftarrow \text{HOUR}(t^{ci}_j)$
3:     return $(\langle d, h, p_j \rangle, n^p_j)$
4: end procedure
```

Reducer

```
5: procedure REDUCE($k, \langle x_1, x_2, \ldots \rangle$)
6:     $v \longleftarrow \text{AVG}(x_1, x_2, \ldots)$
7:     return $(k, v)$
8: end procedure
```

Fig. 3. MapReduce job for the initialization of the $\text{ATO}(d, h)$ using the available historical data.

Mapper

```
1: procedure MAP($r_j = \langle t_j^{ci}, p_j, u^{id}, n_j^p \rangle$)
2:     return $(u^{id}, \langle t_j^{ci}, p_j, n_j^p \rangle)$
3: end procedure
```

Reducer

```
4: procedure SETUP
5:     $\Pi_{tt} \longleftarrow$ retrieve from cache
6:     $V_{map} \longleftarrow \emptyset$
7: end procedure

8: procedure REDUCE($k$, $\langle x_1, \dots x_n \rangle$)
9:     $\langle x'_1, \dots, x'_n \rangle \longleftarrow$ SORT$(x_1, \dots, x_n)$    ▷ by timestamp
10:    for $j \in [1, n-1]$ do
11:        $vt_j = t_{j+1}^{ci} - t_j^{ci} - \pi_{tt}(p_j^c, p_{j+1}^c, m)$
12:        $V_{map}$.ADD$(\langle p_j, n_j^p \rangle,\ vt_j)$
13:    end for
14: end procedure

15: procedure CLEANUP
16:    for $(\langle p_j, n_j^p \rangle, \langle v_1, v_2, \dots \rangle) \in V_{map}$ do
17:        return $(\langle p_j, n_j^p \rangle$, AVG$(v_1, v_2, \dots))$
18:    end for
19: end procedure
```

Fig. 4. MapReduce job for the initialization of the AVT(n^p) using the available historical data.

than one reducer can be executed, each one working on a set of values regarding the same user. Before the execution of any reducer, a SETUP procedure is performed which loads in memory a data structure containing the travel time between any given pair of POIs using a particular travel model (denoted as Π_{tt}). In each iteration, a reducer sorts the user records by timestamp and determines the visiting time for each POI considering the timestamp between consecutive visits and the travel time between them. Then the reducer updates a map which contains for each POI and possible number of visits, the computed visiting time. Finally, in the CLEANUP procedure, the computed visiting times related to the same POI and number of visits is averaged, producing the AVT.

The above definition may suggest that the system does not adapt to the estimated level of crowding as more recommendation are provided by the system, since ATO(d, h) is computed offline. We will show in Sect. 4.6 that the actual ATO$'(d, h)$ profile used by the recommendation engine contains a dynamic variable component, which is continuously updated during the day. The system, therefore, is able to tailor the recommendation to the estimated level of crowding.

4.3 Exploration of the solution space

Looking for an exact solution of the optimization problem defined in Eq. (5) is computationally expensive, thus we need to resort to well known heuristics. Our solution builds trip recommendations using a dominance-based Multi-Objective Simulated Annealing (MOSA) [28] technique.

Multi-Objective Simulated Annealing. At each step of the simulated annealing procedure, the current solution is replaced with a random one with a probability that depends both on the difference between the corresponding objective values and a global parameter T (*temperature*), which is progressively decreased during the process. This behaviour avoids the stuck on local optima, which is the main drawback of many heuristics proposed for the solution of the optimization problem. It has been proven that, using a simulated annealing approach, the solution converges to the global optimum if annealed sufficiently slow [35].

The exploration of the solution space is based on the comparison between the current solution τ_{curr} with a new potential solution τ_{new}, obtained through a *perturbation* of the current solution τ_{curr}. The perturbation could be, for instance, a POI removal or addition, or a change in the order of the POI. The comparison is done by considering the objective function $\bar{f} = \langle f_n, f_d, f_{tt}, f_{td}, f_{sr}, f_{sm} \rangle$ defined in Eq. (4). As stated in Sect. 3, with multi-objective optimization, we can define a partial order on the solution based on the concept of dominance: a trip τ_{new} dominates another trip τ_{curr}, denoted as $\tau_{\text{new}} \prec \tau_{\text{curr}}$, if it is better in at least one objective function and equivalent in the remaining ones.

Trips τ_{curr} and τ_{new} are mutually non-dominating if and only if neither dominates the other. The set of mutually non-dominating solutions is called *Pareto-set*, and it is denoted by $\mathcal{S}$. A solution not dominated by any other solution is called *Pareto-optimum*. From the Pareto-set $\mathcal{S}$ we can compute the *Pareto-front* $\mathcal{F} \subseteq \mathbb{R}^6$, which is the set of points in the objective space, i.e., $\mathcal{F} = \{\bar{f}(\tau) \mid \tau \in \mathcal{S}\}$

The goal of a MOSA algorithm is to move the current Pareto-front towards the optimal Pareto-front (the Pareto-front of the Pareto-optimum set) while encouraging the diversification of the candidate solutions. In particular, the probability of making a transition from the current solution τ_{curr} towards a new solution τ_{new} is specified by an acceptance probability function $P(\tau_{\text{curr}}, \tau_{\text{new}}, T)$ which depends upon the global parameter T (*temperature*) and the energy of the two solutions. The energy of a solution τ, denoted by $E(\tau, \mathcal{F})$, measures the portion (number of solutions) of the current Pareto-front that dominates τ, i.e.,

$$E(\tau, \mathcal{F}) = |\{v \in \mathcal{F} \mid v \prec \bar{f}(\tau)\}|. \quad (7)$$

Note that the energy of a solution τ belonging to the Pareto-front is 0. Given two solutions τ_{curr} and τ_{new}, where τ_{curr} is part of the Pareto-set, and therefore $\bar{f}(\tau_{\text{curr}})$ is part of the Pareto-front $\mathcal{F}$, we can compute the energy difference between τ_{curr} and τ_{new} by considering the extended Pareto-front $\mathcal{F}' = \mathcal{F} \cup \bar{f}(\tau_{\text{new}})$ as following:

$$\Delta_E(\tau_{\text{new}}, \tau_{\text{curr}}, \mathcal{F}') = \frac{E(\tau_{\text{new}}, \mathcal{F}') - E(\tau_{\text{curr}}, \mathcal{F}')}{|\mathcal{F}'|} \quad (8)$$

The acceptance probability $P(\tau_{\text{curr}}, \tau_{\text{new}}, T)$ is then

$$P(\tau_{\text{curr}}, \tau_{\text{new}}, T) = \min\left(1, \exp\left(-\frac{\Delta_E(\tau_{\text{new}}, \tau_{\text{curr}}, \mathcal{F}')}{T}\right)\right) \quad (9)$$

Note that it is possible to escape local optima, since a candidate solution τ_{new} that is dominated by one or more members of the current estimated Pareto-front, may still be accepted with a probability defined in Eq. (9). On the other hand, candidate solutions that move the estimated front towards the true front are always accepted, since

$P(\tau_{\text{curr}}, \tau_{\text{new}}, T) = 1$. The temperature T is actually a monotonically decreasing function, that decreases at every iteration at a very slow rate till it reaches a minimum value.

Basic building block: TRSA Algorithm. The exploration of the solution space is based on a building block called TRSA (Trip Recommendation Simulated Annealing) and illustrated in Fig. 5. Starting from a given Pareto-set $\mathcal{S}_{\text{init}}$ and a trip $\tau \in \mathcal{S}_{\text{init}}$, the algorithm looks for potential new trips to be added to $\mathcal{S}_{\text{init}}$ in order to advance the Pareto-front $\mathcal{F}$.

TRSA

```
1:  procedure TRSA($\mathcal{S}_{\text{init}}$, $\tau$, $\text{TD}_{\max}$, $T_{\min}$, $T_{\text{init}}$)
2:      $\mathcal{S} \longleftarrow \mathcal{S}_{\text{init}}$
3:      $\mathcal{F} \longleftarrow$ ComputeParetoFront($\mathcal{S}$)
4:      $T \longleftarrow T_{\text{init}}$
5:      while $T > T_{\min}$ do
6:          $\tau' \longleftarrow$ Perturb($\tau$, $\text{TD}_{\max}$)
7:          $\mathcal{F}' \longleftarrow \mathcal{F} \cup \bar{f}(\tau')$
8:          $\Delta_E \longleftarrow$ ComputeEnergyDiff($\tau', \tau, \mathcal{F}'$)
9:          $P \longleftarrow \min(1, \exp(-\Delta_E/T))$
10:         if $rand(0,1) < P$ then
11:             RemoveDominated($\mathcal{S}, \tau', \mathcal{F}, \bar{f}(\tau')$)
12:             $\mathcal{S} \longleftarrow \mathcal{S} \cup \tau'$
13:             $\mathcal{F} \longleftarrow \mathcal{F} \cup \bar{f}(\tau')$
14:             $\tau \longleftarrow \tau'$
15:         end if
16:         UpdateTemperature($T$)
17:     end while
18:     return $\mathcal{S}$
19: end procedure
```

Fig. 5. TRSA algorithm.

The function ComputeParetoFront() uses the input Pareto-set $\mathcal{S}$ for initializing the Pareto-front $\mathcal{F}$. As long as the temperature T is greater than the minimum value T_{min}, the algorithm explores the solution space by *perturbating* the current solution τ. The possible perturbations can regard any POI except the first one (since it's the starting point declared by the user) and they can be:

- adding a POI in a random position;
- removing a POI;
- replacing a POI with another one;
- shifting the position between two POIs.

Given the set of transformations, it is clear that, even if we start from a set of popular trips, it is possible to explore freely the solutions space, and any POI not present in the starting point can be included during such exploration, and it could be part of the final recommendation.

The function Perturb(), while looking for a new potential trip τ', evaluates its total trip time, $\lambda_{\tau'}$, and considers only the trips for which $\lambda_{\tau'} < \text{TD}_{\max}$.

The algorithm then computes the energy of τ' (line 8), and the probability of accepting τ' (line 9). If τ' is accepted, we remove from $\mathcal{S}$ the trips dominated by τ', and from $\mathcal{F}$ the corresponding points (line 11), we add the trip to $\mathcal{S}$ and continue to explore. At each iteration, the temperature is updated according to a cooling strategy, such as the ones defined in [36].

Using the TRSA Algorithm. The initial Pareto-set $\mathcal{S}_{\text{init}}$ contains a set of equivalent solutions, and the aim of TRSA is to look for better solutions starting from a trip $\tau \in \mathcal{S}_{\text{init}}$. This exploration can be done on a single machine. In parallel, other machines may try to improve the Pareto-front $\mathcal{F}$ starting from other trips $\tau' \in \mathcal{S}_{\text{init}}$. Therefore TRSA represents the basic building block of the overall parallel computation. In Sect. 4.5 we will show how these parallel computation is done with MapReduce. Before, we need to determine the main input of the TRSA algorithm: the initial Pareto-set $\mathcal{S}_{\text{init}}$. Such input can be determined in parallel using the MapReduce framework.

4.4 Initial Pareto-set

The initial Pareto-set $\mathcal{S}_{\text{init}}$ is built using the popular trips computed offline (see Sect. 4.2) and stored in the main database. The evaluation of these potential solutions includes the verification of the main constraint, i.e., the duration of the selected trips cannot be longer than the total trip duration $\text{TD}_{\max}$. This is done with the help of the profiles $\text{ATO}(d, h)$ and $\text{AVT}(n^p)$ stored in the cache. Moreover, we need to check that each potential trip is not-dominated by the trips currently inserted in $\mathcal{S}_{\text{init}}$.

The evaluation of the potential trips to be added to $\mathcal{S}_{\text{init}}$ may be expensive. Nevertheless, it can be done easily in parallel, since each trip is independent from the others (see algorithm in Fig. 6). Given a query $\mathcal{Q}$ that defines the start point, the travel mode and the mandatory maximal duration, we extract all the popular trips that have a compatible duration: the duration interval of the popular trips has been computed offline considering as a starting point the first POI of the trip, and a visiting time for each POI equal to the average visiting time computed over all the visits at that POI. For each of these popular trips, the Map method checks if the trip satisfies $\mathcal{Q}$, it actually computes the total duration considering the starting point specified in the query and the time at which the trip will start. If the trip satisfies the query, it is added to $\mathcal{S}$. The addition is done with the help of the function Update$(\mathcal{S}, \tau)$, which ensures that $\mathcal{S}$ does not contain duplicates and that dominated values are removed. Since multiple Map calls may be executed by the same JVM, we return the $\mathcal{S}$ in the CleanUp method called at the end of the task.

The Reduce method collects the partial Pareto-sets computed by the Map tasks, and merge them using the Update$(\mathcal{S}, \tau)$. Since it is necessary to verify that the merged Pareto-sets do not contain dominated solutions, there could be only one reducer. Nevertheless, most of the work is done by the mappers, then the reduce simply compares the proposed solutions.

4.5 Stochastic parallel search of the optimum

As reported in Sect. 2, different solutions have been proposed in literature for parallelizing the Simulated Annealing algorithm with MapReduce. The approach we adopt is similar to the one presented in [23]: we use different mappers for executing independent iterations of the TRSA algorithm, starting from different solutions $\tau \in \mathcal{S}_{\text{init}}$, and we then use the reducer to compute the final result.

Mapper

```
 1: procedure SETUP
 2:     $\mathcal{S}_{\text{map}} \longleftarrow \emptyset$
 3: end procedure

 4: procedure MAP(id, $\tau$)
 5:     if $\tau$ satisfies $\mathcal{Q}$ then
 6:         $\mathcal{S}_{\text{map}} \longleftarrow$ UPDATE($\mathcal{S}_{\text{map}}, \tau$)
 7:     end if
 8: end procedure

 9: procedure CLEANUP
10:     return $(\mathcal{Q}, \mathcal{S}_{\text{map}})$
11: end procedure
```

Reducer

```
12: procedure REDUCE($\mathcal{Q}$, $\langle \mathcal{S}_1, \mathcal{S}_2, \dots \rangle$)
13:     $\mathcal{S}_{\text{init}} \longleftarrow \emptyset$
14:     for $\mathcal{S}_i \in \langle \mathcal{S}_1, \mathcal{S}_2, \dots \rangle$ do
15:         for $\tau \in \mathcal{S}_i$ do
16:             $\mathcal{S}_{\text{init}} \longleftarrow$ UPDATE($\mathcal{S}_{\text{init}}, \tau$)
17:         end for
18:     end for
19:     return $(\mathcal{Q}, \mathcal{S}_{\text{init}})$
20: end procedure
```

Fig. 6. MapReduce job for the initialization of the Pareto-set $\mathcal{S}_{\text{init}}$. The key *id* read by the mapper is a generic identifier associated to each row and can be safely ignored.

The initial Pareto-set $\mathcal{S}_{\text{init}}$ computed in Fig. 6 is stored both in the cache (so that all mappers can access to it) and in a parallel data structure that makes easy to access to each $\tau \in \mathcal{S}_{\text{init}}$ in parallel by the different mappers. The MapReduce pseudo-code is shown in Fig. 7.

Each mapper performs an execution of the TRSA algorithm, i.e., it explores the solution space starting from a trip τ. The output of each mapper contains the improved Pareto-set S_i, and these sets are combined together by a single reducer to eliminate dominated and redundant solutions. Note that, also for this job, most of the work is done by the mappers, which explores the solution space through perturbations.

4.6 Closing the loop: Profile update

The recommendation system takes as input a set of popular trips and compute the best solutions for a user query $\mathcal{Q}$. To this aim, since the query contains the maximum trip duration $\text{TD}_{\max}$, the system uses the profiles stored in the cache to compute the duration of the potential solutions. If we use the profiles $\text{ATO}(d, h)$ and $\text{AVT}(n^p)$ (defined in Sect. 4.2) computed offline, we obtain a static system that redistribute the users according to average values. This approach may be still important, since not all tourists make use of the recommendation system, therefore the averages may be a good indication of the level of crowding.

Nevertheless, with the diffusion of smartphones, we may expect that an increasing number of tourists will use the recommendation system, therefore we should be able to

Mapper

```
1: procedure MAP(id, $\tau$)
2:     $\mathcal{S}_{\text{map}} \longleftarrow \emptyset$
3:     $\mathcal{S}_{\text{init}} \longleftarrow$ retrieve from cache
4:     $\mathcal{S}_{\text{map}} \longleftarrow \text{TRSA}(\mathcal{S}_{\text{init}}, \tau, \text{TD}_{\max}, T_{\min}, T_{init})$
5:     return $(\mathcal{Q}, \mathcal{S}_{\text{map}})$
6: end procedure
```

Reducer

```
 7: procedure REDUCE($\mathcal{Q}$, $\langle \mathcal{S}_1, \mathcal{S}_2, \dots \rangle$)
 8:     $\mathcal{S} \longleftarrow \emptyset$
 9:     for $\mathcal{S}_i \in \langle \mathcal{S}_1, \mathcal{S}_2, \dots \rangle$ do
10:         for $\tau \in \mathcal{S}_i$ do
11:             $\mathcal{S} \longleftarrow$ UPDATE($\mathcal{S}, \tau$)
12:         end for
13:     end for
14:     return $(\mathcal{Q}, \mathcal{S})$
15: end procedure
```

Fig. 7. MapReduce job for the execution of the TRSA algorithm. The key *id* read by the mapper is a generic identifier associated to the trip and can be safely ignored.

update the profiles to reflect the actual tourist distribution over the POIs. In particular, we consider the Average Time Occupancy $\text{ATO}(d, h)$ profiles. While building offline these profiles, it is possible to identify the two main components for such profiles: the occupancy due to (i) registered and (ii) anonymous users. While we can not have any impact on the anonymous users, we may be able to influence the registered users, since they are the tourists that make use of the recommendation system. Therefore, in our system we consider the following *Average Time Occupancy* profiles:

$$\text{ATO}'(d, h) = \text{ATO}_{\text{anon}}(d, h) + \text{ETO}_{\text{reg}}(d, h) \qquad (10)$$

where $\text{ATO}_{\text{anon}}(d, h)$ is the component due to anonymous users, and $\text{ETO}_{\text{reg}}(d, h)$ is the *Estimated Time Occupancy* computed considering the registered users and the recommendations done so far by the system.

The profiles $\text{ETO}_{\text{reg}}(d, h)$ are reset at the beginning of each day with the average behaviour of the user in the past. As the system issues recommendations, it records the choices of the users, and it updates the estimation of the POI occupancy assuming that the user will follow the recommended trip, spending the estimated time in each POI and for traveling from one POI to the next. Even if the actual user behaviour may vary, the overall estimation of the POI occupancies should not be significantly affected, since they are the results of aggregated values. In Sect. 5 we will show how different combinations of $\text{ETO}_{\text{reg}}(d, h)$ and $\text{ATO}_{\text{anon}}(d, h)$ can produce different effects on the system.

4.7 Discussion

The current recommendation system considers a set of objective functions in the search for the optimal solution. Using the Avergage Time Occupancy $\text{ATO}'(d, h)$ defined in Eq. (10), for a given trip τ we can estimate the number of visitors inside a POI at the time the tourist should reach the POI (and this estimate includes the recommendations done so far), and we can compute the time spent inside the POI from the AVT(n^p) profile. The level of crowding influences the number of visitors inside a POI, and the corresponding computation of the visiting time. Selecting POIs without considering the current level of crowding may result in spending time in crowded POIs, which in turn influences the number of POIs that can be visited. Since one of the objective functions includes the number of visited POIs, optimizing using such objective function means finding the solution where the time spent in the visited POI is at its minimum. In particular, in the following section we study the effectiveness of the proposed method by considering different behaviours of registered users. For instance, we try to determine the impact on the overall system of users variability in following the recommendation.

We can consider additional objective functions that take into consideration the affinity with the user interests, or the constraints related to the tourist budget, or the relevance of a POI using rankings. All these extensions are straightforward to implement, and we consider here only the functions defined in Eq. (4) in order to show the effectiveness of our approach.

5 Case Study and Experiments

We evaluated the recommendation system described in this paper using real-world traces collected for registered tourists visiting the city of Verona, Italy.

5.1 Available dataset

The tourist office of the city of Verona offers a *sightseeing city pass* called "VeronaCard": for a given fee, the tourist may visit up to 22 POIs around the city within a specific time-frame (e.g., 24 hours, or 48 hours). Every time a tourist with the VeronaCard enters in a POI, a record is created: it contains the VeronaCard number (unique identifier), the timestamp of the entrance and the POI identifier. The dataset includes approximately 1,200,000 records that spans 5 years. We use such a dataset for our experiments. We were not able to find publicly available datasets similar to the one described here that could be used to study our solutions in different contexts. From the Verona's tourist office dataset, we derive a set of data and measurements that we use in our experiments. We start by building the trips followed by the tourists with a VeronaCard, i.e., the sequence of visited POIs, obtaining approximately 250,000 trips. For each trip, given two consecutively visited POIs, with the help of the Google APIs, we compute the travel time and distance of the path connecting such two POIs. Since Verona is a small city and all the POIs are within walking distance, we assume walking as main travel mode. Knowing the travel time, we derive the visiting time for each POI, at different times of the day. In addition, we compute the number of tourists inside each POI[1].

Table 1 shows some statistics related to trips. We grouped trips with the same number of visited POIs, $|\tau|$: for each group, we show the number of trips with that number of POIs, the average duration of the trips (considering the first POI as the starting POI), the average travel time and travel distance – we first sum the travel times and travel distances of the paths for each trip, and then compute the average.

TABLE 1
Statistics about the collected trips. The columns report: the number of visited POIs, the number of trips with such number of POIs, the average duration of the trip (hour:min), the average travel time, and the average travel distance.

$\lvert\tau\rvert$	# trips	λ_τ	avg trav. time	avg trav. dist.
2	14,520	04:10	00:10	750m
3	31,455	04:20	00:17	1,5Km
4	40,878	06:00	00:26	2,0Km
5	37,900	07:50	00:34	2,7Km
6	28,261	09:00	00:42	3,4Km
7	16,139	10:30	00:51	4,0Km
8	7,823	11:30	00:60	4,7Km
9	3,060	12:00	01:10	5,5Km

We use the processed dataset to build the POI *profiles* defined in Sect. 4.2: for any POI, we compute the Average Time Occupancy ATO(d,h) and the Average Visiting Time AVT(n^p). Fig. 8-9 show sample ATO(d,h) and AVT(n^p) profiles for two POIs called "Casa di Giulietta" and "Castelvecchio", respectively. As for the average time occupancy (Fig. 8), we show the curves for July's Sundays (the average number of visitors computed considering the Sundays in July). As expected, there are two peak hours, in the morning and the afternoon. Interestingly, the peak hours for the two POIs in the afternoon are slightly different.

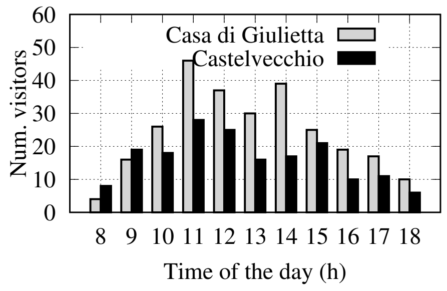


Fig. 8. Average Time Occupancy ATO(d,h) for two POIs in Verona. The averages have been computed considering the Sundays in July.

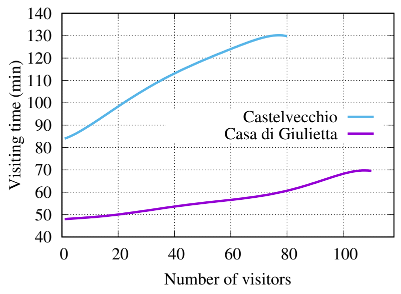


Fig. 9. Average Visiting Time AVT(n^p) for two POIs. The averages are computed considering the whole dataset.

Relatively the average visiting time (ATV curves shown in Fig. 9), we notice an increasing visiting time as the

1. The information about the exact number of tourists visiting a POI is available partially for a subset of POIs, therefore, for the purpose of our experiments, we prefer to compute in the same way the number of visitors with the described approach for all the POIs.

number of visitor increases, which indicates the impact of crowding in the visiting time.

As a final step, our offline processing collects the most popular trips, which will be used by the recommendation engine as a starting point when a new query is submitted.

5.2 Experimental methodology

In order to test our recommendation system, we need to provide a set of queries. To this aim, we consider our dataset and the trips we built from such a dataset. For a given day, we consider the trips collected that day: for each trip, we create a query where (i) the initial coordinates at which the trip starts are the coordinates of the first POI, (ii) the time at which the trip begins is the time of the access to the first POI, and (iii) the maximum trip duration is given by the computed trip duration time augmented with an estimated duration of the last visit[2]. For that set of queries, we observe the output from two possible perspectives: the POI and the trip viewpoint. From the POI viewpoint, we record for each POI the number of visitors over time, and we build the time occupancy curve for that day. From the trip viewpoint, we record the values of the objective functions defined in Eq. (4).

We compare three different approaches:

- *No recommendation*: we simply consider that the user follows the trip as built from the dataset, i.e., the query result is actually the trip from which the query has been derived, which is the trip performed by the user autonomously;
- *Recommendation based on averages*: we use the static version of the ATO(d, h) profiles, i.e., the recommendation are based on the average occupancy of the previous observation interval (e.g., last year);
- *Adaptive Recommendation*: we use the ATO$'(d, h)$ profiles, which are updated after every recommendation based on the recommendation given so far.

For the last case, we initially assume that half of the users recorded in each day are anonymous i.e., $\text{ATO}_{\text{anon}}(d, h) = 0.5 \cdot \text{ATO}(d, h)$. Then, in order to evaluate the dynamic effects produced by the $\text{ETO}_{\text{reg}}(d, h)$ profile on the $\text{ATO}'(d, h)$ one (i.e., on the level of crowding), for each query we perform different experiments by varying the percentage of anonymous users w.r.t. to the total number of users. In this way, we can evaluate the impact of non-registered users on the recommendation system. Moreover, we study the robustness of the method w.r.t. some variability on the user behaviour. To do so, we first assume that the users actually behave as expected, i.e., if we estimate a visiting time for a POI or a travel time for a path, it will take exactly those estimated times to visit that POI or to travel along that path. Then, we introduce some variability by considering the case in which some delays are collected during the visits.

The MapReduce TRSA algorithm has been implemented using SpatialHadoop [37], an extension of Apache Hadoop [38] which provides a native support for spatial data, in terms of spatial data types, operations and indexes. SpatialHadoop has been successfully applied in order to efficiently perform spatial analysis and validation of huge amount of geographical data [39], [40], [41].

2. For any trip, we are not able to know the visiting time of the last POI, since we do not have the next visited POI used to compute such value. The estimated duration of the last visit is simply the average visiting time for that POI.

5.3 Results

POI viewpoint. Figure 10 shows the number of visitors over time for the POI called “Casa di Giulietta” on February 14th, 2015, with or without a recommendation system. It is interesting to note that a static recommendation simply changes the peak hour with respect to a system with no recommendation, since it uses the average peak hour of the past days, but it does not adapt to the estimated number of users in the POI. Instead, our dynamic recommendation spread the tourists over time. In this case the percentage of registered users has been set to 50% of the total number.

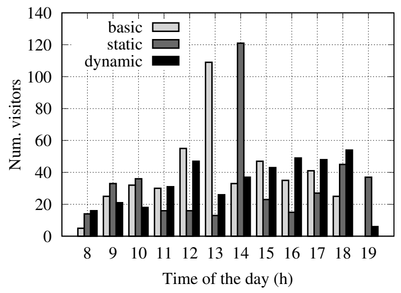
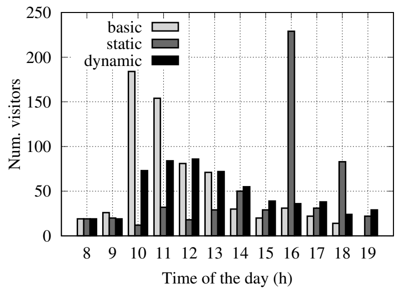


Fig. 10. Number of visits at “Casa di Giulietta” considering the behavior of the tourists without recommendation (basic) and with the two approaches based on average (static) and adaptive (dynamic) profiles for POIs with a percentage of registered users equal to 50%.

The same observations can be made when looking to other POIs. Figure 11 shows two famous POIs in Verona, namely the Arena and the Sant’Anastasia church. Also in these cases, we notice that our solution is able to spread the tourists more evenly over the visiting time.

As a further confirmation, we consider all the POIs of the datasets, and compute the average number of visitors over the day, as well as the minimum and the maximum number of visitors for that day. Figure 12 shows such values when tourists has no recommendation (basic), have static recommendation (static) and our dynamic recommendation (dynamic) – the POIs are ordered by average number of visitors. The effect of an ideal recommendation system is to avoid peaks, so that the tourists will not spend time in the waiting line or in a crowded place. This means that the difference between the maximum number of tourists and the average number of tourists should be minimum, as our dynamic approach obtains. In the basic and in the static scenarios, instead, the gap between the average number of visitors and the maximum number of visitors is is some case extremely wide.

Trip viewpoint. Fig. 13 illustrates three different alternative trips: the red one is an original trip performed by a user without any recommendation. It starts from the POI named “Torre dei Lamberti” at 11:06, then it stops at “Casa di Giulietta” at 12:33, and finally it reaches “Arena” at 14:15. The blue path is a solution produced by the TRSA algorithm using only *recommendation based on average* crowding



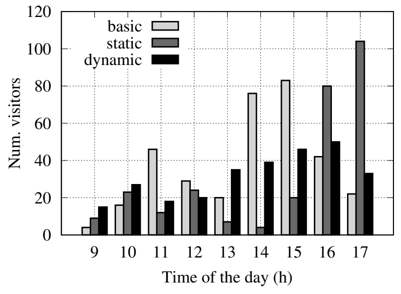


Fig. 11. Number of visits at "Arena" (top) and "Sant'Anastasia Church" (bottom), considering the behavior of the tourists without recommendation (basic) and with the two approaches based on average (static) and adaptive (dynamic) profiles for POIs.

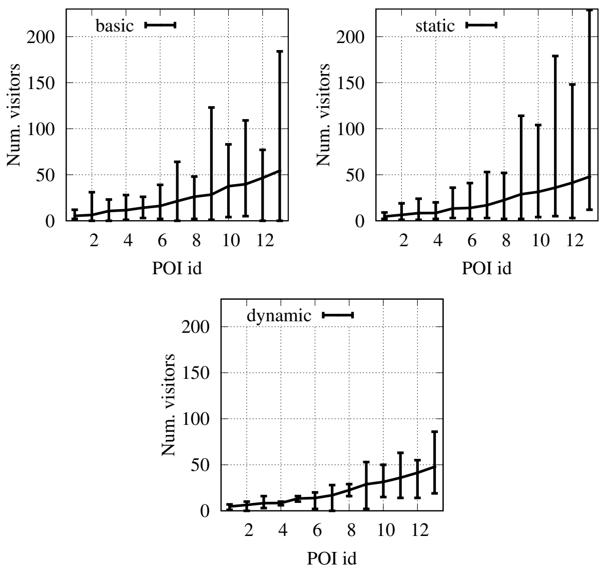


Fig. 12. Average, minimum and maximum number of visits at different POIs (identified by id and ordered by average number of visits), considering the behavior of the tourists without recommendation (basic) and with the two approaches based on average (static) and adaptive (dynamic) profiles for POIs.

information. As you can notice, the trip starts from the same POI and at the same time (query parameters), then it stops at "City Sightseeing" at 12:16, it proceeds towards "Casa di Giulietta" at 13:15, and it finally arrives at "Arena" at 14:15. In this case, the tourist arrived at "Casa di Giulietta" during the peak hour (13:15) for this specific day, since the algorithm considers only average historical static information about the POI occupancies. Finally, the green line represents the trip produced by the TRSA algorithm considering *adaptive recommendation* with 50% of registered users.

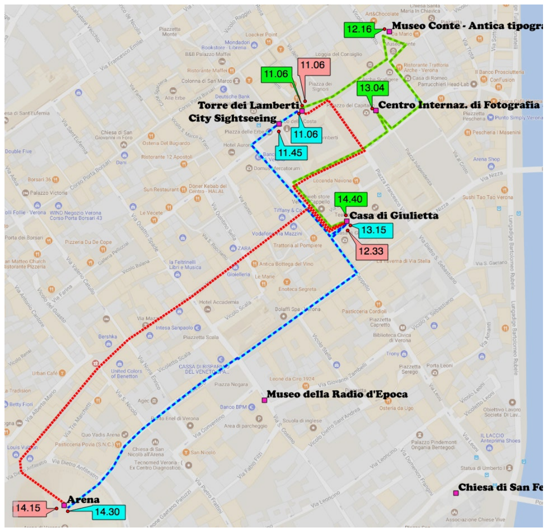


Fig. 13. An original path (the red one) together with two paths produced by our TRSA algorithm, the blue one obtained by using only historical statistical data about the level of crowding, while the green one considering also dynamic and adaptive information about the level of crowding.

In this case, the trip starts from the same POI and at the same hour, but it proceed towards "Museo Conte" at 12:16, then it visits "Centro Internazionale di Fotografia" at 13:04 and it arrives at "Casa di Giulietta" at 14:40 when the peak hour is passed. Notice that, the recommendation system has improved the values of the objective functions, indeed the blue trip enhanced f_n, since it includes an additional POI, while the green trip enhanced more objective functions: it has an additional POI, f_{tt} is decreased of 36% and f_{td} of 49% with respect to the red trip.

Fig. 14 illustrates the effects of considering the presence of scenic routes as an objective function. In particular, the green line represents a trip τ_g which covers three POIs: the amphitheater "Arena" and two churches, "Complesso del Duomo" and "Chiesa di Santa Anastasia". It has length of 1.7Km and contains a scenic route (i.e., the red portion) which passes near the Adige river. Conversely, the blue line represents a shorter trip τ_b. It connects the same POIs of τ_g and has a length of 1.5Km, but it does not contain any scenic route. τ_g survives in the Pareto set, since it includes a scenic route, while τ_b does not. Thus, since the cumulative objective function includes also f_{sr} (taking into account the scenic routes of the trips), τ_g is not dominated by τ_b and by any other trip with no scenic routes despite having better values of the other objective functions.

Similarly, Fig. 15 illustrates the role of the smoothness function in the optimization process. The blue and the green paths represent trips having approximately the same length (1.1Km), but the blue one contains one comeback, while the green one is a circular path that can be a preferable choice for a tourist. Again the same mechanism described before allows the survival of the green path in the Pareto set.

Table 2 reports some data about the quality of the produced recommendations considering a 50% of registered users. More specifically, we have considered the three POIs that have been most frequently chosen as the starting point for a trip: "Arena" (p_s^1), "Casa di Giulietta" (p_s^2) and "Castelvecchio" p_s^3. For each of these POIs, the table reports

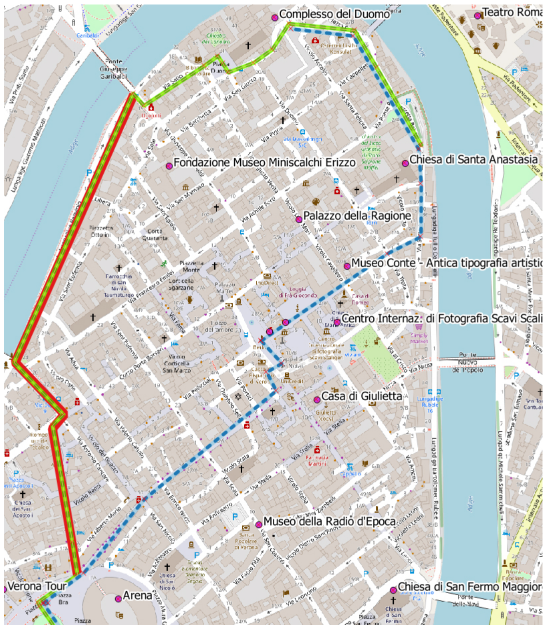

Complesso del Duomo
Teatro Roma
Fondazione Museo Miniscalchi Erizzo
Chiesa di Santa Anastasia
Palazzo della Ragione
Museo Conte - Antica tipografia artisti
Centro Internaz. di Fotografia Scavi Scali
Casa di Giulietta
Museo della Radio d'Epoca
Verona Tour
Arena
Chiesa di San Fermo Maggior

Fig. 14. Effect of including scenic routes as an objective function to be optimized. The green line is a path which covers three POIs: “Arena”, “Complesso del Duomo” and “Chiesa di Santa Anastasia” and contains a scenic route (i.e., the red portion), while the blue line is a shorter path connecting the same POIs but does not contain any scenic route.

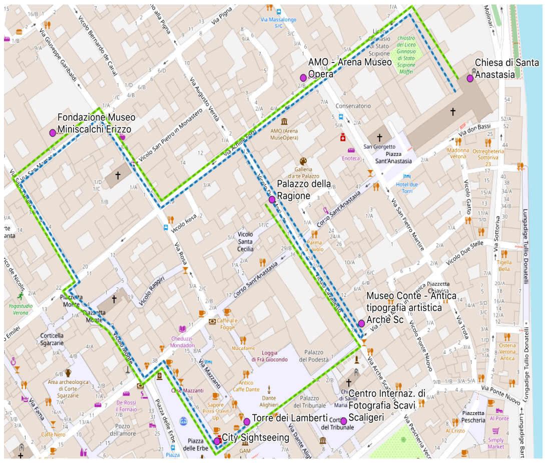

AMO - Arena Museo Opera
Chiesa di Santa Anastasia
Fondazione Museo Miniscalchi Erizzo
Palazzo della Ragione
Museo Conte - Antica tipografia artistica Arche Sc
Centro Internaz. di Fotografia Scavi Scaligeri
Torre dei Lamberti
City Sightseeing

Fig. 15. Effect of considering the smoothness as an objective function to be optimized. The blue and the green paths have approximately the same length (1.1Km), but the blue one contains one comeback, while the green one is a circular path that can be a preferable by a tourist.

the number of historical records, the size of the initial estimated Pareto-front built from these records, the percentage of improvement of the various objective functions w.r.t. the original trips. The table shows that each component of the objective function is improved by the TRSA algorithm. Moreover, both the improvements of each objective function and the overall improvement increase with the number of available historical trips. Table 3 reports the improvement of the objective functions by grouping the obtained trips w.r.t. their initial total time (λ_τ) instead of the starting points. In particular, such perspective reveals that the improvement is greater when the overall duration of the trip increases, see rows 2 and 3. This is reasonable since, if the available time increases, there is more room for improvements. For instance, as regards to f_{tt}, for short trips it is likely that the original users have already reduced the travel time as much as possible by themselves, while for longer trips the system can help in finding shorter trips not considered by the users. Similarly, relatively to f_{sr}, as the amount of available time increases, the insertion of more scenic routes into a trip becomes easier, thus improving the trip choice.

Finally, Table 4 shows some additional statistics. In particular, it groups the suggested trips by the number of POIs they contain, reporting the average duration, the average travel time and the average travel distance for each group. Notice that, comparing these statistics with the content of Table 1, which contains the same values computed on the original trips, all the quantities decrease in Table 4 considering group o trips with the same number of POIs; moreover, we can see that the suggested trips contain more POIs with respect to the original ones with the same or similar duration. Thus confirming the improvement obtained by the recommendation system.

TABLE 2
Statistics about the recommendations produced with a 50% of registered users. For each starting point we show the number of available trips, the size of the initial Pareto-front, the percentage of improvement of each objective function w.r.t. its original trip, the percentage of cases in which at least two objective functions are improved.

p_s	$\#\tau$	$\lvert\mathcal{F}\rvert$	f_n	f_d	f_{tt}	f_{td}	f_{sr}	f_{sm}	$\%2f_{\text{impr}}$
p_s^1	124,800	11,300	99%	71%	95%	89%	15%	58%	99%
p_s^2	28,000	3,930	99%	48%	94%	86%	14%	34%	97%
p_s^3	21,175	3,359	99%	38%	91%	82%	11%	35%	93%

TABLE 3
Statistics about the recommendations produced with a 50% of registered users. In this case the original trips are not grouped by the starting points as in Table 2, but by considering their initial total time (trip duration). In particular, the first column denotes the considered intervals of duration (in hours).

λ_τ	$\#\tau$	$\lvert\mathcal{F}\rvert$	f_n	f_d	f_{tt}	f_{td}	f_{sr}	f_{sm}	$\%2f_{\text{impr}}$
< 6	59,262	6,332	99%	22%	90%	80%	10%	29%	93%
$6 - 8$	44,971	4,805	99%	50%	97%	94%	15%	54%	99%
> 8	69,742	7,452	99%	99%	96%	89%	18%	53%	97%

TABLE 4
Statistics about the recommendations produced with a 50% of registered users. The columns report: the number of visited POIs, the number of trips with such number of POIs, the average duration of the trip (hour:min), the average travel time, and the average travel distance.

$\lvert\tau\rvert$	# trips	λ_τ	avg trav. time	avg trav. dist.
4	18,643	03:58	00:14	948m
5	40,619	04:16	00:23	1,5Km
6	23,492	06:10	00:31	2,1Km
7	21,479	07:20	00:42	2,8Km
8	14,942	08:43	00:48	3,2Km
9	19,235	10:19	00:59	3,9Km
10	13,326	11:37	01:05	4,3Km
11	12,694	12:07	01:12	4,7Km
12	9,5454	12:14	01:13	4,9Km

5.4 Sensitivity analysis

Variability in the number of registered users. The previous experiments consider a percentage of registered users equal to 50%. In order to evaluate the effects of the number of registered users w.r.t. the number of anonymous ones, we evaluate the various test queries considering different percentage of $\text{ETO}_{\text{reg}}(d, h)$. Fig. 16 illustrates the number of visitors over time for POI called "Casa di Giulietta" on February 14th, 2015, obtained by considering different percentages of registered users. As you can notice, as such percentage increases, the quality of the system improves. When the percentage becomes equal to 30% the dynamic component has limited effects and some pick hours remain, while a percentage equal to 50% is enough to obtain a good result and increasing such percentage means improving the balancing effect. More specifically, considering the relative standard deviation (RSD) as a measure of the variability in the POI occupancy, with a percentage of registered users equal to 30%, the RSD is 64%, while it becomes 47% with a 50% of registered users and finally it is only 25% with a 70% of registered users.

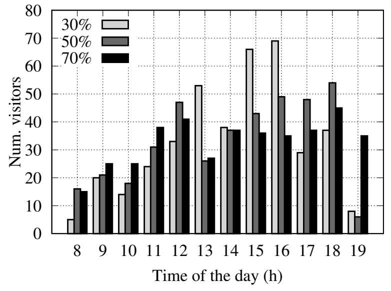


Fig. 16. Number of visits at "Casa di Giulietta" considering different percentages of registered users.

Variability in user behaviour. In the previous experiments, we assume that all registered users follow the recommendations provided by the system, namely they visit all the prescribed POIs spending the estimated visiting and travelling time. In order to evaluate the robustness of the system in presence of variability in users' behaviour, we consider the case in which some delays are collected during the POI visits. We then plot the user presence in the POIs to see if new peaks may appear.

In particular, we assume that a percentage of the registered users accumulates a delay uniformly distributed between zero and one hour during their day. Fig. 17 illustrates the number of visitors over time for POI called "Casa di Giulietta" on February 14th, 2015, obtained by considering a 60% of registered users and varying the percentage of them that collect a random delay in the recommended trip. The results confirms the robustness of the technique to some modifications in the users' behaviour (e.g., 30%), some differences with respect to the reference case are visible only when the number of registered users that does not follow the suggestions are bigger than 50-60%. In particular, considering again the RSD on the POI occupancy, it was equal to 31% for the original case when all registered users follows the suggestions (reference), it becomes 40% when a 30% of registered users accumulate some dalays and it is 50% when the percentage of error is 70%.

Incentives for limiting the variability. The variability in the number of registered users and in user behaviour may have

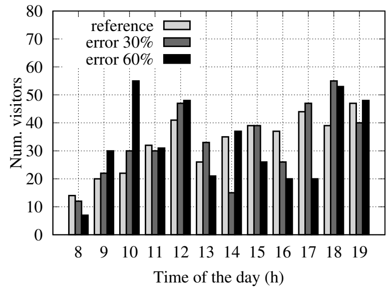


Fig. 17. Number of visits at "Casa di Giulietta" considering different percentages of delay collected during the visits of each POI in the recommended trips.

an impact on the quality of the recommendations. In order to limit such variability, it would be interesting to explore the benefits that a set of incentives might introduce. The design of the recommendation application, in fact, may consider the "gamification" approach [42]: the tourists might receive some gains (e.g., discounts at the shops of the PoIs) if they use the recommendation system – i.e., they become registered users – and if they follow the recommended trip. Since this approach is complementary to our proposed system, we will study its impact in our future work.

6 Conclusion and Future Work

Personalized trip recommendation systems tailor the suggestions to the users based on their constraints and requirements. Nevertheless, they do not consider the impact of the recommendations on the level of crowding of the POIs they recommend to visit. In this paper, we took a step to fill this gap. We proposed a system that efficiently searches the solution space through a MapReduce implementation of the multi-objects optimization problem and balances the users among the POIs by including the predicted level of crowding. We evaluate our implementation using a real dataset, showing consistent improvements over the paths usually autonomously chosen by the tourists.

We also evaluate the robustness of the proposed method w.r.t. to both variability in the number of registered users, i.e., the percentage of users that follows the recommendation, and in the user behaviour, considering possible changes in the suggested paths. Our road-map includes a more extensive evaluation of the impact that errors may have on the predictions of the level of crowding, and the corresponding quality of the recommendations.

References

[1] Y. Yu and X. Chen, "A survey of point-of-interest recommendation in location-based social networks," in *Workshops at the 29th AAAI Conference on Artificial Intelligence*, 2015.

[2] X. Wang, C. Leckie, J. Chan, K. Lim, and T. Vaithianathan, "Improving personalized trip recommendation by avoiding crowds," in *Proc. of ACM CIKM*, 2016, pp. 25–34.

[3] E.-C. Lu, C.-Y. Chen, and V. Tseng, "Personalized trip recommendation with multiple constraints by mining user check-in behaviors," in *Proc. of ACM SIGSPATIAL*, 2012, pp. 209–218.

[4] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. Thalmann, "Time-aware point-of-interest recommendation," in *Proc. of ACM SIGIR*, 2013, pp. 363–372.

[5] K. Varadhan, R. Govindan, and D. Estrin, "Persistent route oscillations in inter-domain routing," *Computer networks*, vol. 32, no. 1, pp. 1–16, 2000.

[6] S. Migliorini, D. Carra, and A. Belussi, "Adaptive trip recommendation system: Balancing travelers among POIs with mapreduce," in *IEEE International Congress on Big Data*, 2018, pp. 255–259.

[7] S. Zhao, I. King, and M. Lyu, "A survey of point-of-interest recommendation in location-based social networks," *arXiv preprint arXiv:1607.00647*, 2016.

[8] E. Cho, S. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proc. of ACM SIGKDD*, 2011, pp. 1082–1090.

[9] H. Gao, J. Tang, X. Hu, and H. Liu, "Exploring temporal effects for location recommendation on location-based social networks," in *Proc. of ACM Conf. on Recommender Systems*, 2013, pp. 93–100.

[10] C. Chen, D. Zhang, B. Guo, X. Ma, G. Pan, and Z. Wu, "TripPlanner: Personalized trip planning leveraging heterogeneous crowdsourced digital footprints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1259–1273, 2015.

[11] H. Gao, J. Tang, X. Hu, and H. Liu, "Content-aware point of interest recommendation on location-based social networks," in *AAAI*, 2015, pp. 1721–1727.

[12] Y. Zheng and X. Xie, "Learning travel recommendations from user-generated GPS traces," *ACM Trans. on Intelligent Systems and Technology*, vol. 2, no. 1, p. 2, 2011.

[13] H. Yoon, Y. Zheng, X. Xie, and W. Woo, "Smart itinerary recommendation based on user-generated gps trajectories," in *Intern. Conf. on Ubiquitous Intelligence and Computing*, 2010, pp. 19–34.

[14] Y. Liu, W. Wei, A. Sun, and C. Miao, "Exploiting geographical neighborhood characteristics for location recommendation," in *Proc. of ACM CIKM*, 2014, pp. 739–748.

[15] J.-D. Zhang and C.-Y. Chow, "iGSLR: personalized geo-social location recommendation: a kernel density estimation approach," in *Proc. of ACM SIGSPATIAL*, 2013, pp. 334–343.

[16] M. Ye, P. Yin, and W.-C. Lee, "Location recommendation for location-based social networks," in *Proc. of ACM SIGSPATIAL*, 2010, pp. 458–461.

[17] G. Adomavicius and Y. Kwon, "Multi-criteria recommender systems," in *Recommender Systems Handbook*. Springer, 2015, pp. 847–880.

[18] J.-D. Zhang and C.-Y. Chow, "TICRec: A probabilistic framework to utilize temporal influence correlations for time-aware location recommendations," *IEEE Trans. on Services Computing*, vol. 9, no. 4, pp. 633–646, 2016.

[19] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M. Damiani, A. Gkoulalas-Divanis, and et al., "Semantic trajectories modeling and analysis," *ACM Computing Surveys (CSUR)*, vol. 45, no. 4, p. 42, 2013.

[20] Y. Yoshimura, S. Sobolevsky, C. Ratti, F. Girardin, J. P. Carrascal, J. Blat, and R. Sinatra, "An analysis of visitors' behavior in the louvre museum: A study using bluetooth data," *Environment and Planning B: Planning and Design*, vol. 41, no. 6, pp. 1113–1131, 2014.

[21] I. Lykourentzou, X. Claude, Y. Naudet, E. Tobias, A. Antoniou, G. Lepouras, and C. Vassilakis, "Improving museum visitors' quality of experience through intelligent recommendations: A visiting style-based approach." in *Intelligent Environments (Workshops)*, 2013, pp. 507–518.

[22] H. Hoos and T. Stützle, *Stochastic local search: Foundations and applications*. Elsevier, 2004.

[23] E. Onbaşoğlu and L. Özdamar, "Parallel simulated annealing algorithms in global optimization," *Journal of Global Optimization*, vol. 19, no. 1, pp. 27–50, 2001.

[24] Z. Czech and P. Czarnas, "Parallel simulated annealing for the vehicle routing problem with time windows," in *Proc. of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, 2002, pp. 376–383.

[25] A. Radenski, "Distributed simulated annealing with mapreduce," in *European Conf. on the Applications of Evolutionary Computation*. Springer, 2012, pp. 466–476.

[26] H. Li and C. Liu, "Prediction of protein structures using a mapreduce hadoop framework based simulated annealing algorithm," in *IEEE Int. Conf. on Bioinformatics and Biomedicine*, 2013, pp. 6–10.

[27] B. Suman and P. Kumar, "A survey of simulated annealing as a tool for single and multiobjective optimization," *Journal of the operational research society*, vol. 57, no. 10, pp. 1143–1160, 2006.

[28] B. Suman, "Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem," *Computers & Chemical Engineering*, vol. 28, no. 9, pp. 1849–1871, 2004.

[29] P. McMullen and G. Frazier, "Using simulated annealing to solve a multiobjective assembly line balancing problem with parallel workstations," *Intern. Journal of Production Research*, vol. 36, no. 10, pp. 2717–2741, 1998.

[30] D. Quercia, R. Schifanella, and L. M. Aiello, "The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city," in *Proceedings of the 25th ACM conference on Hypertext and social media*. ACM, 2014, pp. 116–125.

[31] P. K. Tripathi, M. Debnath, and R. Elmasri, "A direction based framework for trajectory data analysis," in *Proc. of ACM PETRA*, 2016, pp. 1:1–1:8.

[32] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of massive datasets*. Cambridge university press, 2014.

[33] S. Moens, E. Aksehirli, and B. Goethals, "Frequent itemset mining for big data." in *BigData Conference*, 2013, pp. 111–118.

[34] A. Belussi, C. Combi, and G. Pozzani, "Towards a formal framework for spatio-temporal granularities," in *International Symposium on Temporal Representation and Reasoning*, 2008, pp. 49–53.

[35] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. New York, NY, USA: John Wiley & Sons, Inc., 1989.

[36] Y. Nourani and B. Andresen, "A comparison of simulated annealing cooling strategies," *Journal of Physics A: Mathematical and General*, vol. 31, no. 41, p. 8373, 1998.

[37] A. Eldawy, "SpatialHadoop: Towards Flexible and Scalable Spatial Processing Using Mapreduce," in *Proc. of the 2014 SIGMOD PhD Symposium*, 2014, pp. 46–50.

[38] T. White, *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 2015.

[39] S. Migliorini, A. Belussi, M. Negri, and G. Pelagatti, "Towards massive spatial data validation with SpatialHadoop," in *Proc. of the 5th ACM SIGSPATIAL Intern. Workshop on Analytics for Big Geospatial Data*, 2016, pp. 18–27.

[40] A. Belussi, D. Carra, S. Migliorini, M. Negri, and G. Pelagatti, "What Makes Spatial Data Big? A Discussion on How to Partition Spatial Data," in *GIScience*, 2018.

[41] A. Belussi and S. Migliorini, "A framework for managing temporal dimensions in archaeological data," in *International Workshop on Temporal Representation and Reasoning*, 2014, pp. 81–90.

[42] F. Xu, D. Buhalis, and J. Weber, "Serious games and the gamification of tourism," *Tourism Management*, vol. 60, pp. 244–256, 2017.

Alberto Belussi received the Master degree in electronic engineering from the Politecnico di Milano in 1992 and the PhD degree in computer engineering from the same university in 1996. Since 1998 he has been working at University of Verona (Italy), where from 2004 he is Associate Professor. His main research interests include: conceptual modeling of spatial databases, geographical information systems, spatial data integration and spatial query optimization.

Damiano Carra received his Laurea in Telecommunication Engineering from Politecnico di Milano, and his Ph.D. in Computer Science from University of Trento. He is currently an Associate Professor in the Computer Science Department at University of Verona. His research interests include modeling and performance evaluation of large scale distributed systems.

Sara Migliorini received the Master degree in Computer Science from the University of Verona in 2007 and the PhD degree in computer science from the same university in 2012. From 2012 she is a post-doc research associate at University of Verona. Her main research interests include: geographic information systems, big data and collaborative and distributed architectures.