Inf Technol Tourism (2017) 17:275–314
DOI 10.1007/s40558-017-0080-9

ORIGINAL RESEARCH

Planning the trip itinerary for tourist groups

Kadri Sylejmani[1] · Jürgen Dorn[2] · Nysret Musliu[3]

Received: 12 October 2016 / Revised: 14 February 2017 / Accepted: 21 February 2017 /
Published online: 6 March 2017

Abstract Sightseeing trips are often done in groups, where tourists enjoy their trip in company with their relatives or friends. Therefore, in this paper, in order to model the case of trips for tourist groups, we introduce a new problem, as an extension of the existing problem in the literature that is used for planning the trip of a single tourist. The new problem extends the existing problem with two additional concepts. The first is the consideration of multiple tourists, where their individual preferences about points of interests are taken into account, and the second is the introduction of the concept of mutual social relationship between the different tourists. For the actual single tourist trip problem, we use an algorithm that obtains comparable results with the state of the art algorithms, whereas for the group trip problem, since no solution has been published before, we design a new algorithm based on tabu search metaheuristic that uses two new unique operators for exploring the search space. As a result, this paper proposes an anytime algorithm that in average takes about 20 s to obtain better personalized itineraries for tourist groups than when scheduling the whole group together.

✉ Kadri Sylejmani
kadri.sylejmani@uni-pr.edu

Jürgen Dorn
juergen.dorn@ec.tuwien.ac.at

Nysret Musliu
musliu@dbai.tuwien.ac.at

[1] Faculty of Electrical and Computer Engineering, University of Prishtina, Rr. "George Bush", p.n., Prishtina 10000, Republic of Kosovo

[2] E-Commerce Group, Institute of Software Technology and Interactive Systems (188), Faculty of Informatics, Vienna University of Technology, Favoritenstrae 9-11, 1040 Vienna, Austria

[3] Database and Artificial Intelligence Group, Faculty of Informatics, Vienna University of Technology, Favoritenstrae 9-11, 1040 Vienna, Austria

Keywords Trip itinerary · Tourist groups · Planning · Tabu search

1 Introduction

Tourist trip itinerary mainly includes a plan with a sequence of visits scheduled to a number of points of interest (POI), which must be finished during a limited trip duration. Additional information might be provided such as the total cost of the trip, details for each visit (e.g. POI description, start and end time, cost etc.), total unused time, orientation on the map, etc. For some selected tourism destinations, tourist trip itinerary planning could be performed automatically by using expert systems such as those introduced in Tumas and Ricci (2009), Vansteenwegen et al. (2011a), Schaller (2011), Kurata and Hara (2013), Brilhante et al. (2015), Gavalas et al. (2015) and Gavalas et al. (2017).

Tourist trip planning systems are expected to react in real time, regardless of their form of implementation (e.g. app on a hand-held device, online system or desktop application). Such systems are called by different names such as Personalized Electronic Tourist Guide (PET), Mobile Tourist Guide (MTG), Personal Navigation Systems for tourism (PNS) and Electronic Tourist Guides (ETG) (Vansteenwegen 2008). Planning a trip for a single tourist is mostly modeled as a Tourist Trip Design Problem (TTDP) (Souffriau et al. 2008). An extensive review of the functionalities of the state of the art systems in tourism is presented in Souffriau and Vansteenwegen (2010). In addition, an extended review of mobile recommender systems in tourism is made by Gavalas et al. (2014a). Further, Masthoff (2015) discusses group recommender system in the context of aggregation of user information with the aim of finding better recommendation strategies for individual users.

In general, in terms of dealing with the personal preferences of multiple tourists, there exist two opposing approaches for planning the trip for tourist groups. The first approach could be that, during the pre trip phase, all tourists make some kind of negotiation process to agree in visiting common POIs Delic et al. (2016), and afterwards they all get set for a joint trip throughout its duration. Whereas, on a second approach, all tourists stick to their personal preferences, so that during the course of the trip, they may separate at certain locations to visit the POIs of their specific interest. The second approach is a more complex optimisation problem, but it satisfies more the personal preferences of the tourists. Therefore, in this paper, we introduce a new problem that can be used to model the scenario of trip itinerary planning for tourist groups, which is based on the existing mathematical models from the literature Souffriau et al. (2013) that are used to model the situation of trip planning for a single tourist. In addition, we solve the new problem via a tabu search algorithm and present four different approaches of trip itinerary planning, namely "Solo" (individual tourists conduct the trip alone), "Subgroups" (tourists with common preferences and mutual social relations travel together), "All together" (all tourists travel together) and "Combined" (tourists at times are together and at some other times are separated). Further, we compare different approaches of trip planning on newly created benchmark instances that model group trip planning problem.

The remaining part of this paper is structured as in the following. Section 2 presents the literature review of the state of the art algorithms in the envisioned filed. In Sect. 3, we give a mathematical formulation of the problem, and then, in Sect. 4, we present a scenario of a tourist group, for which the trip itinerary will be prepared. In Sect. 5, we describe our solution approach, whereas, in Sect. 6, we present the experimental results. Our conclusions and ideas for further work are outlined in Sect. 7.

2 State of the art

The Orienteering Problem (OP) (Tsiligirides 1984) can be used to model the simplest variant of TTDP, which is made of a single tour that is itself limited to a specified duration. In OP, a number of locations (i.e. POIs) are specified along with a starting and an ending point. Each location has a score, which, in TTDP context, represents the level of tourist interest for visiting it. In order to maximize the total collected score (i.e. tourist satisfaction), the objective is to visit, in a limited period of time, a subset of these locations, starting from the start point and ending at the end point.

The Team Orienteering Problem (TOP) (Chao et al. 1996) enables modeling TTDP with multiple tours (days), whereas TOP with Time Windows (TOPTW) (Vansteenwegen et al. 2009) is used to model opening hours of POIs. The Time-Dependent OP (TDOP) Fomin and Lingas (2002) employs the dependency concept within the travelling times between POIs, thus being suitable for modelling situations of multi-modal transportation. In addition, TDOP with Time Windows (TDOPTW) Zenker and Ludwig (2009) makes it possible to consider operating hours of POIs. Further, the Multi Constraint TOPTW (MCTOPTW) (Garcia et al. 2009), allows for inclusion of additional knapsack constraints, such as maximum tour budget or maximum number of POIs of a certain category or type (e.g. maximum three architectural POIs, maximum four mosques, etc.). In some occasions a POI is characterized by a number of features (e.g. beauty, cultural background, historical relevance, etc.) and as a result a range of scores for each of the features is associated with that POI. This situation can modeled by using Generalized Orienteering Problem (Ramesh and Brown 1991), which allows modeling multiple scores for each POI.

In Vansteenwegen et al. (2009) authors tackle TOPTW by developing their approach based on the Iterated Local Search (ILS) framework. The algorithm combines an insert step with a shake step to escape from local optima. The insert step adds new points into the tours in consecutive manner. The insertion feasibility is evaluated with a quick mechanism that records two values for each already included point, namely *Wait* and *MaxShift*. The *Wait* value represents the waiting time at a point, in case the arrival occurs before the opening of time window, whereas the *MaxShift* value indicates the maximum possible shift of a visit to a point without making other visits into the tour infeasible. The shake step removes one or more visits from each tour. The place of removal and the number of consecutive visits to be removed are determined by two controlling parameters,

namely R_d respectively S_d. Due to different tour lengths, the value of S_d is different for different tours. This makes the shake step more effective in escaping from local optima. Later, Garcia et al. 2009 extend ILS algorithm of Vansteenwegen et al. (2009) to solve the MCTOPTW, while Souffriau et al. (2013) hybridize ILS with a greedy Randomized Adaptive Search Procedure to solve MCTOP with Multiple TW (MCTOPMTW). A complete survey of most efficient algorithms for OP and its extensions, is presented in Vansteenwegen et al. (2011b), where various meta-heuristics are discussed in terms of solution quality and computation time.

In Gavalas et al. (2013) authors tackle TOPTW by using two related approaches from the group of cluster-based heuristics called Cluster Search Cluster Ratio (CSCRatio) and Cluster Search Cluster Routes (CSCRoutes). The presented approaches aim at encouraging visits to topology areas that consist of a high density of "good" points. Both heuristics employ a clustering procedure to organize points into clusters based on topological distance. This increases the probability that more visits would take place inside individual clusters, and as a result, two positive effects could arise, the decrease of duration of tours and minimization of the number of transfers between different clusters. Both CSCRatio and CSCRoutes employ the global *k-means* algorithm (Likas et al. 2003) to build the clusters of points. In the initialization phase, in order to start from diverse positions in the search space, the k tours of the solution are constructed by using points from different clusters. Further, in the local search phase, both approaches use *insert* and *shake* operators that are originated by Vansteenwegen et al. (2009) for exploring the neighborhood of current solutions. CSCRatio heuristic is designed to favor tours with more points inside individual clusters, whereas CSCRoutes heuristic is designed to construct tours that visit each cluster at most once. This approach improves in terms of quality in comparison to ILS (Vansteenwegen et al. 2009), while keeping the computation time in comparable level. An extended review of algorithmic approaches for solving tourist trip design problems can be found in the paper authored by Gavalas et al. (2014b).

To the best of the authors knowledge, there is no any publication that describes or solves trip planning problem for the scenarios of tourist groups. Hence, we extend the existing MCTOPTW problem to MC Multiple TOPTW (MCMTOPTW) problem to model the case of multiple trips with multiple tours, so that it emulates the problem of trip planning for tourist groups. Consequently, the existing test instances of the MCTOPTW problem are extended with additional attributes to model the trip planning problem for tourist groups (more details are given in Sect. 6).

Table 1 presents the above described models, along with the newly proposed model for tourist groups, in a summarized view, where, for each model, the following details are outlined: time dependency (TD), number of tours (M), number of time windows (TW), number of knapsack constraints (MC) and number of tourists. Further, every model, depending on the number of tours and tourists it uses, is tagged with one of the three tourist trip names, such as *tourist trip*, *tourist trips* or *tourist group trips*, which are used interchangeably to refer to such models in the rest of this paper.

Table 1 Possible theoretical models for tourist trip planning

Model abbreviation	TD	Number of				Trip naming
		M	TW	MC	T	
OP	No	1	0	0	1	Tourist trip
TDOP	Yes	1	0	0	1	
TDOPTW	Yes	1	1	0	1	
TOP	No	>1	0	0	1	Tourist trips
TOPTW	No	>1	1	0	1	
MCTOPTW	No	>1	1	>1	1	
MCTOPMTW	No	>1	>1	>1	1	
MCMTOPTW	No	>1	1	>1	>1	Tourist group trips

3 Mathematical modeling

The mathematical definition of the new proposed tourist group trips problem represents an extension to the existing tourist trips problem, which is formulated in Garcia et al. (2009). The tourist group trips problem is formulated as a mixed-integer linear problem by using the following decision variables: $x_{ijmp} = 1$ if a visit to point i is followed with a visit to point j in tour m of person p, 0 otherwise; $y_{pnm} = 1$ if person p visits point n in tour m, 0 otherwise; $z_{pqnm} = 1$ if person p and q visit together point n in tour m, 0 otherwise; $s_{nmp} =$ the start time of the visit to point n in tour m of person p. In tourist group trips problem, a number of P persons make a multiple day trip to a geographic area. The visited area has N points, where each of them is characterized with its coordinates x_n and y_n, visiting duration T_n, opening O_n and closing time C_n, entry fee b_n and an array of Z category attributes e_{nz} that specify the different categories the point belongs to. The details known for each person p include: maximum budget B_p, satisfaction factor with each point S_{pn}, social relationship factor R_{pq} with each other person q in the group, and a range Z of E_{pz} values that specify the maximum number of points of each point category z to be visited. Each person p conducts exactly M tours that have the same duration T_{max}. A person can visit a point at most once. The starting point 1 and ending point N are fixed for each tour. The traveling times t_{ij} between points i and j are known for all points. The objective is to prepare a multiple day trip itinerary for all persons so that their overall satisfaction, both with visited points and with each other's company, is maximized. Next, we show the expressions that formulate the objective function and define specific hard constraints that are associated with tourist group trips problem.

$$Max \sum_{p=1}^{P} \sum_{m=1}^{M} \left(\sum_{n=2}^{N-1} S_{pn} y_{pnm} + \sum_{\substack{q=1 \\ q \neq p}}^{P} R_{pq} z_{pqnm} \right), \tag{1}$$

$$\sum_{j=2}^{N} x_{1jmp} = \sum_{i=1}^{N-1} x_{iNmp} = 1;\ \forall m = 1,\ldots,M;\ \forall p = 1,\ldots,P, \tag{2}$$

$$\sum_{i=1}^{N-1} x_{ikmp} = \sum_{j=2}^{N} x_{kjmp} = y_{pkm};\ \forall k = 2,\ldots,N-1;\ \forall m = 1,\ldots,M;\ \forall p = 1,\ldots,P, \tag{3}$$

$$s_{imp} + t_{ij} - s_{jmp} \le Q(1 - x_{ijmp});\ \forall i,j = 1,\ldots,N;\ \forall m = 1,\ldots,M;\ \forall p = 1,\ldots,P, \tag{4}$$

$$\sum_{m=1}^{M} y_{pnm} \le 1;\ \forall p = 1,\ldots,P;\ \forall n = 2,\ldots,N-1;\ \forall m = 1,\ldots,M, \tag{5}$$

$$\sum_{n=1}^{N} e_{nz} y_{pnm} \le E_{pz};\ \forall p = 1,\ldots,P;\ \forall m = 1,\ldots,M;\ \forall z = 1,\ldots,Z, \tag{6}$$

$$O_i \le s_{nmp} \le C_i;\ \forall n = 1,\ldots,N;\ \forall m = 1,\ldots,M;\ \forall p = 1,\ldots,P, \tag{7}$$

$$\sum_{i=1}^{N-1} \left(T_i y_{imp} + \sum_{j=2}^{N} t_{ij} x_{ijmp} \right) \le T_{max};\ \forall m = 1,\ldots,M;\ \forall p = 1,\ldots,P. \tag{8}$$

Equation (1) expresses the objective function of the problem, which is the maximization of the overall tourists' satisfaction, both in terms of satisfaction with points, by considering each and every tourists' satisfaction with POIs, and with each other's company, by considering tourists' mutual social relationship factor. Constraint (2) enforces the start of each tour at point *1* and the end of it at point *N*, whereas, in practice, the starting and the ending point could be the same (e.g. the hotel). Constraint (3) enforces the continuity of a tour by making sure that any visit to a point is followed by another visit to some other point, so that no break up occurs in any of the tours. Constraint (4) makes sure that consecutive visits are aligned in a timely manner so that no overlapping occurs between consecutive visits and traveling times between them. Coefficient Q in Constraint (4) represents a constant. Constraint (5) makes sure that any person could visit a particular point at most one time, whereas Constraint (6) limits the number of points of certain point category that could be visited by a person to the given maximal value E_{pz}. Constraint (7)

forces the start of a visit to a point to occur only during its respective time windows. Constraint (8) limits the duration of any tour to the maximal specified value T_{max}.

4 Scenario description

In order to didactically describe the problem at hand, we present a simple scenario of a tourist group, which we later use to explain the applicability of the proposed approach in this paper. The scenario consists of a group of three friends, two women (T1 and T2) and one man (T3), who make a trip to a given city *X*, which has 10 POIs, as described in Tables 2 and 3). The time is expressed in units of minutes, where the earliest time is 8:00 am, which is equivalent to 0 min. The given POIs might belong to one or more of the five attraction categories, namely archeology (C1), architecture (C2), religious art (C3), nature (C4) and shopping center (C5). Table 4 presents the budget tourists are willing to spend, as well as their mutual social relationship that is expressed in Likert scale from 0 to 5. They have a close relationship between each other and mostly share common interests for visiting POIs that belong to the category of architecture (C2) and nature (C4) (as given in Table 5), while each of them sets a limit about the maximum number of POIs of certain categories she/he is willing to visit (as presented in Table 6). In addition, the man is also interested in visiting POIs of religious art (C3), therefore he would not like to visit more than one POI of the category of architecture (C2). They jointly decide to have a trip of two days and make sightseeing visits that start from 9 am until 11:30 am on each day.

5 Solution approach

In general terms, the overall process for planning the trip for a group of tourists, can be described by outlining three major algorithms. The first algorithm (denoted as *single trip planner*) is about planning the trip by considering only the preferences of

Table 2 POI details

Point	Entry fee	Open time	Close time	POI category				
				C1	C2	C3	C4	C5
P1	5	0	90	1	0	0	0	0
P2	10	0	145	1	0	0	0	0
P3	7	30	120	0	1	0	0	0
P4	9	0	145	0	1	0	0	0
P5	15	0	120	0	0	1	0	0
P6	10	0	145	0	0	1	0	0
P7	12	0	120	0	0	0	1	0
P8	5	0	130	0	0	0	1	0
P9	6	40	200	0	0	0	0	1
P10	8	0	80	0	0	0	0	1

Table 3 Distances between POIs (in unit of minutes)

Point	Start-end point	Distance									
		P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
P1	10	–	15	18	25	30	25	42	32	45	35
P2	30	15	–	8	12	20	24	18	30	24	26
P3	18	18	8	–	16	23	10	28	30	18	28
P4	30	25	12	16	–	30	19	15	35	15	44
P5	25	30	20	23	14	–	13	28	19	15	14
P6	12	25	24	10	10	13	–	33	16	26	12
P7	16	42	18	28	27	38	33	–	17	5	45
P8	22	32	30	30	25	19	16	17	–	9	20
P9	10	34	24	18	27	15	13	5	9	–	24
P10	30	18	26	28	22	14	12	45	20	24	–

Table 4 Budget and tourists' relationship

Tourist	Budget	Social relation		
		T1	T2	T3
T1	50	–	4	5
T2	40	5	–	5
T3	45	5	4	–

Table 5 Satisfaction factors with POIs

Tourist	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
T1	0	0	40	45	0	0	40	40	0	0
T2	0	0	40	45	0	0	40	40	0	0
T3	0	0	50	50	0	0	40	40	20	45

Table 6 Maximal number of points allowed per POI category

Tourist	Categories of POIs				
	C1	C2	C3	C4	C5
T1	3	2	3	2	1
T2	3	2	3	2	1
T3	3	1	0	3	2

a single tourist, whereas the second algorithm (denoted as *tourist group builder*) clusters the tourists in smaller groups by considering their personal tourism preferences and their mutual social relationship. The third algorithm (denoted as *group trip planner*) uses the *single trip planner* and the *tourist group builder*) to plan the trip for a group of tourists by taking into account preferences of multiple

tourists. In the following subsections, a more detailed description is given for the individual algorithms.

5.1 Single trip planner

In the tourism domain, the tourist information systems are expected to react to user requests in real time. Hence, when it comes to planning the itinerary for tourists, the proposed algorithm should prepare it within a short period of time, which is usually some tens of seconds. In such circumstances, meta heuristic approaches are often applied in the literature, while in this paper, the solution approach is based on Tabu Search (TS) meta-heuristic, which is a local search technique that is used for solving highly constrained combinatorial problems. TS was initially proposed by Glover and McMillan in Glover and McMillan (1986) and then further formalized by Glover in Glover (1989a) and Glover (1989b). The main idea behind TS stands in storing information about the previous search experience and then, based on that information, making decisions to either intensify the search process towards some promising search space areas, or diversify the process towards less explored regions of the search space. The search experience is stored in the so called tabu memories, which keep information about moves that are forbidden or tabu for a certain period of time (i.e. number of iterations).

We developed the *single trip planner* Sylejmani et al. (2012) as part of the solution to a state of the art tourist trips problem, which can be used to model situations where the trip itinerary is prepared when considering the preferences of a tourist alone. In the following, we shall give a summarized description of the single trip planing algorithm.

The solution to this problem is represented as a sequence of numbers, where individual members represent the indexes of POIs that are scheduled to be part of the itinerary. The process of initialization is done by randomly inserting POIs into the sequence, until there is spare time within any of the tours in the trip, and subject to enforcing the corresponding limitations regarding the individual POIs and the tourist himself.

The quality of a given solution is measured by summing up the respective satisfaction factors of the POIs residing within the corresponding itinerary. Meanwhile, the neighbourhood exploration is done by using three different operators, as described in the following:

- *Insert* which inserts a new POI into the itinerary from the group of POIs that are currently left outside,
- *Replace* which replaces one POI from the itinerary with a POI off the itinerary
- *Swap* which swaps two POIs within the itinerary regardless whether they are in the same tour or in different tours

Algorithm 1 Pseudocode of single trip planner

Require: Problem instance I, TLS, MI, DOF, PF, RIF and PEF
Ensure: Best found solution S_b

```
OperatorList ← {Replace, Insert, Swap}
S_c ← Generate initial solution
Evaluate S_c
S_b ← S_c
i ← 1
j ← 0
while i ≤ MI do
    CurrentOperator ← Select operator
    S_c ← Apply CurrentOperator in S_c
    if S_c better then S_b then
        S_b ← S_c
        j ← 0
    else
        j ← +1
    end if
    if j equals DOF then
        S_c ← Apply DeleteOperator in S_c
    end if
    if j equals PF then
        S_c ← Apply PerturbateOperator in S_c
    end if
    if j equals RIF then
        S_c ← Generate initial solution
    end if
    if j equals PEF then
        Penalize frequently used moves
    end if
    if j equals 0.3 * MI then
        Quit search
    end if
    i ← +1
end while
```

The fine tuning of the algorithm is facilitated by means of six parameters, which are presented in Table 7, where the first column presents the parameter abbreviation, the second one its whole name, while the third one presents a short description of the parameter.

At the start, as presented in Algorithm 1, the procedure that creates a random initial solution is executed and tabu memories are initialized to an empty state. After the evaluation of initial solution, the algorithm enters the loop that is controlled by *MI* parameter. The neighbourhood exploration mechanism alternates between the above described operators in such way that in every second iteration, the algorithm

Table 7 Single trip planner parameters

Abbreviation	Name	Description
TLS	Tabu list size	Specifies the number of iterations a certain move cannot be used
MI	Maximum iterations	Defines the number of iterations the algorithm shall run
DOF	Delete operator frequency	Determines how often the delete operator needs to be applied
PF	Perturbation frequency	Sets the frequency of perturbation from actual best found solution
RIF	Random initialization frequency	Indicates the frequency of re-initialization from random solution
PEF	Penalization frequency	Defines the frequency of penalization of frequently used moves

uses the *Replace* operator, and then in between, in turn, it uses either *Insert* or *Swap* operator. In a single iteration, the best tabu and non-tabu solution is obtained among all possible combinations that are generated as a result of applying the current running operator. The best non tabu solution is considered first for adoption as a new current solution. If the non-tabu solution is not better than the current solution, then the best tabu solution is tested whether it fulfils the aspiration criteria. In the implementation at hand, a tabu solution is considered to be fulfilling the aspiration criteria only if it is better than the best found solution so far. Normally, the adoption of a better solution is done whenever a new, better one, is found, otherwise the number of iterations without improvement is recorded. In occasional iterations, when the number of running iterations is equal to one of the values of control parameters (*DOF, PF, RIF* and *PEF*), the process of search diversification is applied by using the respective operator (*Delete, Perturbate, Restart* and *Penalise*) accordingly.

The algorithm quits either when it reaches the maximum iterations allowed, or when the number of iterations without improvement becomes greater than *30%* of the value of *MI* parameter. For a detailed description of the algorithm and its associated computational experiments, the reader is referred to the original paper presenting the approach (Sylejmani et al. 2012).

5.2 Tourist group builder

The *k-means algorithm* is a rather simple approach for organizing a number of objects of a dataset into k groups. This algorithm has been introduced into several disciplines by many authors, most remarkably by Lloyd in Lloyd (1957) and Lloyd (1982).

The algorithm works on a dataset of d-dimensional vectors, $D = \{x_i | i = 1\ N\}$, where x_i represents the ith data point. It starts by picking k points in the dataset as initial group representatives also called "centroids", where the methods used to select them vary from random to more sophisticated ones. Then, the algorithm

alternates between two steps until it converges. The first step includes assignment of each data point to the closest centroid, which results into portioning of the dataset. The second step, deals with relocation of the centroids, by calculating a new central point for the data points assigned to individual groups. The algorithm converges when no more data points change their subgroups. The calculation of distance between two different data points is an issue to resolve and subject to different domains of implementations. The default distance measure is the Euclidean distance.

In the implementation at hand, the data set consists of tourists' data, where each tourist p is represented by four types of attributes:

- Maximum budget (B_p),
- Satisfaction factor about points ($S_{pn}; n = 1, \ldots, N$),
- Social relationship factor with other tourists ($R_{pq}; q = 1, \ldots, P;\ p \neq q$),
- Maximum number of points for each point category z ($E_{pz}; z = 1, \ldots, Z$)

For the given four types of tourist data, the total number of attributes is $n_v = 1 + N + P - 1 + Z = N + P + Z$. These attributes are used to represent the individual tourists into a n_v dimensional coordinate system (see Fig. 1). The measurement of the degree of closeness between any two tourist i and j is calculated by using Eq. 9 that is based on the Euclidean distance.

$$d_{ij} = \sqrt{b + s + r + e}, \tag{9}$$

where:

$$b = (B_i - B_j)^2$$

$$s = \sum_{n=1}^{N} (S_{in} - S_{jn})^2$$

$$r = \sum_{p=1}^{P} (R_{ip} - R_{jp})^2$$

$$e = \sum_{z=1}^{Z} (E_{iz} - E_{jz})^2$$

In the initiation phase, the k-means algorithm (see Algorithm 2) randomly selects k centroids from the represented tourists in the n_v dimensional coordinate system. Then, the grouping is done by assigning the remaining tourists to their closest centroid. Afterwards, the algorithm iteratively alternates between adoption of new centroids for individual groups by choosing the most central point inside the group, and then formation of new groups based on the distance from newly adapted centroids. The algorithm terminates and returns the result when no more tourists change their groups.

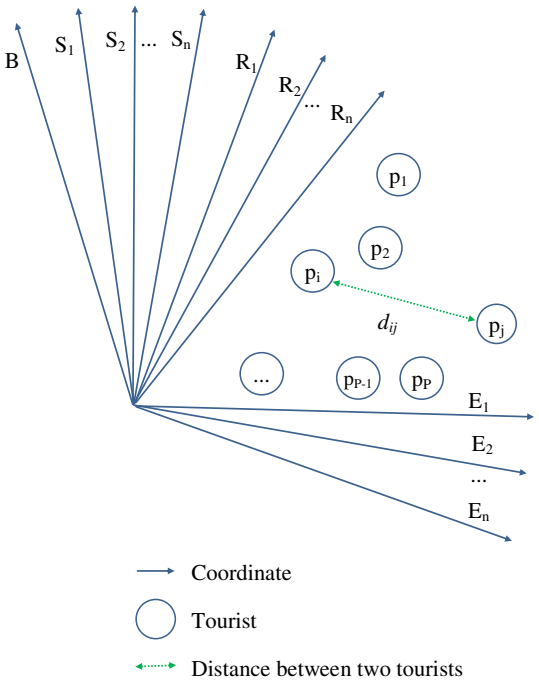


Fig. 1 Representation of tourists in a n_v dimensional coordinate system

The *k*-means algorithm alone is associated with the disadvantage of keeping the number of *k* groups fixed. In order to overcome this disadvantage, we implemented an iterative approach (see Algorithm 3) that uses the *k*-means algorithm to create the groups, whilst it employs an evaluation framework to evaluate the quality of grouping for groups with different number of tourists.

The cluster analysis method of Caliński and Harabasz (1974) is used to evaluate which is the best possible grouping configuration (i.e How many tourists should each group have?). This method is denoted as *pseudo F-static* F_{ch} function and it is based on statistical analysis of groups. In the following, we give details about the evaluation function and its comprising components, along with their corresponding variables.

$$F_{ch} = \left(\frac{SST}{SSE} - 1\right)\left(\frac{P - n_c}{n_c - 1}\right), \tag{10}$$

$$SST = \sum_{i=1}^{n_c}\sum_{j=1}^{n_i}\sum_{k=1}^{n_v}\left(V_{ij}^k - \overline{V^k}\right)^2, \tag{11}$$

$$SSE = \sum_{i=1}^{n_c}\sum_{j=1}^{n_i}\sum_{k=1}^{n_v}\left(V_{ij}^k - \overline{V_l^k}\right)^2, \tag{12}$$

where: *SST* is the total sum of squared distance to the overall mean, *SSE* is the sum of squared distance of the tourists to their own group means, *P* is the number of tourists, n_c is the number of groups, n_i is the number of members in group *i*, n_v is the

number of attributes used for representing tourists, V_{ij}^k is the value of the kth variable of the jth tourist, $\overline{V^k}$ is the mean of the kth variable, $\overline{V_l^k}$ is the mean over all observations of the kth variable in group i

Algorithm 2 Pseudocode of k-means algorithm

Require: MI
Ensure: Tourist groups
Select random centroids for each group
Create groups around each centroid
while some tourist changes their groups **do**
    Select new centroids for each group
    Create new groups around each centroid
end while

In order to increase the probability of obtaining the best possible number of subgroups, the algorithm executes for a number of iterations, as specified by *Maximum Iterations* (MI) parameter. In the course of a single iteration, the *k-means* algorithm is executed for each possible number of groups, starting from 2 up to the maximum possible number of groups, and the best evaluating configuration, based on *pseudo F-static* F_{ch} function, is recorded. After a series of executions, the algorithm returns the solution with the best evaluating configuration. Based on the empirical experiments, for test instances with a maximum of 10 tourists, it is sufficient to run the algorithm for 20 iterations to obtain the best evaluating configuration that has an optimal number of groups.

Algorithm 3 Pseudocode of tourist group builder

Require: MI
Ensure: Best tourist groups S_b
$i \leftarrow 1$
while $i \leq MI$ **do**
    $MinGroups \leftarrow 2$
    $MaxGroups \leftarrow NumberOfTourists - 1$
    $CurrentGroups \leftarrow MinGroups$
    $S_b \leftarrow null$
    while $CurrentGroups \leq MaxGroups$ **do**
        $S_c \leftarrow$ Apply K-means($CurrentGroups$)
        Evaluate S_c
        if S_b is $null$ or S_c better then S_b **then**
            $S_b \leftarrow S_c$
        end if
        $CurrentGroups \leftarrow +1$
    end while
    $i \leftarrow +1$
end while

As indicated in Eq. 9, all the tourist attributes are considered as equally important. This will allow to consider the differences of tourists in all aspects, and not only their mutual social relationship. This is important because some tourists might be related in terms of their social mutual relationship, but might differ in other aspects (i.e. budget, satisfaction with POIs, relation with other tourists and maximum number of POIs of certain categories) and, since the maximum constraints of each and every tourist must me enforced, this might be restricting to the overall group satisfaction, which is not in line with the objective function (see Eq. 1), which aims maximizing the overall tourists' satisfaction. A negative aspect of this approach, though, could be in cases where some tourists are very much related, but due to their differences in other aspects do no get placed in the same cluster. Nevertheless, a solution to such situations could be done in the preprocessing phase, where such tourists could be merged into a single virtual tourist by using the merging procedure (see Eqs. 13, 14 and 15) explained in the next section.

5.3 Group trip planner

The trip itinerary for tourist groups should be planned by considering the preferences of individual tourists and the social relationship factors between them. Nevertheless, in order to start the search process with a rather good initial solution, we initially use the *single trip planner* algorithm to create a joint trip for all tourists. However, since the *single trip planner* algorithm can consider only one pair of preferences, the tourist group data is merged into a single virtual tourist, where the corresponding virtual tourist's attributes for budget limitation (B_v) and maximum number of points for different point categories (E_{vz}), are set to the minimal values of the respective attributes by using Eqs. 13 and 14, respectively. On the other hand, the satisfaction factor of the virtual tourist with points (S_{pn}) is calculated by using Eq. 15, which represents the average value of satisfaction factors of all tourists in the group, respectively. Note that the *single trip planner* algorithm is not designed to work with the attribute of social relationship between tourists, therefore such attributes are not considered when merging the tourist's data in to a single virtual tourist.

$$B_v = Min\{B_1, B_2, \ldots, B_X\}, \tag{13}$$

$$E_{vz} = Min\{E_{1z}, E_{2z}, \ldots, E_{Xz}\};\ \forall z = 1, \ldots, Z, \tag{14}$$

$$S_{vn} = Average\{S_{1n}, S_{2n}, \ldots, S_{Xn}\};\ \forall n = 1, \ldots, N, \tag{15}$$

The main part of itinerary optimization starts when the algorithm enters its iterative phase, where it applies a novel neighbourhood mechanism for exploring the search space, with the aim of optimizing the itinerary by personalizing it based on specific preferences of tourists and their mutual social relationship. In the following subsections, we initially show the solution representation, then discuss the operators we

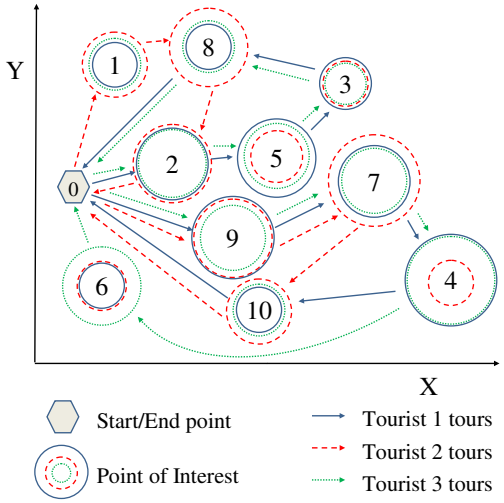
use as part of the neighbourhood exploration mechanism, and at the end we describe the general structure of the proposed nested tabu search algorithm.

5.3.1 Solution representation

The representation of a candidate solution is done by using an array of P members, where each member represents the itinerary for a particular tourist p and it consists of $M + 1$ lists. The first M lists contain the points that are scheduled to be visited on the respective M tours, whereas the last list keeps a record of points that are not part of the itinerary for the corresponding tourist (denoted as *TourOff*). Considering the example from scenario in Sect. 4, which is a problem consisting of $P = 3$ tourists, $N = 10$ points and $M = 2$ tours, a candidate solution might be represented as shown in Fig. 2). In general, the score of a point is illustrated with the size of the corresponding circle, hence the larger the circle the higher the score. Particularly, the score of a particular POI for *Tourist 1* is illustrated with a solid line circle, whereas for *Tourist 2* the illustration is done with a dashed line circle, and lastly the score for *Tourist 3* is illustrated with a dotted line circle.

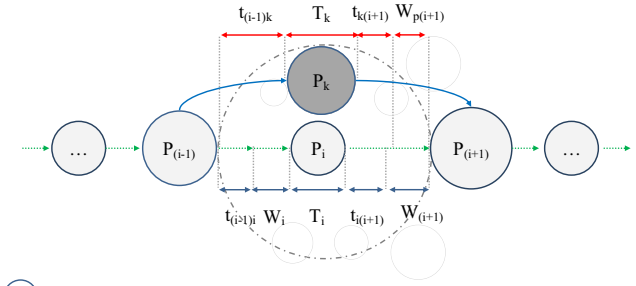
5.3.2 Neighbourhood exploration mechanism

The exploration of the neighbourhood of a given current solution S_c is done by using three different operators, namely *Separate, Join* and *Insert*. The first two operators are used as part of the neighborhood generation procedure in the framework of the

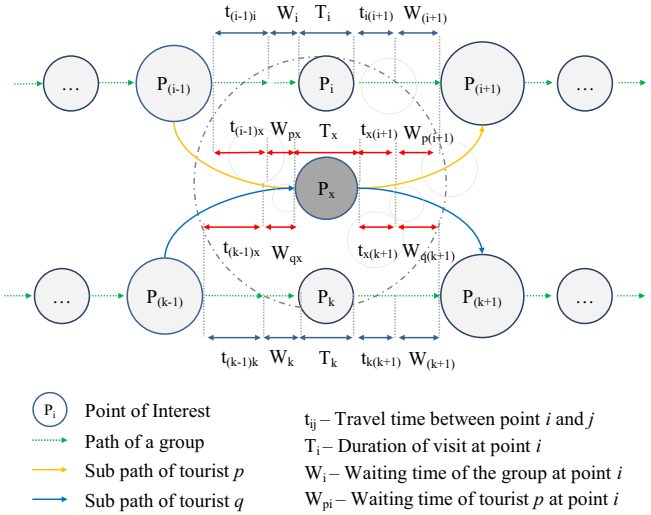


Solution representation		
Tourist1	Tour1	{2, 5, 3, 8}
	Tour2	{9, 7, 4, 10}
	TourOff	{1, 6}
Tourist2	Tour1	{1, 8, 2}
	Tour2	{9, 7, 10}
	TourOff	{3, 4, 5, 6}
Tourist3	Tour1	{2, 5, 3, 8}
	Tour2	{9, 7, 4, 6}
	TourOff	{1, 10}

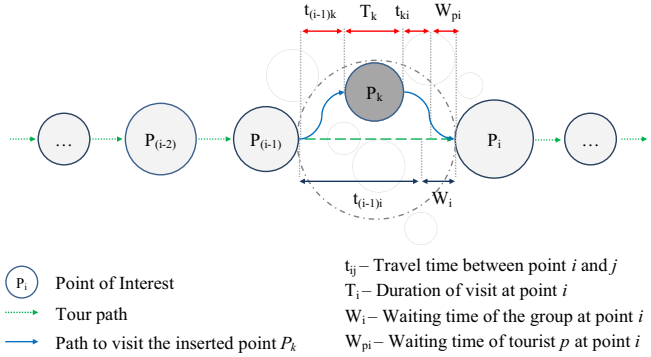
Fig. 2 A sample representation of a candidate solution with three tourists and two tours



(a) Separating tourist p at point P_k.



(b) Joining tourist p and q at point P_x.



(c) Inserting point P_k in the path of the tour of tourist p.

Fig. 3 Neighbourhood operators

tabu search meta-heuristic, whereas the last one serves as an operator for the greediness part of the proposed approach.

Separate operator tries to divide a tourist p from his current group that is at present planned to visit a given point i in tour m (see Fig. 3a). The set of all possible neighbours of the current solution (i.e. number of combinations to try out) is computed by considering every tourist p (belonging to a group with at least two members) for separation from his current own group at every point i of every tour m of the itinerary.

Join operator considers joining any two tourists p and q at present belonging to groups that visit different points on the same tour (see Fig. 3b). The joining point can be either a new point or one of the points currently scheduled to be visited by any of the two tourist. The neighbourhood exploration size (i.e. set of all possible neighbours) of the *join* operator is determined by the product of the triplet made by *number of tourists*, *number of tours* and *number of points in each tour*.

Insert operator examines the spare time of individual tourists, which might be available at some point in the middle or at the end of a particular tour that is part of the itinerary (see Fig. 3c). Depending on the amount of the spare time, the resulting tours are expanded with one or more new points. At this stage, the operator takes a greedy approach by trying to insert first new non included points that have the highest satisfaction factor for the respective tourists.

5.3.3 Tabu memories

For experimentation purposes, three different tabu memories are designed, which differ in the level of restrictiveness they pose to the search process. The first two memories are two-dimensional, namely the *tourist based memory* that is used to record information about two different tourists participating in an operation (i.e. separation or join) and the *point based memory* that keeps record of information whenever an operation (i.e. separation or join) between two particular points occurs. Meanwhile, the third memory, denoted as *tourist and point based memory*, has three-dimensions and it is used to save information about two particular tourists being involved in an operation at a certain point.

In situations from the practice, the number of tourists is usually much less than the number of available points in a tourism destination, therefore the *tourist based memory* is more restrictive than the *point based memory*, since there is a higher probability that more information about tourists than about points would be included into respective memories. On the other hand, the *tourist and point based memory* can be classified as the least restrictive one, since there is a lower probability that two particular tourists would be often subject of an operation at the same point.

These tabu memories are updated whenever either the *Separate* or *Join* operator is applied. In the case of *Separate* operator, all the cells that match the separated tourist with her/his current group members from whom she/he has

Table 8 Group trip planner parameters

Abbreviation	Name	Description
TLS	Tabu list size	Specifies the number of iterations a certain move cannot be used
MI	Maximum iterations	Defines the number of iterations the algorithm shall run
RIF	Re-initialization frequency	Indicates the frequency of re-initialization from a new random solution
OSF	Operator switching frequency	Sets the frequency of switching between *Separate* and *Join* operators

been separated, are updated. Whereas, in the case of *Join* operator, only the cell, which matches the two particular tourists being joined, is updated. Further, if the *tourist and point based memory* are utilized, then the cells/cell of respective matrix that represents the point where the separation or joining of tourists occurs, are/is updated.

5.3.4 Tabu search implementation

The *group trip planner* nests the *single trip planner* inside its framework. The nesting occurs either when the initial solution is generated, or when the algorithm needs to restart its search process from a new random solution in the search space.

Besides the tuning parameter of the *single trip planner*, the *group trip planner* has its own fine tuning parameters as described in Table 8, where the first column shows the abbreviation, the second column represents the full name, whereas the third column describes the purpose of using a particular parameter.

The procedure of the algorithm (see Algorithm 4) starts by setting the respective tabu memory to an empty state and defining a list of possible operators (i.e. *Separate* and *Join*). The preferences of all group members are aggregated to create a single virtual tourist by utilizing the respective formulas described above (see Eqs. 13, 14 and 15). Then, the *single trip planner* is used to create an initial solution (denoted as S_c), which represents a joint trip for all tourists, where the aggregated preferences, represented by a virtual tourist, are taken into account. In this case, the evaluation of the solution quality is done by using the objective function expressed by Eq. 1, where, in distinction to the *single trip planner*, the algorithm at hand takes into account also the mutual social relationship between individual tourists.

Algorithm 4 Pseudocode of group trip planner

Require: Problem instance I, TLS, MI, RIF and OSF
Ensure: Best found solution S_b
$Operators \leftarrow \{Separate, Join\}$
$S_c \leftarrow$ Generate initial solution by using *single trip planner*
Evaluate S_c
$S_b \leftarrow S_c$
$i \leftarrow 1$
$j \leftarrow 0$
$k \leftarrow 0$
while $i \leq MI$ **do**
    $Index \leftarrow (i/OSF)\%2$
    $CurrentOperator \leftarrow Operators[Index]$
    $S_c \leftarrow$ Apply $CurrentOperator$ in S_c
    $S_c \leftarrow$ Apply $Insert$ in S_c
    if S_c better then S_b **then**
        $S_b \leftarrow S_c$
        $j \leftarrow 0$
    else
        $j \leftarrow +1$
    end if
    if S_c has changed **then**
        $k \leftarrow 0$
    else
        $k \leftarrow +1$
    end if
    if k equals RIF **then**
        $S_c \leftarrow$ Generate initial solution by using *single trip planner*
    end if
    if j equals $0.3MI$ **then**
        Quit search
    end if
    $i \leftarrow +1$
end while

The iterative phase duration is specified by *MI* parameter, where, in the course of a single iteration, the neighbours could be generated either by applying the *Separate* or *Join* operator. Since, the initial solution (itinerary) is the same for all tourists, it becomes obvious that in order to start separating tourists, with the aim of personalizing individual itineraries based on specific preferences, the *Separate* operator should be applied in the first couple of iterations (as specified by the dedicated *OSF* parameter). In general, for every *OSF* iterations the neighbourhood generation mechanism alternates between applying *Separate* or *Join* operator. In each iteration, a best tabu and non-tabu solution is returned by the respective operator, where the best non tabu solution is considered first for adoption as a new current solution. In case the non-tabu solution is not better than the current solution,

the best tabu solution is checked whether it fulfils the aspiration criteria, which, in this implementation, is fulfilled only if the new current solution is better than the best found solution so far.

Whenever *Separate* or *Join* operator is applied, it might happen that some tourists at certain points get scheduled to wait for some time for the other tourists (in their groups) so they can start the next visit together. In order to try to reduce or eliminate this waiting time, we apply the *Insert* operator with the aim of inserting new points in the itinerary instead of waiting.

Further, in order increase the probability of not getting caught in local optima of any area in the search space, a random re-initiation procedure foresees restarting the search process from a new initial solution by re-using the *single trip planner* algorithm. This procedure is repeated every *RIF* iterations. As discussed earlier, the solution that is returned by the *single trip planner* algorithm is a joint itinerary that is optimized based on the aggregated preferences and constraints of all tourists. Based on the experimental results of *single trip planner* given in Sylejmani et al. (2012), the fined tuned value of the corresponding *MI* parameter is 46,000. Nevertheless, since the returned solution represents only a starting point in the optimisation process, the *single trip planner* algorithm is employed with its reduced value of *MI* parameter, which, in this implementation, is adapted to be 10,000 that is approximately one forth of the respective fined tuned value. This is reasoned based on two different aspects. The first, it shortens the overall computation time, and the second, since the returned solution is used as a starting solution, it does not fully optimize the starting solution based on the objective function of *single trip planner*, thus retaining a degree of diversity in the initial solution for different algorithm callings.

The algorithm quits either if it reaches 30% of iterations without any improvement or if it makes the foreseen maximum number of iterations, as specified by *MI* parameter.

For a more detailed description of the specific components of the algorithm and the pseudo-code itself, the reader is refereed to the Ph.D. thesis of Sylejmani (2013).

5.4 Types of approaches for itinerary planning

By combining the existing algorithm from the literature for planning the trip itinerary for a single tourist (denoted as *single trip planner*) and the two new algorithms presented in this paper, namely *tourist group builder* and *group trip planner*, it is possible to lay out four different approaches of itinerary planning (see Fig. 4).

Individual unique itineraries (denoted as “solo” mode) can be prepared by executing only the *single trip planner* once for each tourist. Meanwhile, the tourists can be clustered into smaller groups by using the *tourist group builder* algorithm, and then, joint itineraries (denoted as “subgroups”) can be prepared for each individual group by executing the *single trip planner* once for each of them. The *single trip planner* alone can also be used to plan the trip itinerary for the complete tourist group (denoted as “all together” mode), where after the process of the aggregation of tourist data, the algorithm can produce a joint itinerary for all

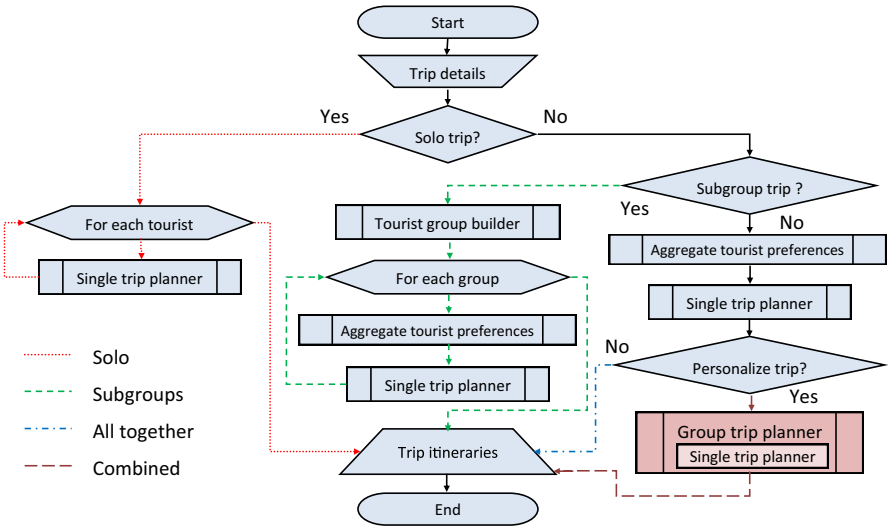


Fig. 4 Possible approaches of itinerary planning

tourists. Finally, the *group trip planner*, through the utilization of the *single trip planner*, can generate personalized trip itineraries (denoted as "combined" mode) for individual tourists.

Nonetheless, each approach is associated with its pros and cons, as explained in the following:

- The *solo* mode itinerary is where the tourists are separated throughout the trip and have their unique trip plan. In this case, the tourists will be able to visit POIs of their personal interests, but their mutual social relations will not be taken into account.
- The *subgroups* mode itinerary, being the only approach utilizing the *tourist group builder* algorithm, places tourists with common preferences in the same subgroup and then prepares a separate itinerary for each subgroup. This method will allow for consideration of social relations between the tourists in the same subgroups, but would make the planning process more constrained as individual constraints of tourists must be enforced.
- The *all together* mode itinerary represents a joint trip plan for all tourists, hence considering the social relations of each pair of tourists. Nevertheless, the preferences of the complete group are aggregated, and this makes it unable to consider the personal preferences of individual tourists, and, in comparison to *subgroups* mode, the planning process becomes even more constrained.
- The *combined* mode itinerary allows tourists to get separated during the course of the trip, so that they are able to visits POIs of their specific interest, and thus the itineraries get *personalized* based on individual preferences of tourists. In this way, tourists would at times be all together and at other times alone or in subgroups. This trip mode tries to "soften" the disadvantages of the three other modes, which are imposed either by not considering mutual social relationships

(*solo* mode) or by the process of aggregation of tourist' preferences (*subgroups* and *all together* modes).

5.5 Preparing the itinerary for a scenario of three tourists

In order to clarify the process of itinerary planning for all four approaches presented above, in this section, we present the results of the execution of the algorithms against the scenario described in Sect. 4, whereas a systematic experimental study is presented in the next section. Further, with the aim of clarifying the process of solution evaluation, we give details about the individual components that take part in the evaluation function. Although the algorithms are run ten times for each mode, since the envisioned scenario represents a very basic example, all the presented algorithms manage to solve it easily by returning always the same results for different executions.

For the running scenario, the algorithm returns the same itinerary for both *solo* and *subgroups* modes. Table 9 shows the corresponding itinerary for these two modes, where it can be seen that the women are scheduled to visit *POI 3* and *POI 7* in the first tour and *POI 4* and *POI 8* in the second tour, while the man is also scheduled to visit *POI 3* and *POI 7* in the first tour, but, in the second tour, he is initially scheduled to visit *POI 10* and then *POI 8*.

Even though the *solo* mode plans the trip itinerary separately for each tourist, still the resulting itineraries are completely the same for both women, whereas the man's itinerary differs in the second tour. On the other hand, the *subgroups* mode puts the two women in the same subgroup, while keeping the man separately. Despite this, the *subgroups* mode, likewise the *solo* mode, produces an identical itinerary.

The similarity in the itineraries of three tourists (especially in the case of the women) can be explained due to their similar preferences about POIs of category of

Table 9 Itinerary of the three tourists in *solo* and *subgroups* mode

Tourist	Tour/POI	Wait time	Start time	End time	Left time	Budget cost
Woman 1	Tour 1		9:00	11:30	0	33
Woman 2						
	POI 3	0	9:18	9:43		
	POI 7	0	9:58	10:47		
	Tour 2		9:00	10:55	35	
	POI 4	0	9:25	9:45		
	POI 8	0	10:04	10:23		
Man	Tour 1		9:00	11:30	0	32
	POI 3	0	9:18	9:43		
	POI 7	0	9:58	10:47		
	Tour 2		9:00	11:11	19	
	POI 10	0	9:35	9:55		
	POI 8	0	10:19	10:39		

nature and architecture. In spite of that, the man, as opposed to the women, is scheduled to visit *POI 10* in the second tour, since he is also interested in visiting POIs that belong to category of religious art and due to his constraint that he does not want to visit more than one POI that belongs to category of architecture (see Table 6). Further, it can be noticed that in the first tour, no-one of the tourists has any left (unused) time at the end of the tour, while in the second tour the women have 35 min left, whereas the man has only 19 min left.

The calculation of the overall tourists' satisfaction (evaluation of the solution) with a proposed itinerary is done by using the objective function (Eq. 1 that is defined in Sect. 3). The satisfaction of a single tourist consists of two parts, which is the satisfaction for visiting POIs and the satisfaction for being accompanied by the other tourists when visiting POIs. Each time a tourist visits a certain POI with a group of other tourists, the score of that POI for that tourist and the sum of her/his social relationship values with the other group members are added into her/his overall satisfaction. Similarly, this process is repeated for all tourists, where the sum of their satisfaction constitutes the overall group satisfaction with the trip itinerary.

The details for calculation of the overall satisfaction of the tourists with the resulting itineraries from the *solo* and *subgroups* mode are given in Table 10. The first column shows all POIs that are in the itinerary, whereas the next four columns present the social relationship values, as presented in Table 4, between the tourists that visit a certain POI together. Whilst, the last two columns depict the satisfaction of tourists for visiting the individual POIs, as well as the satisfaction with accompaniment between tourists visiting the individual POIs together.

In the following, the details of the calculation of the satisfaction of the individual tourists (denoted as S_{w1}, S_{w2} and S_m), as well as of the overall satisfaction of the group (denoted as S_t) in the scenario, are given.

Table 10 Description of the process of solution evaluation for the *solo* and *subgroups* modes

		Social relationship			Satisfaction	
POI	Tourist	Wo. 1	Wo. 2	Man	POIs	Accomp.
POI 3	Woman 1		4	5	40	9
	Woman 2	5		5	40	10
	Man	5	4		50	9
POI 4	Woman 1		4		45	4
	Woman 2	5			45	5
POI 7	Woman 1		4	5	40	9
	Woman 2	5		5	40	10
	Man	5	4		40	9
POI 8	Woman 1		4		40	4
	Woman 2	5			40	5
	Man				40	0
POI 10	Man				45	0
Subtotal		191	195	193	505	74
Total						579

$$S_{w1} = [40 + (4 + 5)] + [45 + (4)] + [40 + (4 + 5)] + [40 + (4)] = 191,$$

$$S_{w2} = [40 + (5 + 5)] + [45 + (5)] + [40 + (5 + 5)] + [40 + (5)] = 195,$$

$$S_m = [50 + (5 + 4)] + [40 + (5 + 4)] + [40 + (0)] + [45 + (0)] = 193,$$

$$S_t = 191 + 195 + 193 = 579$$

Note that the procedure for evaluation of the itinerary is the same also in the cases of the *all together* and the *combined* mode of trip planning, therefore, we do not present it in such details in the remaining part of this section.

In regard to *all together* mode, the algorithm makes a joint itinerary for all tourists (see Table 11). In this case, based on the procedure for merging tourists' data, by using Eq. 15), the score of a given POI is calculated as the average value of the satisfaction of all tourists with that POI. Moreover, in order to meet the constraints set by each tourist regarding the budget limit and the maximum number of POIs of a certain category, the minimal values of the data of the respective tourists are calculated by using Eqs. (13) and (14). Considering such constraints, the *all together* mode of trip planning prepares the itinerary that contains *POI 3* and *POI 7* in the first tour and *POI 10* and *POI 8* in the second tour. The negative side of the itinerary returned by *all together* mode is evident in the case of women, who, although they are not interested in visiting POIs of category of religious art, they are still scheduled to visit *POI 10*, due to the presence of man in the group, who has expressed interest in visiting POIs of such category. The overall satisfaction of all tourists for the *all together* mode is 527, while the left time for the first tour is zero and for the second tour it is 19 min.

Finally, Table 12 presents the itinerary that is returned by the *combined* mode of algorithm execution. It can be noticed that the returned itinerary is a combination of the previous itineraries returned by *solo* and *subgroups* mode on one side, and the itinerary returned by *group* mode on the other side. The women visit their most preferred POIs (*3, 7, 4* and *8*), whereas also the man visits his most preferred POIs (*3, 7, 10* and *8*). In this way, the women are not scheduled to visit *POI 10* (which is out of their interest) as in the case of *all together* mode, whereas the man is not

Table 11 Itinerary of the three tourists in *all together* mode

Tourist	Tour/POI	Wait time	Start time	End time	Left time	Budget cost
Woman 1	Tour 1		9:00	11:30	0	32
Woman 2						
Man						
	POI 3	0	9:18	9:43		
	POI 7	0	9:58	10:47		
	Tour 2		9:00	11:11	19	
	POI 10	0	9:35	9:55		
	POI 8	0	10:19	10:39		

Table 12 Itinerary of the three tourists in *combined* mode

Tourist	Tour/POI	Wait time	Start time	End time	Left time	Budget cost
Woman 1	Tour 1		9:00	11:30	0	33
Woman 2						
	POI 3	0	9:18	9:43		
	POI 7	0	9:58	10:47		
	Tour 2		9:00	11:11	19	
	POI 4	0	9:25	9:45		
	POI 8	15	10:19	10:39		
Man	Tour 1		9:00	11:30	0	32
	POI 3	0	9:18	9:43		
	POI 7	0	9:58	10:47		
	Tour 2		9:00	11:11	19	
	POI 10	0	9:35	9:55		
	POI 8	0	10:19	10:39		

scheduled to travel alone in the second tour as in the cases of *solo* and *subgroups* mode. The *combined* approach takes advantage of the left time at the end of the second tour, by scheduling the women to wait (15 min) for the man when they arrive at *POI 8*, so that they can make the visit together, hence their overall satisfaction will be increased. In this way, the *combined* mode of itinerary planning increases the overall satisfaction of the three tourists by assuming that none of them finds it inconvenient to wait for the other group members in order to make a joint visit. In the *combined* mode, the overall satisfaction of all tourists with the generated itinerary is 598, which is greater than in any other mode, while the left time for both tours is 19 min.

6 Computational experiments

In this section, in order to systematically compare the solutions obtained by different modes of itinerary planning, a new test set is introduced. Next, the results of the *single trip planner* are compared against the results of the state of the art approaches. Afterwards, the details for fine tuning the parameter values of the algorithm precede the computation results comparing the different variants of itinerary planning. The algorithm is coded by using Java 1.7. All experiments are done by using an Intel i3 2.2 GHz processor with 2 GB of RAM memory. A single experiment is conducted by making 10 runs for each instance in the test set.

6.1 Test bed instances

The existing test set of 148 test instances for tourist trips problem, created by Souffriau et al. (2013), is used to model the problem of the single tourist trip

itinerary. We further extend this data set to enable modelling the case of trip planning for tourist groups. In accordance to the test set origin, the instances are divided into two subsets, namely Solomon (1987) and Cordeau et al. (1997) based. The Solomon-based subset consists of 116 instances, while the Cordeau-based one consist of 32 instances.

Since, we deal with multiple tourists, we add an additional attribute P that specifies the number of tourists that are present on the trip. The value of attribute P varies from 2 to 10 in various instances. Further, the preferences of extra tourists for particular POIs are modelled by adding additional satisfaction factors (S_{pn}, $p = \{2, \ldots, P\}, n = \{1, \ldots, N\}$). The newly added satisfaction factor values get a random value in the range of 0–50 as in the original test instances.

Moreover, the budget limit ($B_p, p = \{2, \ldots, P\}$) and the maximum number of points of certain categories (E_{pz}, $p = \{2, \ldots, P\}, p = \{1, \ldots, Z\}$) for the additional tourists, are modelled. The budget for the newly added tourists is set as a random value in the range of the budget of the existing tourist in the instance, with a possible deviation of up to plus-minus 100. In addition, the values of the new tourists for the maximum number of points of certain categories, are also random values in the range of the values for the existing tourist, with a possible deviation of plus-minus 3.

In order to model the *relatedness* between the tourists, we added a matrix of asymmetric social relationship values ($R_{pq}, p = \{1, \ldots, P\}, q = \{1, \ldots, P\}, p \neq q$) that determines how much individual tourist are related/friends to each other. We model this relationship values as asymmetric, as in general, it is considered that two persons might not have the same relationship/friendship level toward each other.

As in the original test set, for different instances, the number of tours in the itinerary ranges from 1 to 4, whereas the number of POIs differs from 50 until 220.

Note that even though the test instances allow modelling of multiple time windows, in our implementation we only utilize single time windows by considering the opening time of the first time window and closing time of the last time window. The newly generate data set and the corresponding experimental results can ba accessed via the following link: https://sites.google.com/site/ushtrimet/tourist-trip-planning.

6.2 Comparing the single trip planner against state of the art approaches

The comparison of the *single trip planner* performance with the state of the art results and with the approach from Souffriau et al. (2013) has been presented in Sylejmani et al. (2012). However, we the aim of having a recapitulated view of the performance of the *single trip planner*, in Tables 13 and 14, we give a summarized presentation of the experimental results. The results are compared against the best known results in the literature for tourist trips that are less constrained problems such as OP, TOP and TOPTW Souffriau and Vansteenwegen (2010), as well as against the approach of Souffriau et al. (2013) for the tourist trips (i.e. MCTOPTW) problem. The comparison is done by using the existing test set for tourist trips (a.k.a. Solomon-based or Cordeau-based instances), which were also used by Souffriau

Table 13 Performance results of the single trip planner (STP)

M	Test set	Gap from best (%)			Gap from ILS-GRASP (%)		
		Best	*Avg.*	*Worst*	*Best*	*Avg.*	*Worst*
1	Solomon	4.07	6.57	13.82	3.82	6.04	12.98
1	Cordeau	6.42	14.97	25.84	2.89	9.06	18.04
2	Solomon	2.59	6.35	11.8	0.69	2.14	5.82
2	Cordeau	5.41	14.5	23.51	−0.8	5.74	12.83
3	Solomon	3.3	7.85	13.09	−0.66	1.96	5.53
3	Cordeau	6.84	14.18	23.67	0.77	5.35	13.19
4	Solomon	3.06	7.85	12.56	−2.2	0.79	4
4	Cordeau	6.28	13.88	20.59	−0.62	5.51	10.86
1–4	Solomon	3.15	7.38	12.71	0.31	2.96	6.97
1–4	Cordeau	6.27	14.25	22.82	0.59	6.28	13.15
	All	3.97	9.19	15.38	0.51	4.0	8.79

Table 14 Computation time of the single trip planner (STP)

M	Test set	Time (s)	
		ILS-GRASP	STP
1	Solomon	2.68	1.74
1	Cordeau	6.83	5.49
2	Solomon	7.69	3.34
2	Cordeau	17.85	10.18
3	Solomon	14.3	5.03
3	Cordeau	32.32	15.28
4	Solomon	22.99	6.71
4	Cordeau	52.13	18.32
1–4	Solomon	11.91	4.2
1–4	Cordeau	27.28	12.32
	All	15.24	5.96

et al. (2013). The results are averaged over the instances having the same number of tours and over a given instance type (i.e. Solomon or Cordeau based test set).

Souffriau et al. (2013) compare their approach with the best state of the art results that were obtained by various algorithms in the literature for the less constrained tourist trip (i.e. OP, TOP and TOPTW) Souffriau and Vansteenwegen (2010). Although the best existing results could not be reached by Souffriau et al. (2013), according to the authors their results are quite good because the problem at hand is more constrained as it extends the original tourist trips (i.e. TOPTW) problem with three additional constraints, which include: allowing multiple time windows, enforcing an extra budget constraint, and utilizing ten type constraints to model the different categories of POIs (e.g. castle, park, mosque, etc.). Comparing the results of the *single trip planner* with those in Souffriau et al. (2013), one can

conclude that the *single trip planner* is still outperformed by Souffriau et al. (2013) regarding the average performance in the given set of instances. Nevertheless, the *single trip planner* finds better solutions than Souffriau et al. (2013) in 70 (out of 149) instances. Since we use different machines (the approach in Souffriau et al. 2013 uses an Intel Xeon 2.5 GHz with 4 GB of RAM memory), we cannot give an accurate comparison regarding the computation time. However, also the *single trip planner* gives good solutions within few seconds (see Table 14). The results show that the *single trip planner* performs well, especially with the Solomon-based instances. The average score gap from the best known solutions is 9.19% and it is only 4.00% from the results obtained by Souffriau et al. (2013). Whereas, in the best case scenario, the average gap of *solo trip planner* from the approach of Souffriau et al. (2013) is only 0.51%. In addition, for 7% the test instances the *single trip planner* did find new best solutions, whereas 18% of the best known solutions were also found.

6.3 Parameter tuning

Based on the preliminary experiments that include a subset of test instances, we select a range of values for tuning the values of the basic parameters of the algorithm. The value of *MI* parameter is fixed to 10,000 iterations. Further increasing the number of iterations only leads to longer execution time.

The *RIF* parameter determines the frequency of re-initialization of the search process from a new starting solution that is generated by using the *single trip planner*. Therefore, the *RIF* parameter will determine how often the *single trip planner* will be called into action. Even though, the *single trip planner* is called with a reduced value of its own *MI* parameter, which equals to 10,000, it still requires an average time of execution of about 1.2 s. Thus, calling the *single trip planner* very frequently causes the *group trip planner* execute more slowly. As result, based on the experiments with the complete data set and in accordance to the selected value for the *RIF* parameter, we identify two different modes of *group trip planner* execution, namely slow and fast mode. The algorithm operates in the slow mode when the value of *RIF* parameter equals 5, whereas if the value of *RIF* parameter equals 40 the algorithm executes in the fast mode. In the next subsection, we give a detailed comparison of the complete data set for the two selected values of *RIF* parameter.

In the following, we present the experimental results about different types of tabu memories used and about the optimal value of the *OSF* parameter, where the results are averaged over 10 executions for each of the instances in the data set. Note that unless stated otherwise, the *group trip planner* is executed in the fast mode by setting *RIF* parameter value to 40.

In Table 15, we present the experimental results that are done by using the three different types of tabu memories, where five different tabu list sizes are considered. The first column shows the type of tabu memory used, whereas the second column displays the size of tabu list. In the next three columns, we present the gap of the *Combined* approach from the three other approaches for the preparation of the itinerary, namely *Solo*, *Subgroups* and *Group*. A positive value of the gap means

Table 15 Type and size of taby memory

Type of tabu memory	TLS	Combined vs. (%)			Average time (s)
		Solo	Subgroups	All together	
Tourist based	5	18.36	14.65	2.70	10.59
	10	18.37	14.66	2.71	10.28
	15	18.29	14.58	2.62	9.48
	20	18.39	14.68	2.73	9.00
	25	18.50	14.80	2.87	9.50
Point based	5	18.58	14.87	2.96	9.56
	10	18.56	14.86	2.94	9.27
	15	18.55	14.85	2.93	9.10
	20	18.45	14.74	2.81	8.76
	25	18.47	14.76	2.83	8.83
Tourist and point based	5	18.26	14.54	2.58	9.43
	10	18.46	14.75	2.81	9.39
	15	18.56	14.86	2.94	9.23
	20	18.58	14.88	2.96	9.03
	25	18.52	14.81	2.88	8.94

that the *Combined* approach has the advantage in comparison to the other compared approach. In the last column, we represent the average execution time of the *Combined* approach.

The results in Table 15 indicate that $TLS = 5$ on point based memory and $TLS = 20$ on tourist and point based memory produce the best results in comparison to the other considered values. In those two cases, the gap of *combined* approach from approaches such as *solo*, *subgroups* and *all together* is 18.58, 14.87 and 2.96%, respectively, except for $TLS = 20$ on tourist and point based memory, which is slightly better (14.88%) when *combined* and *subgroups* approaches are compared. Further, since for *TLS=20* of tourist and point based memory, the average execution time of the algorithm is 9.03 s, which is for 0.53 s faster than in the case of $TLS = 5$ of point based memory, we conclude that utilizing a tourist and point based memory with a TLS value of 20 is the most favorable combination out of all other considered combinations of tabu memories and tabu list sizes. In addition, the results show that a tourist based memory, besides producing worse solutions than the other two memories, it also causes the algorithm to execute a little slower, especially when the size of tabu list is short ($TLS = 5$ and $TLS = 10$).

Further, Table 16 shows the comparison of *combined* mode versus the three other modes, when different values of *OSF* parameter are considered.

The experimental results in Table 16 lay out that, in terms of quality of solutions, the higher range of values (from 50 to 110) for *OSF* parameter yields considerably better results than the lower range of values (from 3 to 40). Nonetheless, the execution time in those cases is much slower, which, at times, it is more than three times slower than on the lower range of values. Such results might also be correlated

Table 16 Operator switching frequency

OSF	Combined vs.(%)			Average time (s)
	Solo	Subgroups	All together	
3	18.50	14.80	2.87	9.35
5	**18.59**	**14.89**	**2.97**	**9.26**
10	18.53	14.82	2.89	9.07
20	18.53	14.82	2.90	8.94
30	18.37	14.66	2.71	8.79
40	18.26	14.55	2.58	8.65
50	19.12	15.44	3.61	19.84
60	19.18	15.50	3.67	22.86
70	19.32	15.65	3.84	24.95
80	19.45	15.79	3.99	23.68
90	**19.49**	**15.83**	**4.04**	**27.60**
100	19.22	15.54	3.72	28.94
110	19.34	15.68	3.87	31.64

The bold values emphasise the two best values of the operator switching frequency (OSF) parameter, where value 5 produces good results in a short period of time (fast mode), whereas value 90 yields to even better results but for a longer time (slow mode)

to the actual value used for *RIF* parameter, which, for this experiment, is set to 40. There is an indication that as soon the *OSF* parameter value surpasses the value of *RIF* parameter, the quality of results increases in the expense of a noticeable increase in terms of computation time. This result suggests that in order to get solutions that are better in terms of quality, the value of *OSF* parameter should be higher than value of *RIF* parameter, although, in that case the computation time increases too. Based on the obtained results, the most favorable value of *OSF* parameter is 90, which is a little more than twice the value of *RIF* parameter.

On the other hand, in case of the lower range of values for *OSF* parameter, the algorithm has the advantage in the aspect of computation time, which is 2.85 times faster than in the case of the higher range of values. Hence, in order to make a trade-off between the high quality of solutions and short computation time, we select two best values for the *OSF* operator, one being the best value on the lower range ($OSF = 5$) and the other one being the best value on the higher range ($OSF = 90$). The two selected *OSF* values are used alternatively every F iterations, where F is set 5% of the overall number of iterations of the algorithm (i.e. $F = (5/100) \times MI$).

6.4 Comparison between different types of approaches for itinerary planning

In the following, we present the results that are obtained when preparing the trip itinerary by using different approaches described above. The results are presented in the percentage gap that show the advantage of *combined* approach over the three other approaches. Further, we also present a comparison between the slow and fast mode of the execution of the *combined* approach, where the respective pros and cons are outlined in terms of computation time and quality of solution.

Table 17 Quality of solutions for different modes of algorithm execution

Compared approaches	M	Test set	Gap (%)	
			$RIF = 5$	$RIF = 40$
Combined vs. solo	1	Solomon	5.30	4.28
	1	Cordeau	21.14	19.81
	2	Solomon	13.36	12.29
	2	Cordeau	25.86	23.40
	3	Solomon	18.73	17.64
	3	Cordeau	25.58	24.20
	4	Solomon	23.49	22.28
	4	Cordeau	28.10	27.40
	1–4	Solomon	18.38	17.24
	1–4	Cordeau	26.09	24.75
	All		20.68	19.48
Combined vs. subgroups	1	Solomon	14.05	13.13
	1	Cordeau	18.28	16.90
	2	Solomon	13.97	12.90
	2	Cordeau	20.65	18.02
	3	Solomon	15.87	14.75
	3	Cordeau	20.18	18.70
	4	Solomon	17.32	16.01
	4	Cordeau	19.44	18.66
	1–4	Solomon	15.92	14.75
	1–4	Cordeau	19.79	18.34
	All		17.08	15.82
Combined vs. all together	1	Solomon	9.15	8.17
	1	Cordeau	11.78	10.29
	2	Solomon	5.05	3.87
	2	Cordeau	9.24	6.23
	3	Solomon	4.04	2.75
	3	Cordeau	7.81	6.11
	4	Solomon	3.34	1.82
	4	Cordeau	6.08	5.17
	1–4	Solomon	4.42	3.09
	1–4	Cordeau	7.91	6.24
	All		5.47	4.03

Tables 17 and 18 present the comparison results of the *combined* approach with the three other approaches, in terms of quality of solutions and time of execution, respectively. The results are aggregated based on the instance types (originated from Solomon or Cordeau) and on the number of tours in the instances. The test instances, which belong to a certain type and have a certain number of tours, are aggregated into a single resulting value, by calculating the average value over ten executions for each instance in the subset.

Table 18 Computation time for different approaches

M	Test set	Time (s)				
		Solo	Subgr.	All tog.	Combined	
					Slow	Fast
1	Solomon	11.08	3.58	1.35	37.27	8.48
1	Cordeau	35.22	11.45	4.59	119.84	18.61
2	Solomon	20.53	7.07	2.53	67.25	12.54
2	Cordeau	68.19	24.04	8.45	230.22	31.85
3	Solomon	31.28	10.78	4.27	101.46	18.17
3	Cordeau	103.82	32.14	13.36	264.17	42.53
4	Solomon	42.57	14.75	5.79	127.51	23.86
4	Cordeau	129.60	42.30	18.36	1028.61	59.50
1-4	Solomon	21.09	9.05	3.49	83.37	15.76
1-4	Cordeau	84.21	27.48	11.19	410.71	38.12
	All	38.87	13.03	5.15	154.15	20.60

In Table 17, the first column depicts the names of the methods being compared, whereas the second and third column show the number of tours and the type of the aggregated test instances, respectively. The last two columns show the average advantage of slow ($RIF = 5$) and fast ($RIF = 40$) mode from the respective approach, over the instances in a particular subset.

In terms of quality of solutions, the obtained results indicate that both modes (slow and fast) of *combined* approach yield to better results than any of the three other approaches. The comparative results of *combined* approach with the *solo* approach show that the difference is quite large (20.68% for slow mode and 19.48% for the fast mode). This can be explained by the fact that *solo* approach does not consider social relationship between tourists in the group, and, as a result, the overall satisfaction of tourists is calculated only based on the satisfaction of tourists with POIs.

The marginal difference between the *combined* approach and the *subgroups* approach is narrower, where the gap is 17.08% in the slow mode and 15.82% in the fast mode. The results are slightly better than in the *solo* case, since the *subgroups* approach takes into account the social relationship between the tourists into the same subgroups. Nevertheless, still the social relationship between tourists in different subgroups is not taken into account, and further the tourists in the same subgroup constrain each other in terms of budget they are willing spend and number POIs of certain category they want to visit.

Lastly, the *combined* approach shows that it can also obtain better results (5.47% for slow mode and 4.03% for the fast mode) than the *all together* approach, which, in the overall, proved to be better than the *solo* and *subgroups* approaches. In the case of the *all together* approach, all tourists go together, hence their social relationship is fully considered, but still, as in the case of *subgroups* approach, the tourist bind each other with their specific constraints.

Table 19 Comparison results between slow and fast mode

M	Test set	Slow vs. fast (%)		
		Best	Average	Worst
1	Solomon	0.86	1.07	1.02
1	Cordeau	1.13	1.66	3.63
2	Solomon	0.65	1.22	2.71
2	Cordeau	2.64	3.22	6.02
3	Solomon	0.87	1.32	2.18
3	Cordeau	0.91	1.82	4.39
4	Solomon	0.41	1.55	2.83
4	Cordeau	**−0.09**	0.96	1.91
1–4	Solomon	0.64	1.37	2.45
1–4	Cordeau	0.94	1.78	3.73
All		0.73	1.49	2.83

The bold value is set to emphasise the only scenario where the fast mode produces better results than the slow mode of the algorithm execution

Further, it is obvious that in comparison to the three other approaches, the *combined* approach (in both modes) makes more improvement in the Cordeau based instances than on the Solomon based instances. Such results, highlight the strength of the *combined* approach in solving difficult instances, such as Cordeau based instances.

In reference to the time of execution, Table 18 shows the comparative results of all four approaches. First and second column of Table 18 present the number of tours and instance type of the test subset, respectively. The next three columns show the average execution time of the approaches that do not utilize the *group trip planner* (i.e. *solo*, *subgroups* and *all together*), whereas the last two columns present the execution time of *combined* approach in its both modes (Slow and Fast).

The experimental data in Table 18 show that the slow mode of *combined* approach is the most intensive in terms of computation time by requiring on average a little more than 150 s per execution, which is several times higher than the time taken by any of the other presented approaches. On the other hand, in average, the fast mode of the *combined* approach is eight times faster than the slow mode and it is also two times faster than the *solo* approach, while being slower than the *subgroups* approach (less than two times) and the *all together* approach (around four times). In addition, for both modes of the *combined* approach, the computation time for the subset of Cordeau based instances is much higher than for the subset of Solomon based instances. The ratio of computation time between Cordeau and Solomon based instances is around 5 for the slow mode and around 2 the for the fast mode.

In terms of quality of solutions, in Table 19, we give a more detailed description between the two modes of the *combined* approach. While the first two columns present the number of tours and instance type, respectively, the last three columns represent the advantage gap of the slow mode against the fast mode.

On average, the slow mode obtains better results than the fast mode with a gap of 1.49%. An other advantage of the slow mode is that it improves the worst case

Table 20 Algorithms evaluation for different number of tourists and POIs

Inst. type	No. of touris.	No. of POIs	Combined (fast) vs. (%)			Combined (slow) vs. (%)			
			Solo	Subgr.	All tog.	Solo	Subgr.	All tog.	Comb. (fast)
Solomon	2	100	10.99	9.96	12.38	13.06	12.02	14.48	1.87
	3	100	4.72	4.54	6.33	5.56	5.38	7.19	0.81
	4	100	6.24	7.56	4.75	7.14	8.48	5.64	0.85
	5	100	6.67	8.09	3.88	8.54	9.99	5.70	1.76
	6	100	12.12	10.79	5.73	13.95	12.61	7.46	1.64
	7	100	17.67	16.64	4.71	19.55	18.50	6.38	1.60
	8	100	21.53	18.42	2.74	23.34	20.18	4.26	1.49
	9	100	25.38	21.46	2.17	26.65	22.69	3.20	1.01
	10	100	32.29	23.38	1.57	34.23	25.20	3.06	1.47
Cordeau	2	48	14.60	14.93	2.92	15.34	15.67	3.58	0.64
	3	96	11.61	7.54	4.44	13.22	9.10	5.95	1.45
	4	144	12.56	8.36	7.67	14.52	10.26	9.55	1.75
	5	192	16.22	14.89	9.05	18.97	17.61	11.63	2.37
	6	240	28.30	23.23	7.89	29.97	24.83	9.29	1.30
	8	72	41.23	20.19	2.16	45.69	23.99	5.39	3.16
	9	144	41.55	25.68	5.77	43.14	27.09	6.96	1.12
	10	216	44.90	31.43	8.51	47.66	33.94	10.58	1.91

solutions by 2.83% on average, whereas the best case solutions are improved as well, but with a gap of only 0.73%. This is expected, because in the worst case executions, the solutions returned by the fast mode are not matured as the solutions returned by the slow mode, due to the shorter time of algorithm execution. In addition, the slow mode is better than the fast mode with a higher margin for Cordeau based instances (average gap is 1.78%) than for Solomon based instances (average gap is 1.37%). The fast mode is slightly better, for 0.09%, only in the case of Cordeau subset of instances where number of tours are four. In overall, the slow mode has the advantage of obtaining solutions with a higher quality than the fast mode, but it requires much more computation effort.

6.5 Analysis of algorithms for various problem sizes

In the previous section, we showed the computation results averaged over different numbers of tours and instance types, whereas in this subsection (see Tables 20, 21), we present the evaluation and computation time of the four trip planning modes averaged over two additional instance particularities, namely the number of tourists and the number of POIs.

In Table 20, we compare the results obtained by the *combined* mode (in both variants) versus the results of three other modes, namely *solo*, *subgroups* and *all together*. As in previous experiments, also here, it is shown that the *combined* mode

Table 21 Algorithms computation time for different number of tourists and POIs

Inst. type	No. of touris.	No. of POIs	Time (s)				
			Solo	Subgr.	All tog.	Comb. (fast)	Comb. (slow)
Solomon	2	100	6.82	7.05	2.97	29.16	206.03
	3	100	12.72	8.27	3.90	30.42	198.02
	4	100	17.78	8.02	3.71	18.62	129.03
	5	100	22.20	11.26	4.00	12.90	77.36
	6	100	22.35	6.54	2.97	9.79	56.49
	7	100	29.27	10.67	3.61	10.67	39.67
	8	100	34.60	8.73	3.53	9.70	35.42
	9	100	37.34	11.57	3.25	11.33	21.39
	10	100	43.06	9.20	3.45	11.86	25.52
Cordeau	2	48	3.82	4.10	1.70	25.47	308.80
	3	96	20.05	13.02	6.60	37.75	675.32
	4	144	45.17	21.07	9.68	38.85	831.67
	5	192	88.45	40.45	15.80	45.77	585.78
	6	240	162.08	54.70	24.63	69.10	530.35
	8	72	31.73	7.88	3.40	10.68	111.53
	9	144	105.17	27.72	10.35	25.42	97.48
	10	216	217.20	50.90	17.37	51.93	144.78

is superior in comparison to the other approaches to any instance type, number of tourists or number of PIOs. A specific behaviours of the *combined* mode against the *all together* mode can be noticed in the Solomon set of test instances, whereas the number of tourists increases, the gap between them starts to narrow down, and this is especially the case with the instances having 8, 9 or 10 tourists. Nevertheless, such a behaviour is not present in the set of Cordeau based instances, where the gap between *combined* and *all together* modes increases with the increase of the number of tourists. Further, the average gap between the slow and the fast versions of the *combined* mode is 1.39 and 1.71% for Solomon and Cordeau based instances, respectively.

As it can be seen in Table 21, since in the *solo* mode the *solo trip planner* is executed once for each tourist, the computation time increases with the increase of the number of tourists, ranging from about 7 to 43 s for the Solomon based instances and from 4 to 217 s for the Cordeau based instances, where it is also clear that the increase in the number of POIs influences the duration of computation time. Whereas, for the *subgroups* mode the computation time scales between 7 to 12 s for the Solomon based instances and between 4 to 45 s for the Cordeau based instances, showing no signs of any correlation between the size of tourist group and duration of computation time. Also, in the *all together* mode, which the fastest among all other modes, the increase in the number of tourists does not produce any noticeable effect in the aspect of computation time, which remains in the scope of at most 4 and 25 s for the Solomon and Cordeau based instances, respectively.

In reference to the results of the *combined* mode, for both its variants and both instance types, a trend of the decrease in the aspect of computation time with the rise of the number of tourists can be noticed. Although, as stated in Sect. 6.3, all the experiments are done by using the same number of maximum iterations (i.e. 10,000), the *combined* approach, for bigger number of tourists, does not need to run that much of iterations. This is because it does not find any better solution after a certain amount of time, which in average for the fast mode, for instances with 5–10 tourists, is around 11 and 40 s for Solomon and Cordeau based instances, respectively. This can be reasoned due to the nature of the *join* operator, which tries out a less number of combinations with the increase of the trip constraints, which arise due to the inclusion of individual tourist constraints into the joint trip (as discussed in Sect. 5.4).

On average, for the faster variant, in the case of the instances with 100 POIs that are Solomon based, the computation time is around 16 s, whereas for instances with the number of PIOs ranging from 48 to 240 that are Cordeau based, the computation time is in average 38 s. Whereas, for the slow mode, the computation time goes to approximately 87 s for Solomon based set, and 410 s for the Cordeau based set.

7 Conclusions

In this paper, we presented an algorithm for planning the trip itinerary for tourist groups, where the construction of the starting solution is done by using the existing algorithm for planning the trip for a single tourist. The proposed algorithm is based on the framework of a nested tabu search meta heuristic with a neighborhood structure that relies on three unique operators, namely *Separate*, *Join* and *Insert*. The solution that is returned by the algorithm creates personalized itineraries for individual tourists in the group, by allowing them to change the companionship with different tourists while on the trip according to their preferences.

Based on the computational experiments that were conducted in a newly generated test set, we outlined two different modes of algorithm execution, which were tagged as slow and fast mode. On average, the slow mode of algorithm execution proved to obtain better results for 1.49% than the fast mode, while being around eight times slower than fast mode. In addition, both modes of algorithm execution outperform the approaches, namely *solo*, *subgroups* and *all together*, that can be generated by using the existing algorithms from the literature, designated for planning the trip itinerary for a single tourist preference list. Nevertheless, the computation time of the *combined* approach is in overall higher, which, for the fast mode of its execution, requires around 20 s that is four times more than the time needed by the *all together* approach.

When comparing the *combined* approach to the *all together* approach, as the two most closest approaches in terms of quality of the solutions they provide, we can conclude that although the *all together* approach is executed faster than the *combined* approach, it cannot obtain better results than *combined* approach, because the itinerary of *all together* mode is calculated by applying the *single trip planner*,

which is not intended primarily to plan the trip for tourist groups. The disadvantages in applying the *single trip planner* for a tourist group are as follows:

The satisfaction factors of all tourists are averaged for each individual point (as explained in Sect. 5.3). This makes the *single trip planner* approach unable to consider the personal preferences of individual tourists.
When tourists are kept in a single group, the personal constraints of individual tourist must be enforced. Therefore, the process of planing becomes more conditional, and along with that the number of possible candidate solutions is decreased.
The *single trip planner*, when applied in the context of tourist groups, from the start of the trip until the end, keeps the group members together, since it has no operators for joining or separating tourists. Hence, the *single trip planner* can not consider the personal preferences of individual tourists.

In general, the newly proposed algorithm, denoted as *group trip planner*, can help obtain better itineraries for tourist groups than the existing algorithms in the literature. This way, the tourists can get divided during the course of the trip, so that sometimes they travel together (or in subgroups) and visit POIs of their common interest, and at other times they travel alone, so that they can visit POIs of their own specific interest.

We conclude that the slow mode of algorithm execution could be applied in planning a combined trip itinerary during the pre-trip phase, where tourists could afford to wait a few minutes in order to get an itinerary with a higher quality, whereas the fast mode could be applicable in the on-trip phase, where tourists might select the option of planning a trip itinerary in a shorter time (in a few seconds), although with a slight decrease in quality.

The *all together* approach aggregates tourist satisfaction toward POIs by finding the average value. It would be interesting to investigate whether other forms of aggregation, such as taking the sum or median (or any other form) of the satisfaction of all POIs, would produce better results for this approach.

Also, as part of a future work, it would be worth analysing neighbourhood mechanisms that apply new operators for solving the problem at hand that might lead to improvements in efficiency or/and effectiveness of exploration of the search space, given that the problem is also highly constrained. For instance, a new possible operator for investigation in future could be swapping one or more tourists between two subgroups that visit two different closely located POIs. Another more general operator could rotate three or more tourists between their own subgroups so that each tourist goes in the subgroup of another tourist based on some order that might be generated randomly or by using a specific heuristic function.

Besides trying out new operators, it would also be interesting to apply other meta heuristic techniques such as for instance memetic algorithms, or hybridization of meta heuristic techniques with other techniques from constraint programming and operations research.

Acknowledgements This work is partially supported by a national research grant from the Ministry of Education, Science and Technology of the Republic of Kosova, as part of the research project entitled Tourist Tour Planning and Social Network Analysis. In addition, the authors would like to thank three anonymous reviewers, whose valuable comments helped in improving the content and presentation of this paper.

References

Brilhante IR, Macedo JA, Nardini FM, Perego R, Renso C (2015) On planning sightseeing tours with tripbuilder.Inf Process Manag 51(2):1–15

Caliński T, Harabasz J (1974) A dendrite method for cluster analysis. Commun Stat Theory Methods 3(1):1–27

Chao I, Golden BL, Wasil EA et al (1996) The team orienteering problem. Eur J Oper Res 88(3):464–474

Cordeau JF, Gendreau M, Laporte G (1997) A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 30(2):105–119

Delic A, Neidhardt J, Nguyen TN, Ricci F, Rook L, Werthner H, Zanker M (2016) Observing group decision making processes. In: Proceedings of the 10th ACM conference on recommender systems. ACM, pp 147–150

Fomin FV, Lingas A (2002) Approximation algorithms for time-dependent orienteering. Inf Process Lett 83(2):57–62

Garcia A, Vansteenwegen P, Souffriau W, Arbelaitz O, Linaza M (2009) Solving multi constrained team orienteering problems to generate tourist routes. **(status: published)**

Gavalas D, Kasapakis V, Konstantopoulos C, Pantziou G, Vathis N (2017) Scenic route planning for tourists. Person Ubiquit Comput **(in press)**

Gavalas D, Kasapakis V, Konstantopoulos C, Pantziou G, Vathis N, Zaroliagis C (2015) The eCOMPASS multimodal tourist tour planner. Expert Syst Appl 42(21):7303–7316

Gavalas D, Konstantopoulos C, Mastakas K, Pantziou G (2014a) Mobile recommender systems in tourism.J Netw Comput Appl 39:319–333

Gavalas D, Konstantopoulos C, Mastakas K, Pantziou G (2014b) A survey on algorithmic approaches for solving tourist trip design problems. J Heurist 20(3):291–328

Gavalas D, Konstantopoulos C, Mastakas K, Pantziou G, Tasoulas Y (2013) Cluster-based heuristics for the team orienteering problem with time windows. In: Experimental algorithms. Springer, Berlin, pp 390–401

Glover F (1989a) Tabu search—part 1. ORSA J Comput 1(3):190–206

Glover F (1989b) Tabu search—part 2. ORSA J Comput 2(1):4–32

Glover F, McMillan C (1986) The general employee scheduling problem. an integration of ms and ai. Comput Oper Res 13(5):563–573

Kurata Y, Hara T (2013) Ct-planner4: toward a more user-friendly interactive day-tour planner. In: Information and communication technologies in tourism 2014. Springer, Berlin, pp 73–86

Likas A, Vlassis N, Verbeek JJ (2003) The global k-means clustering algorithm. Pattern Recognit 36(2):451–461

Lloyd S (1957) Least squares quantization in pcm. In: Unpublished Bell Lab. Tech. Note. portions presented at the Institute of Mathematical Statistics Meeting Atlantic City

Lloyd S (1982) Least squares quantization in pcm. IEEE Trans 28(2):129–137

Masthoff J (2015) Group recommender systems: aggregation, satisfaction and group attributes. In: Recommender systems handbook. Springer, Berlin, pp 743–776

Ramesh R, Brown KM (1991) An efficient four-phase heuristic for the generalized orienteering problem. Comput Oper Res 18(2):151–165

Schaller R (2011) Planning and navigational assistance for distributed events. In: Proceedings of the 2nd workshop on context aware intelligent assistance, Berlin

Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper Res 35(2):254–265

Souffriau W, Vansteenwegen P (2010) Tourist trip planning functionalities: state-of-the-art and future. In: Current trends in web engineering. Springer, Berlin, pp 474–485

Souffriau W, Vansteenwegen P, Vanden Berghe G, Van Oudheusden D (2013) The multiconstraint team orienteering problem with multiple time windows. Transp Sci 47(1):53–63

Souffriau W, Vansteenwegen P, Vertommen J, Berghe GV, Oudheusden DV (2008) A personalized tourist trip design algorithm for mobile tourist guides. Appl Artif Intell 22(10):964–985

Sylejmani K (2013) Optimizing trip itinerary for tourist groups. Ph.D. thesis, Vienna University of Technology, Faculty of Informatics, Austria

Sylejmani K, Dorn J, Musliu N (2012) A tabu search approach for multi constrained team orienteering problem and its application in touristic trip planning. In: 2012 12th international conference on hybrid intelligent systems (HIS). IEEE, pp 300–305

Tsiligirides T (1984) Heuristic methods applied to orienteering. J Oper Res Soc 35:797–809

Tumas G, Ricci F (2009) Personalized mobile city transport advisory system. Inf Commun Technol Tour 2009:173–183

Vansteenwegen P (2008) Planning in tourism and public transportation attraction selection by means of a personalised electronic tourist guide and train transfer scheduling. PhD thesis, Katholieke Universiteit Leuven, Centre for Industrial Management, Belgium

Vansteenwegen P, Souffriau W, Berghe GV, Oudheusden DV (2011a) The city trip planner: an expert system for tourists. Expert Syst Appl 38(6):6540–6546

Vansteenwegen P, Souffriau W, Oudheusden DV (2011b) The orienteering problem: a survey. Eur J Oper Res 209(1):1–10

Vansteenwegen P, Souffriau W, Vanden Berghe G, Van Oudheusden D (2009) Iterated local search for the team orienteering problem with time windows. Comput Oper Res 36(12):3281–3290

Zenker B, Ludwig B (2009) Rose: assisting pedestrians to find preferred events and comfortable public transport connections. In: Proceedings of the 6th international conference on mobile technology, application and systems. ACM, p 16