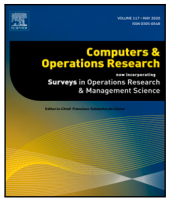
Computers & Operations Research 167 (2024) 106633

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor



Centralized multi-visitor trip planning with activity reservations in crowded destinations

Joris Slootweg [a,b,c,*], Rob van der Mei [a,c], Caroline J. Jagtenberg [d], Frank Ottenhof [b]

[a] CWI, Stochastics, Science Park 123, Amsterdam, 1098 XG, Netherlands
[b] The Driving Force, J.N. Wagenaarweg 6, Uithoorn, 1422 AK, Netherlands
[c] Vrije Universiteit, Mathematics, De Boelelaan 1111, Amsterdam, 1181 HV, Netherlands
[d] Vrije Universiteit, School of Business and Economics, Operations Analytics, De Boelelaan 1111, Amsterdam, 1181 HV, Netherlands

ARTICLE INFO

Keywords:
OR in service industries
Combinatorial optimization
Tourist trip design
Orienteering problem
Practical applications

ABSTRACT

We study the problem of centralized planning of leisure trips in congested areas for visitor groups with reservations for activities. We develop an algorithm that through a combination of customization and coordination can improve average happiness considerably. Extensive numerical experimentation with both synthetic and real-life data show that our algorithm strongly outperforms the classical First-Come-First-Served reservation policy, both in terms of *visitor happiness* and in terms of *fairness* among visitors. Moreover, our results show that our algorithm leads to good solutions for small-sized problem instances (with errors typically within 5%–10% from an optimal solution obtained via Integer Linear Programming). Finally, the computational effort with regard to number of visitors is bounded by the capacity and the number of activities, while the increase in computation time for the number of attractions is bounded by the average number of activities that fit into a trip. As a result, our approach leads to good solutions within minutes in realistic settings with more than 10 thousand visitors a day.

1. Introduction

Several cities suffer from negative effects of over-tourism. In those cities residents or visitors feel like the quality of experiences deteriorates because of the number of visitors (Goodwin, 2017). In Dichter and Manzo (2017) five methods to cope with over-tourism are mentioned: (1) smooth visitors over time, (2) spread visitors across sites, (3) adjust prices to balance supply and demand, (4) regulate accommodation supply, and (5) limit access and activities. We will focus on an algorithmic approach to facilitate the first two methods that can be used in a personal electronic travel guide. For optimal spreading of visitors over time and space a centralized decision agent is needed. Since industry partners think such a centralized agent is possible if the benefits are clear, we assume such an agent exists.

Personal electronic travel guides can assist with this task by providing recommendations, generating a route and allowing visitors to customize such a route (Kenteris et al., 2011). The route-generation functionality of these applications is studied as the tourist trip design problem. This problem is often modeled as a profitable tour or an orienteering problem (Gavalas et al., 2014). In many of these applications only a *single trip* at a time is considered. However, to facilitate spreading visitors over time and locations our algorithm should consider *other visitors* as well. Hence, effective trip generation functionality should consider the interest of *all* visitors, the availability of the activities, and the congestion generated by their trips. When generating these trips, a proposed algorithm should *customize* based on individual preferences, while at the same time *coordinate* between them. Although trip generation has been studied quite extensively, algorithms that coordinate between visitors have received far less attention.

Besides considering multiple people at the same time, such a centralized system needs to be able to do reservations as well. Doing reservations for slots ensures that people can do their recommended trip and at the same time this can provide an incentive to join the system by providing users priority access. Therefore, a trip-planning system should be able to *interact with booking systems*. In tourism, slot management is already a widely adopted mechanism. Hence, an algorithm that generates trips while spreading visitors over time and attractions should also be able to perform this task based on slot management. In this case, a trip generated by an algorithm can directly result in a set of bookings at the activities. To the best of our knowledge, the *combination* of orienteering multiple agents with time slot reservations has not been studied yet.

* Corresponding author at: CWI, Stochastics, Science Park 123, Amsterdam, 1098 XG, Netherlands.
E-mail address: joris.slootweg@cwi.nl (J. Slootweg).

https://doi.org/10.1016/j.cor.2024.106633
Received 31 July 2023; Received in revised form 25 March 2024; Accepted 26 March 2024
Available online 28 March 2024

Finally, these algorithms can only be implemented in practice if they scale very well to large numbers of visitors. Since popular cities or theme parks have millions of visitors per year (Rubin, 2019; Robino, 2019), handling ten-thousand or more visitors should be possible. To the best of our knowledge three papers studied multi-agent orienteering (Luo et al., 2022; Wang et al., 2017; Chen et al., 2014). However, the algorithms discussed in these papers focused on small scale instances and require about an hour to compute solutions for 40 visitors. Therefor, scalability for these type of algorithms is an important research gap that this paper aims to fill.

Motivated by this, the contribution of this paper is three-fold. First, we formulate a mathematical model for multiple-visitor route generation with activity reservations at crowded destinations. Second, we propose an algorithm that solves this model effectively and efficiently for large number of agents and attractions. Finally, we show the benefits of a centralized approach with a case study based on tourists in Amsterdam in cooperation with industry partners.

The remainder of this paper is organized as follows. In Section 2 we discuss previous research on algorithms for tour generation in the tourist-trip design problem with a special focus on variants that include congestion. Section 3 formulates the model as an ILP and Section 4 describes our solution heuristic in detail. In Section 5, the experimental results are discussed with respect to accuracy and computation times, using the classical FCFS policy as a benchmark for the large real-life instance and an ILP for a small example. In Section 6 we summarize our findings and discuss open research areas within this topic.

2. Literature review

Tourist-trip-generation and electronic tourist guides have received much attention in literature. In Gavalas et al. (2014), a survey of algorithms used for trip-generation is provided. Frequently, trip generation is modeled as a *profitable tour* or an *orienteering problem* with additional constraints, depending on the requirements of the specific problem. In orienteering problems, the goal is to collect as much *reward* as possible by visiting nodes with individual rewards within a travel-time limit, where each node can only be visited once (Gunawan et al., 2016). In the profitable tour, rewards are also obtained for visiting nodes, but the travel time or cost contributes negatively to the objective function. For many applications, including tourist-trip planning, additional restrictions to these problems are necessary, e.g., to include the opening hours of facilities. In our case, we consider the orienteering model more appropriate since the travel time is guaranteed to be part of the constraints. Although reducing travel time or cost will make visitor happier, this is indirectly achieved by optimizing the number of visits in the orienteering problem.

Excellent surveys of variants and solution methods for the orienteering problem are provided in Gunawan et al. (2016) and Ruiz-Meza and Montoya-Torres (2022). Some of the more relevant variants are the orienteering problem with time windows, the time-dependent orienteering problem, and the orienteering problem with functional profits. The orienteering problem with *time windows* is of particular interest because it shares the similarity that visitors might have to wait before they can start their visit. Righini and Salani (2009) use dynamic programming with state-space relaxation and smart initialization on this problem, which is capable of solving many instances to optimality. For large instances, their algorithm can turn any incumbent infeasible solution to a feasible, though possibly sub-optimal, solution by removing double visits. As an alternative, Ant Colony Optimization (ACO) was used in Verbeeck et al. (2017, 2014), which can deal with a considerable variety of problems. The solution quality is generally good. However, it is never known how close to optimal an incumbent solution is, which makes early termination harder. Several researchers use Iterated Local Search (ILS) to solve variants of the orienteering problem (Vansteenwegen et al., 2009b). This meta-heuristic delivers solutions of decent quality very fast. However, it is highly dependent on an appropriate combination of local search and shake methods. For the problem described here, especially local search is more complicated due to the rigidness imposed by the slot reservations. For extensions of the problem the *time-dependent* orienteering problem is of interest since it allows varying travel times for instance based on the arrival and departure time of public transport. An example is given in Gavalas et al. (2015), who use a more extensive time-dependent travel-time table. Finally, the orienteering problem with *functional profits* is of interest, since it allows for varying the profit dependent on the expected crowds of a location at a given time or obtaining an associated reward when one of the locations in a set is visited (e.g., a visit to the beach). This problem is solved using ACO in Mukhina et al. (2019).

Algorithms that consider the capacity of the visited activities are studied far less extensively. Usually, these are implemented by generating a single trip but adding a queuing time to the visit time, depending on the time of arrival. An example of this approach is Testa and Dozier (1999), where evolutionary computing is used to generate theme-park tours with queue-awareness based on the arrival time at an attraction. However, this study does not consider the effect that scheduled or suggested trips have on the queues. In Chen et al. (2014) and Varakantham et al. (2015), the effect of scheduled trips on the queues of the activities is considered by focusing on the calculation of a *Nash equilibrium* with multiple self-interested agents. Although these agents have more knowledge than regular visitors on the effect of their and others' decisions on the outcome, the final result still does not necessarily maximize the average (global) happiness. Optimization of the *average trip happiness* is the objective in Wang et al. (2017). In this study, three methods are investigated to solve the problem: (1) a sequential algorithm, (2) an iterated local search, and (3) a probabilistic version of ILS. However, the last two of these algorithms are not able to handle large numbers of visitors, since the computation time is typically more than one hour for 100 visitors and seems to scale exponentially. The sequential algorithm is able to deal with a large number of visitors. Furthermore, the sequential algorithm proposed in that paper produces decent results on random scenarios. However, in hard instances the solution quality is considerably worse compared to the two other algorithms especially since it does not consider the heterogeneity in visitor preferences. Luo et al. (2022) expand upon the research done by Wang et al. (2017) by improving the computational efficiency of their exact algorithm and proposing constructive heuristics based on Vansteenwegen et al. (2009a) and Labadie et al. (2011) and combining it with a Variable Neighborhood Search. However, their proposed algorithms are only able to handle very limited numbers of visitors.

A different approach is taken by Ruiz-Meza et al. (2021) where they simultaneously make a pre-defined number of trips and match tourists to them. To prevent congestion each trip has a maximum number of participants and each location can only be visited by one trip. However, for the goal of our algorithm their approach has several limitations. The main one is that the algorithm does not increase the utilization of the locations by distributing visitors over time; and second, the algorithm cannot handle a large number of trips, making individual customized tour generation nearly impossible. Finally, we refer to Ruiz-Meza and Montoya-Torres (2021) for an algorithm that creates trips for groups of tourists with *heterogeneous preferences* within the group. This study only creates one trip, hence is less relevant in the multi-agent setting that we study.

3. Problem definition and model formulation

Before visiting an area, each group of visitors indicates which *activities* (the set of all activities is referred to as A) they would like to do by assigning a *reward* to each of them, and with how many people they are visiting. We assume a homogeneous preference within a group, but the preferences between different groups are heterogeneous. Furthermore, the other input variables are (1) a *travel-time table* indicating the travel

time between all pairs of activities, (2) the start times of all possible reservation slots, (3) their respective capacities, (4) the overall capacity of each activity when people from multiple slots can be present at the same time, and (5) their duration. The goal is to generate a trip for each visitor group where the *happiness* is considered to be the total reward of all the visited activities. Formulating this as an ILP leads to the following model.

Let U be the set of visitor groups where each group k has size S_k. Each group $k \in U$ has a start location L_k^{start}, an earliest start time $t_k^{\min}$, an end location L_k^{end}, and an end time $t_k^{\max}$ when he has to be there. Note that we assume that it is always possible to reach the final destination in time from the start position for all visitors to prevent infeasibility. Each group k specifies a score $P_{i,k}$ for each activity i, when they visit this activity this amount is added to their level of happiness with their trip.

The travel-time matrix contains entries $T_{i,j}^{\text{mode}}$ indicating the travel time from activity i to activity j for a specific mode. Furthermore, we assume that modes can be switched at any stage of the trip. This is a realistic assumption when walking, bike sharing systems, taxis or frequent public transport are considered. We also note that visitors do not like all modes of transport equally. So using a not preferred mode of transport comes at a penalty. Which is very small compared to the rewards obtained by visiting activities. Finally, this matrix is not necessarily symmetric since activities can have different start and end locations. However, for our experiments we do assume the triangle inequality holds. Our ILP formulation only considers one mode of travel since including multiple travel modes and a travel time penalty has too much impact on computation time.

Furthermore, we define a set of time slots or batches for each activity B_i, where each slot $s \in B_i$ has capacity $C_{i,s}$, start time $t_{i,s}^{\text{start}}$ and end time $t_{i,s}^{\text{end}}$. All the time slots (T) from which people will be present at activity i at the moment time slot s starts are in $J_{i,s}$, at each of those moments the activity can have at most $CT_{i,t}$ people present. Notice that $C_{i,s}$ is not sufficient since allowing the maximum number of people all the time can lead to overcrowding and reducing $C_{i,s}$ leads to a reduction of daily capacity due to the slower inflow of people at the start of the day and less flexibility. Hence, it is not desirable. Furthermore, we have opted to make both flexible in time in our model to allow a part of the capacity to be reserved for other purposes.

The ILP formulation has two types of decision variables: (1) a binary variable $x_{i,j,k}$ that indicates that visitor k travels from i to j, and (2) another binary variable $y_{i,m,k}$ that indicates that visitor k visits activity i at time slot m. This results in the following ILP formulation:

$$\max \quad \sum_{i\in A}\sum_{s\in B_i}\sum_{k\in U} S_k P_{i,k} y_{i,s,k} \tag{1}$$

Subject to:

$$\sum_{i\in A\cup L_k^{\text{end}}} x_{L_k^{\text{start}},i,k} = \sum_{j\in A\cup L_k^{\text{st}}} x_{j,L_k^{\text{end}},k} = 1 \qquad \forall k \in U \tag{2a}$$

$$\sum_{i\in A\cup L_k^{\text{start}}} x_{i,m,k} = \sum_{j\in A\cup L_k^{\text{end}}} x_{m,j,k} = \sum_{s\in B_m} y_{m,s,k} \qquad \forall k \in U, \forall m \in A \tag{2b}$$

$$\sum_{s\in B_i} y_{i,s,k} \le 1 \qquad \forall k \in U, \forall i \in A \tag{2c}$$

$$\sum_{s\in B_i} y_{i,s,k} t_{i,m}^{\text{end}} + T_{i,L_k^{\text{end}}} \le t_k^{max} \qquad \forall k \in U, \forall i \in A \tag{2d}$$

$$t_k^{\min} + T_{L_k^{\text{st}},i} \le \sum_{s\in B_i} y_{i,s,k} t_{i,j}^{\text{start}} \qquad \forall i \in A, \forall k \in U \tag{2e}$$

$$\sum_{s\in B_i} y_{i,s,k} t_{i,m}^{\text{end}} + T_{i,j} \le \sum_{s\in B_j} y_{j,s,k} t_{j,m}^{\text{start}} + (1 - x_{i,j,k})M \qquad \forall i,j \in A, \forall k \in U \tag{2f}$$

$$\sum_{k\in U} y_{i,s,k} S_k \le C_{i,s} \qquad \forall i \in A, \forall s \in B_i \tag{2g}$$

$$\sum_{j\in J_{i,t}}\sum_{k\in U} y_{i,j,k} S_k \le CT_{i,t} \qquad \forall i \in A, \forall t \in T \tag{2h}$$

$$x_{i,j,k}, y_{i,s,k} \in \{0,1\} \qquad \forall i,j \in A, \forall k \in U, \forall s \in B_i$$

The objective function (1) is the total visitor happiness; note that the group size is added to the objective to prevent a strong preference for small groups of visitors on capacity-scarce activities. In our numerical experiments we validated that this resulted in no correlation between group size and trip happiness (see the end of Section 5.2). Constraint (2a) ensures that each trip starts and ends at the right locations. Constraint (2b) ensures that a location of a scheduled time interval is in the route as well and that only the locations where an event is scheduled are part of the trip. Constraint (2c) prevents multiple bookings from one visitor for one event. Constraint (2d) ensures that the final destination is reached in time, while constraint (2e) ensures that a visitor reaches his or her first activity in time. Constraint (2f) ensures that an activity can be reached in time from the previous activity. Furthermore, as a side effect this constraint also prevents sub- and self-tours, since it enforces strictly increasing start times. The maximum travel time plus the last end time of an activity is sufficiently large to serve as big M for this constraint. Constraint (2g) ensures that the capacity of an interval is not exceeded. Constraint (2h) ensures that at the sum of visitors at the same time never exceeds the capacity of the activity.

4. Solution approaches

The inspiration for the algorithm comes from the following insights based on many years of practice, consultation of experts in the tourism industry, and early experiments with a FCFS policy:

1. *Leisure trips* with activities tend to include a *small number of activities*.
2. There is usually significant *heterogeneity* in the preferences between groups of visitors. This opens the way to *customize* and optimally *coordinate* trips for different visitor groups.
3. People accept the fact that in congested settings they cannot do everything they want to do.
4. Allowing visitors to book a congested activity which they do not value highly hurts total visitor happiness when it prevents others who value it highly from visiting that activity

Based on these realistic assumptions, we propose an iterative scheme in which at every iteration routes of visitors are expanded with only one activity from a small subset of their top rated activities. For this scheme, we need a trip-generating algorithm with route expanding heuristics that are fast while delivering decent quality solutions. Furthermore, we require computationally efficient methods to escape local optima in a given trip.

4.1. Efficient single agent route expanding heuristics

We use two greedy route expansion heuristics: (1) Insert an activity at the first feasible place in the route given the already planned activities, capacities and start and end location of the visitor. The activities are inserted based on their ranking, inserting the one with the highest reward first. (2) A novel minimum blocking heuristic where the time slot is chosen such that we minimize the loss of options for unscheduled activities (explained in more detail in the next paragraph).

When we plan activity i for visitor group k, we want to pick the time slot that gives the highest number of possibilities for scheduling other activities. This is equivalent to selecting the time slot that blocks the minimum number of time slots at other activities for this visitor. When time slot t of activity i is booked, all time slots of activity j that start between $t_{i,t}^{\text{start}} - T_{ji} - d_j$ and $t_{i,t}^{\text{end}} + T_{ij}$ will become unavailable. We take the fraction of slots that fall in this interval relative to the total number

of available time slots of activity j to calculate a blocking fraction b_j. Since in practice the number of activities of interest to a visitor group is quiet limited. This algorithm is made more efficient by considering only activities (A_k^{rel}) that are not yet in the trip and have a positive contribution to their happiness ($S_{i,k} > 0$). This allows us to calculate a blocking score for a time slot $B_{i,t}$ using the following equation:

$$B_{i,t} = \sum_{j \in A^{\text{rel}}} S_{i,k} b_j^{\alpha_j} \tag{3}$$

In this equation A^{rel} is the set of activities which the visitor wants to visit, $S_{i,k}$ is the reward for activity j, b_j is the fraction of time slots of activity j that is blocked by selecting time slot t at activity i. Finally, α_j is a system parameter that can be used to make blocking more/less severe depending how detrimental blocking part of your possibilities is for that activity. For highly crowded activities $\alpha_j \in (0, 1)$ will increase the severity of blocking the available options, for uncrowded activities $\alpha_j > 1$ will decrease the severity of blocking, since it probably is possible to book this as long as there is still a time slot available. In our experiments we used $\alpha_j = 1$.

To construct a full trip for a single agent using this algorithm we use the following stepwise procedure:

Step 1: Sort activities based on their reward
Step 2: Discard events with a reward lower than a pre-determined cut-off point
Step 3: Calculate $B_{i,t}$ for each time slot t of the first activity i using Eq. (3)
Step 4: Choose the time slot t where $B_{i,t}$ is minimal
Step 5: Remove Blocked time slots from all remaining activities
Step 6: Remove activities that cannot be booked anymore from the list
Step 7: Start at Step 3 again until the list is empty

4.2. Multi-agent algorithm approach

In instances where the capacity is very tight it could be wise to do an assignment of activities to visitors without considering routing and time scheduling as an initialization of our algorithm. This assignment can easily be formulated as the following ILP:

$$\max \sum_{j \in A} \sum_{k \in U} S_k P_{i,k} y_{i,k} \tag{4a}$$

subject to:

$$1 \leq \sum_{i \in A} y_{i,k} \leq n_k \qquad \forall k \in U \tag{4b}$$

$$\sum_{k \in U} S_k x_{i,k} \leq C_i \qquad \forall i \in A \tag{4c}$$

$$y_{i,k} \in \{0, 1\} \qquad \forall k \in U, \forall i \in A$$

In this ILP, we have binary decision variables $y_{i,k}$ that determine whether visitor k goes to activity i. The input S_k is the group size of visitor group i with, $P_{i,k}$ is the reward visitor k obtains by visiting activity i like before. Constraint (4b) ensures that everyone will visit between 1 and n_k events, where the maximum number of assigned visits can be based upon the visitor level and the duration of the trip. Constraint (4c) ensures that the daily capacity C_i of activity i is respected. We calculate this daily capacity based on the available slots, where we maximize the number of total visitors while respecting $C_{i,s}$ and $C_{i,s}$. In the objective function, Eq. (4a), we include the group size to prevent the assignment from having a preference for small groups. It is worth noting that including a visitor level in the objective allows for diversification between different visitors if necessary. Although this is a potentially large ILP, our numerical experiments show negligible computational effects (see Section 5.4). Furthermore, since it is only used for initialization it is possible to drop the integer constraint on $x_{i,j}$. In this case, solving the relaxed version and rounding the variables up would speed this process up. Although this probably leads to an infeasible solution of this assignment problem, this will not be an issue for the algorithm since the addition of routing and scheduling constraints mean that some assigned activities need to be dropped anyway for some visitors.

For the final solution our goal is to have a trip for each visitor with a high average visitor happiness. Furthermore, it is considered beneficial to prevent especially bad trips since those visitors can in practice result in bad publicity. The outline of the algorithm is shown in Algorithm 4.1 below.

Algorithm 4.1 Multi-Agent Trip Planing Algorithm

```
1: Assign Activities to Visitors Using ILP
2: for all Visitors do
3:    Create Trip With Assigned Activities
4: end for
5: for i in { i_start, ..., i_max } do
6:    Sort Visitors on Trip Happiness
7:    for all Visitors ∈ VisitorList do
8:       Expand trip with one activity from the top i Activities
9:       if Expansion Not Successful and i ≥ I_max then
10:         Remove Visitor From VisitorList
11:      end if
12:   end for
13:   if VisitorList empty or Activities Fully Booked then
14:      break
15:   end if
16: end for
17: Sort Visitors on Trip Happiness
18: for all Visitors do
19:   Optimize Trip
20: end for
```

We solve the assignment problem (line 1) and use it to create a set of initial trips (line 3). Next on every iteration we expand this route with one activity they would like to visit (line 8). We start this expansion with i_{start} activities available, since computational experiments showed that allowing slightly more flexibility at the start (as an example, the top 3 instead of only the top 1) often resulted in better final results. The maximum i_{max} can be used to limit the computational effort. However, we used the number of activities in all our experiments to get the maximum solution quality. In case this fails we stop trying to improve this visitor if we are past iteration I_{max}, which is a system parameter (line 10). This prevents using a lot of computation time on trips that are impossible to improve further. After performing this operation for all visitors, we rank the visitors based on their current trip quality (line 6). In case we want to diversify between visitors, the visitor ranking is also used for sorting. In the next iteration we do the exact same operation for all visitors that still can be improved, until we either hit our maximum number of iterations (line 5), or there is no more capacity left (line 14) or all trips are fully booked. Finally, we go over all trips again to see if we can improve them further by using another preferred mode of transport or switching slots of activities to reduce (perceived) travel time (line 19).

An important part of our proposed algorithm is the expansion of trips from visitors at line 8 of Algorithm 4.1 which is done in every iteration. Therefore, it is important that this step is computationally very efficient. However, we should also prevent being stuck in a local optimum too soon. For that reason, we use the logic described in Algorithm 4.2 to expand trips. First, we always try to insert an extra activity using one of our insertion heuristics. Only in case this fails, we try to escape the local optimum using either full enumeration or ILS which are discussed in more detail in Section 4.3. Our computational experiments showed that for small instances full enumeration results in an optimal trip while being computationally efficient. In contrast, ILS only explores a limited set of solutions starting at our initial trip and is therefore

computationally bounded when there are more solutions possible. To that end, we use full enumeration if $i \leq I_{\text{enum}}$ and ILS otherwise. Based on computational experiments we usually set the threshold $I_{\text{enum}} = 6$, since at that point they require comparable computational efforts, while full enumeration quickly explodes for more activities.

4.3. Shake mechanisms

Greedy heuristics to create a route can potentially get stuck in a local optimum based on choices made earlier. Hence, we need a way to try to improve a route by changing some previous choices. For very small instances, the quickest and easiest way is full enumeration of all possible permutations of the itineraries. For larger instances, we apply the shake mechanism often seen in ILS since it is applicable to many variants of the orienteering problem as shown in Vansteenwegen et al. (2009b) and combines decent solution quality with good computational efficiency.

For a route with five destinations there are $5! = 120$ possible permutations of the visit and since not all activities have to be performed $\sum_{k=0}^{5} \binom{5}{k} k! = 326$ possible trips if we disregard variation in time slot selection. The resulting trip is the permutation with the highest reward for consecutive visits that can still be transformed into a feasible tour for visitor j. To transform a permutation into a tour, we use the last activity after which we can still reach the final destination in time. So if in permutation $A_j^1 \rightarrow A_j^2 \rightarrow A_j^3 \rightarrow A_j^4 \rightarrow A_j^5$ we cannot visit A_j^5 and reach L_j^{end} in time the resulting route will be $L_j^{\text{start}} \rightarrow A_j^1 \rightarrow A_j^2 \rightarrow A_j^3 \rightarrow A_j^4 \rightarrow L_j^{\text{end}}$. However, for computational performance we apply a simple early pruning principle that is guaranteed to not miss candidate solutions: If in a permutation as a result of visiting the activity at position i it is not possible to reach the final location in time all permutations with the same starting sequence including this activity are considered explored. This will not skip any candidate solutions since skipping the visit of this activity will be considered by another permutation that has this activity later in the sequence. Example: if in permutation $A_j^1 \rightarrow A_j^2 \rightarrow A_j^3 \rightarrow A_j^4 \rightarrow A_j^5$ activity A_j^3 cannot be reached in time the resulting route will be $L_j^{\text{start}} \rightarrow A_j^1 \rightarrow A_j^2 \rightarrow L_j^{\text{end}}$ and the permutation $A_j^1 \rightarrow A_j^2 \rightarrow A_j^3 \rightarrow A_j^5 \rightarrow A_j^4$ will not be considered. However, all options with A_j^1 and A_j^2 later in the route will be explore by other permutations.

In our algorithm we use ILS as described in Vansteenwegen et al. (2009b) on an existing route when insertion fails for all additional activities. Since the computational costs of this method can easily be controlled by limiting the number of times we execute this procedure. This method is discussed in more detail in Appendix.

Finally, we included an implementation of ACO as described by Verbeeck et al. (2017). This algorithm is known to be able to handle the same set of problem variants for a single agent as our proposed algorithm and usually produces good quality results. However, since this algorithm makes is harder to bound the computational effort without suffering too much in solution quality we did not use it in our proposed solution. We do use this mechanism to construct trips on a FCFS basis and include these as a benchmark.

In our proposed solution someone has to enter their preferences in order for our solution to work. We assume that in a system without coordination a visitor would at this moment book his entire trip. Hence, our FCFS uses the order of entering preferences as the order for FCFS. Note that this is not necessarily equal to order of arrival at the site. In our computational experiments this order will be reflected by the index of the visitor group.

Algorithm 4.2 Trip Expansion

```
Try Greedy Expansion Algorithm
if Expansion Not Successful then
    if i ≤ I_enum then
        Use ILS
    else
        Use Full Enumeration
    end if
end if
```

5. Computational results

This section consists of five parts. First, we investigate the solution quality and computational efficiency of our trip-construction heuristics. For this test we use our data from the case study in Amsterdam with random tourist preferences. Second, in Section 5.2 the algorithm is used on small instances where we compare the results of our method to a FCFS policy and a global optimum computed using ILP. In Section 5.3, we report on a large case study based on tourists in Amsterdam. In Section 5.4 we perform a set of experiments on randomly generated instances to see the impact of different input sizes on the algorithm performance. Finally, in Section 5.5 we study some examples where we expect our algorithm to be outperformed by FCFS. To ensure that the values of the rewards $r_{i,j}$ were similar between several visitors we used a ranked wish list for each visitor to compute $r_{i,j}$, the reward visitor j gets for visiting activity i as follows:

$$r_{i,j} = 100 \exp\left(\frac{x_{i,j}^2}{-B}\right) \tag{5}$$

In this equation $x_{i,j}$ is the *position on the wish list* and B is a parameter that determines how quickly the rewards deteriorate based on the position of the wish list. In our computational experiments, we used a value of $B = 20$, since commercial partners stated that this would generally reflect how visitors would feel about a scheduled trip. We only differed for the ILP where we used $B = 5$ since higher differences between the rewards allowed the ILP to prune more efficiently and compute optimal solutions for small instances.

Our solutions were implemented in C++. Gurobi (Gurobi Optimization, 2020) was used to solve (I)LPs. All experiments where run on a personal notebook with 16 GB of RAM and a Intel Core i7-7700HQ CPU.

5.1. Trip-construction

Since our algorithm needs to construct many trips it is important to compare several trip-construction algorithms with regard to solution quality and computational performance. Fig. 1 shows the solution quality for ACO, ILS, full enumeration with the top 6 (enum6) or top 8 (enum8) activities as discussed in Section 4.3 and our greedy insertion (insert) algorithm and blocking algorithm as discussed in Section 4.1. Furthermore, we also show the computational effort of these algorithms in Fig. 2. It is clear that full enumeration of the top eight activities results in the best results. However, this is also computationally the most expensive method. Furthermore, both enumeration algorithms benefit a lot from the method we use for computing the rewards. This means that the performance with regard to solution quality will not generalize well if the rewards are more equally distributed over a larger set of activities. The median solution quality of ACO, ILS, enumeration of the top six and our proposed blocking algorithm are very close, although the variance varies a bit. Furthermore, it is important to note that the blocking algorithm is computationally at least a factor ten less expensive than those others. Finally, the greedy insertion heuristic gives the worst solutions. However, it also requires the least computational effort.

In our large computational experiments, we have chosen the greedy insertion heuristic as our expansion method since the computational effort is by far the lowest. For short trips (with less than six possible activities), we use full enumeration as a shake mechanism since the computational effort is comparable to the other algorithms while guaranteeing an optimal trip given the small subset of activities allowed to

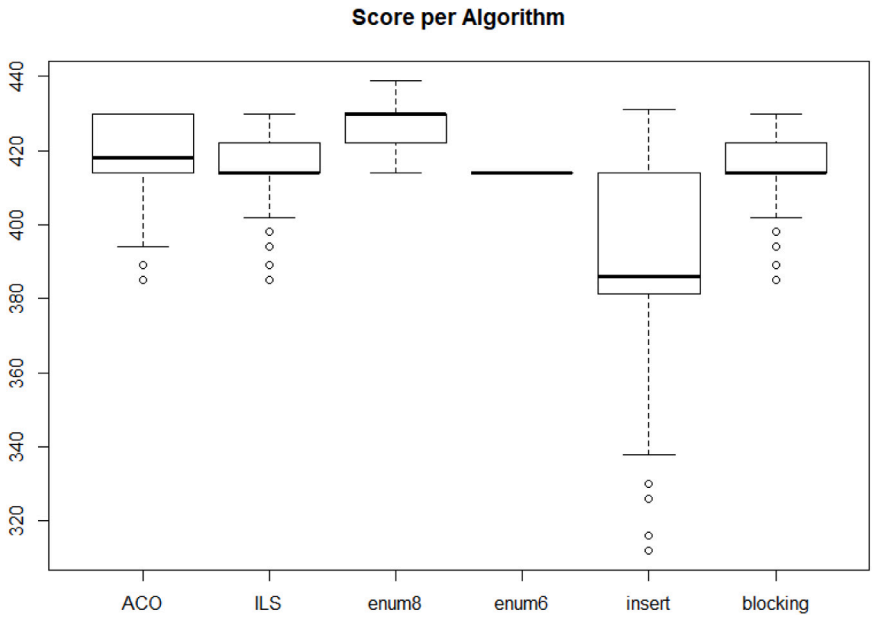


Fig. 1. Visitor happiness for several algorithms.

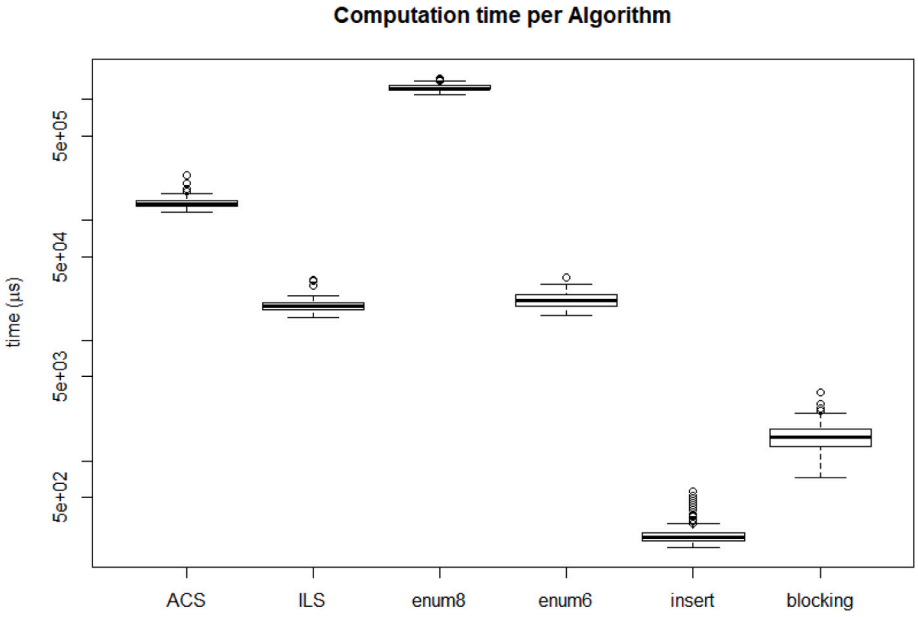


Fig. 2. Computation time for several algorithms.

be planned. We decided to use ILS for longer trips, since it uses the previously created trip and is computationally inexpensive. Although the blocking algorithm is creating comparable trips when used as a construction mechanism it cannot use the existing trip as a starting point for further improvements.

5.2. Comparison to optimal solutions

For a comparison between our algorithm and an optimal solution computed with ILP we used a small set of activities within the city of Amsterdam with real travel times on foot between those locations. Since this instance is relatively small we reduced the capacity and the number of slots available so the visitor capacity ratio remains similar to realistic instance sizes. A summary of our instance set up is shown in Table 1. The mentioned slots are evenly spread between the first and the last service and the duration is in minutes. All visitors have no travel time to their first activity or from their last activity. All trips start at 9:00 and end at 13:00, so all slots are available for all visitors. For our comparison we generated rewards for 30, 40 and 50 groups of travelers all traveling with a group size uniformly distributed between 1 and 3. For the test instance with 50 groups of travelers we doubled the capacity mentioned in Table 1. We generated a ranked wish list for the five activities for each group of visitors and calculated the associated rewards with Eq. (5). We generated these ranked wish lists in two ways. First, we generated these independent uniformly at random. Second, we generated them independent random but with similar preferences so that it was more likely that people had the same activities high on their wish list.

The results of the previous experiment for the ILP, our iterative algorithm *with* and *without* an initial assignment as described in Model (4a)–(4c) and FCFS are shown in Table 3. The FCFS policy delivers the worst performance on all instances. However, depending on the

Table 1
A summary of the instance used for the comparison.

Location	#Slots	First service	Last service	Duration	Slot capacity
ScheepvaartMuseum	3	11:00	12:20	40	10
Rijksmuseum	2	09:00	10:15	120	10
Museum Ons'Lieve Heer op Solder	4	10:00	12:15	30	10
Hermitage	1	11:00	11:00	45	20
Stedelijk Museum	4	09:00	10:30	90	10

Table 2
The objective function for the optimization model on random instances.

	ILP			Assignment			Iterative		
Instance	Found	Bound	FCFS	Objective	Gap Bound	Gap Found	Objective	Gap Bound	Gap Found
5-30-0-0	9364	10672	8866	9327	12.60%	0.40%	9352	12.37%	0.13%
5-30-0-1	9493	10478	8864	9358	10.69%	1.42%	9299	11.26%	2.04%
5-30-0-2	8932	9853	8475	8834	10.34%	1.10%	8606	12.66%	3.65%
5-30-0-3	9428	10583	9154	9399	11.19%	0.31%	9371	11.45%	0.60%
5-30-1-0	8794	9062	8095	8635	4.71%	1.81%	8169	9.86%	7.11%
5-30-1-1	8108	8484	7834	8356	1.51%	−3.06%	7829	7.72%	3.44%
5-30-1-2	8540	8836	8131	8095	8.39%	5.22%	8065	8.73%	5.57%
5-30-1-3	9032	10055	8314	8642	14.05%	4.32%	8652	13.96%	4.21%
5-30-2-0	9371	10094	8795	9355	7.32%	0.17%	9131	9.54%	2.56%
5-30-2-1	9799	9980	8920	9880	1.00%	−0.83%	9340	6.41%	4.68%
5-30-2-2	8684	9856	8381	8522	13.53%	1.86%	8692	11.81%	−0.10%
5-30-2-3	8610	9907	8108	8717	12.02%	−1.24%	8655	12.64%	−0.52%
5-50-0-0	12750	12876	11690	12687	1.47%	0.49%	12299	4.48%	3.54%
5-50-0-1	13385	13516	11877	13237	2.07%	1.11%	12589	6.86%	5.95%
5-50-0-2	12922	13181	12079	13015	1.26%	−0.72%	12470	5.39%	3.50%
5-50-0-3	12897	12994	11494	12592	3.10%	2.37%	12429	4.35%	3.63%
5-50-1-0	11843	11960	9180	11707	2.11%	1.15%	11786	1.45%	0.48%
5-50-1-1	11794	11912	8794	11500	3.46%	2.49%	11181	6.13%	5.20%
5-50-1-2	11253	11365	9101	11182	1.61%	0.63%	11139	1.99%	1.01%
5-50-1-3	11507	11622	8907	11152	4.05%	3.09%	10885	6.34%	5.41%
5-50-2-0	12389	12513	10881	12160	2.82%	1.85%	11616	7.17%	6.24%
5-50-2-1	12388	12512	10781	12204	2.46%	1.49%	12174	2.70%	1.72%
5-50-2-2	11959	12078	10723	12052	0.22%	−0.77%	11720	2.97%	2.00%
5-50-2-3	11684	11800	10262	11551	2.11%	1.13%	11756	0.38%	−0.62%
Average Gap					5.59%	1.07%		7.44%	2.98%

instance, the pure iterative algorithm or the algorithm that allocates first performs second best. On the tight instances with 40 visitors allocation first clearly outperforms sequential only. On the other instances, it varies which one of the two performs better. On these instances, both algorithms perform within 5% of the optimum found by the ILP.

To verify that our results are not caused by the network structure we also tested on randomly generated instances with 5 activities and 30 or 50 visitor groups. We generated 7 locations on a 100 by 100 grid of which one was used as start location and the other as final location. The time to travel in minutes between two locations is the euclidean distance. The other 5 indicated the locations of the activities. For each activity we randomly chose the capacity of the activity between 10 and 15, the capacity of a slot between 5 and 10, the duration and the time between slots where independently chosen between 15 and 120 min, the opening time between 8:00 and 10:00 and the closing time (last entry) between 13:00 and 14:00. All visitors visit between 8:00 and 17:00.

All instances were solved using the FCFS approach, the pure iterative strategy and the Assignment and than Iterative strategy. Each instance described in the table is first the number of activities, than the number of visitors, the network/activity instance and the visitor instance. The ILP was solved using Gurobi with a time limit of 4 h or an optimality gap of 1% whichever was reached first on the Dutch National Supercomputer Snellius. In Table 2 we show the bound and the best solution found for the objective function ($\sum_{i\in A}\sum_{m\in B_i}\sum_{k\in U} S_k P_{i,k} y_{i,m,k}$) for the ILP and compare that to FCFS and our two approached methods. For our methods we present also the gap between the bound and the best found solution of the ILP. These results are comparable to the results shown on the small instance based on our case study. It also shows that our approach works also well in other instances, it could even indicate that our created instance was relatively hard.

Table 3
Comparison of the algorithms for several instances.

# Visitor Groups		30	40	50	30	40	50
Wishlist		similar	similar	similar	random	random	random
# Visitors		63	78	99	63	78	99
FCFS	Score	8810	8699	15522	9973	10314	16225
	Gap	18.26%	19.39%	4.14%	10.85%	19.97%	10.81%
Pure Iterative	Score	10401	9919	15534	10755	11734	17864
	Gap	3.50%	8.08%	4.07%	3.86%	8.95%	1.80%
Iterative +	Score	10268	10167	15789	10704	12159	17840
Allocation	Gap	4.73%	5.78%	2.49%	4.32%	5.65%	1.93%
ILP	Score	10778	10791	16193	11187	12887	18192

To put these numbers in perspective, we compare them to Wang et al. (2017) and Luo et al. (2022) since their goal was similar to ours. The solution quality of Wang et al. (2017) seems to deteriorate with the number of visitors. Furthermore, their best performing algorithm is 6.1% from the optimal solution with just 20 visitors when the have the same preferences. However, the sequential algorithm described in that paper performs considerably worse than their Probabilistic ILS method on instances with variable rewards and 20 to 40 visitors. Since their sequential algorithm is the only one that scales well enough to handle problem instances, we consider the results of our proposed method very promising. Luo et al. (2022) only report on optimal solutions for the instances with 7 nodes (5 activities) and 10 visitors. For these instances, they have an optimality gap between 0 and 20%. Hence, our solution approach results in similar results on small scale instances. Furthermore, our approach scales considerably better than

Table 4
A summary of the configuration of the events used for testing.

Activity	Location	D	First Entry	Last Entry	SF	DC
Old maps of Amsterdam	Hermitage	60	10:00:00 A.M.	05:00:00 P.M.	20	880
Treasures from the Golden Age	Scheepvaartmuseum	40	10:00:00 A.M.	05:00:00 P.M.	20	1760
Rembrandt and the Golden Age	Rijksmuseum	120	09:00:00 A.M.	05:00:00 P.M.	20	2500
Portrait Gallery of the Golden Age	Hermitage	60	10:00:00 A.M.	05:00:00 P.M.	15	1160
Discover a church in the Attic	Museum Ons'Lieve Heer op Solder	30	10:00:00 A.M.	06:00:00 P.M.	10	2940
Discover the world in a painting	Hermitage	45	11:00:00 A.M.	02:00:00 P.M.	60	120
The influence on maritime history	Scheepvaartmuseum	90	10:00:00 A.M.	05:00:00 P.M.	15	1160
Architecture by Boat	Lovers	60	09:00:00 A.M.	10:00:00 P.M.	15	1750
Avondje Rembrandt	Rijksmuseum	120	07:00:00 P.M.	08:00:00 P.M.	60	300
Best Windmill in Amsterdam	Gooyer	15	09:00:00 A.M.	05:00:00 P.M.	20	250
Discover the invisible life at Micropia	Micropia	75	10:00:00 A.M.	05:00:00 P.M.	15	2900
Verzetsmuseum Amsterdam	Dutch Resistance Museum	120	10:00:00 A.M.	05:00:00 P.M.	15	900
Vintage in Foam	Foam Amsterdam	90	10:00:00 A.M.	06:00:00 P.M.	20	5000
The Secret Annex of Anne Frank	Anne Frank House	45	09:00:00 A.M.	07:00:00 P.M.	15	2050
Portraits of Modernity	Huis Marseille	90	11:00:00 A.M.	06:00:00 P.M.	15	4250
Exhibition 'Welcome Today'	Stedelijk Museum	45	10:00:00 A.M.	06:00:00 P.M.	5	4850
The inspirator of van Gogh	van Gogh Museum	75	09:00:00 A.M.	06:00:00 P.M.	15	1640
A multi-headed Snake	Cobra Museum	60	11:00:00 A.M.	05:00:00 P.M.	20	950
The Netherlands in World War II	Dutch Resistance Museum	90	10:00:00 A.M.	05:00:00 P.M.	10	435
Plantage Area Walk	Dutch Resistance Museum	45	10:00:00 A.M.	05:00:00 P.M.	30	450
Banksy Laugh Now	Moco Museum	60	09:00:00 A.M.	08:00:00 P.M.	10	3450
Visit the Batavia	BataviaLand	60	09:00:00 A.M.	05:00:00 P.M.	10	980
Batavia Museum	Batavia Museum	60	09:00:00 A.M.	05:00:00 P.M.	10	980
Dutch Windmills	Zaanse Schans	20	09:00:00 A.M.	05:00:00 P.M.	20	1000
Dutch Flowers	Keukenhof	20	09:00:00 A.M.	05:00:00 P.M.	20	1000

Table 5
The highlighted tour, #WL indicates the position of the activity on the wishlist.

	Activity	#WL	Location	Start	End	Walk time
1.	Discover the invisible life at Micropia	2	Micropia	10:15	11:30	11 min
2.	Discover the world in a painting	1	Hermitage	12:00	12:45	12 min
3.	Vintage in Foam	6	Foam	13:00	14:30	14 min
4.	Banksy Laugh Now	7	Moco Museum	15:30	16:30	1 min
5.	The inspirator of van Gogh	3	Van Gogh Museum	16:45	18:30	28 min
6.	The Secret Annex of Anne Frank	4	Anne Frank House	19:00	19:45	

their proposed methods. Finally, we confirmed that the correlation between happiness and group size in our solution was close to zero in all solution methods. Thus, confirming that none of the algorithms optimized unfairly in this regard.

5.3. Real-life use case: The city of amsterdam

In cooperation with commercial partners in the region of Amsterdam we got a realistic set up for tourist activities in the region. A summary of the activities is shown in Table 4. In this table, the duration of the activity in minutes is in column D, SF denotes the interval of the arrival slots in minutes and DC is the maximum daily capacity. The last four tourist destinations in the table are outside of Amsterdam (in a range of 20 to 60 km from the city center) to test the multi-modal functionality of the algorithm. We used the walking time between the locations according to OSRM[1] as travel time. For our multi-modal test, we added a transport option. For this option, we took the travel time by car from the same source and added flat 15 min. We used our algorithm to schedule 2500 till 10000 groups of tourists with group sizes varying between 1 and 4. We generated their preferences at random with a large preference for the Rijksmuseum, the Anne Frank House and the Van Gogh museum, which reflect the true preferences of tourists.

In Fig. 3, we show twenty of the generated tours on a map of Amsterdam with one trip highlighted in orange. For clarity, we show the activities (including their position on the wish list) and travel time of that trip in Table 5. Based on this table, we can see that the trip scheduled a lot of activities the visitor wanted to do. The only thing skipped from the top 7 of their wish list is 'The influence on maritime history' in the Scheepvaartmuseum which was ranked number 5. Finally, the route between the activities is sensible, which will increase the acceptance of the suggested trip.

In Fig. 4, we show a comparison of FCFS and our scheduling algorithm. It clearly shows that with FCFS the first visitors get very nice trips. However, the quality of the trips quickly deteriorates as events get fully booked. As a result, the average happiness is considerably lower for FCFS than with our approach. Furthermore, the distribution of visitor happiness is also a lot more fair since the minimum happiness is a lot higher and low happiness occurs less frequent. Fig. 5 illustrates very well that the distribution of visitor happiness is also more fair. As a result, the variability in happiness is reduced, three quarters of the visitors are happier with their trip than the median using a FCFS policy, and only a few negative outliers of our algorithm perform in the worst 25% range of FCFS which results in a far lower maximum of the minimum.

To validate that our algorithm also works for *multi-modal* transport options, we ran a test where we included the last four activities of Table 4. In this case, the algorithm is slightly slower taking about 25 s instead of around 20 s for 2500 visitors. This can be explained by multi-modal trip generation taking slightly longer, but is also caused by the higher capacity due to the four extra activities. However, the results we obtained are similar as can be seen in Fig. 5, where we see that the distribution of visitor happiness for both FCFS and our proposed algorithm is similar to the single mode case used for the Amsterdam region. In this case, the algorithm performs even slightly better in comparison since it makes better use of the extra capacity and opportunity to diversify provided by the extra activities. We also used our approach on an instance with required visits and using our greedy blocking heuristic for the initial trip creation. We got similar results with regards to computational effort and solution quality.

[1] see http://project-osrm.org/.

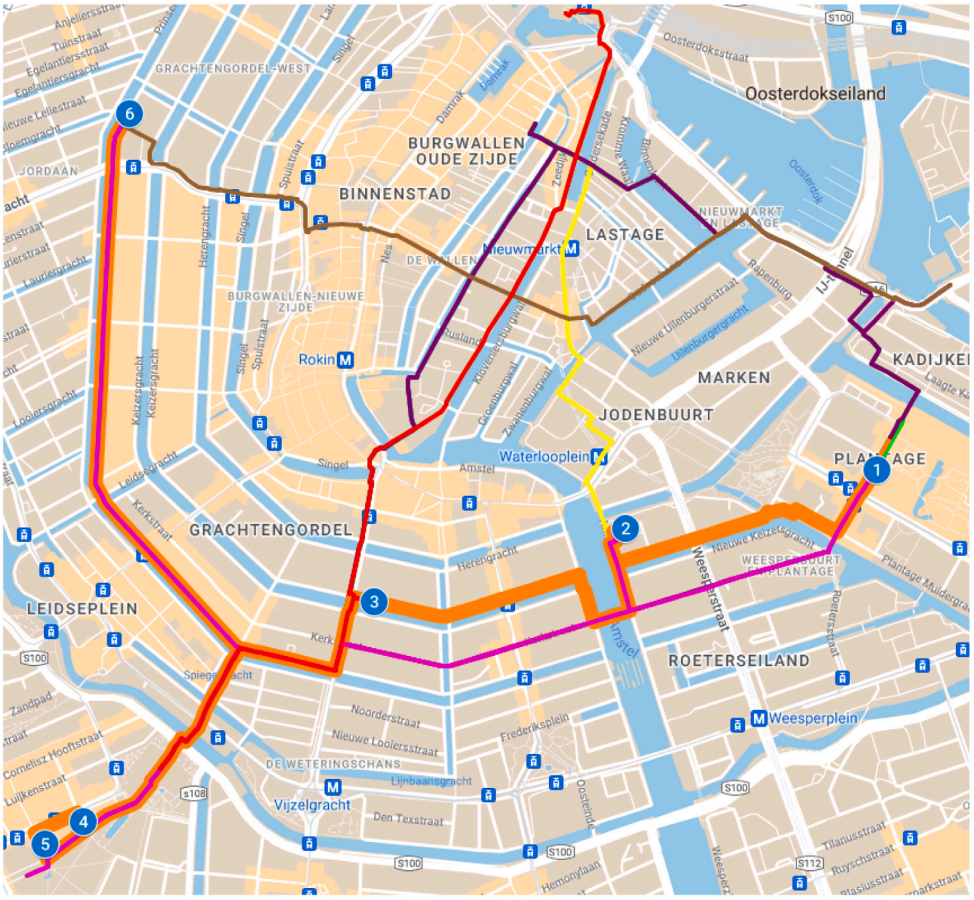


Fig. 3. A selection of generated trips (each in different colors), the trip in fat orange is explained in Table 5, the numbers indicate the activities of that trip. Map data ©2022 Google.

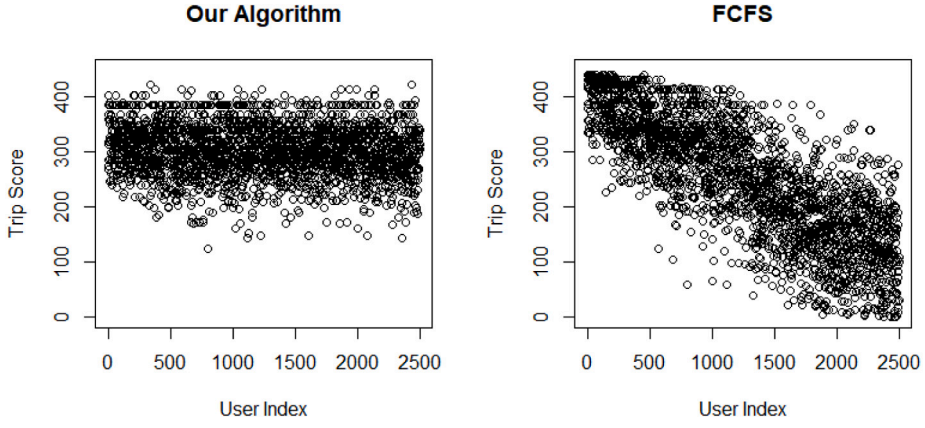


Fig. 4. The Scores per visitor (note that in FCFS the trips are generated in order of index).

5.4. Algorithm performance on varying instance sizes

One of the focus points of our approach is scalability. Hence, we investigate the computational performance for varying input sizes. To that end we generated activities with associated parameters randomly and used our algorithm on them. The results for increasing the number of activities while keeping the number of visitors at 500 is shown on the left in Fig. 6. This figure makes clear that adding activities has almost no effect on the computation time. Since our algorithm aims to insert activities in order of their reward most of these extra activities can be ignored by individual insertion since visitors do not have time to attend them. The extra overhead for keeping track of the reservations and capacity is negligible. So the available time per trip can be considered bounding for the computational effort.

Even more important is the effect of the number of visitors on the computation time, as for most applications these could vary quiet a bit on a daily basis. To test this effect we generated activities with random but regular slot patterns and random travel times. Furthermore, we generated visitors with random preferences and kept track of the computational effort to generate all tours. In Fig. 6 we notice that computational effort first increases linearly and stabilizes at some point and becomes constant. When the number of visitors becomes extreme compared to the capacity, the computational effort even starts to decrease. The intuition behind the first effect is that the computational

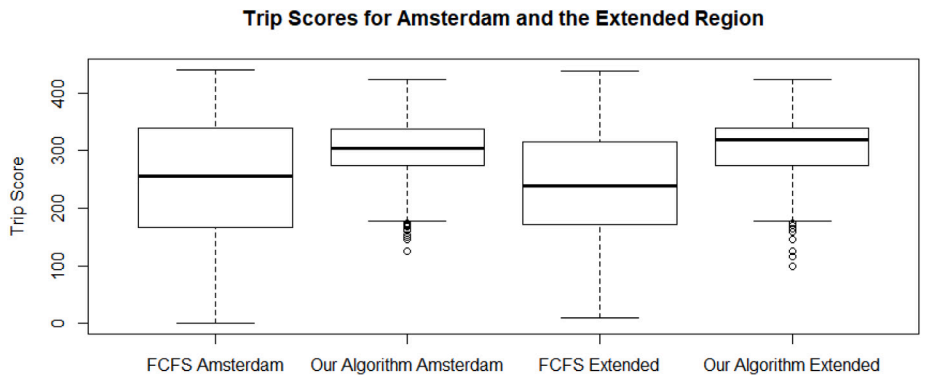


Fig. 5. The boxplot for the trip happiness using FCFS and our proposed scheduling algorithm in Amsterdam and an extended area where multi-modal transport is used.

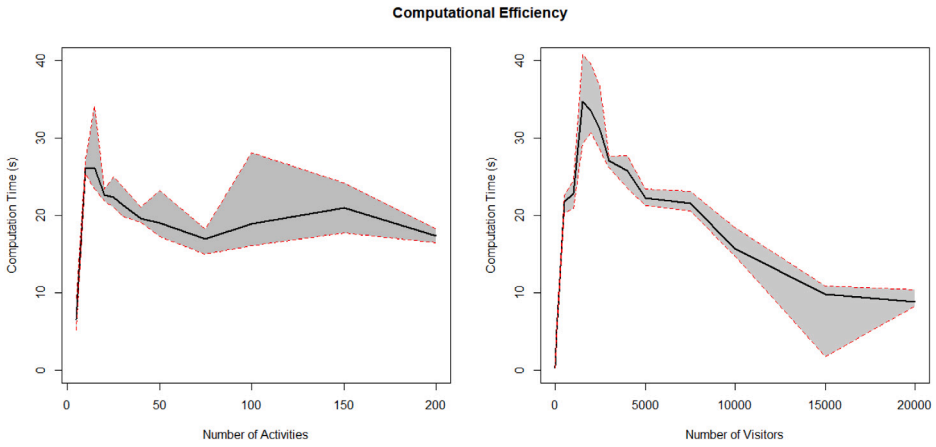


Fig. 6. The effect of the problem size on computation time.

effort is limited by the capacity since at one point all events are fully booked and it is not possible to improve the tours of any visitor anymore. The second effect happens since improving short tours is computationally a lot easier than expanding longer tours. Therefore, the computational effort of our algorithm decreases when the tours become extremely short due to overcrowding. In our computational experiment we witnessed the same effect for higher capacity problems when the number of visitors got sufficiently high. In practice this will mean that although the algorithm might be faster on very quiet days it will not take excessively long on busy days since on those days capacity will be the limiting factor.

5.5. Hard instances

The algorithm we propose uses specific properties of the problem to reach good results. First, we assume that there is variability in the preference of visitors. Second, we assume that capacity is scarce. However, in situations where these conditions do not apply our algorithm can be further from optimal or even be outperformed by a FCFS policy. In situations with homogeneous preferences our algorithm will be comparable or even slightly worse to FCFS with regards to average visitor happiness since the reward on each activity is exactly the same for each visitor. However, our method should still performs better with regard to fairness since it will try to ensure that everyone can at least visit some activities. The result of an instance with homogeneous preferences is shown in Fig. 7 where we show the trip scores per visitor. As expected the first visitors with FCFS are happy, but scores start to go down when activities get fully booked. In this case the average trip happiness score is almost identical, but our proposed solution is clearly fairer.

To illustrate what will happen when capacity is not scarce, we use our approach on an instance with abundant capacity. In this instance, it would probably be better or just as good to construct full tours after each other in a FCFS manner since early bookings will not block later arrivals at all. To illustrate this we did a test with just 100 visitors for our Amsterdam instance. The result of this test is shown in Fig. 8, where there is indeed almost no difference between FCFS and our proposed algorithm. In cases like these using more sophisticated and computationally intensive methods to create individual trips could make this difference more significant.

6. Conclusion and future research

In this paper we outlined an approach to generate leisure trips at crowded destinations which include reservations for activities. The main advantages of our approach are as follows:

1. Average visitor happiness increase substantially compared to a FCFS benchmark
2. On small instances our approach produces good results compared to optimal solutions (optimality gap of about 5%)
3. Our solution scales well with regards to increases in the problem size with regards to number of activities and visitors

Furthermore, we propose a novel greedy blocking algorithm to solve the orienteering problem with fixed time slots. We show that this algorithm performs very well with regard to efficiency while still providing good solutions. In tight instances, our approach benefits from an initial assignment of activities to visitors, where the assigned activities are the only activities a visitor is allowed to use in his trip in the first round of our iterative approach. Finally, we have used our proposed solution

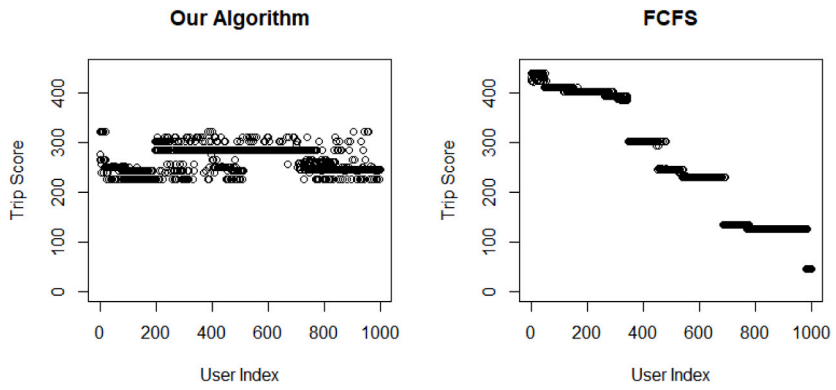


Fig. 7. Our algorithm compared to FCFS when everybody has the exact same preferences.

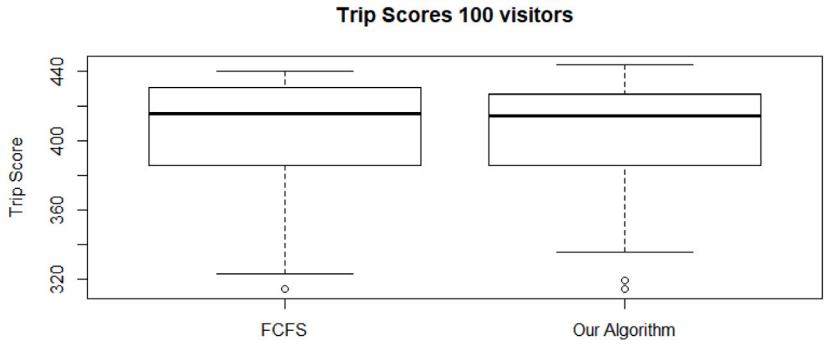


Fig. 8. Visitor happiness with only 100 visitors.

approach on several variations of the tourist trip design problem such as required visits, multi-modal transport and entries or exits with limited capacity (e.g. ticket booth or ferry). In these cases the approach works with slight modifications to the initial trip creation (Algorithm 4.1 line 3) or the trip expanding method (Algorithm 4.1 line 8), but the conclusions with regard to solution quality compared to FCFS and computational complexity still stand.

Our approach shows that a combination of coordination and customization for leisure trips has potential to increase visitor satisfaction. In practice this combination could be used in cities that suffer from over-tourism, theme parks and large events to facilitate a proper spread of visitors. Furthermore, we note that this approach will remain valid if only part of the capacity is reserved for a centralized agent to still allow non-participants to be in the system. Since our trip design mechanism was tested with required visits it can also work with tourist groups that already made some reservations before joining the system. In this case the system will book a trip around their existing reservations.

In practice algorithms that facilitate spreading visitors over time can be combined with recommendation systems to steer demand away from over-crowded highlights in the preparation phase. While people are on their trip it would also be nice if an application can continuously guide them. Furthermore, having an algorithm in these applications that also facilitates coordination and customization makes it possible to continuously improve trips. Since in reality there are always no-shows, last minute cancellations, and early and late arrivals, it would help out visitors as well as service providers if their trips can be updated in real time. Our approach is fast enough that it can be used by such systems.

These options leave open possible research opportunities for online variants of our solution approach. This approach also allows creation or updates of trips at the moment visitors make their booking. Furthermore, such mechanisms can also be used to facilitate coordination of visitors through suggestions. Since at the moment a good trip for a visitor considering other preferences is known, it is possible to steer them towards that trip. Since leisure visitors often do not want to have their entire day planned they usually want a combination of flexibility and guarantees. Our current solution gives a lot of guarantees, but has no flexibility. This is another point that can be improved upon by online variants that throughout the day continuously extend the trip of a visitor.

CRediT authorship contribution statement

Joris Slootweg: Writing – review & editing, Writing – original draft, Software, Methodology, Conceptualization. **Rob van der Mei:** Writing – review & editing, Supervision, Conceptualization. **Caroline J. Jagtenberg:** Writing – review & editing, Supervision. **Frank Ottenhof:** Writing – review & editing, Funding acquisition, Conceptualization.

Data availability

Data will be made available on request.

Acknowledgments

The authors thank Jesse Nagel for his useful comments to earlier drafts. We would also like to thank the anonymous reviewer for his comments that helped improve the paper significantly. Part of this work was sponsored by the Dutch Research Council (NWO), Netherlands, project number 18938.

Appendix. Iterated local search

In our implementation of ILS which is based on Vansteenwegen et al. (2009b), we use the insertion mechanism described in 4.1 to construct and modify trips. Without an initial trips, the insertion procedure generates an initial trip. This initial trip is the start of the ILS algorithm. When it is not possible to add any activity to the trip, a local optimum is reached. When this happens the algorithm performs a shake step.

During the shake step activity on position S till $S + R$ are removed from the trip. All activities scheduled after position $S + R$ are moved as far as possible to the start of the trip. Example: if we have a tour with 4 activities with these starting times and we remove activity 1 and 2 $A_j^1(9:00) \rightarrow A_j^2(11:30) \rightarrow A_j^3(13:00) \rightarrow A_j^4(15:00)$ we will get $A_j^1(9:00) \rightarrow A_j^4(15:00)$ on which we try to move A_j^4 as much as possible to the start while remaining feasible with respect to travel time and capacities. This could for example result in $A_j^1(9:00) \rightarrow A_j^4(10:30)$, which can be used to expand upon again and explore a different part of the solution space. On such a trip we use our insertion heuristic again till a new local optimum is reached. The outline of the algorithm is provided in Algorithm A.1.

Algorithm A.1 Iterated Local Search

```
S = R = 1
bestRoute=route⟵input
bestScore=score⟵input
for ITERATIONS do
  while not local optimum do
    route,score=insert
  end while
  if score>bestScore then
    bestScore=score
    bestRoute=route
    S = R = 1
  end if
  while S ≥ route.size do
    S− = route.size
  end while
  if S + R > route.size then
    R = 1
  end if
  route,score=shake(S, R)
  S+ = R
  R++
end for
```

References

Chen, C., Cheng, S.-F., Lau, H.C., 2014. Multi-agent orienteering problem with time-dependent capacity constraints. Web Intell. Agent Syst. Int. J. 12 (4), 347–358. http://dx.doi.org/10.3233/WIA-140304.

Dichter, A., Manzo, G.G., 2017. Coping with Success: Managing Overcrowding in Tourism Destinations. World Travel & Tourism Council, London.

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. http://dx.doi.org/10.1016/j.eswa.2015.05.046, URL https://linkinghub.elsevier.com/retrieve/pii/S0957417415003826.

Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G., 2014. A survey on algorithmic approaches for solving tourist trip design problems. J. Heuristics 20 (3), 291–328. http://dx.doi.org/10.1007/s10732-014-9242-5, URL http://link.springer.com/10.1007/s10732-014-9242-5.

Goodwin, H., 2017. The challenge of overtourism. Res. Tour. Partnersh. 4, 1–19.

Gunawan, A., Lau, H.C., Vansteenwegen, P., 2016. Orienteering problem: A survey of recent variants, solution approaches and applications. European J. Oper. Res. 255 (2), 315–332. http://dx.doi.org/10.1016/j.ejor.2016.04.059, URL https://linkinghub.elsevier.com/retrieve/pii/S037722171630296X.

Gurobi Optimization, L., 2020. Gurobi optimizer reference manual. URL http://www.gurobi.com.

Kenteris, M., Gavalas, D., Economou, D., 2011. Electronic mobile guides: A survey. Pers. Ubiquitous Comput. 15 (1), 97–111. http://dx.doi.org/10.1007/s00779-010-0295-7.

Labadie, N., Melechovskỳ, J., Wolfler Calvo, R., 2011. Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. J. Heuristics 17, 729–753. http://dx.doi.org/10.1007/s10732-010-9153-z.

Luo, X.-G., Liu, X.-R., Ji, P.-L., Shang, X.-Z., Zhang, Z.-L., 2022. Trip planning for visitors in a service system with capacity constraints. Comput. Oper. Res. 148, 105974. http://dx.doi.org/10.1016/j.cor.2022.105974.

Mukhina, K.D., Visheratin, A.A., Nasonov, D., 2019. Orienteering problem with functional profits for multi-source dynamic path construction. PLoS One 14 (4), e0213777. http://dx.doi.org/10.1371/journal.pone.0213777, URL https://dx.plos.org/10.1371/journal.pone.0213777.

Righini, G., Salani, M., 2009. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. Comput. Oper. Res. 36 (4), 1191–1203. http://dx.doi.org/10.1016/j.cor.2008.01.003, URL https://linkinghub.elsevier.com/retrieve/pii/S030505480800004X.

Robino, D.M., 2019. Global destination cities index 2019. MasterCard. Available online: https://www.mastercard.com/news/media/wexffu4b/gdci-global-report-final-1.pdf. (Accessed on 16 December 2022).

Rubin, J., 2019. TEA/AECOM 2019 Theme Index and Museum Index: The Global Attractions Attendance Report. Themed Entertainment Association (TEA).

Ruiz-Meza, J., Brito, J., Montoya-Torres, J.R., 2021. A GRASP to solve the multi-constraints multi-modal team orienteering problem with time windows for groups with heterogeneous preferences. Comput. Ind. Eng. 162, 107776. http://dx.doi.org/10.1016/j.cie.2021.107776.

Ruiz-Meza, J., Montoya-Torres, J.R., 2021. Tourist trip design with heterogeneous preferences, transport mode selection and environmental considerations. Ann. Oper. Res. 305 (1–2), 227–249. http://dx.doi.org/10.1007/s10479-021-04209-7.

Ruiz-Meza, J., Montoya-Torres, J.R., 2022. A systematic literature review for the tourist trip design problem: Extensions, solution techniques and future research lines. Oper. Res. Perspect. 100228. http://dx.doi.org/10.1016/j.orp.2022.100228.

Testa, L., Dozier, G., 1999. Evolving efficient theme park tours. J. Comput. Inf. Technol. CIT 7, https://hrcak.srce.hr/150201.

Vansteenwegen, P., Souffriau, W., Berghe, G.V., Van Oudheusden, D., 2009a. A guided local search metaheuristic for the team orienteering problem. European J. Oper. Res. 196 (1), 118–127. http://dx.doi.org/10.1016/j.ejor.2008.02.037.

Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D., 2009b. Iterated local search for the team orienteering problem with time windows. Comput. Oper. Res. 36 (12), 3281–3290. http://dx.doi.org/10.1016/j.cor.2009.03.008, URL https://linkinghub.elsevier.com/retrieve/pii/S030505480900080X.

Varakantham, P., Mostafa, H., Fu, N., Lau, H.C., 2015. DIRECT: A scalable approach for route guidance in selfish orienteering problems. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. AAMAS '15, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, pp. 483–491, http://dl.acm.org/citation.cfm?id=2772942.

Verbeeck, C., Sörensen, K., Aghezzaf, E.-H., Vansteenwegen, P., 2014. A fast solution method for the time-dependent orienteering problem. European J. Oper. Res. 236 (2), 419–432. http://dx.doi.org/10.1016/j.ejor.2013.11.038, URL https://linkinghub.elsevier.com/retrieve/pii/S0377221713009557.

Verbeeck, C., Vansteenwegen, P., Aghezzaf, E.-H., 2017. The time-dependent orienteering problem with time windows: A fast ant colony system. Ann. Oper. Res. 254 (1–2), 481–505. http://dx.doi.org/10.1007/s10479-017-2409-3, URL http://link.springer.com/10.1007/s10479-017-2409-3.

Wang, W., Lau, H.C., Cheng, S.-F., 2017. Exact and heuristic approaches for the multi-agent orienteering problem with capacity constraints. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, Honolulu, HI, pp. 1–7. http://dx.doi.org/10.1109/SSCI.2017.8285329, URL http://ieeexplore.ieee.org/document/8285329/.