A Mathematical Model and GRASP for a Tourist Trip Design Problem

D. R. Santos-Peñate[1(✉)], J.A. Moreno-Pérez[2], C.M. Campos Rodríguez[2], and R. Suárez-Vega[1]

[1] Dpto de Métodos Cuantitativos en Economía y Gestión/TIDES, Universidad de Las Palmas de G.C., 35017 Las Palmas de Gran Canaria, Spain
{dr.santos,rafael.suarez}@ulpgc.es

[2] Instituto Universitario de Desarrollo Regional, Universidad de La Laguna, San Cristóbal de La Laguna, Spain
{jamoreno,ccampos}@ull.edu.es

Abstract. We consider a tourist trip design problem with time windows and recommended occupancy levels at the points of interest. A 3-objective optimization model is formulated where the objectives are to maximize the total score and minimize total over occupancy and time gap. The multiobjective optimization problem is modeled as a mixed integer linear mathematical program. A GRASP is proposed to solve the problem.

Keywords: Tourist trip design problem · Occupancy · Mathematical programming · Multiobjective optimization · Heuristic

1 Introduction

In tourist routing problems and at certain points of interest (POIs) crowds of visitors can occur. The satisfaction experienced by the tourist and the image projected by the POI can be negatively impacted in such situations.

We study a problem of designing a tourist route with time windows where, in addition to maximizing tourist satisfaction and given recommended levels of occupancy at the POIs, the aim is to minimize both over occupancy and *lost time* in the route. Lost time occurs when the amount of time used to do the route is higher than the length of the route, defined as the minimum time required to do the route (travel time plus visit time in the best conditions). Lost time can be interpreted as waiting time. We formulate a mixed integer linear programming model to solve a TTDP with time windows and recommended occupancy levels for POIs.

The background to address the problem is the research that has been conducted on the Tourist Trip Design Problem (TTDP) found in the literature. Some surveys of the TTDP can be found in [4,6,7]. A review of algorithms proposed to solve the TTDP can be found in [3,6].

This research is partially financed by Gobierno de España, grant GOB-ESP2019-07, and Gobierno de Canarias, grant COVID-19-04.

R. Moreno-Díaz et al. (Eds.): EUROCAST 2022, LNCS 13789, pp. 112–120, 2022.
https://doi.org/10.1007/978-3-031-25312-6_13

The construction of algorithms that provide solutions in a very short time is especially relevant. Some articles in this respect include [1,2]. Proposals to balance the visits to the POIs and avoid congestion are presented in several works found in the literature [5,8] but, dislike our work, recommended occupancy levels for the POIs are not explicitly considered in the model.

2 Problem Statement and Mathematical Model

Consider a network $N(V, E)$ where $V = \{v_1, ..., v_n\}$ is the set of nodes and $E \subseteq V \times V$ is the set of edges. The travel time between nodes v_i and v_j is denoted $\bar{t}_{ij}$, it is the length of the shortest path (faster path) from v_i to v_j. A route is a sequence of nodes, $v_1, v_{i1}, ..., v_{ir}, v_n$, nodes v_1 and v_n are the initial and final points in the route. For a route, the initial and final points can be the same. Each node v_i, $1 < i < n$, is a point of interest (POI) and it has a score $s_i > 0$ which represents its attractiveness as a point to be visited. The POI v_i has opening and closing times L_i and U_i respectively, t_i is the duration of the visit and cap_i is the capacity of POI v_i. The recommended occupancy at POI v_i is c_i, it is desirable that at most c_i visitors enter point v_i at time τ ($c_i = \alpha_i \times cap_i$ where $0 \leq \alpha_i \leq 1$). And ρ_i is the contribution of a new visit to occupancy at POI v_i.

The length of a route, denoted T_r, is the summation of the total travel time and the total visit time. The time budget or available amount of time to do the route is T_{max}. Moreover, initial and final times, T_I and T_F, are given, so that the route should be done in the time interval $[T_I, T_F]$. We have that $T_{max} \leq T_F - T_I$, and τ_1, τ_n represent the starting route time and ending route time respectively. The lost time or gap is the difference between the duration of the route, given by T_r, and the time used to do it, $T_u = \tau_n - \tau_1$.

The function $\gamma_i(\tau)$ represents the occupancy level at instant τ. In order to define the occupancy function for a POI v_i in the interval $[L_i, U_i]$, we consider that the occupancy for a discrete set of instants $L = \overline{\tau}_1 < \overline{\tau}_2 <, ..., \overline{\tau}_q = U$ are known. The occupancy function is given by a piecewise linear function built from the known occupancy values.

We want to find the route from the initial point v_1 to the final point v_n which maximizes the total score, minimizes the total over occupancy, minimizes the lost time, and satisfies the time limitations constraints.

2.1 The Model

To state the model we use the following sets, and variables.

Index sets:

$I = 1, ..., n$: index set corresponding to node set V.
$I_1 = I \setminus \{1, n\}$: index set I excluding initial and final nodes in the route.
$K = \{1, ..., q\}$: index set for time partition.

Variables:

$x_{ij} = 1$ if in the route we go from point v_i to point v_j, otherwise $x_{ij} = 0$.
τ_i: instant at which POI v_i is reached.
$\gamma_i(\tau)$: occupancy at v_i in time τ.
u_i: variables introduced in order to avoid subtours.
z_i: over occupancy at v_i.
a_{ik}: variable used to define the occupancy function, for v_i and $k \in K$.
y_{ik}: binary variable for the occupancy function, for v_i and $k = 1, ..., q-1$.
Then the problem is stated as follows:

$$\min\Big(-\sum_{(i,j)\in I_1\times I} s_i x_{ij},\ \sum_{i\in I_1} z_i,\ (\tau_n - \tau_1) - \sum_{i,j\in I}(t_i + t_{ij})x_{ij}\Big) \tag{1}$$

$$\sum_{j\neq 1} x_{1j} = 1 \qquad \sum_{i\neq n} x_{in} = 1 \tag{2}$$

$$\sum_{i\neq n} x_{ik} = \sum_{j\neq 1} x_{kj}, \qquad \forall k \neq 1, n \tag{3}$$

$$\sum_{i\neq n} x_{ik} \leq 1, \qquad \forall k \neq 1, n \tag{4}$$

$$\tau_n - \tau_1 \leq T_{max} \tag{5}$$

$$L_i \leq \tau_i \leq U_i, \qquad \forall i \neq 1, n \qquad T_I \leq \tau_i \leq T_F, \qquad \forall i \tag{6}$$

$$\tau_i + t_i + t_{ij} \leq \tau_j + M(1 - x_{ij}), \qquad \forall i, j \tag{7}$$

$$\tau_i = \sum_{k=1}^{q} a_{ik}\overline{\tau}_k,\ \forall i \tag{8}$$

$$a_{i1} \leq y_{i1},\ \forall i \quad a_{ik} \leq y_{i,k-1} + y_{ik},\ \forall i,\ k = 2, ..., q-1 \quad a_{iq} \leq y_{i,q-1},\ \forall i \tag{9}$$

$$\sum_{k=1}^{q} a_{ik} = 1,\ \forall i \qquad \sum_{k=1}^{q-1} y_{ik} = 1;\ \forall i \qquad \gamma_i = \sum_{k=1}^{q} a_{ik}\gamma_i(\overline{\tau}_k),\ \forall i \tag{10}$$

$$z_i \geq (-c_i + \gamma_i + \rho_i) - M\Big(1 - \sum_{j\in I} x_{ij}\Big),\ \forall i \neq 1, n \tag{11}$$

$$u_i - u_j \leq (n-1)(1 - x_{ij}) - 1, \qquad i, j \neq 1, i \neq j \tag{12}$$

$$2 \leq u_i \leq n, \qquad \forall i \neq 1 \tag{13}$$

$$\begin{array}{ll} x_{ij} \in \{0,1\},\ \forall i, j; & 0 \leq z_i \leq cap_i - c_i,\ \forall i \neq 1, n, \\ a_{ik} \geq 0,\ \forall i, k \in \{1, ..., q\}, & \gamma_i \geq 0, \forall i \\ y_{ik} \in \{0,1\},\ \forall i, k \in \{1, ..., q-1\}. & \end{array} \tag{14}$$

Algorithm 1. Construction phase

Set μ with $0 < \mu < 1$
$R = [1, n]$
$I_1(R) = \{1\}$
$\tau_1 = \tau_n = T_I$
$T_u = T_r = 0$
while *a new POI can be inserted* **do**
  $CL = \emptyset$
  for $i \in I_1 \setminus I_1(R)$ **do**
    $CL = CL \cup \{(j(i), i)\}$
  end
  if $CL = \emptyset$ **then**
    a new insertion is not possible. Break While
  end
  Set $p^* = \max_{\{i:\exists(j,i)\in CL\}} p_i$
  Construct the Restricted Candidate List RCL as follows:
  $RCL = \left\{(j, i) \in CL : p_i \geq \mu \times p^*\right\}$
  while *an insertion is not done and* $RCL \neq \emptyset$ **do**
    Select at random an element (j, i) from RCL
    if *insertion of* i *is feasible* **then**
      Update route R by insertion of i after node j, and
      $I_1(R) = I_1(R) \cup \{i\}$
    else
      $RCL = RCL \setminus \{(j, i)\}$
    end
  end
end

If the initial and final points coincide, in the formulation v_n represents node v_1 considered as the final point in the route. Expression (1) represents the objectives: maximizing the total score, minimizing over occupancy, and minimizing the lost time or gap. Constraints (2) indicate that v_1 and v_n are the initial and final points, respectively. Expression (3) represents the flow conservation conditions. Constraints (4) indicate that a POI is visited at most once. The time budget limitation is included in constraint (5) and the time windows restrictions are constraints (6). The conditions on the sequence of times are incorporated in expression (7). Constraints (8) to (10) define the occupancy function, and constraints (11) define z_i as the over occupancy at point v_i, with M representing a large number. Constraints (12) and (13) are included to avoid subtours. Expressions (14) are the domain constraints.

Algorithm 2. Insertion procedure
Insertion of node i between j and k in route R which includes arc (j, k)

$\tau'_i = \max\{\tau_j + t_j + \bar{t}_{ji}, L_i\}$, $\Delta T_u = (\tau'_i + t_i + \bar{t}_{ik}) - \tau_k$
$T'_r = T_r + t_i + \bar{t}_{ji} + \bar{t}_{ik} - \bar{t}_{jk}$, $T'_u = T_u + \Delta T_u$
if $T'_r > T_{max}$ **then**
  infeasible insertion. Stop Algorithm 2
end
for *k1 in route R* **do**
  if $\tau_{k1} \leq \tau_j$ **then**
    $\tau'_{k1} = \tau_{k1}$
  else
    $\tau'_{k1} = \tau_{k1} + \Delta T_u$
    if $\tau'_{k1} > \min\{U_{k1}, T_F\}$ **then**
      infeasible insertion. Stop Algorithm 2
    end
  end
end
Set R' the route R with insertion of i between j and k and times τ'
for *POI k*1 *from position(i) to position(n) in route R'* **do**
  Calculate $\gamma(\tau'_{k1}) = \gamma_{k1}(\tau'_{k1}) + \rho_{k1}$
  if $\gamma(\tau'_{k1}) > c_{k1}$ **then**
    calculate $\tau^* = \min\{\tau : \tau > \tau'_{k1} \text{ and } \gamma(\tau) = c_{k1}\}$
    if $\tau^* > \min\{U_{k1}, T_F\}$ **then**
      infeasible insertion. Stop Algorithm 2
    end
    $\Delta T_u = \tau^* - \tau'_{k1}$
    $\tau'_{k1} = \tau^*$
    $T'_u = T'_u + \Delta T_u$
    for *POI k*2 *from position(k1)+1 to position(n)* **do**
      $\tau'_{k2} = \tau'_{k2} + \Delta T_u$
      if $\tau'_{k2} > \min\{U_{k2}, T_F\}$ **then**
        infeasible insertion. Stop Algorithm 2
      end
    end
  end
end
insertion of i between j and k is feasible
$\tau_k = \tau'_k, \forall k$, $T_r = T'_r$, $T_u = T'_u$

2.2 Problem Resolution

To solve the 3-objective optimization problem we apply a constraint method. We maximize the total score constrained to over occupancy and gap limitations and fix an upper bound for both over occupancy and the gap. The problem is

$$\begin{array}{l} \max \sum\limits_{(i,j)\in I_1\times I} s_i x_{ij} \\ \text{subject to } (2)-(14),\ z_i \le \zeta_i,\ (\tau_F-\tau_I)-\sum\limits_{i,j}(t_i+t_{ij})x_{ij} \le \beta \end{array} \tag{15}$$

where ζ_i is the highest over occupancy value admitted for POI i and β is the upper bound for the gap.

3 A Heuristic Algorithm to Solve the Problem

In this section we present a GRASP to solve the problem posed in Sect. 2. We consider that a solution to the problem can be represented by a list

$$R = [(v_1,\tau_1),(v_{i_1},\tau_{i_1}),(v_{i_2},\tau_{i_2}),...,(v_{i_q},\tau_{i_q}),(v_n,\tau_n)] \tag{16}$$

where the nodes are pairwise different except perhaps the initial and final points which can be the same and $\tau_1 < \tau_{i_1} < \tau_{i_2} < ... < \tau_{i_q} < \tau_n$. We use notations (v_i,τ_i) and (i,τ_i) indistinctly. For simplicity, we can omit times in (16).

Algorithm 3. Pushing algorithm
Pushing operation for route
$R = [(v_1,\tau_1),(v_{i_1},\tau_{i_1}),(v_{i_2},\tau_{i_2}),...,(v_{i_q},\tau_{i_q}),(v_n,\tau_n)]$

for *position* $i = n-1$ *to* $i = 2$ **do**
  $\delta = t_{i-1} + \bar{t}_{i-1,i}$
  if $\tau_i - \tau_{i-1} > \delta$ **then**
    Calculate $\tau^* = \max\{\tau : \tau_{i-1} \le \tau \le \tau_i - \delta,\ \text{and } \gamma_{i-1}(\tau) \le c_{i-1}\}$
    $\tau_{i-1} = \tau^*$
  end
end

For the route R, we denote $I_1(R)$ the set made of the POIs in R and the initial node (v_1). That is, $I_1(R) = \{1, i_1, i_2, ..., i_q\}$. The score of the route is $S(R) = \sum\limits_{i\in I_1(R)} s_i$ and the gap is $T_u - T_r = (\tau_n - \tau_1) - T_r$. For $i \in I_1 \setminus I_1(R)$ we define the unit profit of i for R as $p_i = \max\limits_{j\in I_1(R)} \dfrac{s_i}{\Delta T_{ij}}$, where $\Delta T_{ij} = -\bar{t}_{jk} + \bar{t}_{ji} + \bar{t}_{ik} + t_i$ and j and k are consecutive points in the route R, that is, we go from node j to node k. Let $j(i) = \arg\{p_i\}$, which is equivalent to $j(i) = \arg\{\min\limits_{j\in I_1(R)} \Delta T_{ij}\}$.

Fixed the parameter values, a route is built by application of a sequence of insertions. Once a route where no other POI can be inserted is built, an improvement procedure is applied. The construction phase is described in Algorithm 1. The insertion of a node in a route has to be done taking into account the occupancy limitations. If a node i is inserted between nodes j and k, times τ from

node k to the final point of the route increase in at least ΔT_{ij} units. Due to occupancy limitations for some POIs, this increase could be bigger and the insertion could be unfeasible.

Algorithm 2 contains the insertion algorithm. For the initial and final points in the route, we consider $t_1 = 0$ and $U_n = T_F$. In order to avoid the algorithm stops too early giving very short routes, the gap time constraint in (15) is relaxed. The solution obtained is improved by application of several procedures such as the insertion of POIs at the end of the route, reduction of the gap with Algorithm 3, where a *pushing operation* is executed, and an exchange algorithm.

Table 1. Computational example

n	T_I (min)	T_F (min)	T_{max} (min)	Score	T_r (min)	T_u (min)	T_{visit} (min)	Gap (min)	$Time_1$ (s)	$Time_2$ (s)
25	600	900	240	130	234	234	150	0	0.01	0.05
				129	234	234	150	0	0.01	0.08
				96	218	218	150	0	0.01	0.05
50	600	900	240	167	212	212	150	0	0.02	0.14
				135	208	208	120	0	0.02	0.10
				185	214	214	120	0	0.02	0.10
75	600	900	240	166	204	204	150	0	0.03	0.19
				183	188	188	150	0	0.04	0.15
				177	196	196	120	0	0.04	0.16
100	600	900	240	161	208	208	120	0	0.06	0.21
				148	218	218	120	0	0.04	0.21
				164	224	224	180	0	0.05	0.24
25	600	1080	480	274	442	442	270	0	0.04	0.05
				276	468	468	300	0	0.04	0.04
				246	460	460	240	0	0.04	0.04
50	600	1080	480	312	432	432	300	0	0.08	0.11
				364	464	464	300	0	0.09	0.09
				370	452	452	300	0	0.12	0.08
75	600	1080	480	335	458	458	330	0	0.14	0.14
				334	396	437	300	41	0.13	0.14
				337	432	432	330	0	0.16	0.13
100	600	1080	480	352	392	433.33	300	41.33	0.21	0.19
				315	468	468	330	0	0.16	0.19
				378	466	466.82	390	0.82	0.21	0.19

4 Computational Example

We solve the TTDP for n POIs randomly generated in a $[0, 100] \times [0, 100]$ square, for $n = 25, 50, 75, 100$ and the metric L_1. For each n we generate three instances. The scores are randomly generated in $[1, 100]$. Times (in minutes) are $(T_I, T_F, T_{max}) = (600, 900, 240)$ and $(T_I, T_F, T_{max}) = (600, 1080, 420)$. The maximum capacity is 100 and $\alpha \in \{0.8, 0.9, 1\}$. The break instants in the occupancy function γ go from 540 to 1080 by steps of 60, and the values are integer numbers randomly generated between 30 and 100. The contribution value is $\rho = 1$ and $\beta = 45$. The initial and final points in the route are the same. The algorithm is executed 3 times for each of the 24 instances. Table 1 shows the best solution (route with maximum score) for each scenario $(n, T_I, T_F, T_{max}, instance)$. The last two columns contain the computational times (in seconds) required to find the solution, the time before the improvement and the time consumed by the improvement procedure, respectively. The rest of times presented in the table are given in minutes.

5 Conclusions

Tourist route design is an important issue in the tourist management field. The problem modelled in this paper is to find a tourist route taking into consideration time windows and occupancy constraints, and 3-objectives. We maximize the total score while the other two objectives are incorporated as constraints. We propose a GRASP to solve the problem heuristically and present some preliminary computational results. Although a deeper study of the problem and the proposed heuristic is required, with a more extensive computational analysis, the results obtained seem promising.

References

1. García, A., Vansteenwegen, P., Arbelaitz, O., Souffriau, W., Linaza, M.T.: Integrating public transportation in personalized electronic tourist guides. Comput. Oper. Res. **40**, 758–774 (2013)
2. Gavalas, D., Kasapakis, V., Konstantopoulos, C., Pantziou, G., Vathis, N., Zaroliagis, C.: The eCOMPASS multimodal tourist tour planner. Expert Syst. Appl. **42**(21), 7303–7316 (2015)
3. Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G.: A survey of algorithmic approaches for solving tourist trip design problems. J. Heuristics **20**(3), 291–328 (2014)
4. Gunawan, A., Lau, H.C., Vansteenwegen, P.: Orienteering problem: a survey of recent variants, solution approaches and applications. EJOR **255**, 315–332 (2016)
5. Migliorini, S., Carra, D., Belusi, A.: Adaptative trip recommendation system: balancing travelers among POIs with MapReduce. In: 2018 IEEE International Congress on Big Data (BigData Congress), pp. 255–259. IEEE, (2018)
6. Ruiz-Meza, J., Montoya-Torres, J.R: A systematic literature review for the tourist trip design problem: Extensions, solution techniques and future research lines. Oper. Res. Perspect. **9**, 1–28 (2022)

7. Vansteenwegen, P., Souffriau, W., Van Oudheusden, D.: The orienteering problem: a survey. EJOR **209**, 1–10 (2011)
8. Wang, X., Leckie, C., Chan, J., Lim, K.H., Vaithianathan, T.: Improving Personalized Trip Recommendation by Avoiding Crowds. In: Proceedings ACM CIKM 2016, pp. 25–34. RMIT University, (2016)