A simulated annealing approach for the tourist trip design problem with the pattern sequence of the points of interest

Vigan Abdurrahmani · Kadri Sylejmani · Lule Ahmedi

1 Introduction

The Tourist Trip Design Problem (TTDP) [2] is about planning the trip itinerary for a tourist, when he/she visits a particular place (e.g. a city) for a certain period of time (e.g. couple of days). It is assumed that the place comprises of a number of Points of Interest (POIs), and the satisfaction factor of the tourist for each of the POIs is a known figure, whereas, each POI is characterized with its own attributes (e.g. opening hours, visit duration, cost of visit, specific types/categories, etc.). In addition, the tourist can enforce constraints, such as amount of money to spent on the trip and maximum number of POIs of certain type/category to visit. In this abstract, we present an extended variant of the TTDP problem, where we allow the tourist to express patterns of the visits to the POIs, in terms of having a certain number of types of POIs visited in a predefined sequence. For example, if the tourist has selected the pattern *monument-castle-museum*, for his/her first day of the tour, then the plan should have at least one *monument*, one *castle* and one *museum* included into the itinerary. Moreover, the specified POIs should be planned in the given sequence, although other ones, of any given type, can be inserted in-between. Our approach for tackling this newly defined variant

V. Abdurrahmani
Faculty of Electrical and Computer Engineering, University of Prishtina, Bregu i Diellit p.n.
Tel.: +383-38-554896
E-mail: vigan.abdurrahmani@uni-pr.edu

🖂 K. Sylejmani
Faculty of Electrical and Computer Engineering, University of Prishtina, Bregu i Diellit p.n.
Tel.: +383-38-554896
E-mail: kadri.sylejmani@uni-pr.edu

L. Ahmedi
Faculty of Electrical and Computer Engineering, University of Prishtina, Bregu i Diellit p.n.
Tel.: +383-38-554896
E-mail: lule.ahmedi@uni-pr.edu

of TTDP problem is based on the simulated annealing meta-heuristic and it utilizes the concept of POI pivoting for construction of good starting solutions.

2 Modeling the Tourist Trip Design Problem

In the literature, the TTDP problem is mostly modeled based on the Orienteering Problem (OP) [5] and its derived variants. In the OP, during a single and limited period of time, among a given number of points, a subset of them has to be visited, with the objective of selecting points whose total satisfaction factor is maximized. The Team OP (TOP)[1] enables multiple periods (e.g. days), whereas OP with Time Windows (OPTW) allows modeling service periods (e.g. opening hours) of points. Further, Time Dependent TOP (TDTOP)[3] makes it possible to consider variability within the distances between the points (e.g. walking or traveling by public transport). The Multi Constrained TOPTW (MCTOPTW)[8] problem represents a specific variant, where it is possible to express certain additional knapsack constraints, such as limiting the total cost to spent for visiting points, or enforcing upper limits about the number of points of certain category that can be visited (e.g. at most three points of the category of *architecture*). Furthermore, Multi Constrained Multiple TOPTW (MCMTOPTW)[9] is used to model multiple periods for multiple users (e.g. multiple day trips for multiple tourists), where each user gets its personalized itinerary, which, at certain points can overlap, hence allowing them to be together for some part of the trip. For an extensive study of the existing variants of OP and the respective approaches used for solving them, the reader is referred to Gunawan et al. [6].

In this paper, we define a new model, tagged as MCTOPTW with Patterns (MCTOPTWP), which enables adding an additional hard constraint that enforces the presence of a pattern in the form of a predefined sequence of points within each period of the itinerary. This model will allow the tourist to make certain types/categories of POIs as mandatory to be included into the itinerary [4], and moreover, the order of visits to such types/categories is enforced.

3 Problem Formulation

In more specific terms, the MCTOPTWP problem consists of the following inputs:

Number of tours. The total number of visiting periods (e.g. days).

Budget. The whole budget available for all tours.

Points of interest (POI). These are locations described by geographic coordinates (latitude and longitude), visit duration, satisfaction score, opening and closing time of the location, cost of the visit and the category of the location (might contain multiple categories).

Max allowed visits per category. This represents maximum number of visits per category type of POI for the tour, example *Museum: 3* would allow us to visit at most 3 museums during our tour.

Patterns. Patterns represent sequence of categories of POIs that should be visited, for example a pattern could consist of the following: *Museum*, *Castle*, *Natural Park*.

In our problem there are no soft constraints and all hard constraints must be satisfied in order to have a valid solution. The problem is defined by the hard constraints depicted below:

Visit once. Each POI could be visited at most once during the whole tour no matter in which day.
Time window. A POI cannot be visited before it's opening time or after it's closing time, however a visit could last after it's closing time if the visit starts before closing time.
Day duration. During each day of the tour we cannot exceed the duration defined by opening and closing time of starting POI.
Starting POI. Each daily visit should start from the first POI and end up at same POI.
No simultaneous visits to POIs. During each visit we can visit at most one location at a time.
Budget limit. Total cost of all days should not exceed the input budget.
Max visit per category. During the whole tour we should not visit more POIs of specific category than the defined number in input.
Pattern sequence. It is important that every day should fulfill its category pattern (sequence) and respect its POI order, although other POIs (of any category) could be visited in between. For example if we have the pattern *Museum*, *Castle*, *Natural Park* for a specific day, during that day we could visit POIs of categories *Museum*, *Castle*, *Seaside*, *Natural Park*. Note that *Seaside* category is not part of the pattern, still it can be visited, since the required order (in the pattern) *Museum*, *Castle* and *Natural Park* is maintained.
Travel duration. During the visit we should calculate also the travel duration from one POI to another. The duration is calculated as Euclidean distance between the geographic locations of any two POIs.

The objective function of the problem is maximizing the total satisfaction score of the tour which consists of sum of the satisfaction scores of each visited POI. We can increase this score by visiting as much POIs as we can while respecting hard constraints and reducing the waiting time between consecutive visits.

4 Search Method

The search method is based on the Simulated Annealing (SA) approach as presented in Algorithm 1. In abstract terms, the proposed approach can be described as in the following:

Search Space. The search space consists of the combinations that allocate all POIs into one of the two subsets. The first one makes the assigned subset, which represents a sequence of POIs (i.e. that make the tourist itinerary), whilst the second subset contains the unassigned POIs (i.e. those that do not get placed into the tourist itinerary). The states that violate the hard constraints are excluded from the search space.

Neighborhood Structure. The neighborhood structure consists of the union of three basic moves:

Insert: Inserts a POI from a unassigned subset into the assigned subset
Remove: Removes a POI from the assigned subset and places it into the unassigned subset, and
Swap: Swaps two POIs between the assigned and unassigned subsets.

In the course of a given iteration, we select a non-pivot POI that is within the itinerary and takes up the larges amount of time, then we replace it with one of the POIs outside of the itinerary. The replacement policy is based on the heuristic that tends to chose POIs of types that are less represented into the itinerary and have high satisfaction factors. After the swap move, we successively apply the insert move aiming to fill empty spaces that might have appeared. Whenever a certain POI is subject to a move, than all the following POIs are shifted forward or backward in time (in the respective itinerary of the affected day) to reflect the changes that occurs due to the process of moving the POI.

Cooling Strategy. The cooling strategy is based on the function defined by Lundy and Mees [7] that is expressed by equation $T_t = T_{t-1}/(1+\beta T_{t-1})$. A low value of β parameter (typically close to zero) makes the temperature change at a slow rate, hence making the algorithm do more exploration into the search space.

Stopping Criterion. The stopping criterion of the algorithm is determined by the minimal temperature parameter, which in this case is set to zero.

Initial Solution. The initial solution is constructed based on two phases, the first adding greediness and the second adding randomness attributes to the initial solution. The first phase uses the so called concept of pivoting, where some selected *pivot* POIs are placed first into the itinerary. For a given day, for each type in the pattern sequence, we make a group of POIs that belong to that specific type, and then we order them (in decreasing mode) based on their satisfaction factor. The POI with the highest satisfaction factor, in a given group, is selected first to act as a pivot POI, hence it will be the first to be considered for insertion into the itinerary of that given day. In case, the first POI does not meet the hard constraints, then the process of selection of a pivot POI is repeated for other candidates in the corresponding group, by using a backtracking strategy. In the second phase, the remaining places are filled randomly with other POIs that are not part of the itinerary yet.

Perturbation. The perturbation mechanism uses all of the three moves in a sequence and it is applied only when a number of iterations without improvement are passed (as defined by *MAX_ITERS* parameter). First, it removes randomly a number (as specified by *MAX_DEL* parameter) of non-pivot POIs from the itinerary. Then, it makes a number (that is selected at random between *MIN_TRIES* and *MAX_MAX*) of consecutive swaps between pivot POIs within itinerary and pivot POIs outside of the itinerary. Finally, it makes several attempts (as specified by *MAX_INS* parameter) to randomly insert unassigned POIs into the itinerary.

Algorithm 1 Simulated Annealing

```
1: procedure SOLVE($input, T_{min}, T_{max}, \beta$, $MAX\_ITERS$, $MAX\_DEL$, $MIN\_TRIES$,
       $MAX\_TRIES$, $MAX\_INS$)
2:     $S, Q, P \leftarrow$ generateInitialSol($input$) // S - solution, Q - unassigned, P - pivots
3:     $best \leftarrow S, t \leftarrow T_{max}, i \leftarrow 0$
4:     while $t > T_{min}$ do
5:         $a \leftarrow$ findPoiWithHighestSpace($S$)
6:         if $a$ is pivot then
7:             $i \leftarrow MAX\_ITERS$
8:         else
9:             $Q \leftarrow$ sort($Q$)
10:            for $b$ in $Q$ do
11:                $R \leftarrow$ clone($S$)
12:                $type \leftarrow$ selectTypeByPolicy($b$)
13:                if swap($a$, $b$, $type$, $R$) then
14:                    $inserted \leftarrow$ fillEmptySpaces($R$, $Q$)
15:                    if eval($R$) > eval($S$) or $e^{\frac{\text{eval}(R)-\text{eval}(S)}{t}} > random[0,1]$ then
16:                        $S \leftarrow R$
17:                        remove($inserted$, $Q$), remove($b$, $Q$)
18:                        add($a$, $Q$)
19:                    end if
20:                    if eval($S$) > eval($best$) then
21:                        $best \leftarrow S, i \leftarrow 0$
22:                    else
23:                        $i \leftarrow i + 1$
24:                    end if
25:                else
26:                    $i \leftarrow i + 1$
27:                end if
28:            end for
29:        end if
30:        if $i > MAX\_ITERS$ then
31:            randomRemove($S$, $Q$, $P$, $MAX\_DEL$)
32:            swapPivot($S$, $Q$, $P$, $MIN\_TRIES$, $MAX\_TRIES$)
33:            randomInsert($S$, $Q$, $MAX\_INS$)
34:            $i \leftarrow 0$
35:        end if
36:        $t = t/(1 + \beta t)$
37:    end while
38:    return $best$
39: end procedure
```

Table 1 Comparison of the results of simulated annealing against Iterated Local Search approach

	Iterated Local Search (ILS)		Simulated Annealing (SA)			
Instance	Time (sec)	Fitness (best)	Time (sec)	Fitness (best)	Fitness (avg)	ILS vs. SA (%)
MCTOPTWP-1-pr04	0.49	433	2.21	369	306.9	14.8
MCTOPTWP-1-c105	0.08	314	2.08	330	305.0	-5.1
MCTOPTWP-2-c108	0.25	654	1.08	570	496.0	12.8
MCTOPTWP-2-pr07	0.27	540	1.81	513	435.1	5.0
MCTOPTWP-2-pr08	0.96	764	2.94	655	546.1	14.3
MCTOPTWP-3-pr01	0.19	586	1.25	550	490.0	6.1
MCTOPTWP-3-pr09	2.82	1133	2.45	838	679.5	26.0
MCTOPTWP-3-c107	0.29	861	1.40	800	721.0	7.1
MCTOPTWP-4-r111	0.61	858	0.94	728	633.9	15.2
MCTOPTWP-4-pr04	1.86	1464	4.30	1120	920.5	23.5

5 Results

The above described model is tested by using a test set of ten instances that are derived based on a large instance set in the literature. Further, the results of our approach [1] are compared against the Iterated Local Search approach of [10] for the TOPTW, but which we adopted for the new model presented in this paper (i.e. MCTOPTWP). The adopted ILS solver can be obtained in GitHub [2]. As it can be seen in Table 1, in terms of fitness, in one (out of ten) instances our algorithm performs better than the ILS approach, and, in average, our algorithm falls behind the ILS for about 12% . In addition, in terms of computation time, our approach has an average computation time of 2.04 sec, while the ILS approach takes, in average 0.78 sec.

6 Conclusion and Future Work

The results presented above show that our approach is quite quick as it is able to produce good solutions in a matter of few seconds, which makes it suitable for use in practical applications. As part of future work, new neighborhood operators will be developed and the algorithm will be tested against a greater set of instances that are derived from the existing test in the literature.

Acknowledgements We thank Bsc. Festim Prebreza for implementing, from scratch, the state of the art algorithm ILS[10], who made it possible for us to have reference results for comparison purposes.

References

1. I-Ming Chao, Bruce L Golden, and Edward A Wasil. The team orienteering problem. *European journal of operational research*, 88(3):464–474, 1996.

[1] GitHub `https://github.com/vigan-abd/mctopp`
[2] GitHub `https://github.com/festimprebreza/iteratedLocalSearch`

2. Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics*, 20(3):291–328, 2014.
3. Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, Grammati Pantziou, and Nikolaos Vathis. Heuristics for the time dependent team orienteering problem: Application to tourist route planning. *Computers & Operations Research*, 62:36–50, 2015.
4. Michel Gendreau, Gilbert Laporte, and Frederic Semet. A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks: An International Journal*, 32(4):263–273, 1998.
5. Bruce L Golden, Larry Levy, and Rakesh Vohra. The orienteering problem. *Naval Research Logistics (NRL)*, 34(3):307–318, 1987.
6. Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315–332, 2016.
7. Miranda Lundy and Alistair Mees. Convergence of an annealing algorithm. *Mathematical programming*, 34(1):111–124, 1986.
8. Kadri Sylejmani, Jiirgen Dorn, and Nysret Musliu. 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)*, pages 300–305. IEEE, 2012.
9. Kadri Sylejmani, Jürgen Dorn, and Nysret Musliu. Planning the trip itinerary for tourist groups. *Information Technology & Tourism*, 17(3):275–314, 2017.
10. Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12):3281–3290, 2009.