Tour4Me: A Framework for Customized Tour Planning Algorithms

Kevin Buchin
TU Dortmund
Dortmund, Germany
kevin.buchin@tu-dortmund.de

Mart Hagedoorn
TU Dortmund
Dortmund, Germany
mart.hagedoorn@tu-dortmund.de

Guangping Li
TU Dortmund
Dortmund, Germany
guangping.li@tu-dortmund.de

ABSTRACT

«««< HEAD The touring problem aims to find an 'interesting' (round) trip of a given length. Here, what is considered interesting depends on the type of the desired route, e.g., a user may be looking for a off-road cycling trip or fast running route. There are two main perspectives on the touring problem, maximizing profit or minimizing cost, which result in very different algorithmic solutions. We provide a framework that allows for straightforward integration of new algorithms for both perspectives on the touring problem. In this demonstration we have included a new exact solver, a heuristic, and two greedy methods. The user can experiment with the algorithms and different profits/costs. The generated tours can be explored in an easy-to-use web interface. ======= In this demonstration paper we provide a framework that allows for straightforward integration of new algorithms for the touring problem. The touring problem aims to find an 'interesting' tour of a given length, an user decides what is considered interesting. There are two main approaches for the touring problem: maximizing attributes or minimizing cost. We provide a framework that allows for straightforward integration of new algorithms for both approaches on the touring problem. In this demonstration we have included a new exact solver, a heuristic, and two greedy methods. Furthermore, a GUI is provided in the form of a webpage, allowing for testing of algorithms, even by end users. »»»> d8fefa629fd1f14c230f98a1332e24075728464e

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Kevin Buchin, Mart Hagedoorn, and Guangping Li. 2018. Tour4Me: A Framework for Customized Tour Planning Algorithms. *J. ACM* 37, 4, Article 111 (August 2018), 2 pages. https://doi.org/XXXXXXX.XXXXXXX

Authors' addresses: Kevin Buchin, TU Dortmund, Dortmund, Germany, kevin.buchin@tu-dortmund.de; Mart Hagedoorn, TU Dortmund, Dortmund, Germany, mart.hagedoorn@tu-dortmund.de; Guangping Li, TU Dortmund, Dortmund, Germany, guangping.li@tu-dortmund.de.

0004-5411/2018/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Most people who do outdoor activities run into the problem of finding an appropriate route. Depending on the activity from hiking and jogging to gravel and road cycling, requirements from individual users can greatly vary. To this end we developed the framework Tour4Me for computing customized round trip. The tool Tour4Me includes four touring algorithms and provides an intuitive web interface to create tours customed to users specific demands and preferences. In this work, we introduce the touring problem, which is defined formally as follows. Let G be a directed graph consisting a vertex set V and a set E of arcs. A cost function $w : E \to \mathbb{N}$ and a profit function $\pi : E \to \mathbb{N}$ assign different cost and profit to each arc of G. We generalize the terms cost and profit to paths. Given a path $P = (e_1, \ldots, e_\ell)$ of G, we denote the cost of P as $w(P)$, which is defined as $\sum_{i=1}^{\ell} w(e_i)$;we write $\pi(P) = \sum_{e \in \{e_1, \ldots e_l\}} \pi(e)$ for the profit of P. Given a cost budget B and a starting vertex s, the objective of touring problem is to find a cycle $P = (e_1, \ldots, e_\ell)$ starting and ending at the vertex s that maximizes the profit sum within the cost budget B. Note that in the touring problem it is allowed to visit an edge multiple times, but its profit is only counted once.

Related Work. In the orienteering problem (OP), given an (un)-directed graph, the goal is to find a path which maximizes the total collected profit while not violating the budget constraint. In the classic model, a cost is assigned to each edge/arcs and a profit is assigned to each vertex. The profit of a path is defined by the sum of collected profit of its visited vertices. The orienteering problem is well-studied since decades and finds applications in the fields . Motivated by cycling routing problem, the variant arc orienteering problem (AOP) was introduced and has been studied in. In this variant, the profit is assigned to each edge and the profit of a path is defined by the sum of collected profits of each visited edge. The touring problem is a variant of the arc orienteering problem where the goal is to find a cycle with a fixed starting point.

Contribution. Our contribution is twofold. We designed a prototype tool Tour4Me for generating customized round trip. Here the user can give their priorities and preferences of road types and then the selected algorithm will compute a round trip with customized weights and profits; see Section 2. In this framework, we implemented four algorithms for touring problems; see Section 3. We selected one simple greedy approach, two representative sophisticated heuristics for AOP problem and one exact solver based on our introduced (ILP) model for the touring problem.

2 SYSTEM

Architecture. Our framework is a one-page web application. For the implementation of efficient algorithms in back-end, we use C++. To display the computed routs in the map, our web interface uses

the JavaScript library *Leaflet*[1] (version x.x.x). say something about the requirements sending....

Data.

Interface. The user interface. shown in Fig., consists of a large background map. The starting point can selected by clicking on a spot directly in the map. say something about the toolbox and the workflow...

3 ALGORITHM

We implemented four representative algorithms in our framework. The simplest approach Greedy Selection is a fast greedy approach which selects edges of high profits. The second greedy algorithm Jogging Tour aims to obtain a cycle consisting of edges of low costs. The two sophisticated algorithms Iterative Local Search and Integer Linear Programming support area maximization. The algorithm Iterative Local Search starts with the greedy solution computed by Greedy Selection and improve the initial solution by an iteratve local search based metaheuristic, which also gives the name to our algorithm. Finally, we introduce the exact solver Integer Linear Programming for the touring problem. As the name suggests, we formulate the touring problem as a linear integer programming problem. To solve our ILP formulation we use one of the state-of-the-art solver GUROBI for mathematical programming.

Greedy Selection. The greedy approach Greedy Selection computes a cycling starting at the vertex s by iteratively picking edges of the highest profit. To make sure the computed path is a valid cycle, we only pick edges, from whose endpoint the vertex s can be reached by the remaining cost budget. We call such edges as valid candidates. It starts by picking an valid edge starting at s of the highest profit. Next, we update the remaining cost budget and repeat this greedy selection from the endpoint of the new selected edge until reaching the starting vertex s or reaching a vertex with no adjacent valid edge. If the process terminates at a vertex v other than s, we add the shorted path from v to s to the computed solution.

Jogging Tour. We further selected one representative existing approach Jogging Tour [**DBLP:conf/wea/GemsaPWZ13**] for the AOP problem. (something goes wrong with the bibtex) In contract to Greedy Selection approach, this approach takes the edge costs into account and tries to select the "efficient" edges, that are edges of high profit and low cost. For this, we introduce the metric Inefficiency on the edges $\gamma : E- > R_+$. For each edge $e \in E$, we set $\gamma(e) = \frac{w(e)}{\pi(e)+0.1}$. Intuitively, this approach aims to obtain an equilateral triangles consisting of three efficient paths, each of which has a cost close to $\frac{B}{3}$, and having s as one of the triangle's edge. Firstly, we run a shortest path computation (with Dijkstra's algorithm) from s using this metric inefficiency. For each vertex u, v, we refer to $P_{u,v}$ as the shortest path of the metric inefficiency in the following. We now determine the vertices whose shortest path from s has cost in the range $[\frac{B}{3} - \frac{B}{10}, \frac{B}{3} + \frac{B}{10}]$. We refer to the set of such candidate vertices as the *ring* of s R_s. Next, for each candidate vertex v of s, we run a shortest path computation from v and compute the ring R_v. Now, we consider the intersection $I_{s,v}$ of two ring sets R_s and R_v. Note that the distances from each vertex $u \in I_{s,v}$ to s and u are in the range $[\frac{B}{3} - \frac{B}{10}, \frac{B}{3} + \frac{B}{10}]$. Then, we check whether the concatenation of the paths $P_{s,v}, P_{v,u}$ and $P_{v,u}$ is a valid solution, i.e., the cost of the computed cycling is bounded by the cost budget B. We repeat this step for each vertex in the ring R_s and collect a set of candidate cycles. Finally, we pick the candidate solution with the highest profit as the final solution. Note that this algorithm may not find any feasible solution.

Iterative Local Search. The Iterative Local Search algorithm is modified from the iterative local search approach proposed for AOP problem, which is proposed in [] and proved to be efficient for real-life instances. The local search Iterative Local Search runs the Greedy Selection approach and starts with the the computed solution as the initial solution. In order to escape form local minima, in each step of improvement phase, the current solution is modified by removing a different partial path. Then, a local movement is applied to connect the endpoints in the current solution and increase the total profit. The partial path for removal is determined by two parameters p and l, where i indicates the position of the current solution to start the removal and l is the length of the path to remove. Both parameters are set to 1 at the start and increased by 1 in every iteration. The parameter p is reset to 1 if the removal reaches the starting vertex s and l is reset to 1 if l reached the length of the whole solution. After removing the path P from a vertex u to a vertex v in the current solution S, we aim to find a better path between u and v which provides higher profit and whose cost is bound by the remaining cost budget, i,e, $B - w(S \setminus P)$. The basic idea applied here is a depth first search. To accelerate the local move, we use a parameter D to restrict the depth of the search. The search starts from the vertex u and explore at most D depth along each branch. Once a path P' reaching the vertex v without violating the budget constraint is found, we check if the total profit of the current solution is increased by replacing P by P'. If so, we stop the search and update the current solution.

Integer Linear Programming. The integer linear program (ILP) gives the optimal solution for an instance of the AOP. The ILP used in Tour4Me is a modified version from Verbeeck et al. The ILP from [] introduces a constraint for every subset of the vertices in order to avoind disconnected components, resulting in $O(2^n)$ constrains.

$$somethingBad \tag{1}$$

The ILP from [] uses Equation 1 to avoid subcycles. Instead we introduce a variable ρ_{kij}, for $1 \le k \le L$ and $1 \le i, j \le m$. Variable ρ_{kij} denotes whether edge e_{ij} is included in the path at location k.

$$\sum_{i=1}^{m}\sum_{j=1}^{m} \rho_{kij} = 1 \qquad \forall 1 \le k \le L \tag{2}$$

$$\sum_{k=1}^{L} \rho_{kij} = \begin{cases} h_{ij} & \text{if } e_{ij} \text{ is an edge} \\ 0 & \text{otherwise} \end{cases} \qquad \forall 1 \le i, j \le m \tag{3}$$

$$2 \cdot \rho_{kij} \le p[k][i] + p[k+1][j] \tag{4}$$

We include Constraint 2 for every $1 \le k \le L$ so that the path only has one edge at every position.

Constraint 3.

4 CONCLUSION

[1] https://leafletjs.com/