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

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.

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

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...

[1]https://leafletjs.com/

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

3 ALGORITHM

Greedy Selection.
Jogging Tour.
Iterative Local Search.
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 $1 \leq k \leq L$ and $1 \leq i, j \leq m$. Variable ρ_{kij} denotes whether edge e_{ij} is included in the path at location k.

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

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

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

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

Constraint 3.

4 CONCLUSION