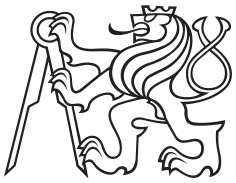
Bachelor Project



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

# Planning routes for recreational cycling

Miroslav Matocha

Supervisor: doc. Ing. Michal Jakob, Ph.D.
June 2019

# Acknowledgements

Thanks to CTU for beeing so great *alma mater*.


# Declaration

I declare that I wrote this paper by myself and that I mentioned all used literature in bibliography section.

In Prague, 6. June 2019

# Abstract

This paper is focused on problem of planning enclosed circular walks in graph constrained by length and optimizing criterion of pleassantnes. It examines various work in this field. Then from this work is chosen one state of the the art algorithm, which is described in more specific theoretical way. Introducing some particular concepts used in solution as is working with walk roundness or geospatial diversity of generated routes. Implementation of chosen algorithm is also part of this work. This implementation is further tested on various instances of problem and its results are further discussed.



**Keywords:** route planning, cycles



**Supervisor:** doc. Ing. Michal Jakob, Ph.D.
Fakulta elektrotechnická, Resslova 307/9, Praha



# Abstrakt

Tato práce je zaměřená na problém nalezení kružnic v grafu, které jsou omezeny délkou a přitom optimalizují kritérium příjemnosti. Zkoumá rozličné postupy v této oblasti. Poté je vybrán specifický algoritmus, který je teoreticky rozebrán ve větším detailu. Během toho práce rozebírá několik specifických konceptů použitých při řešení problému, jako je třeba práce s kulatostí vybrané kružnice, nebo geografická diversita mezi nalezenými řešeními. Součástí této práce je také implementace vybraného algoritmu. Tato implementace je dále testována na rozličných instancích problému a její výsledky jsou diskutovány.



**Klíčová slova:** plánování cest, cykly

**Překlad názvu:** Plánování tras pro rekreační cyklistiku

# Contents

# Figures



# Tables

# Chapter 1

# Introduction

People are nowadays pretty used to planning and navigating their trips with the help of various map applications, thank to the rapid spread of GPS technology especially in mobile phones in the past years. The problem of getting from A to B by the best route possible is, thanks to this intensive usage and various achievments in this field, practically completely solved. These are also the reason why people are starting to get into other areas of planning trips and trying to apply obtained knowledge to wider set of problems. This for example includes multicriterial planning or planning in dynamic networks.

In this paper I am focused to break down the problem of using algorithmically planned routes for recreational activities, which is specific for several reasons. For example people which are interested in using generated routes for jogging or cycling are just rarely interested in specific route in terms of visited points or area, they prefer to get several suggestions based on total cost of route instead (in terms of total length or travel time). Actually the most common use case is to plan these routes to form cycles containing the starting point. This is also the use case which I am focused on the most in this paper. Another property of these routes should be optimization of total or mean plesantness. This is whole another aspect of the problem as pleasantness of the edge can mean anything from type of communication through interesting surroundings of the edge to the proximity to some points of interest. It can even mix several criterions and can be personalized to specific user.

In the next chapters I will try to analyze the work which was done in this field in the past, take apart the individual parts of the problem in attempt to formalize it properlly and finally introduce the solution and its implementation which I have created.

# Chapter 2

# Related work

One of the most general papers laying down the theory needed for solution of this problem is problem of finding the minimum mean cycle in the graph. [2] This problem is defined as finding the circle in the graph which sum of weights through all edges divided by count of edges is minimal. Karp provides formula which yields the minimum mean of the mentioned cycle with linear complexity. Moreover it shows a simple technique to obtain not only this number but also the actual cycle from the same calculation. Although helpful, this technique ignores the length or cost constraint which is crucial to our application.

The next paper from Gemsa [4] introduces several techniques to generate feasible enclosed jogging routes. First it focuses on solving the problem through dual graph which consists of faces from original graph as nodes connected by edges only if the coresponding faces share one or more of theirs outlining edges in original graph. This of course requires simple preprocessing step based on right hand deep searches in original graph. After the dual graph is obtained the BFS is run over it starting from node which corresponds to the face incident with starting node. The result of BFS is enclosed jogging route in every iteration. This jogging route is constructed from the border edges of the search tree. This holds as far as two conditions are met. To get simple cycle from border edges the search tree has to be connected. And it has to exclude at least one face incident with starting node to ensure that starting node is part of that border. Both of the properties are constantly checked for during algorithm execution. BFS is finnished as soon as the length criterium is fullfilled. Gemsa introduces one more technique to optimize badness of the jogging route. Another metric influencing the execution of the BFS is added to the mix. This metric consists of force field which assigns a vector to every point in the plannar graph. This vector is counted from simple formula using individual face badness normalized by distance to the power of two. This formula is iterated and summed over all faces in the graph. BFS is then edited in manner of expanding the face which center has the biggest force in the direction of extension. They call this solution Greedy Faces.

Second approach which is introduced is called Partial Shortest Paths. This approach is working with computing triangles(or rectangles) which fulfill the length criterion and have starting node as one of its vertices. It does so by

computing the search tree of acceptable candidates from starting node. This search tree is called ring and it is constrained by third of the length of the route. From these results one via node is selected and its ring is computed with the same parameters. Then we can take the intersection area and pick the third point from there. Last step is to compute shortest paths between all three mentioned nodes, this yields one of feasible cycles. All the results from this proccess are then filtered to provide just the ones with best mean badness.

This approach provide a space for further improvements. For example we could limit the ring bound distance to one fourth of the desired cycle length and pick two points instead of one. From these points we can backtrack a little bit (to improve smothness of the resulting path) and run another two ring searches. In the intersection lies points which are on the path which includes nodes originating the search as well as the nodes from which we backtracked. These nodes are the fourth candidate vertices of the rectangle. Now we can compute shortest paths to complete this rectangle. Results are then filtred in the same manner as before. Algorithm can be further improved by implementing bidirectional search to found the fourth point in the rings intersection. This is also great for paralelization of the algorithm.

One of the most helpful papers for this work was Generating constrained length personalized bicycle tours [1]. This paper provides definition of Cycling Problem. This is the problem of optimizing closed walks in graph not only by cost of the edges itself but in combinaton with roundness of the whole walk as well. Here the input is length interval in which the total length of this walk has to be, starting node and two factors which control how will the algorithm evaluate the roundness part of the cost. The algorithm than works in two parts. First is called forward routing and the goal of this part is to find a set of suitable turning points for second part. Candidates are searched for by Dijkstra algorithm focused on cost of the edges and constrained by one criterion. Length of the path plus distance from the starting point has to be between the bounds of entered length interval. From this candidates one or more are choosed at random, or by some predefined strategy. This candidate is becoming the turning point of the tour. Second part of the algorithm is to run another search from every node on the optimal path between starting point and turning point that is targeted for starting point. These searches will yield paths which in combination with trimmed forward paths forms closed walks. This walks are further filtered by length criteria and optimized by cost criteria. The paper is than focused on several optimalizations of the algorithm, as is joining edges or computing reaches to limit the number of canidates.

Another approach to the problematic which I studied was proposed by Maervoet[3]. This approach doesn't lack the common denominator of computing partial shortest paths, although the way of getting the turning nodes is different. This approach is based on routes optimizing combined cost of visited nodes and edges. This algorithm starts by creating feasible window around starting node. This window is constrained by choosed length of the

tour. In this window the most valuable nodes (POIs) are localized and added to set of candidates. The size of the set is bounded by two numbers. If the set is too small algorithm proceeds by counting the summative cost of edges associated with nodes in feasibile window and promoting the ones with best results to candidates. On the contrary when size of the set exceeds the second number the candidates are filtered by spatial location. Grid is created on top of the window and just one candidate from each grid tile is chosed. Next step is to find several subsets of size two from candidates and run mentioned partial shortest path algorithm in both directions - clockwise and anticlockwise. This results in closed walk which contains starting node and choosed candidates. This proces does not run for every combination of feasible candidates, as this would be resource exhaustive. Author mentions running competitive learning system on top of this triangle search to pick several promising combinations. Lastly the found closed walks are filtered by length constraints and optimized by total cost of POIs and traveled edges.

# Chapter 3

# Problem

We introduce some notation in order to define the problem formally. Our problem is based on undirected graph $G = (V, E)$. Where every node represent intersection of some sort and edges represents roads in between them. Every one of these edges has its nonegative length. Formally we introduce function $l : E \to \mathbb{Z}^+$ which represents this length. We than define path or route of edges which is simply a sequence of edges $P = [e_1, e_2, ..., e_n]$ where holds that $SN(e_{i+1}) = EN(e_i)$ for all $i$ in $\{2, 3, ..., n\}$ where $SN$ denotes starting node and $EN$ ending one. The length of this path is counted as $w(P) = \sum_{i=1}^{n} l(e_i)$. Cycle is special type of path, where $SN(e_1) = EN(e_n)$.

## 3.1 Formal description

Input to our problem is a mentioned graph $G = (V, E)$. One starting node $s \in V$ and the desired length of the final cycle $l_{des} \in \mathbb{R}_+$. It is impractical and almost impossible to aim for single number as desired length. So we will choose a parameter $\varepsilon$ which will define interval $L = (l_{des} - \varepsilon, l_{des} + \varepsilon)$. We will search for cycles $C$ which fulfill $w(C) \in L$ and $s \in C$. And which optimizes the tour cost. This cost is derived from two components. Pleassantness of the edges themself and tour roundness which is introduced in the next section.

## 3.2 Tour roundness

Another property which we are interested in is the tour roundness. We could just ban revisitng the edges, but this hard constraint could disqualify a whole range of problem solutions. For example every instance of the problem where starting point is located in dead-end street has to be treated specially. As well as instances with large neigbourhoods accessible by one edge only. These would be completely invisible to algorithm with implemented hard constraint. Instead we will describe a functional roundness metric which assigns value representing roundness of the whole walk, this value can be later added to the minimalizing term of the algorithm.

This metric is based on sinusoidal relation between displacement and distance of two point on the circle (as shown in Fig. 3.1). We obtain

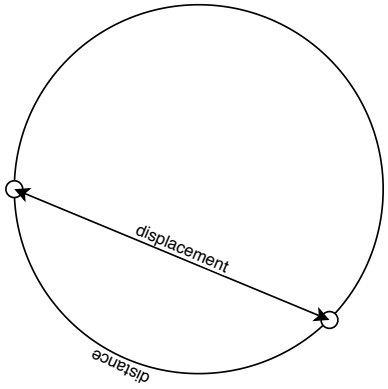



**Figure 3.1:** Ilustration of distance and displacement.

displacement between edges by computing distance as crown flies between two GPS locations representing their centers. We will mark this distance as $d_{real}$. Then we will define $d_{exp} : E^2 \to \mathbb{R}_+$ which represents the expected displacement of these centers based on distance of shortest path between these two edges. To use the sinusoidal relation to compute the $d_{exp}$ we would need to know the final length of the tour, which is not known yet and lies somwhere in $L$. Instead of using one of interval bounds we will choose different approach - we can approximate the relation itself to make it picewise linear. This function is then clearly symetric by $\frac{l}{2}$ mark, where $l$ stands for total circumference of the circle. In other words we get a function for which holds $d_{exp}(x) = d_{exp}(l - x)$. This means we can always count with the shorter distance by circumference and first half of our function. The last step is to exactly count this funtion, we get $d_{exp}(x) = \frac{2x}{\pi}$, which means that this function is not dependent on total circumference - that is important fact for continuous resolution of tour roundness. We define $d_{exp}$ for edges as well as for individual points around the tour. The expected displacement of two edges will be defined by expected displacement of theirs center points. This metric has one more important feature, it can be customized to different scales of penalization for not completely round tours. This envolves defining the cost function as

$$p(e_1, e_2) = \begin{cases} \frac{\sigma d_{exp}(e_1,e_2) - d_e(e_1,e_2)}{\sigma d_{exp}(e_1,e_2)} & \text{if } d_e(e_1, e_2) < \sigma d_{exp}(e_1, e_2) \\ 0 & \text{otherwise} \end{cases}$$

Where $d_e$ stands for actual geometric distance between center points. $\sigma \in [0, 1]$ parameter is called strictness and in pratcise define what is the same region for the penalisation of the algorithm. For example with sigma close the zero only the really close edges in the same path would be penalized and therefore it is not hard to find even less round solutions with penalization 0. On the other hand, $\sigma$ set at 1 means penalizing all the edges where is their actual distance smaller than should be by expectations. Finally the total penalization for whole tour $\pi$, $p(\pi) = \sum_{i=0}^{N} \sum_{j=0}^{N} \frac{p(e_i,e_j) l(e_i) l(e_j)}{l(\pi)^2}$ this function should approximate the average penalty between two points randomly chosen on the tour. This also means that it gets more accurate when the number of

edges and length of the tour increases.

# Chapter 4

# Solution proposal

As mentioned earlier we will try to optimize two components of the cost. This means we have to fit the roundness into the optimalizing term. We will furthermore introduce factor $\lambda$ which will basically identify the emphasis on the roundness in comparison with total cumulative cost. Our optimizing term therefore looks like this: $w_{avg}(\pi) = c_{avg}(\pi) + \lambda p_{avg}(\pi)$. It is easy to spot by plain sight, that $\lambda$ close to zero will not take roundness into account much, the main component of the final cost is practically only the cost of individual edges in the walk. However greater values of $\lambda$ can lead to another extreme where penalties for roundness overshadows edge cost and algorithm searches for perfect round tour rather than optimalizing edge cost.

The algorithm itself will run in two stages. The first stage is meant to find acceptable candidates. From this set individual candidates are picked and used for planning of the cycle. This stage mostly consists of Dijkstra search which is executed from starting node with summative edge cost used for ordering of the queue. Furthermore only nodes with combined length of path and distance from starting node smaller than upper bound of $L$ are added to the queue. If also holds that this sum is smaller than lower bound of $L$ node is considered as candidate. Search is complete when we encounter empty queue. The result of this first stage is set of suitable candidates for second stage.

From this set we pick several candidates for second stage of the algorithm. This subset is picked with emphasis on the best cost to length ratio and geospatial diversity of the picked candidates, because this strategy yields the best results in overall geospatial diversity of found cycles. We will get more into how to choose this candidates in the next section. For every candidate from this subset we will proceed to steps outlined in second stage of the algorithm

Now when we have obtained the single candidate we will extract the path by which we got from the starting node to the picked candidate from search tree of the first stage. This path is obviously optimized by total cost. Then we will go by the path node by node and from the ones for which holds that length of the path plus its distance from starting point is in $L$, we will run another Dijkstra search. We will denote this point as turning point $t$. This search is targeted towards starting point and it cost function is changed

to reflect not only edge cost but also the roundness of the resulting cycle. Specifically for forward path $\pi_f$ we got edge penalty for backward Dijkstra as $w(e) = c(e) + \frac{2\lambda}{l_{max}} \sum_{e' \in \pi_f} p(e, e')l(e)l(e')$. This will will result in two paths $\pi_f(\pi_{s \to t})$ and $\pi_{t \to s}$ which together form a cycle. This cycle is optimalized to cost and roundness by parts. However we have no gurantee, that length of this cycle lies in $L$ therefore we have to filter found cycles by this criterion and sort them by its total cost. We return only the best one as result of this second stage of the algorithm.

We typically run the second stage several times with different candidates to obtain more than one cycle to use for multiple choice.

## 4.1 Geospatial diversity

# Chapter 5

# Implementation details

## 5.1 Implementation description

## 5.2 Performance results

# Chapter 6
# Conclusion

Lorep ipsum [1]

# Bibliography


[1] Stroobant, P., Audenaert, P., Colle, D., & Pickavet, M. (2018). *Generating constrained length personalized bicycle tours.* 4OR, 16(4), 411–439. https://doi.org/10.1007/s10288-018-0371-9

[2] Karp, R. M. (1978). *A characterization of the minimum cycle mean in a digraph.* Discrete Mathematics, 23(3), 309–311. https://doi.org/10.1016/0012-365x(78)90011-0

[3] Maervoet, J., Brackman, P., Verbeeck, K., De Causmaecker, P., & Vanden Berghe, G. (2013). *Tour Suggestion for Outdoor Activities.* In Web and Wireless Geographical Information Systems (s. 54–63). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-37087-8_5

[4] Gemsa, A., Pajor, T., Wagner, D., & Zündorf, T. (2013). *Efficient Computation of Jogging Routes.* In Experimental Algorithms (s. 272–283). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-38527-8_25

[5] Juraska, J., Nykl J. (2016). *Multi-criteria Route Planning with Emphasis on Geospatial Result Diversity* https://www.semanticscholar.org/paper/Multi-criteria-Route-Planning-with-Emphasis-on-Juraska-Nykl/96165d905691de3b8e9b0af8b83fc2e68941b59c