technische universität
dortmund

Masterthesis

Customizable Roundtrips with Tour4Me

Meta-heuristic Approaches for Personalized Running and Cycling Routes

Lisa Salewsky

July 2024

Supervisors:
Prof. Dr. Kevin Buchin
Mart Hagedoorn, M. Sc.

Technische Universität Dortmund
Fakultät für Informatik
Algorithm Engineering (LS-11)
http://ls11-www.cs.tu-dortmund.de

Contents

Chapter 1

Introduction

Algorithms for optimal tours are an important and much studied part of computer science. It is a topic that directly influences the lives of many people. Better routing algorithms can help reduce travel times by car, bicycle or even on foot. There is also considerable work on optimizing public transportation [**?**, **?**] and managing traffic jams [**?**, **?**]. Examples are Dijkstra (uni- and bidirectional) [15, 23, 28], A* search (also uni- and bidirectional) [15, 23, 28], greedy algorithms [15, 28], branch-and-bound algorithms [13], the Bellman-Ford-Moore algorithm [8] and many more [9, 23].

All of these have in common that they always look for the shortest or quickest path between two different points. However, when planning a tour, the goal might not be to simply get to a location as quick as possible. In particular, in many cases people plan round-trips. Especially for running and cycling - for training towards a specific goal or even as a pastime hobby - it is often desired to have roundtrips of a certain length. Additionally, people typically enjoy running or cycling on more appealing paths in nature rather than between high buildings and on softer ground rather than on asphalt. So, a lot more information have to be taken into account when trying to find good running or cycling roundtrips. Which means, these algorithms become useless for these scenarios. The shortest path from the starting point back to it will always be to never leave. So, a different approach is needed for these kinds of routes [11].

Considering the benefits of running and cycling (for overall health [19, 22, 25], the cardiovascular system[17], as a measure against many different diseases[19] as well as for social[16, 18, 27] and psychological benefits[3, 7, 24, 27] and also further benefits of touristic cycling for cities [4]), the problem at hand becomes all the more important. Not only are there many joggers and cyclists, who would profit from a tool that returns a roundtrip for their preferred personal optimal route, but having such a tool at hand could help convince more people of starting one or both of the two activities. This could result in an overall larger population doing some exercise and profiting from the previously mentioned benefits of physical activity outdoors. Not only does such a web app lower the effort it takes to start running or cycling (as route planning is coupled with effort), it also helps to show people better or more appealing routes and encourage participation in outdoor activities.

Additionally, as already stated in examples for benefits of running and cycling, such an app can prove useful for tourism purposes as well. People typically enjoy running or cycling along enticing, exiting routes, which are hard to find - especially in unfamiliar areas. For any kind of holiday trip, planning new roundtrips for either exercise purposes or event for several-day roundtrips, this app can be very useful.

1.1 Goal and Methodology

The goal of this thesis is to create a usable application for computing running or cycling roundtrips of (almost) arbitrary length. Useful in this case means an app that can be used in real time, that produces results of the desired length and prioritizes paths according to the users' input. To achieve this, the thesis will be built on the already existing prototype Tour4Me and eventually add meta-heuristic approaches that have been deemed the most fitting for its purpose.

First, an interface for testing the new approaches has to be built. This also needs an overlay for adding in user options like the length of the desired roundtrip, as well as other preference inputs. Based on this interface, different algorithms can be added and compared with each other to find the ones that will produce optimal results. These optimal results can have very different definitions of optimal. An ideal algorithm would be fast, always generate a route and use all the users' preference inputs. However, it is not possible to achieve all these goals with just one algorithm. Therefore, different approaches will be implemented and analyzed according to how well they fulfill the previously mentioned criteria.

Some of the possible approaches include different implementations of genetic algorithms [1], of ant colony or anthill algorithms [2, 1, 26] as well as possible hybrid versions. These hybrids can either be hybrids of one of the meta-heuristics with - for example - local search algorithms [1, 26] or hybrids of these two joined together. Furthermore, if there is enough time left, it is also possible to include the new algorithms into already implemented ones to improve those.

When the best algorithms for this application have been determined, they will be integrated into the already existing Tour4Me application. The aim is for the app to calculate a high-quality tour for any typical roundtrip requests for running and cycling.

In addition to finding suitable algorithms that allow for fast and reliable computation of all typical roundtrips, working on the interface and data used also improves the usefulness of the app. It can be equally important to improve the interface, add more options like elevation data, include more information (for example previously used routes) etc. There are several opportunities and options to improve the app not only by changing the used algorithms but also adding and upgrading the GUI and user selection options. This is an alternative approach towards the goal of making Tour4Me more usable. It is another option to put more work into improving the app aside from adding more or faster algorithms.

1.2 Structure

Chapter 2

Fundamentals and Background

As stated in the introduction, most routing algorithms focus on shortest paths between two or more points. Many of those have been reviewed in several different surveys [15, 28]. Additionally, there have been many more heuristic approaches, like local search variants [5, 12, 21] or different neighborhood based ideas [5, 12, 21]. Much research has been done and is still ongoing for these kinds of problems, stemming from the fact that many routing problems (for example the traveling salesman problem (TSP) [1] or the vehicle routing problem [5, 12]) are NP-hard [20]. Furthermore, finding a shortest path is important in various parts of daily life. Whether it is the best way to get to work or to a supermarket by car or bike, a good way to minimize travel time by bus or any other trip from one place to another. Additionally, shortest paths are not limited to real-world networks but can also prove useful for social networks or any form of digital network. [15]

2.1 Shortest Path algorithms

For calculating shortest path trips from one starting point s to a destination d, several different approaches can be used. This problem has been well-studied and still continues to advance in terms of quality of the returned paths as well as in optimizing the running time of algorithms. Thus, the number of ideas to solve it is enormous.

Generally most shortest path problems are in one of two categories: they are either single-source shortest paths (SSSP) or all-pairs shortest paths(APSP). The first only uses a starting point and tries to find the one shortest path between it and all other vertices. The second aims to find shortest paths between all vertices of a graph, which can be necessary for transportation networks and similar use cases. Aside from these two categories, many more can be found to describe and sort types of approaches. Madkour et al propose a taxonomy to help classify the different algorithms into specific categories. [15]

Which of these algorithms performs best is typically dependent on the type of graph it is being used on, the graph's structure and the specific problem to be solved. A graph can be categorized as planar or not, directed or undirected, weighted or not (and carry only non-negative weights or allow negative ones as well), they can contain cycles or be acyclic and many more. These different types determine which algorithms can be used as well as which will return better results.

2.1.1 Single Source Shortest Paths

2.1.2 All Pairs Shortest Paths

2.1.3 Heuristic Approaches

Additionally to exact approaches, heuristics can be used to improve the runtime of an algorithm. A heuristic is a technique that is based on experience or statistical insights. The downside of using such an approach is, that there will no longer be a guarantee that the result is the global optimum, as heuristics specifically only find partial or approximate solutions to a given problem. In many cases where it would take too much time or space to find the actual optimal solution, heuristics can be used to find the best possible solution within the given bounds.

For these, several different ideas have been formed. These can then be categorized into construction heuristics, improvement heuristics and meta-heuristics [21]. Sometimes, a fourth category for two-phase heuristics is included as well (see [**?**]).

#TODO is this correct for heuristics in general? The paper refers to heuristics for VRP

Construction heuristics build their solution from a starting point until a certain boundary is reached. They typically don't have a separate improvement phase. Improvement heuristics try to improve an already existing solution. They perform improvement steps several times until a specified boundary is reached. These boundaries can be e.g. a time limit or reaching the threshold for a good enough approximation. (Iterative) Local Search and Neighborhoods are examples of improvement heuristics that can be used to reach a more optimized solution. [**?**, 21]

2.2 Meta-heuristics

Meta-heuristics are a form of heuristic approaches. As such, they also try to find an approximate solution to a problem that is as optimal as possible. The distinction between classical heuristics and meta-heuristics is, that the latter are combined with additional strategies. These are used to enable the meta-heuristics to not produce only solution that are locally optimal, but to broaden the search space they can use for finding optima.

Classical heuristics oftentimes carry the inherent risk of only finding a local optimum that can be far from the actual global one. To reduce this risk, higher level approaches are necessary. These can include using several neighborhood structures to broaden the search space or entirely new concepts like the Ant Colony approach or Genetic Algorithms. [1]

Some of these meta-heuristic ideas that will be used in this thesis will be explained in the following subsections.

2.2.1 Ant Colony

Ant Colony is a meta-heuristic approach that is based on biological ants, ant colonies and how they search food. Real ants start off by walking around on random paths starting from their nest. When they discover a food source, they pick up the food and walk back to their nest. On this way, they distribute a substance called pheromones. These can then be detected by other ants and indicate to them, that a path leads to a potentially good food source. Other ants then are more likely to follow a path with more pheromone placed on it and will in turn lay down their pheromone as well, leading to an accumulation of these on good paths. Over time, the pheromones dissipate and when they aren't renewed, will evaporate completely, decreasing the attractiveness of the corresponding path [10, 1].

Furthermore, pheromone distribution also inherently leads to using shorter paths. When several ants have to choose between paths, they will first select at random. However, as soon as one ant discovers the food, turns around and distributes its pheromone on the way back, it increases the likelihood of it's path being taken. Here, the shorter paths will be first to receive more pheromones as the ants returning will be quicker. Due to the faster accumulation, more ants will choose this shorter path and thus place even more pheromone on it, leading to a self-reinforcing loop that converges when all ants choose the best path only. Then, all worse paths will loose all their pheromone over time and leave the best result as the only remaining path [10, 1].

To illustrate pheromone distribution, an example illustrates in figure 2.1 how real ants find food and establish the best path towards the source. In part a on the left side, there are many ants that fun between two points A and E. These could be the nest and an interesting food source. In part b in the middle, an obstacle has been added. This now leaves the ants with a choice, which path to follow. In the beginning, the likelihood of picking either path will be around 50%. While taking the path, the ants distribute pheromones on it. On the shorter route, the ants will end up reaching the food source earlier, thus returning quicker than the ones who took the long path and distribute more pheromone on the shorter path. For the first few ants, there will be almost no change in the attractiveness of either path. However, the more ants take the short tour and return quicker, the more pheromone will accumulate on that path. This leads to a shift in the attractiveness, making the shorter path more likely to be chosen by later ants. These ants will in turn again increase the amount of pheromones placed, making the path even more attractive. So, the ants create a self-reinforcing loop of positive feedback through their pheromones which eventually leads to a state where all ants always choose the shorter option.

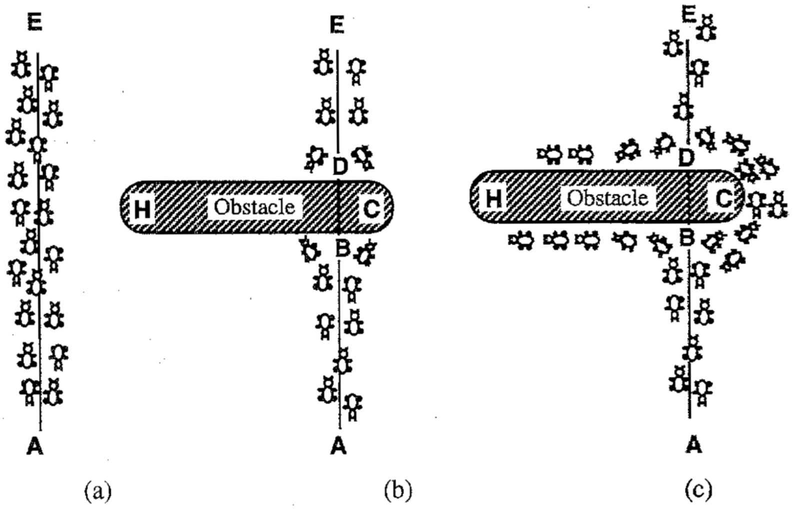

Figure 2.1: This figure shows an example of pheromone distribution with real ants. Taken from *Ant System: An Optimization by a Colony of Cooperating Ants*[10]

This behavior can be replicated in virtual graphs for various routing problems. Ant system has been first introduced in 1990 by Dorigo et al[10]. In the paper, the authors describe how to use ants for solving the traveling salesman problem (TSP). This is different from the question of finding a roundtrip with a certain length (plus additional user preferences). However, in the paper, they stress the adaptability of ant system approaches, showing both versatility and robustness on different example problems[10].

2.2.2 Calculations

To transform the analogy of real ants into an algorithm, some formulas and calculations are needed. Ants are very simple agents. They can only do two things: Pick the next node to move to and place pheromone on a path. They communicate with other ant agents through the pheromone trails, making it a decentralized way of communication without the need for a central agent. For the algorithm, a set amount of m ants moves through the graph, tries to find a good tour and places pheromones on edges. Every ant has a defined amount of pheromone to place. How much of it will be laid on a path can be calculated in several different ways. Dorigo et al propose the following three ideas[10]:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if (i,j)} \in \text{tour described by } tabu_k(1) \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} Q & \text{if the kth ant goes from i to j between time} \\ & \text{t to t+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{d_{ij}} & \text{if the kth ant goes from i to j between time} \\ & \text{to t+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

Here, equation 2.1 is the default the authors used for solving the TSP. Q is a constant that has to be picked according to the problem in question. L_k describes the length of the whole tour. This property makes sense for the TSP setting, but is relatively useless for the case of tours with a fixed length, as it will be the same value for every ant and every run made. In this case, where the user defines the length of the tour, L_k will only scale the values picked for Q[10].

Equation 2.2 only uses the constant Q to describe pheromone placement. Here, neither the full tour length nor individual edge costs are taken into account. Pheromone is placed evenly on all edges. This equates to a not-scaled version of equation 2.1 with the given use-case of a set length for the tour[10].

The last equation 2.3 divides the constant by the length - or the cost - of each edge when it is used. Doing this reduces the amount of pheromone placed on longer edges proportionally to shorter edges. While this equation is not influenced directly by the fixed length, this property can still cause the equation to be less useful for tours with a specified length than for TSP. Since tours that are meant to cover a fixed distance are different from the TSP, where a shortest path that visits all selected cities is to be found, the last equation seems like the least promising candidate for useful pheromone distribution[10].

It is possible to define other ways to calculate the pheromone placement. Which option turns out to be the best fitting one will be described in the evaluation chapter 5.

Using a suitable formula to calculate the pheromone distribution, this value can then be used to calculate the overall distributed pheromone for each edge (i, j) that was placed by all ants during one iteration. This value is described by $\Delta\tau_{ij}$ as follows[10]:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{2.4}$$

This overall value can then be used to calculate the so called "insensity" of the placed pheromone trail. Since pheromones evaporate over time, this property has to be modeled as well. To do this, a new parameter ρ needs to be introduced. It describes how much of the pheromone stays on the trail between two time steps. So, the overall pheromone intensity can be described by

$$\tau_{ij}(t+n) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}^k \tag{2.5}$$

where $\tau_{ij}(t)$ is the previous pheromone intensity and t+n describes the next time step after one full tour was created in n steps[10].

Using these calculations, the pheromone intensity on all paths can be represented. What's left is determining the probability with which ants will choose a certain edge over the other options. To do this, two more properties are needed: the visibility of an edge and a tabu-list (or rather a list of allowed nodes). The tabu-list contains all nodes that have been visited before. Since roundtrips should - per default - be round rather than the same path run in two directions, this property is needed to ensure no city is visited more than once. In chapter 5, different configurations are tested to represent different shapes and allow for more options users can define. Thus, for other shapes, this list is not needed. The visibility ν_{ij}^k is calculated using the length of the edge d_{ij} as follows:

$$\nu_{ij}^k = \frac{1}{d_{ij}} \tag{2.6}$$

And the transition probability is given by

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\nu_{ij}]^\beta}{\sum_{k \in allowed_k} [\tau_{ij}(t)]^\alpha \cdot [\nu_{ij}]^\beta} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

using all previously defined values to calculate visibility ν_{ij}^k, trail intensity τ_{ij}, pheromone distribution $\Delta\tau_{ij}$ and $\Delta\tau_{ij}^k$. Here, α and β are parameters that influence the weight of visibility and trail intensity. Higher values of α increase the significance of the pheromones on the trail (setting α to 0 would lead to completely ignoring the pheromone placed) and higher values of β increase the importance of the visibility of an edge (making longer edges less attractive as a result)[10]. These parameters will be experimented with and their influence will be evaluated in chapter 5.

In their paper, Dorigo et al suggest middling values for α and β in a range of $[0.5, 5]$. They furthermore stated that the best tour was achieved using $\rho = 0.5$ and $Q = 100$. Overall, the results of experimenting with different parameter configurations showed that for very high or very low values of α, no good results could be generated [10].

TODO add fomulas and desription how they help constructing paths # TODO add how to use for my work

2.2.3 Genetic Algorithms

2.2.4 Simulated Annealing

Chapter 3

Related Work

Much research has been done for shortest path algorithms and their optimization, however, for the - more complicated [11] - problem of finding a round trip with additional conditions, not much work has been done yet. While there are a few tools that can be used to calculate round trips, most of them only focus on cycling or create a very limited set of trips that do not satisfy the needs of most people, or both. Some examples for these tools are RouteLoops [1] and RouteYou [2] which both do not allow for much customization of preferences.

Adding new options for user inputs that enable a higher degree of customization can vastly improve the usability of a tool. The usefulness is not only determined by the implemented algorithms, but also by the interface, the data used, and the selection options presented to the user.

As both RouteLoops and RouteYou are commercial programs, it was not possible to obtain the necessary details about any used algorithms, heuristics, meta-heuristics or even the language they used for programming these solutions. All gathered information are collected from exploring the functionality of the two tools by hand and reading both the general information and the FAQ pages provided by the websites.

3.1 Tour4Me

The tool which this thesis will be based off, Tour4Me[3] [6], incorporates many of these points in its web interface. It is possible to choose the preferred ground type as well as make selections about preferred route types. Furthermore, the user can also mark certain types as unpreferrable (rather than just keeping them neutral or preferring them). This allows for much more customization. What the tool does not incorporate yet is the option to make selections about the preferred elevation or route complexity. However, the route can be optimized for a circular route when using the covered area of the tour.

[1] https://www.routeloops.com/
[2] https://www.routeyou.com
[3] http://tour4me.cs.tu-dortmund.de/

It implements a solution for the "touring problem", which is used to describe the task of finding appealing and ideally interesting roundtrips. To achieve an optimal solution, two factors are taken into consideration. First is the total profit, that can be collected within the given length restriction for the tour. Second is an additional quality function that assures for a relatively round tour by maximizing the area that is surrounded by the created roundtrip. Tour4Me presents a selection of four different algorithms to calculate the tour as well as some additional customization options. The offered choices include a Greedy Selection approach, Integer Linear Programming, MinCost with Waypoints, a shortest paths variant, and Iterative Local Search [6].

The Greedy Selection is the simplest algorithm which only ensures that the chosen route is a roundtrip. It builds it's path by iterating over the valid edges and picking the most profitable of these until the cycle is finished or no candidate is left. A valid edge is determined by checking whether the start- and endpoint s can still be reached if that edge is picked next [6].

For Integer Linear Programming, the touring problem must be stated in an appropriate form. To do so, a single instance can be encoded as $\mathcal{I}(G, w, \pi, B, v_0)$, containing the Graph G, edge costs w, the profit function π, the budget (length restrictions) B and the starting (and end-) point v_0. Given this encoding, cycles $P = (v_0, ..., v_i, ..., v_0)$, which are always at most of length L, can be built. For the current definition, a few additional variables an be introduced to encode whether or not an edge is part of a solution (and how many times it occurs), whether or not an edge is the k-th edge of the solution and whether or not a vertex is the k-th vertex of a solution. Using these, constraints can be built to describe the desired behavior of the algorithm [6].

The MinCost algorithm needs the waypoints because it is typically meant to solve shortest path problems. Thus it would always choose not leaving the starting position without the added points. Even though this algorithm is not originally meant to solve roundtrip problems, it takes into account the cost and profits of edges to create a solution tour, which makes it more suited to the task than simple greedy search. To create an optimized tour, the inefficiency of paths has to be measured. This is done by calculating the quotient of the edge costs and the profit the edge yields. Using this inefficiency, a ring of candidate points R_s surrounding the start-point s can be calculated. All points that are part of this ring have a shortest path distance of at most π. From these, new rings R_r with the same requirements can be calculated. The solution path is then obtained through intersecting the sets of all circles and selecting all those that intersect with R_s. To ensure the highest profit tour is returned, all possible combinations are calculated and the optimum is returned[6].

Building from this solution, the Iterative Local Search can be applied to improve the found tours. From the returned roundtrip, the algorithm removes partial paths P from the current best solution S and tries to iteratively add new parts that improve the solution profit while always staying within the given budget $(B - w(\frac{S}{P}))$. Since searching for viable edges is performed using a depth first approach, bounding the maximum depth of this step can drastically speed up the algorithm. To keep track of the added length and profit, two variables (l and p respectively) are introduced. These start with an initial value of one and are raised by a single increment for each iteration. p is reset when the starting point is reached by the removal step. l is reset when the maximum length for the solution is reached. The best solution is improved constantly until the user selected time limit is reached [6].

#TODO add more citations –> see Tour4Me paper

3.2 Roundtrip paths

As already stated above, existing tools leave out certain data like elevation or path types. This impacts the quality of the created routes for users or even user groups. For example, people who prefer running with little to no elevation can end up with a route that takes them uphill through a park for half of the route. Which still may be a good choice for other users - joggers who prefer more challenging routes or people who want to hike and enjoy ascending. However, others could prefer running through the city over a park when the elevation matches their preferences better in the city. For these users, the created route would be highly unfavorable, even though it matches other constraints for what is considered a nice roundtrip. Therefore, it can be crucial to the usefulness of an app to give the user as many options to customize as possible.

3.2.1 RouteLoops & RouteYou

RouteLoops has two text fields for entering the starting point and the length of the trip. Aside from that, no real customization is possible. It does have a few features to show more information about the route like showing distance markers or elevation, however, these can not be used as inputs to get a route with - for example - as little elevation as possible. Apparently it can also show route difficulty for the United States, however even when creating a route in the United States, no result was shown. RouteLoops also does not actually create loops but rather picks a route that has high value (for example with a river in a park) and lets the user run along that path, turn around at the end and run back the same way.

To crate a roundtrip, some "waypoints" are created. These can be removed or more can be added in when editing the tour. Between the waypoints, it seems like a shortest path is tried

RouteYou offers several different options that will return varying results, however, picking the same option again will also give different results every time. Here, the roundtrips are more round than with RouteLoops, but again, elevation or difficulty are not taken into account. Also, while both do offer the possibility to edit the returned roundtrip, this editing changes the length of the route arbitrarily. Furthermore, it is not possible to specify directly what type of underground or surroundings etc. are preferred.

3.2.2 Computing Running Routes

The problem of calculating good running roundtrips is not new. In addition to the commercial applications, there also are research papers on this subject. One of these papers is "Efficient Computation of Jogging Routes" [11], presents two ideas to handle the new routing problem which the authors labeled "Jogging Problem". It is split up into two variants: One being the simple version, that only aims to build a cycle that contains the starting point s and has the desired length. The other is a more complex version, that allows for some flexibility regarding the length of the final tour during optimization. Hence, it is named "Relaxed Jogging Problem". This relaxation allows to take more factors into account to also optimize for the resulting shape, the area surrounding the tour and/or the simplicity of the path [11].

The second problem is chosen as the one to optimize, since it enables the addition of other conditions than just the length of a tour. For this, two different ideas are proposed. The first approach - "Greedy Faces" - is based on extending previous cycles. It starts with a cycle containing the starting point s that can be selected by the user. This roundtrip then can be extended to gradually approach the user specified length. The second algorithm was named "Partial Shortest Paths" and uses via-vertices. These are a number of new points that can be connected with shortest paths. When the via-vertices are connected with each other and the start, they form a roundtrip [11].

#TODO maybe stop the description here already

For both algorithms, the authors measure the badness of paths, the number of edges that are shared as well as the number of turns. The badness is used to take the additional constraints into account. To reduce the possibility of having a roundtrip which turns at the end and uses all paths twice, the shared edges have to be minimized. The number of turns corresponds to the complexity of the tour and is measured by a percentage of a full u-turn [11].

#TODO how often do I need to reference the paper?

Greedy Faces Greedy Faces is built from an already existing circle by extending it. For this, blocks outside the tour that are adjacent to the current path are used. The previous circle encloses the chosen block and thus extends the previous route. New blocks are picked until the desired length is reached. To ensure only blocks that correspond to faces are picked, a preprocessing phase is introduced that identifies faces of the graph. During this step, first, dead-ends are removed, so the resulting graph will be two-connected. Faces then are defined by the edges that surround them. While identifying all faces, a dual graph $G^\star = (V^\star, E^\star)$ for $G = (V, E)$ is built as well.

The Greedy Faces algorithm then works on the dual graph $G^\star$, selects a face f from $V^\star$ which has a surrounding path that contains the starting point s. Then, a Breadth First Search Tree T is built, starting at f, until the desired length (a relaxed version $(1+\varepsilon)L$) is exceeded. The resulting tour will be a simple path iff all vertices in V without the ones in T are connected and contain s. The final jogging path can be extracted by taking the cut edges between the tree T and the remaining vertices. This always forms a circle and thus builds a roundtrip.

For building a path which optimizes all constraints, the three introduced measures for badness, number of shared edges and the number of turns are used. The badness function is incorporated into a different force function which can assign positive and negative badness values to edges. Furthermore, the force function uses the cost of the face and a vector $\vec{d} = \vec{p} - \vec{C}(f)$ which is built from the geometric center $\vec{C}(f)$ of a face to any point $\vec{p}$

After the tour has been created, it will be smoothed to reduce the complexity.

Partial Shortest Paths

Computational Complexity Aside from introducing two methods to calculate roundtrip tours for running, this paper also presents a proof for the computational complexity of their Simple and Relaxed Jogging Problems. The authors show NP hardness by reduction of Hamiltonian Cycle to the optimization problem corresponding to their original problems.

TODO add the actual proof?

TODO other jogging route paper

3.2.3 Computing Cycling Routes

TODO cycling paper Tour4Me

TODO other cycling paper

3.3 Apps that assist with sports

Aside from Tour4Me, RouteLoops and RouteYou, another prototype for running route recommendations has been developed. In the corresponding paper[14], the authors express the problems with existing apps, some of which have already been identified in the introduction of the two websites (see 3.2.1). They also stress, that most research either concentrates on shortest paths or - if it is research and app development specifically for running - on the assistance with the training itself rather than finding a good route. Apps like *Runtastic*[4], *Sprtractive*[5] or *Strava*[6] are designed to help runners track the tours they already ran. They measure pace, position, height meters and several other stats to then be able to present the user feedback of the run they've done. Planning a route is not one of the features these apps offer. And even apps that are meant to assist with the training and which create a plan like *Trainingpeaks*[7] or *SportTracks*[8] do not offer a feature to create routes or roundtrips with a set of preferences[14].

A German app that is meant to provide suitable routes for a variety of different outdoor sports - *Komoot*[9] - does offer a route selection. However, it relies only on tours other users have planned and added. No customization or route creation is offered here. As the authors of the paper "Recommending Running Routes: Framework and Demonstrator"[14] pointed out in their user study, it is very important to take user preferences into account. No participant of the study decided to try out a route another member had recorded, which further stresses the importance of personalized route generation[14].

[4] https://www.runtastic.com/
[5] http://sportractive.com/
[6] https://www.strava.com
[7] https://www.trainingpeaks.com/
[8] https://sporttracks.mobi/
[9] https://www.komoot.de/

Chapter 4

Implemented Changes

This work extends Tour4Me, which is an application written in C++ and HTML. The implemented interface uses C# as its programming language to enable easy porting of the webapplication to a desktop or mobile application. To improve the query times, a spatial database was added. Reasons for and positive effects of this decision are described in the following section.

Furthermore, not only the language and data access was changed. New options and parameters to improve the customizability of preferences for a generated tour were added as well. These changes had to be incorporated into an upgraded frontend design (see sections 4.1.3 and 4.3) as well as into the backend and all solvers (see section 4.2).

4.1 Application

4.1.1 New Architecture

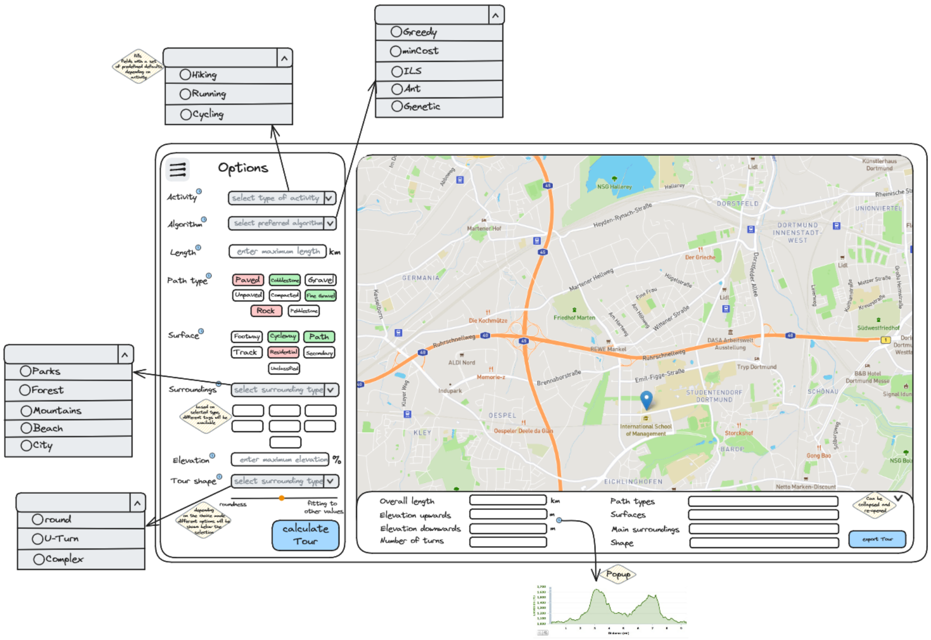

Figure 4.1: Design concept for the frrontend view, including descriptions for drop-downs and pop-ups

4.1.2 Database

4.1.3 Interface and Frontend changes

4.2 Algorithmic changes

4.2.1 Ant Colony

4.2.2 Genetic Algorithms

4.2.3 Simulated Annealing

4.3 Parameter changes

Chapter 5

Evaluation

Chapter 6

Conclusion

6.1 Results

6.2 Future Work

Appendix A

Source Code

Bibliography

[1] *Handbook of Metaheuristics.*

[2] Babaoglu, O., H. Meling and A. Montresor: *Anthill: a framework for the development of agent-based peer-to-peer systems.* In *Proceedings 22nd International Conference on Distributed Computing Systems*, pages 15–22. IEEE. ISSN: 1063-6927.

[3] Biddle, Stuart J. H.: *Psychological benefits of exercise and physical activity.* 2(2):0099–107. Number: 2.

[4] Blondiau, Thomas, Bruno van Zeebroeck and Holger Haubold: *Economic Benefits of Increased Cycling.* 14:2306–2313.

[5] Bräysy, Olli and Michel Gendreau: *Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms.* 39(1):104–118. Number: 1 Publisher: INFORMS.

[6] Buchin, Kevin, Mart Hagedoorn and Guangping Li: *Tour4Me: a framework for customized tour planning algorithms.* In *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, pages 1–4. ACM.

[7] Cekin, Resul: *Psychological Benefits of Regular Physical Activity: Evidence from Emerging Adults.* 3(10):710–717. Number: 10.

[8] Cherkassky, Boris V., Andrew V. Goldberg and Tomasz Radzik: *Shortest paths algorithms: Theory and experimental evaluation.* 73(2):129–174. Number: 2.

[9] Delling, Daniel, Peter Sanders, Dominik Schultes and Dorothea Wagner: *Engineering Route Planning Algorithms.* In Lerner, Jürgen, Dorothea Wagner and Katharina A. Zweig (editors): *Algorithmics of Large and Complex Networks: Design, Analysis, and Simulation*, Lecture Notes in Computer Science, pages 117–139. Springer.

[10] Dorigo, M., V. Maniezzo and A. Colorni: *Ant system: optimization by a colony of cooperating agents.* 26(1):29–41.

[11] Gemsa, Andreas, Thomas Pajor, Dorothea Wagner and Tobias Zündorf: *Efficient Computation of Jogging Routes.* In Bonifaci, Vincenzo, Camil Demetrescu and Alberto Marchetti-Spaccamela (editors): *Experimental Algorithms*, Lecture Notes in Computer Science, pages 272–283. Springer.

[12] Irnich, Stefan, Birger Funke and Tore Grünert: *Sequential search and its application to vehicle-routing problems.* 33(8):2405–2429. Number: 8.

[13] Lawler, E. L. and D. E. Wood: *Branch-and-Bound Methods: A Survey.* 14(4):699–719. Number: 4 Publisher: INFORMS.

[14] Loepp, Benedikt and Jürgen Ziegler: *Recommending Running Routes: Framework and Demonstrator.*

[15] Madkour, Amgad, Walid G. Aref, Faizan Ur Rehman, Mohamed Abdur Rahman and Saleh Basalamah: *A Survey of Shortest-Path Algorithms.* Issue: arXiv:1705.02044.

[16] Mueller, Florian 'Floyd', Shannon O'Brien and Alex Thorogood: *Jogging over a distance: supporting a "jogging together" experience although being apart.* In *CHI '07 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '07, pages 2579–2584. Association for Computing Machinery.

[17] Nystoriak, Matthew A. and Aruni Bhatnagar: *Cardiovascular Effects and Benefits of Exercise.* 5:135.

[18] O'Brien, Shannon and Florian "Floyd" Mueller: *Jogging the distance.* In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 523–526. ACM.

[19] Oja, P., S. Titze, A. Bauman, B. de Geus, P. Krenn, B. Reger-Nash and T. Kohlberger: *Health benefits of cycling: a systematic review.* 21(4):496–509. Number: 4 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1600-0838.2011.01299.x.

[20] Reinelt, Gerhard: *The Traveling Salesman: Computational Solutions for TSP Applications.* Springer. Google-Books-ID: fa6rCAAAQBAJ.

[21] Ropke, Stefan: *Heuristic and exact algorithms for vehicle routing problems.*

[22] Ruegsegger, Gregory N. and Frank W. Booth: *Health Benefits of Exercise.* 8(7):a029694. Number: 7 Publisher: Cold Spring Harbor Laboratory Press.

[23] Sommer, Christian: *Shortest-path queries in static networks.* 46(4):1–31. Number: 4.

[24] Szabo, Attila and Júlia Ábrahám: *The psychological benefits of recreational running: A field study.* 18(3):251–261. Number: 3 Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/13548506.2012.701755.

[25] Vina, J, F Sanchis-Gomar, V Martinez-Bello and Mc Gomez-Cabrera: *Exercise acts as a drug; the pharmacological benefits of exercise.* 167(1):1–12. Number: 1 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1476-5381.2012.01970.x.

[26] Wang, Xueyang, Chonghua Liu, Yupeng Wang and Chengkai Huang: *Application of ant colony optimized routing algorithm based on evolving graph model in VANETs.* In *2014 International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pages 265–270. Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1882-5621.

[27] Wankel, Leonard M. and Bonnie G. Berger: *The Psychological and Social Benefits of Sport and Physical Activity.* 22(2):167–182. Number: 2 Publisher: Routledge.

[28] Wayahdi, Muhammad Rhifky, Subhan Hafiz Nanda Ginting and Dinur Syahputra: *Greedy, A-Star, and Dijkstra's Algorithms in Finding Shortest Path.* 2(1):45–52. Number: 1.

List of Figures

Algorithmenverzeichnis

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den January 29, 2024

Lisa Salewsky