Doctoral School of Economics

THESIS OF DISSERTATION

Sándor Apáthy M.

A Model of a Touristic Recommender System

Supervisor:

Péter Tallos, CSc

Budapest, 2016

Corvinus University of Budapest
Department of Mathematics

THESIS OF DISSERTATION

Sándor Apáthy M.

A Model of a Touristic Recommender System

Supervisor:

Péter Tallos, CSc

Table of Content

1. Introduction

There are few worse situations I could imagine than getting lost in a foreign city. It is even worse if the language barriers keeps us away from the chance of getting help. Though it has happened to me several times, what really drives my research efforts is rather the aim of designing the model of a touristic recommender system, that might later be the base of a mobile application. Such an instrument would possibly be the best partner during our journeys feeding us with filtered and tailor-made information as a decision support engine. I have always dedicated my research to solve practical problems, so advocating someone in discovering a new place or an immersive experience would be considered as the ultimate success measure of my work.

1.1 Research objectives

In my life I have the pleasure to wander so many cities, although it has always been a challenge to create such a tour plan that efficiently uses my disposable resources (namely my time and budget). The objective of my dissertation is to design the theoretical foundations of a touristic recommendation system, that is sufficiently precise and capable of designing tailor-made itineraries in a foreign city, taking into account the user's needs and constraints. Implementing the user preferences and constraints into the model is nontrivial, however we attempt to design a model that considers the user’s preferences. First of all we develop a statistical method to estimate the user’s velocity while accounting for the steepness of the ground, the user’s current physical conditions and many other circumstances. Then we design a model that is capable of providing personalised recommendation for tourists based on minimal information given previously by them. We use techniques of recommender systems to map the preferences of users accurately and transform them to evaluations of touristic hotspots. By using above information we designed a routing algorithm that creates personalised city tour plans for users and offers visits in desired POIs organised into effective routes.

In this paper we intend to answer questions below:

1. How can we prepare raw GPS tracklogs that allows us to perform our analysis? Is there a digital elevation model (DEM) that approximates well the actual height values?

2. Is there a function that describes more precisely than the current solutions the relationship between the steepness of the ground and the velocity of the hiker

3. How can we design a method to estimate the hikers travel time more precisely than the current solutions?

4. How can we create a touristic recommender system that provides individualised recommendations based on minimal information given by the user?

5. Is the a possibility to classify the users based on the information provided by them? How can we design the questionnaire of the recommender system?

6. How can we accurately describe the user preferences related to tourist attractions by a utility function. What kind of objective function leads our routing algorithm to practically acceptable route plans?

7. How can we design an algorithm to solve the Team Orienteering Problem (TOP) that keeps run time low and enables its implementation in a mobile application?

1.2. Thesis Structure

In this paper according to our previously stated research objective we follow ternary structure. We dedicate the 2nd chapter to the Estimated time of Arrival problem and we present a novel method to estimate the travel time of a hiker considering many circumstances. We utilise more than 2400 tracklogs to perform our estimations that is unique according to the literature. Based on NASA data on Earth's surface we developed a DEM model to substitute GPS elevation data that is rather unreliable. We smoothen longitude-latitude pairs of the raw GPS tracklogs with Kalman-filter to eliminate GPS noises. We follow Tobler's earlier results estimate a function describing the relation between ground steepness and hiker's velocity. Later we use this function to our two novel hiking time estimation method.

To fulfil the needs of tourist and to recommend tailor-made tourist attraction lists we have to be able to understand their preferences and capabilities. The 3rd chapter provides a brief introduction to the theory and history of Recommender systems. Then we perform an empirical study to obtain sufficient information from the users to determine classification of tourist types that enables us to give more accurate recommendations. For this purpose we designed a list of touristic factors (such as Museum, Art, History, etc.) to describe each POI with the linear combination of these factors.

This content based method helps us to provide recommendations to users based on minimal information from the beginning.

In the 4th chapter we present a brief summary on the types of routing problems and the history of the algorithmic approaches. By using the this theoretical background we design a heuristic routing algorithm to solve the TOP. Our added value lies in the novel objective function which enables us to adopt the user's preferences rather collecting profit units at each vertex. The graph that symbolise the city is created based on the information learnt during the previous chapters: at each vertex we can earn a profit determined by the recommender system, while the cost of each arc is calculated based on our travel time estimation method. To facilitate the understanding of the model's structure we created a data and process flow chart (*fig. 1*)

The results of this paper are summarised in the 5th chapter where we also designate the directions of our future research.

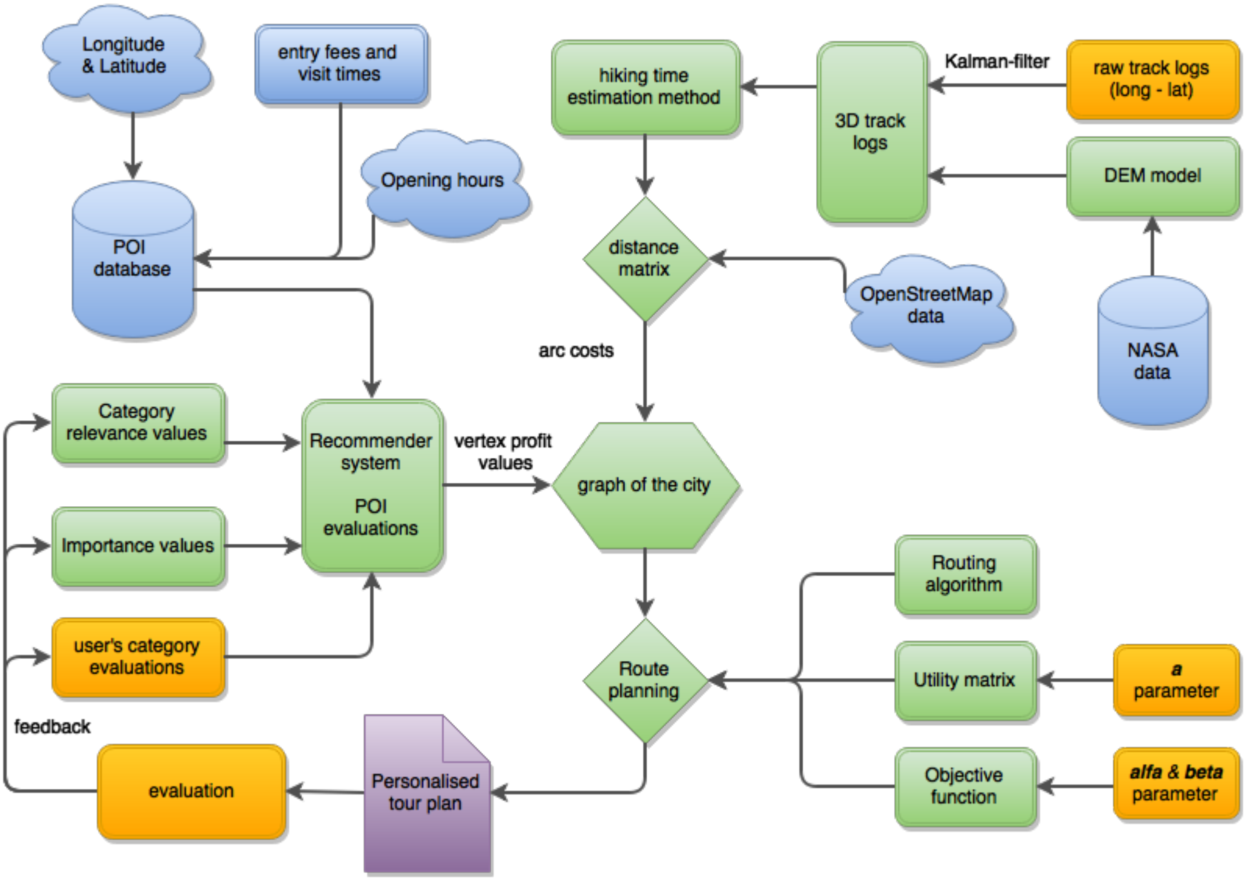


Figure 1: Data and Process flow of the Touristic Recommender system

2. Hiking time estimation

2.1. Problem formalisation

Since ancient Roman Empire there has been several attempts to estimate the speed of hikers. The most commonly used rule of thumb dates back to 1892, when a Scottish mountaineer, William Naismith described his estimation [1]: a man with average physical conditions can walk 3 miles (4827,9m) in 1 hour on flat ground , while he needs additional 1 hour for every 2000ft (632m) elevation. So practically 1 unit of elevation equals to 7,92 units of horizontal walk. There has been many attempts to refine Naismith's estimation (Langmuir [2], Aitken [3]), including Waldo Tobler [4], who conjectured exponential relationship between steepness of the ground and the velocity of the hiker. We compared the hiking time estimations on *Figure 2*.

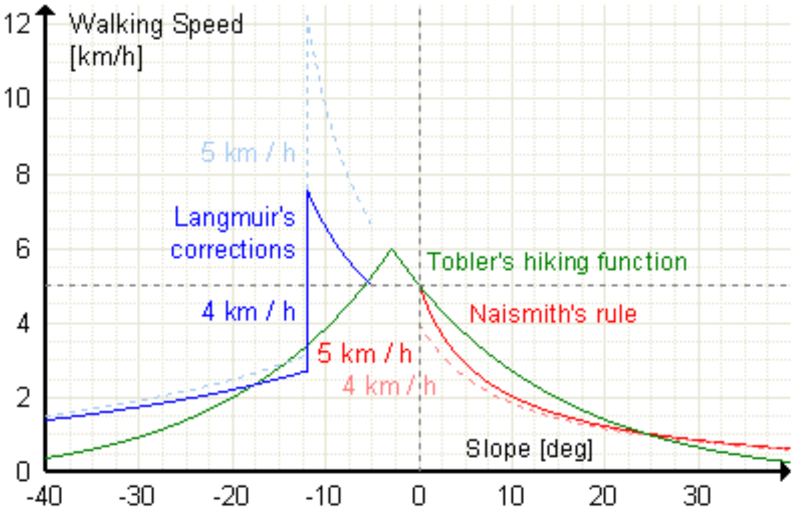


Figure 2: Comparison of hiking time estimations

Based on Tobler's concept we performed a novel function that refines Tobler's results and enables us to have more accurate estimation of hiking times.

Our database consists of 2400 tracklogs (after removing outliers). In case of each tracklog we removed the raw GPS elevation data and substituted with the elevation values calculated by our DEM. The DEM is based on NASA Earth's surface data (that assigns a single elevation value to every 30mx30m square of the surface). Our DEM model uses bilinear interpolation (that we summarised in *Appendix A*) to calculate elevation estimation and as a result it provides approximation of the Earth's surface. According to our comparison between our DEM and Google Elevation API the average difference between the two estimations is 13cm. It is an excellent result (by assuming Google as a good benchmark) considering that error's of mobile devices can be higher than 150m. The Longitude and Latitude values of raw GPS tracklogs have been smoothened by Kalman-filtering method. Thus we obtained adjusted 3D tracklogs, and standardised to 20 second long logs. Based on these logs steepness (*m*) and velocity (*v*) values we calculated the average speed of hikers in every 1/8 degree of steepness and fitted function in R with linear model package (that uses QR matrix decomposition method).

$$v(m) = \begin{cases} e^{2.3203m+0.4462} \mid m \in (-\infty; -0,15) \\ p(m) \mid m \in [-0,15; 0,15] \\ e^{-2.4672m+0.3769} \mid m \in (0,15; \infty) \end{cases}$$

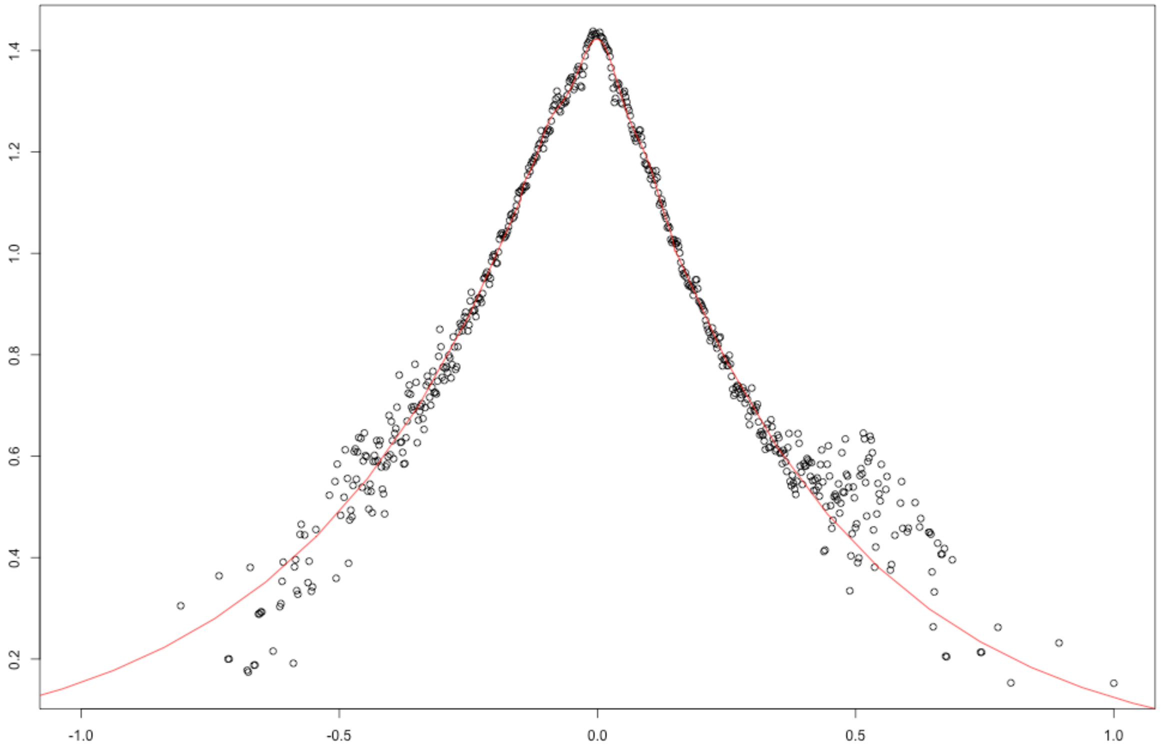


Figure 3: Fitted steepness-velocity function

Based on the fitted v(m) function we developed two novel hiking time estimation method calculated as follows:

2.2. The steepness based method

Let m_i denote the steepness of the ground of the ith log of a trail. Let us estimate the velocity of the first log with $v(m_1)$ then after measuring back the actual results (v_1)on the first log, adjust our estimations with the ratio of actual and estimated values, that is $b_1 = v_1/v(m_1)$. We consider this as the estimated personal fitness factor of a particular hiker, that enables us to adjust our estimations dynamically through the trail. After observing the fitness factor of the second log, we can estimate the 3rd log's velocity with $(b_1+b_2)v(m_3)/2$, i.e we calculate the overall fitness factor as the arithmetic mean of the observed fitness factors assigned to each log. More generally we estimate the velocity of the nth log as follows:

$$v_n^* = \frac{\sum_{i=1}^{n} b_{i-1}}{n-1} v(m_i)$$

By adjusting our initial estimation with the dynamically calculated fitness factor we can amend the estimated hiking and account for other circumstances, as the actual physical state of the hiker or the weather. Though during the first logs (namely in the first 10%) we observed that the estimation has poorer results due to the high variance of the fitness factors, after obtaining a stable fitness estimation we performed far better results comparing to earlier estimation methods (e.g. Pitman et al. [5]).

2.3. Mean velocity based method

The idea of this method is extremely simple: we estimate the first 20% of the trail with the *v(m)* fitted steepness-velocity function and gather the observed velocity values of each log. After this test period we estimate the velocity of the nth log of the trail with the arithmetic mean velocity of the *n-1* previous trails:

$$v_n^* = \frac{\sum_{i=1}^{n} v_{i-1}}{n-1}$$

During the first 20% of the trail this estimation has typically yield poor estimation, hence we substituted the mean velocity based estimations with the *v(m)* values.

We compared our results to the existing solutions (we considered Tobler's curve as the most sophisticated and accurate one) and performed a test on the available 2400 tracklogs (splitting it to a 75-25% learning and test dataset). On the learning dataset we fitted the *v(m)* curve, then we used it on the test dataset to estimate hiking time) and we performed 10 iterations. To measure the goodness of our estimations we partitioned each tracklog separately to 100 equal pieces and for the pth percentile of the tracklog we calculated the estimated hiking time of the remaining trail (denoted r^*_{ip} for the ith tracklog) comparing with the actual remaining time (r_{ip}). We used the Mean Absolute Relative Error (MARE) measure for our estimations:

$$MARE(p) = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{r_{ip} - r^*_{ip}}{r_{ip}}\right|$$

In *Table 1.* we summarised the results of the test (highlighting the mean of 10 iterations). The Wech-tests performed (based on 100 iterations) has rejected the null hypothesis that the MARE value's means of any of the above 3 methods are equal, hence the proposed two hiking time estimation methods provided significantly better results that the earlier estimations (Pitman's method was based on a multivariate regression and resulted in 18% mean MARE value). On fig. 4 we illustrated the results (red: steepness based model, black: mean velocity based model, blue: Tobler's estimation). Our methods seems to perform relatively better on the 15-80% percent of the trail. At the beginning do to the variance of the estimated factors the errors are higher, and at the beginning the smaller errors seem to be relatively high as we compare them to a smaller value of the remaining hiking time.

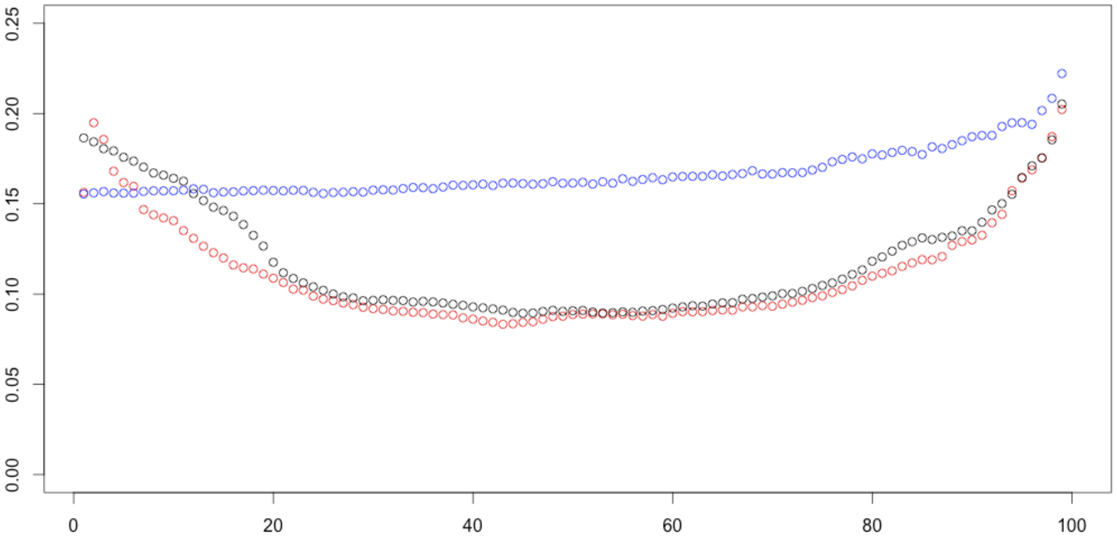


Figure 4: MARE values of the 3 estimation methods

We also tested whether we can reach better MARE results by applying the steepness based method with the Tobler-curve instead of our fitted *v(m)* function. According to the test results our method's mean MARE value (11,12%) was significantly better than the Tobler-curve based solution (13,48%), see *Figure 5*.

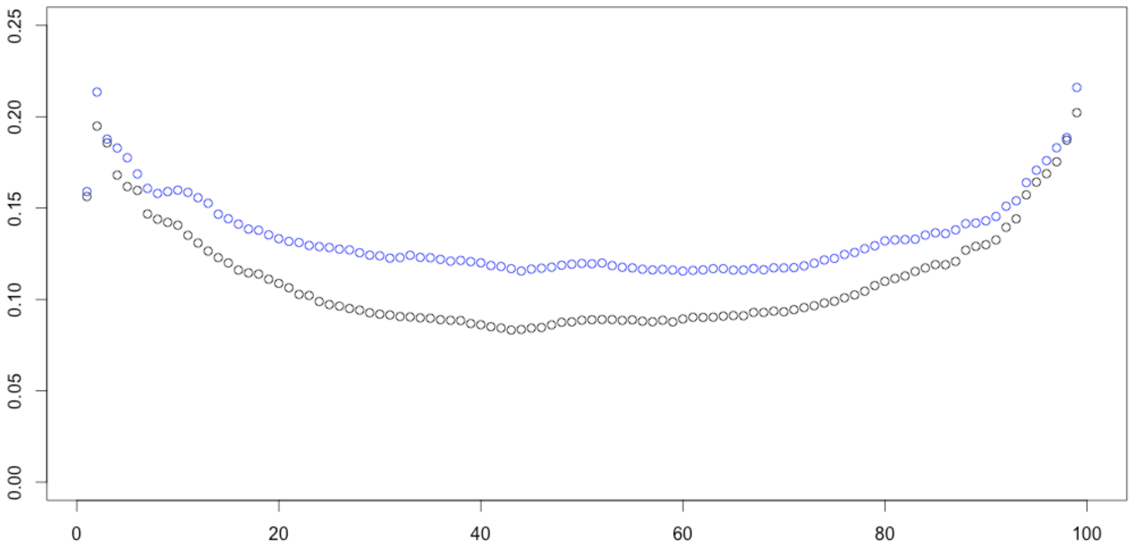


Figure 5: MARE values comparison for *v(m)* and Tobler-curve based methods

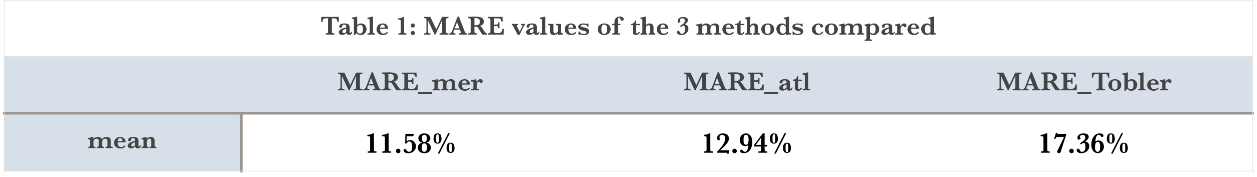


Table 1: MARE values of the 3 methods compared

	MARE_mer	MARE_atl	MARE_Tobler
mean	11.58%	12.94%	17.36%

2.4. Conclusions and future work

By following Tobler's initial idea to conjecture a relationship between steepness of the ground and velocity of the hiker we refined his earlier results. We developed two novel methods to estimate hiking time using only our fitted steepness-velocity curve and the observed velocity values of previous logs. The tests were performed on an outrageously large dataset (2400 tracklogs) and resulted in 11,58% and 12,94% Mean Absolute Relative Error compared to Tobler's result of 17,36%. In the near future we intend to broaden the list of explanatory variables used in our model to enhance the accuracy of our estimations. In case we would be able to acquire a larger dataset where we could also identify tracklogs related to the same user, we could probably design a collaborative estimation method where could utilise previous trails of the user and other users who are similar to him to adjust our predictions.

3. Recommendation Systems

3.1. Problem formalisation

In our everyday life we make decisions countless times, sometimes even unnoticed. How to dress up in the morning considering our schedule? Which menu to pick in the restaurant? Which television to buy? Thousands of these everyday questions have to be answered besides couple of crucial ones day by day. To make the correct decision in some cases we ask our friend's opinion or an expert, but there are other options that raised recently to support us in our decisions. We not necessarily need to nag the librarian or the vendor at the bookstore finding a book that would fit to our mood and preferences, if we have the possibility using Amazon to recommend us a book based on our previous readings. Youtube also recommends audiovisual content to its users based on their previous views. The advantage of using Youtube's recommendation engine instead of asking our friends lies in the immense content it sees through, while our friends' overall knowledge is just a fraction of that. I would like to avoid even the semblance of advising the substitution of our friends and family with an engine, perhaps I would urge those who are open to innovation to use these websites and applications as decision support tools that might spare some time to them or facilitates new experiences.

Based on dozens of recommendation system definitions ([6], [7], [8]) I have encountered during my research hereby I intend to determine a **definition** that synthesise the previous one, though also eliminates unnecessary aspects: **a Recommendation system is an information filtering system that supports the user to make a decision in a given situation by narrowing the set of available options and ranking options considering the context. Ranking can be done based on the user's preferences expressed either explicitly or implicitly, and it can take into account similar user's behaviour observed previously in similar situation.**

3.2. The proposed model

As we intend to build a touristic recommendation engine from scratch we are facing with the well known problem of cold start, i.e. the scarcity of information. Thus we decided to build a hybrid system consisting of two modules:

- Content based module requires the designer to understand the structure of the product to be recommended. Maybe the best example is the Music Genome Project [9] dates back to the late

90s that where the project scope was to understand the structure and factors of music that influences people when deciding which one they prefer or dislike. There has been more than 400 individual factors identified during the project and Pandora Radio's recommender system is based on this achievement even nowadays. By following this idea we initially identified 17 individual factors to describe with their linear combinations Point of Interests (POI) related to tourism (see *Table 2*). For this purpose we also assigned relevance values for each factor in case of each POI, e.g. The Hungarian National Gallery is described as Museum/art, Top sight, History/culture, with relevance numbers of 3, 3, 2 respectively. As per our experience almost every POI can be described by 3 factors at most. Thus we created a test instance of 150 POIs using the touristic attraction of Budapest (and 350 further POIs for London and Paris) containing the locations (longitude - latitude values), the factors and corresponding relevance values (0 to 3), the importance values (0 to 3) to signal how frequently a POI is visited. The database has been used while we performed a questionnaire on our website (www.travelschedule.org), though this is related to the second module.

Table 2: Touristic factors

top sight	church/religious place	museum/art
history/culture	architecture/facade	monument
street/square	vista point	park/nature
university/science/technology	amusement/family/children	bath/sport/recreation
market/local food	cafe/restaurant	theatre/entertainment/cinema
music/bars/nightlife	shopping/fashion	

- Knowledge based module [10] serves our needs of gathering information from the users on their preferences. On our website we asked the test subjects to evaluate the 17 factors according to their preferences, then evaluate the 4 recommended list of POIs (15 per each) on a scale 1 to 10, that have been calculated based on 4 different methods:

    - KPI1: we calculate the scalar product of the factor evaluation vector with the relevance vector multiplied by the importance number.

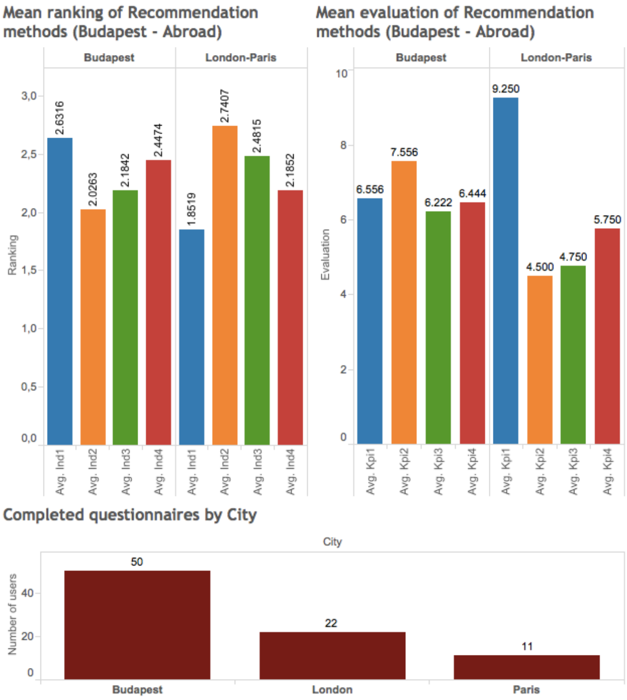
    - KPI2: we multiply the value of KPI1 in case importance number equals to 2, then divide the value by the number of factors with positive relevance number.

    - KPI3: here we multiply the value of KPI2 with the number of factors that has the highest relevance number (3) and the highest corresponding factor evaluation (3) at the same time.

- KPI4: the first 7 items of the list is identical with the fist 7 items of KPI1, the remaining 8 fields are filled with items of KPI2, avoiding any duplications.

3.3. Experimental Results

Hence KPI1 usually recommends the popular places, while KPI2 and KPI3 aims to come up with some less known ones. KPI4 is clearly a mixture the two concept. The questionnaire has been completed by 50 user for Budapest and 33 for London and Paris (see *Fig. 6*). In case of Budapest clearly KPI2 performed the best, while for Paris and London most of the users found KPI1 to recommend the most attractive places. Considering almost all the test subjects were Hungarian I presume they preferred to see the well known attractions instead of the latent sights.


Mean ranking of Recommendation methods (Budapest - Abroad)

Mean evaluation of Recommendation methods (Budapest - Abroad)

Completed questionnaires by City

Figure 6: Evaluation results

Based on the evaluations we have calculated the correlation matrix of the 17 factors and grouped factors that are in close relation with each other. As an interpretation of the grouped factors we have identified 6 different tourist types described by preferring the corresponding factors (see *Table 3*).

We investigated the evaluations of the segmented user groups that resulted in different recommendation strategies for the tourist types (see the summary of KPI evaluations in *Table 4*).

Table 3: Identified tourist types

Touristic factors	Culture lover	Nature lover	Families	Young	Mundane	Gourmet
museum/art	x					
park/nature		x				
architecture/facade	x					
history/culture	x					
shopping/fashion				x	x	
vista point		x				
top sight				x		
music/bars/nightlife				x	x	
market/local food				x		x
street/square				x		x
monument	x					
bath/sport/recreation				x		x
theatre/entertainment/cinema			x	x	x	
cafe/restaurant				x	x	x
church/religious place	x					
university/science/technology			x	x		
amusement/family/children			x			

Table 4: Ranking of recommendation methods for tourist types

Touristic factors	Eljárások	Culture lover	Nature lover	Families	Young	Mundane	Gourmet
Budapest	KPI1	3	4	2	4	4	4
	KPI2	1	1	1	3	3	2
	KPI3	2	2	4	2	2	3
	KPI4	4	3	3	1	1	1
London - Paris	KPI1	1	1	1	1	1	1
	KPI2	4	4	4	3	4	3
	KPI3	3	3	3	4	2	2
	KPI4	2	2	2	2	3	4

By applying the key takeaways of the KPI evaluations of the user groups we created a simple decision process which KPI to use:

- In case of a city abroad, pick KPI1

- In case of inland, and the mean overall evaluation of factors above 2.2 points: pick KPI1 again
- For Culture lovers pick KPI2 (still inland)
- If the person is not culture lover but young, pick KPI4
- For everyone else in inland recommend according to KPI2.

With this method we 86,4% success rate (i.e. picked the best KPI in 76 cases out of 88).

3.4. Conclusions and future work

We have created a hybrid (Feature augmenting) recommender system (see Melville et al. [11]) consists of a Knowledge based and a Content based module. Thanks to their characteristics we have overcome the cold start problem and have the ability to recommend users touristic attractions with minimal information shared in the beginning. By using the identified 6 tourist types we have been able to refine the accuracy of our initial recommendations.

As part of our future research plan we would like to review the results of the existing 4 KPIs and probably design some more competitive versions. In case we would be able to acquire a larger evaluation dataset we consider refining or initial 17 factors and identified 6 user groups by using matrix factorisation technique. Additional evaluation dataset would also enable us to build a collaborative module to our recommender system to enhance the accuracy of our current recommendations.

4. Route Planning

4.1. Introduction

In the 3rd pillar of this paper we present a novel, heuristic routing algorithm that aims to provide personalised tour plans for the users. We strongly build on the achievements of previous chapters, namely the routing algorithm optimises tours on an individualised graph where the profits collected at each vertex are derived from the previously designed recommender engine and the cost of each arc (travel time) has been determined by our hiking time estimation method.

We also consider important to highlight that as of our knowledge there is no such existing solution, that is able to deliver route plan for P days, assigned and optimised to a dedicated hotel. Besides the algorithm itself we also designed a utility function and an objective function to adjust route plans to user preferences.

4.2. Problem formalisation

We intend to solve the Team orienteering problem (formalised first by Butt and Cavalier in 1994 [12]), though as we do not necessarily agree with the maximisation of sum of profit points by visiting each arc, we recommend a new approach. This objective function has been inherited from the Traveling Salesman Problem (formalised and solved first by Karl Menger in the late 20s [13]), where we can attach particular meaning to collecting profit points by visiting the vertices. As per my understanding tourists are not interested in attraction they have evaluated poorly (namely assigning 4 or less points on a scale 1 to 10). Hence we decided to design an objective function as an extension of the broadly applied version. Let the utility function be *u(s_i ,a),* where s_i is the evaluation of the vertex *i* by a certain user, and and *a* is a parameter that measures how much the user appreciates an attraction evaluated as *s* comparing to *s-1*. The utility function is designed as follows:

$$u(s_i, a) = \begin{cases} \frac{1-e^{-a(s_i - s^*)}}{a} \mid a \neq 0 \\ s_i - s^* \mid a = 0 \end{cases}$$

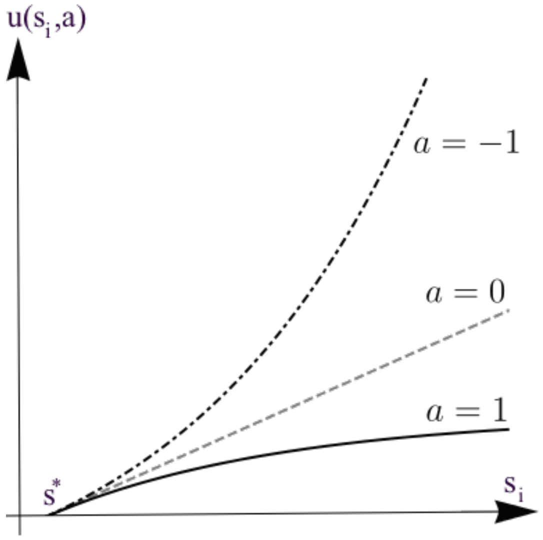


Figure 7: Utility function

The effect of parameter *a* on the evaluation is depicted on *Fig. 7*. By using the utility function determined above we have the opportunity to rule out POIs that are undesirable for the user (namely with evaluation under *s**). Thus we avoid visiting POIs that are very close to our planned route, despite their poor evaluation. We use the following denotement in the problem formalisation:

- *N* - number of vertices in graph *G(N,E)*
- *E* - number of arcs in graph *G(N,E)*
- *P* - number of available days
- *B* - daily budget constraint (the user only have to meet with the overall budget constraint *BP*)

- T_{max} - daily time limit (the user has to meet with the daly time limit every day, and he can only exceed it with 5%)
- s_i - evaluation of vertex i
- v_i - visit time of vertex i
- t_{ij} - travel time from vertex i to j
- τ_{ijp} - is equal to 1 if in the pth path the vertex j is visited right after vertex i, 0 in any other cases
- θ_{ip} - is equal to 1 if in the pth path the vertex i is visited, 0 in any other cases
- β - parameter of "laziness"
- α - parameter of the objective function to balance the importance of utility and visiting time - travel time ratio
- R - denotes the set of planned routes for P days
- b_i - entrance fee for vertex i

The objective is to design *P* routes for *P* days that maximises the objective function and observe the constraints. All vertices of the graph can be visited not more than once, except the hotel. Each route starts at the hotel and ends at the hotel (vertex number 1 and N denotes the hotel). The problem is formulated as follows:

$$max_{\tau_{ijp}} \left(\frac{\sum_{p=1}^{P} \sum_{i=1}^{N} \theta_{ip} v_i}{\sum_{i=1}^{N-1} \sum_{j=2}^{N} \tau_{ijp} t_{ij}^{\beta}} \right)^{\alpha} \times \left(\sum_{p=1}^{P} \sum_{i=1}^{N} \theta_{ip} u(s_i, a) \right)^{1-\alpha}$$

$$\sum_{p=1}^{P} \sum_{j=2}^{N} \tau_{1jp} = \sum_{p=1}^{P} \sum_{i=1}^{N-1} \tau_{iNp} = P$$

$$\sum_{p=1}^{P} \theta_{kp} \leqslant 1; \forall k = 2, ..., N-1$$

$$\sum_{j=2}^{N} \tau_{kjp} = \sum_{i=1}^{N-1} \tau_{ikp} = \theta_{kp}; \forall k = 2, ..., N-1; \forall p = 1, ..., P$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} \tau_{ijp} t_{ij} + \sum_{i=1}^{N} \theta_{ip} v_i \leqslant T_{max}; \forall p = 1, ..., P$$

$$\sum_{p=1}^{P} \sum_{i=1}^{N} \theta_{ip} b_i \leq PB$$

$$h_{ip} - h_{jp} + 1 \leqslant (N-1)(1 - \tau_{ijp}); \forall i, j = 2, ..., N; \forall p = 1, ..., P$$

$$2 \leqslant h_{ip} \leqslant N; \forall i = 2, ..., N; \forall p = 1, ..., P$$

$$\tau_{ijp}, \theta_{ip} \in \{0, 1\} \, \forall i, j = 1, ..., N; \forall p = 1, ..., P$$

The interpretation of the lines:

1. Objective function to be optimised, where the two considered factors are the sum of visit times divided by the sum of travel times and the sum of utilities earned by visiting vertices.
2. Every route starts at vertex 1 and ends at vertex N (both denotes the same hotel).
3. Every vertex is visited once at most.
4. Every route is connected severally.
5. The routes are meeting with the time constraint for each day.
6. The routes are meeting with the budget constraint for the overall tour of *P* days.
7. and 8. together grants avoiding cycles in the route according to Miller–Tucker–Zemlin [14]
9. The value set of τ_{ijp} and θ_{ip} is 1 or 0.

Where our problem differs from the literature is the objective function which is clearly an extension of the usual profit collecting (since it yields the same formula for $\alpha=a=0$).

4.3. Routing algorithm

We start presenting our routing algorithm with two functions:

Lexicographical ranking: We determine the rank of a set of vertices (C_1) to another set of vertices (C_2) considering the effectiveness of the constraints and the score threshold. More formally $L(C_1, C_2, sc, s^*)$, where *sc* denotes the resource that is more scarce (the estimation is going to be presented in the 3rd step of the algorithm) and C_1 is the set of vertices to be ranked based on the set of vertices in C_2. Let us calculate the two below measures:

$$\frac{\frac{u(s_i,a)}{u(s^*+1,a)}}{d^*(c_i,C_2)+v_i} \qquad \frac{\frac{u(s_i,a)}{u(s^*+1,a)}}{b_i}$$

where $d^*(c_i,C_2)$ denotes the mean distance between vertex c_i (element of C_1) and the elements of C_2. The nominator interprets the utility of s_i in the units of the utility of s^*+1 (which is the lowest score assigned to a vertex we consider to visit, see the first step of the algorithm). In case of the second measure we divide the utilities by the corresponding entrance fee. *$L(C_1, C_2, sc, s^*)$* first determines the scarce constraint, e.g. let it be the time. In this case it rank the vertices of C_1 based on the first measure, cuts the list into 6 equal pieces (the number of elements in the last group can differ), then it rerank the vertices within each group based on the second measure.

Identifying outliers: *$O(H, cr)$* determines the outliers for a given Hamiltonian-cycle considering the *cr* threshold parameter. First we calculate for each vertex of the Hamiltonian-cycle the sum of travel times of the inbound ($t_{i,be}$) and outbound ($t_{i,ki}$) arcs. Vertex i is considered to be an outlier in case:

$$t_{i,be} + t_{i,ki} > t_H^* + cr\sigma_H$$

where t_H* is the mean of inbound and outbound travel time and σ_H is the standard deviation. The value of *cr* differs in some cases, that we indicate later.

The algorithm

1. Simplification of the problem space: let us eliminate all vertices with evaluation less than or equal to *s**. We call the remaining vertices as the set of relevant points, denoted by C_r

2. Fixed vertices: let us appoint mandatory vertices, practically vertices with maximum evaluation ($\max\{s_i\}$).

3. Grouping: Let us first estimate which is the more scarce resource. As a rule of thumb we calculate the below two measures:

   - Take the sum of visiting times of relevant points and the average distance between them (once for each) and divide it by the overall time constraint PT_{max}

   - The sum of entry fees for all relevant points divided by the budget constraint *BP*

The higher is considered to be the more scarce resource (so *sc* is determined). Let us choose the vertices with the highest score (as those are mandatory) then for all the remaining ones use *$L(C_1, C_1, sc, s^*)$* and pick the first 5*P* points and add them to the mandatory vertices. Now determine an optimal Hamiltonian-cycle (including the hotel) on the vertex set, then apply *$O(H,1)$* to eliminate outliers.

4. Initial daily routes: The remaining vertices to be separated into P groups (without any overlap) by Hartigan-Wong-algorithm. For each cluster we determine the shortest Hamiltonian-cycle (including

the hotel), and choose the best result after 10 iterations. We consider a solution to be better in case the objective function's value is higher for the *P* routes altogether.

5. Refill: In case of any free capacities in each day we refill the routes with the remaining points from C_r, hence we use $L(C_r, C_i, sc, s^*)$ for each day i, then fill the routes until we reach any of the resource constraints.

6. Switch: We determine for each point in all existing routes the mean of the 3 lowest distances from the points of the regarding route (we also calculate this measure for the route contains the given point), and we assign the point to the route where the value of this measure is the lowest, and repeat it in 10 iterations.

7. Cut: In case a day exceeds the time constraint with more than 5%, we apply $L(C_i, C_i, sc, s^*)$ and eliminate the vertices from the last in the rank until we meet with the constraint.

8. Refill: If we still have free capacities at day i, we apply $L(C_r, C_i, sc, s^*)$ and insert points from C_r from the top rank until we meet with any of the constraints.

4.4. Experimental results

We have performed our experiment on a POI list consists of 150 touristic attractions in Budapest, including the visit times, entry fees, locations (longitude and latitude values) and the evaluations given by the user (that we assumed to be calculated based on our recommender system). The travel times are derived from the location applying OpenStreetMap API to calculate the distance matrix for 150 POIs and the hotel assigned to the user. Hereby we summarise our experiences:

- We obtain large bypasses in case of positive a parameter values, moreover the routes disintegrate when we apply low parameter values for α and β.
- In case $\alpha < 0.5$ we can only obtain acceptable routes if $\beta > 1.5$ and $a < -0.5$
- Generally we obtain relatively good results in case $a < - 0.5$; $\alpha > 0.5$ and $\beta > 1.5$
- Considering the chargeability and the visiting time - travel time ratio as a measure of goodness we found $\alpha = 0.75$; $a = -1$ and $\beta = 2$ parameter-set optimal in case of 1-2-3 and 4 days. We present in *Appendix B* a tour of 4 days calculated applying these parameter values. The profit collection case has shown poor results: though the chargeability is almost equal to our solution (95% average)

and the computation time is lower by 1 second, the visit time - travel time ratio is 1.43 comparing to 3.8 in case of our parameter values.

- We ran the algorithm 100 times to calculate a 3 day tour, and we obtained 3.73 seconds mean computation time (we ran our tests in R software on a laptop with following details: 3.8 GB RAM, Intel Core i3-3217U CPU, 1.80GHz × 4 proc). Though we cannot compare our results to Vansteenwegen et al. [15] or Gavalas et al. [16], as we solved a different problem, as a point of comparison we mention the results of Sylejmani and Dika [17], who presented a tour planner algorithm in 2011. The designed Taboo Search algorithm has created a tour of 3 days on a graph of 40 vertices with a mean computation time of 81.7.

- We also present the results of the test runs for each parameter combination (performing 20 iterations each). As we can see the in Fig. 8. the mean computation time increasing significantly between the tours designed for 2 and 3 days. The average visit time - travel time ratio and the average chargeability slightly decreases by increasing the number of days.

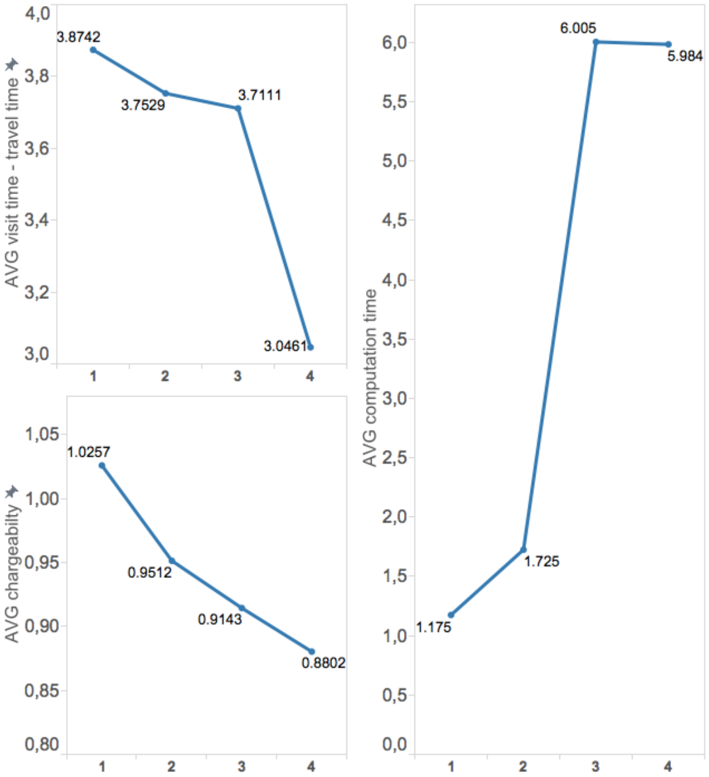


Figure 8: Test results of the routing algorithm

4.5. Conclusions and future work

We presented a modified version of the TOP, that - at least according to our intentions - focusing more on a practical problem formulation related to the Touristic trip design problem. As part of our endeavour we designed an objective function that adjust the problem formulation according to the user's preferences. A novel heuristic solution has been introduced to solve the TOP which provides an opportunity implementing in a mobile tour planning application considering its low computation time. As a future goal we would like to extend the algorithm with the ability to handle time windows which we consider the next milestone towards a practically useful solution. We also plan to refine the algorithm with looking over more hotels and appointing the one where the value of the objective function is maximum for the *P* days tour. We plan to replace the first clustering step with a method that identifies P vertices in sufficient distance from each other and builds trees including the hotel.

5. Thesis summary

1. **Designing two DEM solutions**

How can we prepare raw GPS tracklogs that allows us to perform our analysis? Is there a digital elevation model (DEM) that approximates well the actual height values?

The correction of the longitude and latitude values of the raw GPS tracklog has been performed by applying Kalman-filter. As the elevation data of the GPS tracklogs are unreliable we created 2 novel solution to approximate the Earth's surface. The firs Digital Elevation Model is based on bilinear interpolation, while the second approximated the surface with triangles. Both solutions provides very similar results to the values of Google DEM, which is considered to be nowadays the market standard.

2. **Refining the steepness-velocity function determined by Tobler**

Is there a function that describes more precisely than the current solutions the relationship between the steepness of the ground and the velocity of the hiker?

The relationship between steepness of the ground and the velocity of the hiker has been determined by Waldo Tobler in 1993 [4], unfortunately it has not been reconsidered or refined until now. Based on 2400 tracklogs we designed a novel function that enables us to valculate more accurate estimations of hiking time than by using the Tobler-curve. We would also highlight that one of the key reason of our success is the quality and quantity of tracklogs we used to our experiments, which is - as per our knowledge - outrageous in the literature.

3. **Designing two hiking time estimation methods**

How can we design a method to estimate the hikers travel time more precisely than the current solutions?

Naismith's rule of thumb [1] to estimate the hiking time has been used since 1892. In the era of mobile applications we have the opportunity to refine these approximations. We proposed two estimation methods that collect information during the trail and adapts the estimations dynamically. The mean velocity based method has resulted in 12.94% mean absolute relative error while the steepness based method was 11.58%. These two are the most accurate solutions as per our knowledge.

4. **Designing a Touristic Hybrid Recommender System**

How can we create a touristic recommender system that provides individualised recommendations based on minimal information given by the user?

As we are lack of sufficient quantity of evaluation at this stage of our research, we have created a POI database (consists of 500 POIs) and determined 17 touristic factors to describe each POI as a linear combination of the factors (content based module). The knowledge based module gathers the users' evaluations on the 17 factors, hence we can perform recommendations to users based on very limited information by applying our Feature Augmenting hybrid Recommender System.

5. **Classification of Tourists**

Is the a possibility to classify the users based on the information provided by them? How can we design the questionnaire of the recommender system?

As part of our experiment we have created a website where the users received 4 different recommendations after evaluating the 17 touristic factors. Based on the ratings related to the recommendations we have identified 6 different types of tourists. These groups has been created considering the high correlation values between the factors assigned to each group. By identifying tourist types we have been able to refine our initial recommendations.

6. **Objective function for the TOP**

How can we accurately describe the user preferences related to tourist attractions by a utility function. What kind of objective function leads our routing algorithm to practically acceptable route plans?

The broadly used objective function has been inherited from the Traveling Salesman Problem (formalised and solved first by Karl Menger in the late 20s [13]), where we can attach particular meaning to collecting profit points by visiting the vertices. As per our understanding tourists are not interested in attraction they have evaluated poorly (namely assigning 4 or less points on a scale 1 to 10). Hence we decided to design an objective function as an extension of the broadly applied version, which is clearly an extension of the usual profit collecting (since it yields the same formula for α=*a*=*0*).

$$u(s_i, a) = \begin{cases} \frac{1-e^{-a(s_i - s^*)}}{a} \mid a \neq 0 \\ s_i - s^* \mid a = 0 \end{cases}$$

$$C(\alpha, \beta, a, R) = \left(\frac{\sum_{p=1}^{P} \sum_{i=1}^{N} \theta_{ip} v_i}{\sum_{i=1}^{N-1} \sum_{j=2}^{N} \tau_{ijp} t_{ij}^{\beta}} \right)^{\alpha} \times \left(\sum_{p=1}^{P} \sum_{i=1}^{N} \theta_{ip} u(s_i, a) \right)^{1-\alpha}$$

7. **Creating heuristic algorithm to solve TOP**

How can we design an algorithm to solve the Team Orienteering Problem (TOP) that keeps run time low and enables its implementation in a mobile application?

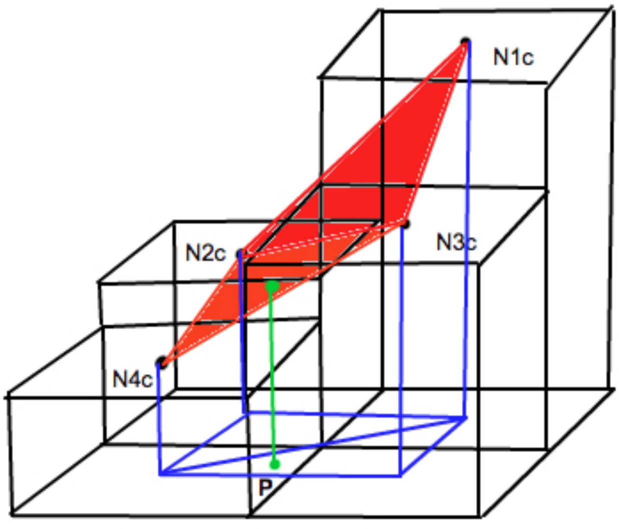
We presented a modified version of the TOP, that - at least according to our intentions - focusing more on a practical problem formulation related to the Touristic trip design problem. As part of our endeavour we designed an objective function that adjust the problem formulation according to the user’s preferences. A novel heuristic solution has been introduced to solve the TOP which provides an opportunity implementing in a mobile tour planning application considering its low computation time.

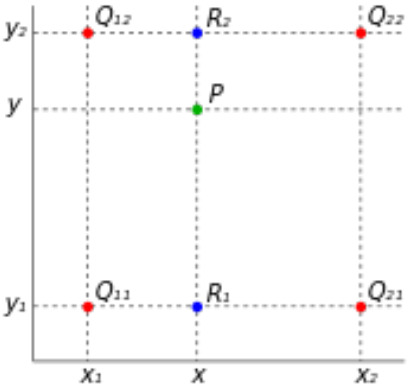
APPENDIX A

Let ($N_{1,c}$,$N_{2,c}$,$N_{3,c}$,$N_{4,c}$) denote the elevation values assigned by the NASA DEM to centers of 4 neighbor squares and ($Q_{1,1}$,$Q_{1,2}$,$Q_{2,1}$,$Q_{2,2}$) are their projections to the ground. Let *f(x)* denote the elevation value assigned to x point of the ground. Thus the bilinear interpolation of point P can be esimated as follows:

$$f(x,y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$
$$f(x,y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

$$\begin{aligned} f(x,y) &\approx \frac{y_2 - y}{y_2 - y_1} f(x,y_1) + \frac{y - y_1}{y_2 - y_1} f(x,y_2) \\ &\approx \frac{y_2 - y}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \right) + \frac{y - y_1}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \right) \\ &= \frac{1}{(x_2 - x_1)(y_2 - y_1)} \left(f(Q_{11})(x_2 - x)(y_2 - y) + f(Q_{21})(x - x_1)(y_2 - y) + f(Q_{12})(x_2 - x)(y - y_1) + f(Q_{22})(x - x_1)(y - y_1) \right) \end{aligned}$$

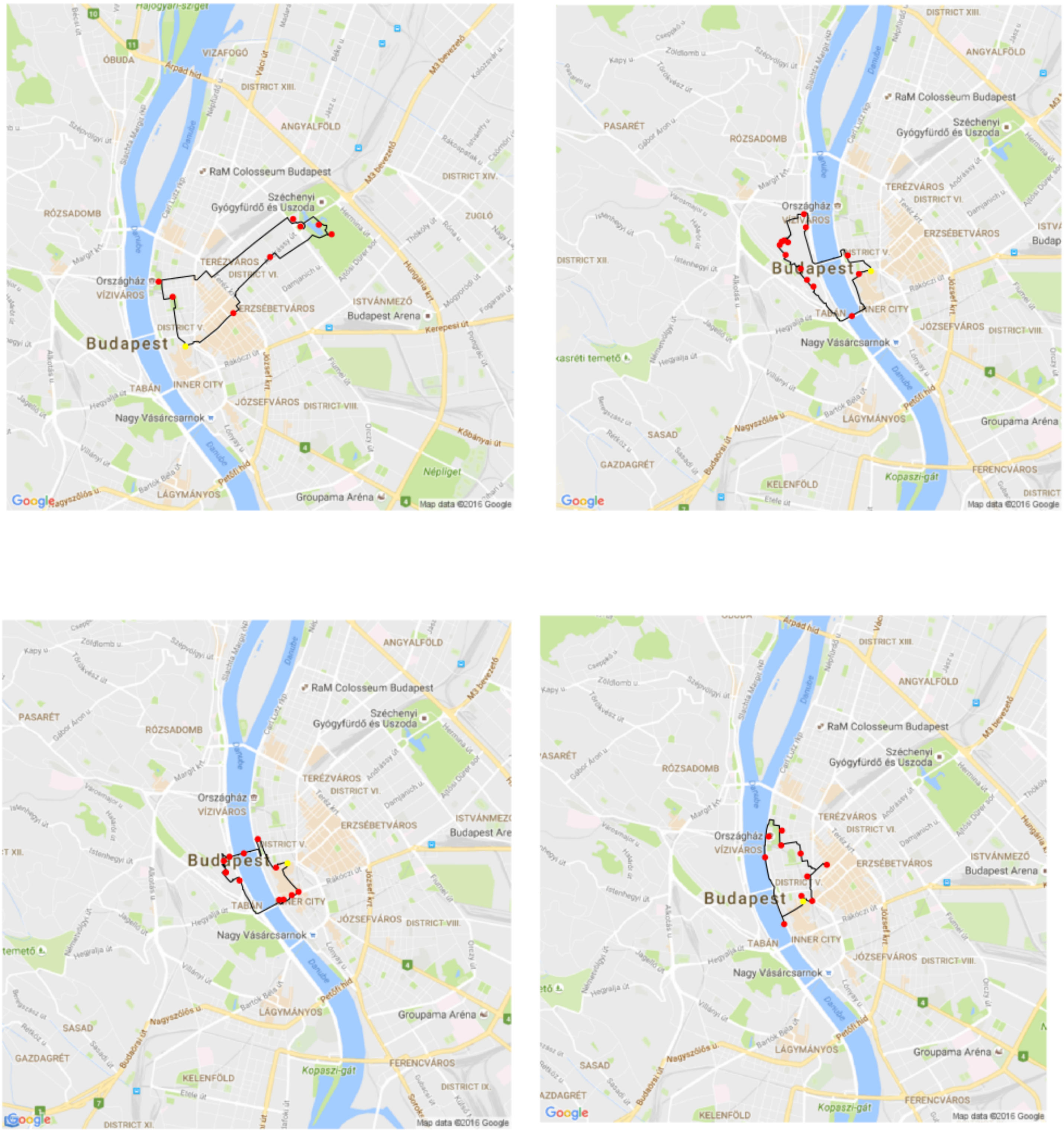




(source: https://en.wikipedia.org/wiki/Bilinear_interpolation)

APPENDIX B

4 day tour (α = 0.75; *a* = -1 and *β* = 2)



References

[1] W. Naismith: *Notes and queries*, [1892] Scottish Mountaineering Club Journal, Vol. 2, p. 133.

[2] R. Aitken (1977): *Wilderness Areas in Scotland*, unpublished Ph.D. Thesis. University of Aberdeen. Aberdeen.

[3] E. Langmuir (1995): *Mountaincraft and Leadership*, 3rd ed. Sportscotland.

[4] W. Tobler (1993): *Three presentations on geographical analysis and modeling: Non-isotropic geographic modeling speculations on the geometry of geography global spatial analysis*, National Center for Geographic Information and Analysis Technical Report, Vol. 93, No. 1, pp. 1–24.

[5] A. Pitman - M. Zanker - J. Gamper - P. Andritsos (2012): *Individualized hiking time estimation*, in Proceedings of the 23rd International Workshop on Database and Expert Systems Applications, pp. 101-105. doi: 10.1109/DEXA.2012.51

[6] U. Hanani - B. Shapira - P. Shoval (2001): *Information filtering: Overview of issues, research and systems*, User Modeling and User-Adapted Interaction, Vol. 11, pp. 203-259. doi: 10.1023/A: 1011196000674

[7] P. Melville - V. Sindhwani (2011): *Recommender Systems*, in C. Sammut - G.I. Webb (Eds.): Encyclopedia of Machine Learning, Springer. pp. 829-838. doi: 10.1007/978-0-387-30164-8_705

[8] http://en.citizendium.org/wiki/Recommendation_system

[9] M.H. Ferrara - M. P. LaMeau (2012): *Pandora Radio/Music Genome Project. Innovation Masters: History's Best Examples of Business Transformation*. Detroit. pp. 267-270. Gale Virtual Reference Library. doi: 10.5860/CHOICE.50-2756

[10] R. Burke (2000): *Knowledge-based Recommender Systems*, Encyclopedia of Library and Information Science, Vol. 69, No. 32, pp. 180-200. doi:10.1.1.21.6029&rank=1

[11] P. Melville - R.J. Mooney - R. Nagarajan (2002): *Content-Boosted Collaborative Filtering for Improved Recommendations*, in Processing of the 18th National Conference on Artificial Intelligence, pp. 187-192. doi: 10.1109/CSNT.2012.218

[12] S.E. Butt - T.M. Cavalier (1994): *A heuristic for the multiple tour maximum collection problem*, Computers and Operations research, Vol. 21, pp. 101-111. *A heuristic for the multiple tour maximum collection problem*

[13] K. Menger (1928): *Ein Theorem über die Bogenlange*, Anzeiger — Akademie der Wissenschaften in Wien — Mathematisch-naturwissenschaftliche, Klasse 65, pp. 264–266.

[14] C. Miller - A. Tucker - R. Zemlin (1960): *Integer programming formulations and travelling salesman problems*, Journal of the ACM, Vol. 7, pp. 326–329. doi:10.1145/321043.321046

[15] P. Vansteenwegen - W. Souffriau - G. Vanden Berghe - D.D. Van Oudheusden (2011): *The city trip planner: an expert system for tourists*, Expert Systems with Applications, Vol. 38. No. 6, pp. 6540–6546. doi:10.1016/j.eswa.2010.11.085

[16] D. Gavalas - M. Kenteris (2011): *A pervasive web-based recommendation system for mobile tourist guides*, Personal and Ubiquitous Computing, Vol. 15, No. 7, pp. 759–770. doi: 10.1007/s00779-011-0389-x

[17] K. Sylejmani - A. Dika (2011): *Solving touristic trip planning problem by using taboo search approach*, International Journal of Computer Science Issues, Vol. 8, Issue 5, No 3, pp. 139-149. doi: 10.1109/HIS.2012.6421351

Publication list

Publications in Hungarian language

Apáthy M., S. [2016]: *Egy heurisztikus útvonaltervező algoritmus többnapos túrák tervezésére*, Szigma Matematikai-közgazdasági folyóirat, Vol. 3-4, accepted

Apáthy M., S. [2016]: Az útvonaltervező algoritmusok történeti áttekintése, különös tekintettel azok turisztikai célú alkalmazásaira, Alkalmazott Matematikai Lapok, Vol. 33, elfogadva

Apáthy M., S. [2016]: A menetidőbecslés alkalmazásai, Közlekedéstudományi Szemle, accepted

Apáthy M., S. [2016]: Turistatípusok azonosítása - Egy lehetséges turisztikai ajánlórendszer, Vezetéstudomány, under review

Other Publications in Hungarian language

Apáthy M., S. [2016]: *Földfelszíni gyalogos közlekedés modellezése*, Innováció és fenntartható felszíni közlekedés Konferencia 2016. Budapest, Magyarország, 2016.08.29 - 2016.08.31, paper 22, ISBN: 978-963-88875-2-8

Apáthy M., S.[2011]: Fejezetek a modern közgazdaságtudományból – recenzió [Móczár József: Fejezetek a modern közgazdaságtudományból. Akadémiai Kiadó, Budapest. 2008. 608 oldal ISBN 9789630585378], Gazdaság és társadalom, Vol. 3, No. 3-4, pp. 189-194. ISSN 0865-7823

Publications in English language

Apáthy M., S. [2016]: *Personalised hiking time estimation*, Pure Mathematics and Applications, under publication

Apáthy M., S. [2016]: *Practical Route Planning Algorithm,* Periodica Polytechnica Transportation Engineeering, Vol 45, No. 2, accepted